# Platten-Scheduling und Stochastische Service-Garantien für Multimediale Daten-Server

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Technischen Fakultät
der Universität des Saarlandes
von
Dipl.-Inform.

**Guido Nerjes**

Saarbrücken
im Juni 1999

| | |
|---|---|
| Dekan der Technischen Fakultät: | Prof. Dr. Wolfgang Paul |
| Vorsitzende der Prüfungskommission: | Prof. Dr. Reinhard Wilhelm |
| Erstgutachter: | Prof. Dr. Gerhard Weikum |
| Zweitgutachter: | Prof. Dr. Kurt Mehlhorn |
| Beisitzer: | Dr. Elmar Schömer |
| Tag des Promotionskolloquiums: | 29. Oktober 1999 |

# Danksagung

Diese Arbeit entstand im Rahmen des von der Europäischen Union geförderten Forschungsprojektes HERMES und meiner Tätigkeit als wissenschaftlicher Mitarbeiter an der ETH Zürich und der Universität des Saarlandes.

Mein persönlicher Dank gilt vor allem Herrn Prof. Dr. Gerhard Weikum, der mich während der gesamten Zeit wissenschaftlich betreute und der durch seine Expertise mit zahlreichen Ratschlägen wesentlich zum Gelingen dieser Arbeit beigetragen hat.

Bedanken möchte ich mich auch bei Prof. Dr. Kurt Mehlhorn, für seine Bereitschaft das Koreferat zu übernehmen.

Ein herzliches Dankeschön auch an alle Kollegen für die angenehme Arbeitsatmosphäre sowie zahlreiche konstruktive Diskussionen und Anregungen. Besonders herausstellen möchte ich hierbei Dr. Peter Muth, der als fachkompetenter Kollege stets wertvolle Hinweise liefern konnte. Des weiteren bedanke ich mich bei meinen Diplomanden Andreas Wendling, Bare Said und Anja Jantke, die an der technischen Umsetzung vieler in dieser Arbeit entwickelter Gedanken und Ideen beteiligt waren.

Für das kurzfristige und dennoch sorgfältige Korrekturlesen der Arbeit möchte ich mich bei Dietmar Meyer und Anette Schröder bedanken.

Abschließend herzlichem Dank auch an meine Eltern für ihre unermüdliche Unterstützung während des Studiums und der Zeit danach.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Kurzfassung

In dieser Arbeit werden neuartige Algorithmen für das Platten-Scheduling in multimedialen Daten-Servern vorgestellt und ein stochastisches Modell zur Vorhersage der vom Server garantierbaren Service-Qualität entwickelt. Angenommen wird, daß die Speicherung der multimedialen Daten auf den Platten eines Disk-Arrays *gemischt* erfolgt. Dies bedeutet, daß sowohl kontinuierliche Daten (Audio, Video) als auch diskrete Daten (Text, Bild, Graphik) gemeinsam auf einer Platte abgelegt werden. Bedingt durch diese gemischte Datenhaltung muß jede Magnetplatte Zugriffe auf kontinuierliche *und* diskrete Daten verarbeiten können.

Die Auswirkungen verschiedener Scheduling-Algorithmen auf die Service-Qualität beim Datenzugriff auf kontinuierliche und diskrete Daten werden experimentell evaluiert. Die Service-Qualität für kontinuierliche Daten ist hierbei durch die Störungsrate bestimmt, d.h. durch die Häufigkeit, mit der der Server zeitweilig die Übertragung des Datenstromes zum Client unterbrechen muß. Die Service-Qualität für diskrete Daten wird durch die Antwortzeit festgelegt.

Grundlage für die Vorhersage der Service-Qualität bildet die stochastische Modellierung des Daten-Servers. Zu diesem Zweck wird die Verteilungsfunktion für die Zeitdauer analysiert, die für eine Menge von Plattenzugriffen benötigt wird. Hierauf aufbauend läßt sich die Wahrscheinlichkeit bestimmen, mit der eine festgelegte Störungsrate bzw. eine festgelegte Antwortzeitschranke überschritten wird. Für letzteres wird ein M/G/1-Warteschlangenmodell entwickelt, das die Verteilungsfunktion der Antwortzeit in Form der Laplace-Transformierten berechnet. Die Vorhersagen des stochastischen Modells bilden die Basis für die Konfiguration des Disk-Arrays und für die Zulassungkontrolle, mit der der Server die vereinbarte Service-Qualität garantieren kann.

# Abstract

This thesis presents new methods for disk scheduling in multimedia servers and derives a stochastic model to predict the quality-of-service that the server can guarantee. It is assumed that all data, i.e. continuous data (audio/video) and discrete data (text/indexes/images), resides for efficiency reasons on a shared disk-array. Thus, accesses to this data induce a mixed workload on each disk.

The impact of different disk scheduling policies on the quality-of-service for both continuous and discrete data is studied, and a framework for describing various policies in terms of a few parameters is developed. The quality-of-service for continuous data is determined by the error rate, i.e. the frequency of the server temporarily suspending the delivery of data to the client. The quality-of-service for discrete data is given by the response time for discrete data requests.

A stochastic model is developed based on deriving the distribution function of the service time for batched disk service under a multi-user load. Using this result it is possible to bound the tail probabilities of the error rate and the response time. The latter can be obtained using an M/G/1 queueing model that derives the Laplace transform of the response time distribution. The results from the stochastic model provide the basis for configuring the disk-array and exerting an admission control such that the server can guarantee its quality-of-service.

# Teil I

# Einleitung

# Kapitel 1

# Einleitung

## 1.1 Motivation

Durch die rasante Entwicklung der Computer- und Netzwerk-Technologie und den damit sinkenden Kosten für Hardware- und Verbindungskosten ist das Internet zum Massenmedium geworden, das seinen Anwendern eine Fülle unterschiedlichster Informationen zur Verfügung stellt. Untersucht man die Entwicklung bis zum heutigen Stand genauer, so lassen sich folgende Trends beobachten, die neue Anforderungen an die zugrundeliegende Infrastruktur stellen.

Die Darstellung der Inhalte beschränkte sich in der Vergangenheit im wesentlichen auf die Verwendung von Texten, Bildern und Graphiken. Durch das Interesse der Informationsanbieter, die Inhalte für den Anwender interessanter und informativer zu gestalten, und durch den Einsatz leistungsfähigerer Computer und Netzwerkverbindungen zeichnet es sich ab, daß zukünftig in verstärktem Maße diese Darstellungformen durch den Einsatz von Audio und Video ergänzt werden. Die Anwendungsgebiete für audiovisuelle Präsentationsformen in Kombination mit Texten, Graphiken und Bildern sind vielfältig. Folgende Anwendungsszenarien sind denkbar bzw. bereits realisiert:

- Tele-Shopping
  Da beim elektronischen Einkauf über das Internet der Kunde das Produkt nicht physisch begutachten kann, ist es notwendig, daß ihm detaillierte Produktionformationen zur Verfügung gestellt werden. Ergänzend zu beschreibenden Texten und Fotos, kann z.B. durch den Einsatz eines Videos die praktische Handhabung eines Produkts vorgeführt werden.

- Tele-Teaching
  Neben dem konventionellen Lehrmaterial in Form von Texten und Bildern, das interaktiv erschlossen werden kann, ergeben sich durch die Verwendung von Videos zusätzliche Möglichkeiten, Inhalte effizient zu vermitteln.

- News-On-Demand
  Neben der graphischen und textuellen Darstellung der Nachrichten, wie sie heutzutage ebenfalls durch die Print-Medien geschieht, wird durch den Einsatz von Gesprächs- und Videoaufzeichnungen die Attraktivität des Angebots erhöht und es können Zusatzinformation vermittelt werden.

Weiterhin läßt sich beobachten, daß die Präsentation der Daten zunehmend durch die Interaktion mit dem Benutzer geprägt wird. Anstelle großer umfangreicher Dokumente, die der Benutzer zusammenhängend mit seinem Web-Browser lädt und für längere Zeit betrachtet, werden zunehmend kurze Teildokumente zur Verfügung gestellt, die z.B. über Hyperlinks oder eine Navigationsstruktur miteinander verbunden sind. Mit der steigenden Anzahl der Verknüpfungen innerhalb der Dokumente und des verringerten Datenumfangs eines Dokumentes steigt darüber hinaus auch die Häufigkeit, mit der der Benutzer Daten vom multimedialen Daten-Server anfordert.

Neben der wachsenden Anzahl Benutzer muß deren gesteigertes Qualitätsbewußtsein berücksichtigt werden. Insbesondere dann, wenn der Benutzer für die bereitgestellten Informationen ein Entgelt zahlen muß, erwartet er neben der inhaltlichen Qualität, daß die fehlerfreie und schnelle Bereitstellung der Daten auch garantiert werden kann.

Die Anforderungen, die sich aus diesen Trends ableiten, lassen sich wie folgt zusammenfassen.

Da Audio und Video im Vergleich zu den konventionellen Daten ein vielfach größeres Datenvolumen besitzen, muß die Speicherung, die Verwaltung und die Verarbeitung dieser Daten besonders effizient durchgeführt werden, um dem Anwender die Daten preisgünstig zur Verfügung stellen zu können.

Durch die steigende Interaktivität der Anwendungen, muß eine verzögerungsfreie schnelle Bereitstellung der Daten trotz steigender Zugriffshäufigkeit ermöglicht werden. Die interaktive Präsentation von Daten kann in der Tat nur sinnvoll erfolgen und ist dann für den Benutzer interessant, wenn die Reaktion auf seine Benutzeraktionen unmittelbar erfolgt. Die Verzögerungen beim Laden und der Wiedergabe der Daten müssen möglichst gering gehalten werden.

Gleichzeitig muß aufgrund der Vielzahl von Benutzern, die gleichzeitig auf die Daten zugreifen, durch eine geeignete Koordination gewährleistet werden, daß den Benutzern eine garantierte Service-Qualität angeboten und diese auch eingehalten werden kann.

Obige Anforderungen gelten nicht nur für das Internet, sondern auch für unternehmensinterne Informationssysteme. Sie stellen eine große Herausforderung für die Konzeption und Realisation eines multimedialen Informationssystems dar. Besonders betroffen ist hiervon der Daten-Server, der als Kernkomponente des Systems die Speicherung und Bereitstellung der Daten übernimmt.

## 1.2 Multimediale Daten-Server

Bevor auf die Problemstellung beim Entwurf eines multimedialen Daten-Servers näher eingegangen wird, soll zunächst ein Überblick über dessen Aufbau und prinzipielle Funktionsweise gegeben werden.

### 1.2.1 Systemarchitektur

Der Daten-Server ist Bestandteil einer Client-Server-Architektur und über ein Netzwerk, z.B. Internet/Intranet, mit dem Client des Benutzers verbunden ist (siehe Abbildung 1.1). Der Daten-Server übernimmt die zentrale Speicherung der multimedialen Daten, wobei als Speichersystem

ein *Disk-Array*, das aus mehreren Magnetplatten besteht, verwendet wird. Die Vorteile dieser zentralen Datenspeicherung sind u.a.:

- hohe Verfügbarkeit
  Der Daten-Server bietet im Vergleich zu einem Client ein hohes Maß an Verfügbarkeit. Die Informationen sind 24 Stunden pro Tag abrufbar.

- effiziente Datenverwaltung und Sicherung
  Durch die zentrale Speicherung der Daten auf einem Server vereinfacht sich die Sicherung und die Aktualisierung der Daten.

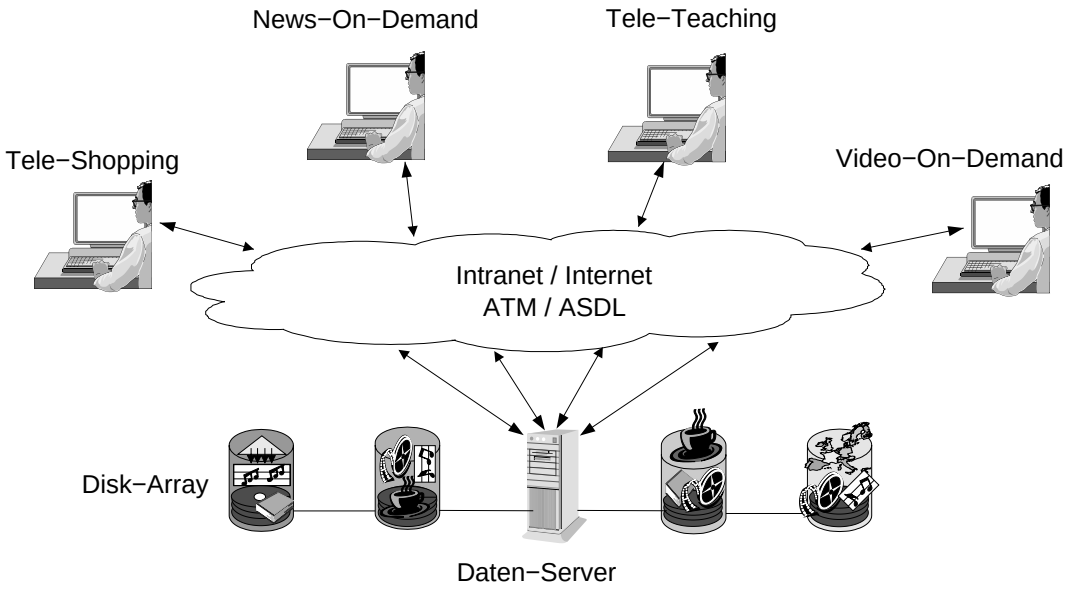



Abbildung 1.1: Systemarchitektur

Aufgrund der heutzutage bereits zur Verfügung stehenden Netzwerk-Technologie (ASDL, ATM) und der in Zukunft zu erwartenden Entwicklungen kann davon ausgegangen werden, daß der Transport auch großer Datenmengen mit hinreichender Geschwindigkeit über die Netzwerkverbindungen durchgeführt werden kann. Des weiteren stehen Transportprotokolle zur Verfügung, die Service-Garantien (maximale Verzögerung, minimale Geschwindigkeit) zusichern können. Der eigentliche Engpaß in der Bereitstellung und Verwaltung der Daten liegt somit beim Daten-Server, auf dem sich deshalb die nachfolgenden Betrachtungen konzentrieren.

Der Daten-Server speichert die Daten auf den Magnetplatten in Form von *Datenobjekten*. Die Datenobjekte lassen sich in zwei unterschiedliche Typen klassifizieren.

### 1.2.2 Typisierung der Datenobjekte

Multimediale Datenobjekte lassen sich in zwei Klassen einteilen. Die erste Klasse umfaßt *kontinuierliche* Datenobjekte, die zweite Klasse umfaßt alle *diskreten* Datenobjekte:

**Kontinuierliche Datenobjekte**

Kontinuierliche Datenobjekte (engl. continuous objects) sind bei ihrer Präsentation an die Zeit gekoppelt und treten z.B. in Form von Audio- oder Videodateien auf. Zu jedem Zeitpunkt wird ein bestimmter Ausschnitt aus dem Datenobjekt dargestellt. Die Dauer der Präsentation ist, falls keine Benutzerunterbrechung stattfindet, im voraus festgelegt. Das Datenvolumen kontinuierlicher Datenobjekte beträgt abhängig von der Wiedergabedauer mehrere Gigabytes.

Der Aufbau eines kontinuierlichen Datenobjektes ist abhängig vom verwendeten Datenformat [Ben97, HPN96, KEGLA98]. Videoobjekte bestehen aus einer Aneinanderreihung einzelner *Bilder*. Die *Bildrate* bestimmt die Geschwindigkeit, in der diese Bilder nacheinander dargestellt werden. Ihr Wert liegt üblicherweise bei 25-30 Bildern pro Sekunde [ITU90]. Audioobjekte bestehen aus einer Folge von Abtastwerten, die als Amplitudenwerte des analogen Tonsignals zu regelmäßigen Zeitpunkten gemessen werden [Pan95]. Die Anzahl der Abtastwerte und damit die Qualität der Wiedergabe wird durch den Abstand der Zeitpunkte, d.h. durch das Abtastintervall festgelegt.

Durch die zeitabhängige Präsentation der kontinuierlichen Daten läßt sich für jedes kontinuierliche Datenobjekt dessen *Datenrate* angeben. Diese Datenrate beschreibt die Menge an Daten, die für eine kontinuierliche Darstellung pro Zeiteinheit benötigt wird. Sie ist somit eine Metrik für die Geschwindigkeit, in der der Client bei der Präsentation auf die Daten zugreift. Tabelle 1.1 [KEGLA98] zeigt die Datenrate für verschiedene Audio- und Videostandards.

| Standard | Datenrate | Komprimierungstyp | Anwendung |
|---|---|---|---|
| G.711 | 64 kbps | konstant | Sprachkodierung |
| G.721 | 32 kbps | konstant | Sprachkodierung |
| G.722 | 48,56,64 kbps | konstant | Sprachkodierung |
| G.728 | 16 kbps | konstant | Sprachkodierung |
| MPEG-1 (audio) | 128-384 kbps | variabel | 2-Kanal Audio |
| MPEG-2 (audio) | 320 kbps | variabel | 5-Kanal Audio |
| H.261 | x*64 kbps | konstant | Telekonferenz |
| H.263 | 32- kbps | variabel | Telekonferenz |
| MPEG-1 (video) | 1.5- Mbps | variabel/konstant | TV-Kodierung |
| MPEG-2 (video) | 2-80 Mbps | variabel/konstant | HDTV-Kodierung |

Tabelle 1.1: Datenrate verschiedener Kodierungs- und Komprimierungsstandards für Audio- und Videodaten

Da bei einigen der Standards unterschiedliche Formate mit unterschiedlicher Wiedergabequalität (Bildgröße, Farbtiefe, Mono/Stereo) wählbar sind, läßt sich aufgrund des unterschiedlichen Datenvolumens nur ein Intervall für die Datenrate angeben. Der Komprimierungstyp bestimmt, ob das in diesem Format kodierte Datenobjekt eine konstante oder variable Datenrate besitzt.

Die *variable Datenrate* entsteht bei der Kodierung der Ausgangsdaten dadurch, daß sich einige Abschnitte hieraus mit weniger Daten beschreiben lassen als andere. Kodiert man anstatt der Bilder eines Videoobjektes die Veränderungen, die sich zwischen zwei aufeinanderfolgenden Bildern ergeben, so lassen sich z.B. Landschaftsszenen, in denen kaum eine Veränderung sichtbar ist, stärker komprimieren als z.B. Musikvideos, in denen sich die Einzelbilder stark unterscheiden.

Bei einer *konstanten Datenrate* wird der Komprimierungsgrad, d.h. die Stärke der Komprimierung dynamisch angepaßt. Datenabschnitte ohne große Veränderung werden hierbei nur geringfügig komprimiert. Bei relativ großen Unterschieden ist die Komprimierung gegebenenfalls mit einem Informationsverlust behaftet, um die komprimierte Datenmenge der angestrebten Datenrate anzupassen. Letzteres führt aufgrund des Informationsverlustes bei der Kodierung zu einer variablen Qualität bei der Präsentation der Daten.

Kontinuierliche Datenobjekte, die durch die Kodierung, z.B. MPEG-2, DVD [Ben97, HPN96, Tay97, KEGLA98], eine zeitlich variable Datenrate (VDR, engl. VBR = variable bit rate) aufweisen, werden im folgenden auch als *VDR-Datenobjekte* bezeichnet. Kontinuierliche Datenobjekte mit zeitlich konstanter Datenrate (KDR, engl. CBR = constant bit rate) werden als *KDR-Datenobjekte* bezeichnet.

#### Diskrete Datenobjekte

Diskrete Datenobjekte (engl. discrete objects) sind zeitunabhängig, d.h., ihre Präsentationsdauer kann frei gewählt werden und ihre Darstellung bleibt zeitlich konstant. Diskrete Datenobjekte liegen auf dem Daten-Server in Form von Texten (z.B. ASCII-, HTML-, Word-Format), Bildern (z.B. JPEG-, GIF-, PNG-Format) und Graphiken (z.B. EPS-, DXF-Format) vor. Die Größe eines diskreten Datenobjektes liegt deutlich unter der eines kontinuierlichen Datenobjektes und bewegt sich typischerweise in einem Bereich von weniger als einem Megabyte.

### 1.2.3 Datenfluß zwischen Server und Client

Die Art und Weise, wie Datenobjekte vom Daten-Server gelesen und zum Client geschickt werden, ist abhängig vom Datentyp. Diskrete Datenobjekte werden, nachdem sie vom Client angefordert worden sind, vom Speichersystem des Servers zunächst vollständig in einen *Server-Puffer* im Hauptspeicher gelesen. Dort erfolgt abhängig vom verwendeten Übertragungsprotokoll [OS97, Sch96] eine Zerlegung in kleinere Transportpakete, die über das Netz zum Client geschickt werden. Im *Client-Puffer* werden die einzelnen Transportpakete wieder vollständig zusammengesetzt und die Daten der Anwendung, z.B. dem Browser, zur Verfügung gestellt.

Kontinuierliche Datenobjekte werden aufgrund ihrer Größe von u.U. mehreren Gigabytes zum Versand an den Client nicht als Ganzes in den Server-Puffer eingelesen. Stattdessen erfolgen mehrere zeitlich versetzte Datenzugriffe auf Teilabschnitte des kontinuierlichen Datenobjektes. Diese Teilabschnitte, nachfolgend *Fragmente* genannt, werden nacheinander aus dem Speichersystem gelesen und jeweils im Server-Puffer bereitgestellt. Hier wird ein Fragment abhängig vom verwendeten Übertragungsprotokoll in Transportpakete zerlegt und über das Netz an den Empfänger verschickt. Sobald die Daten eines Fragments übertragen worden sind, kann der im Server-Puffer beanspruchte Platz für nachfolgende Fragmente freigegeben werden. Damit der

Client-Puffer nicht das vollständige Datenobjekt aufnehmen muß, wird auf der Empfängerseite der Daten analog verfahren. Der Client liest aus dem Client-Puffer kurz nach dem Eintreffen des ersten Transportpaketes und beginnt mit der Wiedergabe. Um Platz für nachfolgende Transportpakete zu schaffen, können bereits präsentierte Daten aus dem Client-Puffer gelöscht werden. Durch dieses Verfahren, das als *Audio/Video-Streaming* bekannt ist, ergibt sich ein *kontinuierlicher Datenstrom* zwischen dem Daten-Server und dem Client [GVK+95].

Neben dem reduzierten Pufferbedarf ergibt sich ein weiterer Vorteil dieses Streaming-Verfahrens. Die Verzögerung zwischen Benutzerinteraktion und dem Start der Wiedergabe ist geringer, da nicht gewartet werden muß, bis das gesamte kontinuierliche Datenobjekt über das Netzwerk übertragen worden ist, sondern die Wiedergabe der Daten bereits nach dem Eintreffen des ersten Transportpaketes erfolgt.

Damit das Audio/Video-Streaming möglich ist, werden einige Anforderungen an die Bereitstellung der Daten gestellt. Die *Übertragungsrate*, d.h. die Geschwindigkeit mit der die Daten dem Client zur Verfügung gestellt werden, muß mindestens der Datenrate des Datenobjektes entsprechen. Geringe Schwankungen können hierbei durch den Server- und den Client-Puffer ausgeglichen werden. Sinkt allerdings die Übertragungsrate für einen gegebenen Zeitraum unter die Datenrate, so leert sich der Client-Puffer schneller als er aufgefüllt wird und die Wiedergabe wird verzögert bzw. sogar unterbrochen, wenn bei einem *Pufferunterlauf* der Client-Puffer keine Daten mehr enthält (siehe Abbildung 1.2).

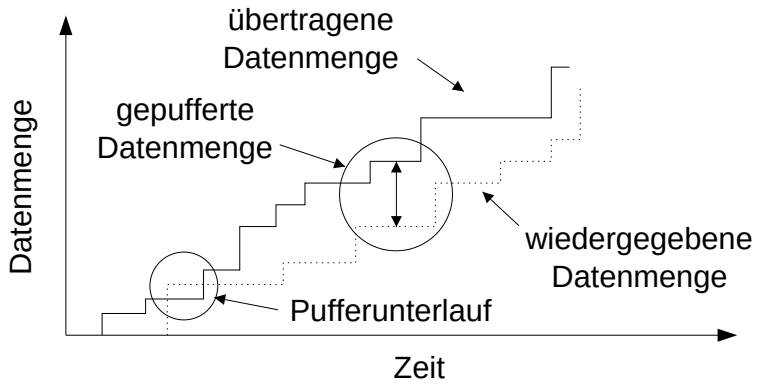



Abbildung 1.2: Pufferbedarf des Clients bei variierender Daten- und Übertragungsrate

Ist die Übertragungsrate größer als die Datenrate, so gehen Daten, nachdem der Client-Puffer vollständig aufgefüllt worden ist, durch einen *Pufferüberlauf* verloren [GC92, GH94]. Ziel muß es sein, die Übertragungsrate so an die Datenrate anzupassen, daß die Größe des Server- und des Client-Puffers möglichst klein gehalten werden kann. Hierbei muß beachtet werden, daß sich aufgrund der stetigen Wiedergabe kontinuierlicher Datenobjekte für jedes Fragment eine zeitliche *Frist* (deadline) ergibt, bis zu der es spätestens im Server-Puffer für den Netztransport zur Verfügung gestellt werden muß. Können aufgrund einer Überlastung die Fristen nicht eingehalten werden, so werden einzelne Fragmente übersprungen und so die Übertragungsrate verringert. Durch diese Überlastkontrolle wird vermieden, daß sich Verzögerungen in der Bereitstellung der Daten fortpflanzen. Durch das Auslassen von Fragmenten werden *Störungen* bei der Wiedergabe hervorgerufen, da der Client als Anpassung an die reduzierte Übertragungsrate die Datenrate und damit die Wiedergabequalität reduzieren muß. Dies kann z.B. durch die

Reduzierung der Bildauflösung oder durch das Wegfallen einzelner Bilder geschehen. Letzteres führt zum kurzzeitigen Einfrieren der Videowiedergabe bzw. zu Lücken in der Tonwiedergabe.

In Abhängigkeit davon, ob dem Daten-Server oder dem Client die Kontrolle dieser Datenflußanpassungen obliegt, lassen sich *Server-Push-* oder *Client-Pull-Architekturen* unterscheiden [RVT96]. Bei letzteren fordert der Client die Daten selbst beim Daten-Server an und kontrolliert die Datenmenge sowie den Bereitstellungszeitpunkt. Bei einer Server-Push-Architektur dagegen bestimmt der Daten-Server Menge und Bereitstellungszeitpunkt und schickt die Daten ohne explizite Aufforderung des Clients. Damit keine Störung in der Wiedergabe durch einen Über- bzw. Unterlauf im Client-Puffers hervorgerufen wird, ist es notwendig, daß der Daten-Server die Datenrate des kontinuierlichen Datenobjektes kennt und hieran die Übertragungsrate in seinem Zuständigkeitsbereich, d.h. bis zur Bereitstellung der Fragmente im Server-Puffer, anpaßt. Die Kontrolle der Übertragungsrate jenseits der Schnittstelle zum Netzwerk wird dem Netzwerkprotokoll und dem Client überlassen.

Für die Anpassung der internen Übertragungsrate innerhalb des Daten-Servers ist es erforderlich, daß der Server über Zustandsänderungen bei der Wiedergabe, wie z.B. das Anhalten oder die Fortsetzung der Präsentation, informiert wird. In dieser Arbeit wird eine Server-Push-Architektur betrachtet, da durch die vielfältigeren Kontrollmöglichkeiten auf Seiten des Servers ein größerer Optimierungsspielraum besteht.

Da ein Push-Server unter der Randbedingung der aufrechtzuerhaltenden Übertragungsrate selbst bestimmen kann, wann er die Fragmente vom Disk-Array liest, ist eine einfache Form der Speicherverwaltung für den Server-Puffer nach dem *Doppelpufferprinzip* möglich. Für jeden Datenstrom werden zwei Pufferbereiche angelegt (siehe Abbildung 1.3). In den ersten Pufferbereich werden die Daten von der Platte gelesen und vom zweiten Pufferbereich erfolgt das Versenden der Daten über das Netzwerk. Sobald der erste Pufferbereich gefüllt und der zweite Pufferbereich geleert ist, werden die Rollen getauscht. Aus dem ersten Pufferbereich erfolgt dann das Versenden der Daten, und der zweite Pufferbereich wird erneut vom Disk-Array gefüllt.

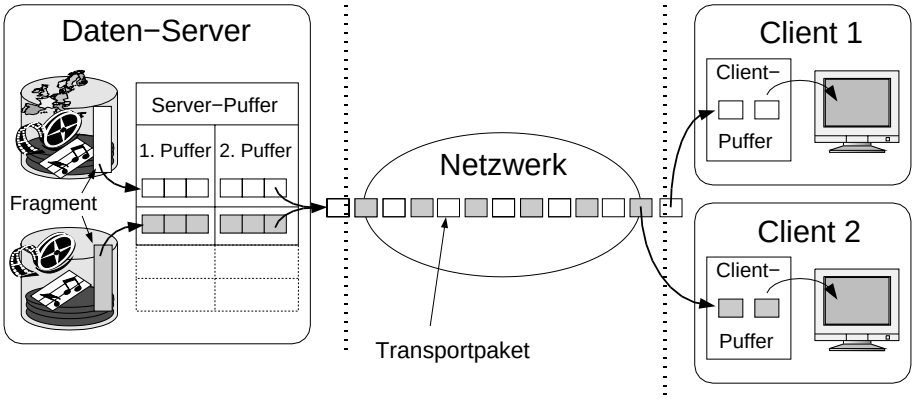



Abbildung 1.3: Datenfluß zwischen Daten-Server und Client

### 1.2.4 Performance-Anforderungen und Service-Garantien

#### Anwendungsorientierte Performance-Anforderungen

Anwendungen stellen an die Bereitstellung von Daten durch den Server bestimmte Anforderungen, die im folgenden als *anwendungsorientierte Performance-Anforderungen* bezeichnet werden. Damit sich anspruchsvolle Anwendungen (und Benutzer) auf den Daten-Server hinsichtlich der Erfüllung dieser Anforderungen verlassen können, ist es zwingend notwendig, daß Zusicherungen von Seiten des Servers garantiert werden können. Dieses geschieht durch die Vereinbarung von *Performance-Garantien* oder auch *Service-Garantien* zwischen Server und Client. In Abhängigkeit vom Datentyp lassen sich Service-Garantien für kontinuierliche und Service-Garantien für diskrete Daten unterscheiden.

**Service-Garantien für die Bereitstellung kontinuierlicher Daten** Die *Service-Qualität* (engl. quality-of-service, Abk.: QoS), mit der der Daten-Server kontinuierliche Daten dem Client zur Verfügung stellt, wird dadurch bestimmt, in welchem Maße die zeitlichen Fristen bei der Bereitstellung der Fragmente im Server-Puffer eingehalten werden. Jede Überschreitung einer Frist hat das Auslassen eines Fragments und damit eine *Störung* in der Wiedergabe, d.h. eine Verminderung der Wiedergabequalität, zur Folge. Die *Störungsrate* eines Datenstroms beschreibt die Häufigkeit, mit der diese Störungen während der gesamten Dauer der Übertragung hervorgerufen werden. Besteht ein kontinuierliches Datenobjekt aus insgesamt $3600$ Fragmenten, von denen $36$ nicht fristgerecht zur Verfügung gestellt werden können, so liegt eine Störungsrate von $36/3600 = 1\%$ vor. Unter den Service-Garantien, die der Daten-Server abgibt, lassen sich zwei Abstufungen unterscheiden. Durch eine *deterministische Service-Garantie* [GC92, VGG95, ORS95, DBB96] garantiert der Server, daß alle Fragmente fristgerecht im Server-Puffer bereitgestellt werden. Hierdurch wird während der gesamten Wiedergabe eines kontinuierlichen Datenobjektes keine Verminderung seiner Wiedergabequalität eintreten, die auf ein Verschulden des Daten-Servers zurückzuführen ist. Durch eine *stochastische Service-Garantie* [NMW97b, CZ94a, VGGG94] wird die Service-Qualität, die vom Server eingehalten wird, stochastisch charakterisiert, und zwar dadurch, daß die Restwahrscheinlichkeit, mit der eine festgelegte Störungsrate überschritten wird, durch eine obere Schranke begrenzt wird. Ein Beispiel für eine solche Service-Garantie sieht wie folgt aus:

$$P[\text{Störungsrate} > 1\%] \leq 1\% \quad .$$

Besteht ein Datenobjekt aus $3600$ Fragmenten, so garantiert der Daten-Server hiermit, daß die Wahrscheinlichkeit, daß während der Gesamtdauer der Übertragung mehr als $36$ einzelne kurzzeitige Störungen auftreten, kleiner ist als ein Prozent. Für die meisten Anwendungen sind solche stochastischen Service-Garantien tolerierbar. Außerdem kann durch eine dynamische Anpassung der Wiedergabequalität, z.B. durch eine Verringerung der Wiedergaberate, die Wahrnehmbarkeit der Störungen durch den Benutzer häufig reduziert werden [TKC+97, HKR97].

Die Einhaltung der Service-Garantien kann der Daten-Server nur gewährleisten, wenn er mit Hilfe einer *Zulassungskontrolle* die Anzahl der gleichzeitig zu bedienenden Datenströme begrenzen kann. Die Zulassungskontrolle entscheidet, ob die Bedienung eines zusätzlichen Datenstromes unter der Voraussetzung möglich ist, daß für die bereits zugelassenen Datenströme die Service-Garantie für die Störungsrate aufrechterhalten werden kann. Als Grundlage dient

hierfür ein mathematisches Modell, das unter Berücksichtigung der Systemparameter, wie Plattenparameter und Fragmentgrößen sowie der einzuhaltenden Service-Garantie die maximale Anzahl bedienbarer Datenströme berechnen kann.

Weitere Metriken für die Service-Qualität bei kontinuierlichen Daten stellen die *Wartezeit* und die *Startverzögerung* (engl. access latency, startup latency) dar. Die Wartezeit beschreibt die Zeitdauer zwischen dem Auftreten eines Auftrages, einen neuen Datenstrom zu starten, und der Zulassung des Datenstromes. Mit der Startverzögerung wird der Zeitraum definiert, der zwischen der Zulassung eines Datenstromes und der Bereitstellung des ersten Fragments im Server-Puffer liegt. Auch für diese beiden Metriken ist die Abgabe stochastischer oder deterministischer Service-Garantien möglich.

**Service-Garantien für die Bereitstellung diskreter Daten** Für diskrete Daten ist die Zeitdauer, die der Client auf die angeforderten Daten warten muß, eine Metrik für die Service-Qualität des Daten-Servers. Die *Antwortzeit* auf der Server-Seite definiert diese Zeitdauer vom Eintreffen eines Auftrages bis zur Bereitstellung der Daten im Server-Puffer.

Der Daten-Server kann die Service-Garantie, die er einem Client hinsichtlich dieser Antwortzeit gibt, ebenfalls deterministisch oder stochastisch charakterisieren. Eine deterministische Service-Garantie wird die maximale Antwortzeit auf eine feste obere Schranke begrenzen. Eine stochastische Antwortzeit-Garantie beschränkt hingegen lediglich die Restwahrscheinlichkeit, mit der die Antwortzeit über einer oberen Schranke liegt, also z.B.

$$P[\text{Antwortzeit} > 1\text{sec}] \leq 1\% \quad .$$

Im Gegensatz zu den kontinuierlichen Daten wird in der Regel keine Zulassungskontrolle für die Bereitstellung diskreter Daten durchgeführt. Hierdurch würde die Funktionalität und die Interaktivität der Anwendungen stark eingeschränkt werden. Stattdessen muß der Daten-Server bei der Konfiguration bereits so dimensioniert werden, daß er die Service-Garantien unter einer vorgegebenen maximalen Last einhalten kann. Für die meisten Anwendungen mit vielen Clients impliziert dies, daß deterministische Service-Garantien zu akzeptablen Kosten nicht möglich sind.

#### Systemorientierte Performance-Anforderungen

Systemorientierte Performance-Anforderungen beschreiben die Anforderungen, die an die Leistungsfähigkeit des Daten-Servers generell gestellt werden und die das Preis-Leistungs-Verhältnis bestimmen. Als systemorientierte Performance-Anforderung kann im wesentlichen die Maximierung des *Durchsatzes* definiert werden. Im Falle diskreter Daten ist damit die maximale Anzahl der während einer Zeiteinheit durchführbaren Aufträge definiert. Im Falle kontinuierlicher Daten beschreibt der Durchsatz die maximale Anzahl parallel bedienbarer Datenströme.

### 1.2.5 Das Speichersystem

Das Speichersystem des Daten-Servers besteht aus mehreren unabhängig voneinander arbeitenden Magnetplatten, die zu einem Disk-Array zusammengefaßt sind. Die Verbindung zum

Server-Puffer im Hauptspeicher und zur CPU des Daten-Servers erfolgt zunächst über den E/A-Bus (z.B. IDE-, SCSI- oder FC-Verbindung) zur E/A-Schnittstelle des Rechners. Diese E/A-Schnittstelle kann über den Systembus (z.B. PCI, ISA, GigaPlane) den Hauptspeicher erreichen. Dadurch, daß sich mehrere Platten diese Datenverbindung zum Hauptspeicher des Servers teilen, muß verhindert werden, daß bei gleichzeitiger Übertragung von allen Platten ein Datenstau auf einem der Verbindungswege entsteht. Dies kann z.B. dadurch geschehen, daß mehrere E/A-Schnittstellen eingesetzt werden, an denen lediglich eine kleine Teilmenge der Platten angeschlossen ist. Im folgenden wird zunächst der Aufbau und die Funktionsweise einer Magnetplatte näher erläutert und daraufhin der Ablauf eines Schreib-Lese-Vorgangs erklärt.

### Mechanischer Aufbau einer Magnetplatte

Der mechanische Aufbau einer Magnetplatte ist in Abbildung 1.4 schematisch skizziert. Sie besteht aus mehreren übereinander angeordneten Magnetscheiben, die beim Betrieb, durch einen Motor angetrieben, um eine gemeinsame Achse rotieren. Über jeder Oberfläche der Scheiben befindet sich ein Schreib-Lese-Kopf, der Daten auf die Magnetscheibe schreiben bzw. von der Magnetplatte lesen kann. Die gelesenen Daten werden über einen von allen Köpfen genutzten *Datenkanal* weitergeleitet. Befestigt sind die Köpfe an einem *Plattenarm*, der ihre Positionierung auf unterschiedlichen Radien der Scheibe erlaubt. Aufgrund der Konstruktion ist eine unabhängige Positionierung jedes einzelnen Kopfes nicht möglich; der vom Plattenarm vorgegebene Radius gilt für alle Köpfe. Während des Betriebs schwebt jeder Kopf auf einem Luftpolster über der sich drehenden Magnetscheibe. Jede Magnetscheibe ist in mehrere konzentrische *Spuren* aufgeteilt, die wiederum aus mehreren *Sektoren* bestehen, die die gleiche Größe, meistens 512 Bytes, besitzen. Mehrere Sektoren werden zu einem *logischen Block* zusammengefaßt (in Abbildung 1.4 besteht ein Block aus genau einem Sektor). Die Menge aller direkt übereinanderliegenden Spuren, die ohne Armbewegung gelesen werden können, wird als *Zylinder* bezeichnet.

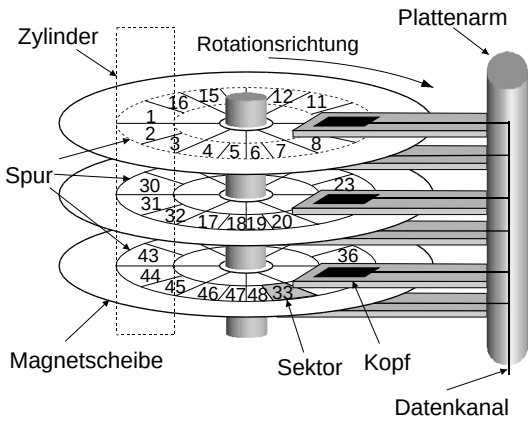



Abbildung 1.4: schematischer Aufbau einer Magnetplatte

Nach außen, z.B. für das Betriebssystem, stellt sich die Platte als eindimensionales Feld logischer Blöcke dar, die mit einer *logischen Blocknummer* versehen sind. Da mit Hilfe dieser Blocknummer die Adressierung für den Blockzugriff erfolgt, stellt ein logischer Block die

kleinste Einheit dar, die von der Platte angefordert werden kann. Die Abbildung der Blocknummer auf Zylinder, Spur und Sektoren der Magnetscheibe ist von den Parametern der Plattengeometrie, z.B. von der Anzahl der Sektoren pro Spur, abhängig und wird von der Plattensteuerung berechnet.

### Ausführung eines einzelnen Plattenzugriffs

Als *Plattenzugriff* wird das Lesen bzw. Schreiben von Daten auf bzw. von der Magnetplatte bezeichnet. Ausgelöst werden Plattenzugriffe durch Lese- oder Schreibaufträge, im folgenden nur als *Aufträge* bezeichnet, die das Betriebssystem über die E/A-Schnittstelle an die Plattensteuerung schickt. Ein Auftrag identifiziert eindeutig den Plattenzugriff durch Angabe einer logischen Blocknummer oder eines Blockintervalls und bestimmt hiermit auf welchen Block bzw. auf welche Sequenz von aufeinanderfolgenden Blöcken zugegriffen werden soll. Zunächst wird mit dem Plattenarm der Zylinder angefahren, in dessen Spur sich der erste Block des Intervalls befindet. Dazu muß der Arm von seiner aktuellen Position bis zur gewünschten Position die *Positionierungsdistanz*, d.h. eine bestimmte Anzahl zu überquerender Zylinder, überwinden. Die Zeit, die hierfür benötigt wird, wird im folgenden als *Positionierungszeit* bezeichnet. Sie setzt sich aus vier Komponenten zusammen [RW94b]:

- der *Beschleunigungsphase*, in der der Arm auf seine maximale Geschwindigkeit beschleunigt wird,
- der *Hochgeschwindigkeitsphase*, in der sich der Arm mit maximaler Geschwindigkeit bewegt,
- der *Abbremsphase*, in der sich der Arm dem anzufahrenden Zylinder nähert und seine Geschwindigkeit verringert wird,
- der *Abstimmphase*, in der der Plattenkopf für den Blockzugriff auf der gewünschten Spur kalibriert wird.

In Abhängigkeit von der Positionierungsdistanz durchläuft der Plattenkopf alle oder nur einen Teil dieser einzelnen Phasen. Bei größeren Distanzen werden alle vier Phasen durchlaufen und es dominiert die Hochgeschwindigkeitsphase mit einer konstant bleibenden Geschwindigkeit des Plattenarmes. In diesem Fall wächst die Positionierungszeit proportional zur Positionierungsdistanz. Sind sehr kurze Distanzen zu überwinden, werden ebenfalls alle vier Phasen durchlaufen. Im Unterschied zum vorangegangenen Fall verbringt der Arm die Zeit aber fast ausschließlich in der Beschleunigungs- und der Abbremsphase und folglich wächst die Positionierungszeit proportional zur Quadratwurzel der Positionierungsdistanz. Die Beschleunigungs-, die Abbrems- und die Hochgeschwindigkeitsphase entfallen praktisch, wenn lediglich ein Wechsel auf einen benachbarten Zylinder (*Zylinderwechsel*) durchgeführt werden muß. In diesem Fall hat die Positionierungszeit einen konstanten Wert, der die Dauer der Abstimmphase beschreibt. Beim Wechsel des Schreib-Lese-Kopfes (*Spurwechsel*) innerhalb eines Zylinders durchläuft der Plattenkopf ebenfalls nur die Abstimmphase. Ursache hierfür ist, daß die Spuren eines Zylinders aufgrund von Fertigungstoleranzen nicht absolut exakt übereinander liegen und deshalb eine Kalibrierung des Plattenkopfes auf der anzufahrenden Spur stattfinden muß.

Nach der Positionierung des Plattenarmes wird gewartet bis sich der gewünschte erste Block der Sequenz unter den Plattenkopf gedreht hat. Diese Verzögerung wird als *Rotationsverzögerung* bezeichnet und ist abhängig von der *Rotationsgeschwindigkeit* der Scheiben sowie vom Abstand des zu lesenden Blockes vom Plattenkopf in Rotationsrichtung. Die Rotationsgeschwindigkeit als lineare Geschwindigkeit steigt im Gegensatz zur konstanten Winkelgeschwindigkeit der Platte mit zunehmendem Abstand von der Plattenmitte an. Befindet sich der Block unter dem Plattenkopf kann der Schreib-Lese-Vorgang beginnen. Die *Transferzeit* bezeichnet dessen Dauer und ist proportional zur Größe und zur Anzahl der zu lesenden oder zu schreibenden aufeineinanderfolgenden Blöcke und umgekehrt proportional zur Rotationsgeschwindigkeit.

Als *Bedienzeit* bezeichnet man die Gesamtdauer des Plattenzugriffs. Sie setzt sich aus den vier oben beschriebenen Zeiten zusammen:

- der Positionierungszeit für die Bewegung des Plattenarmes auf die gewünschte Spur,
- der Rotationsverzögerung bis der erste zu lesende bzw. schreibende Block unter dem Plattenkopf liegt,
- der Transferzeit für das Lesen/Schreiben der Daten von der Magnetscheibe.

Sollte sich, anders als in den vorangegangenen Erläuterungen angenommen, die Blocksequenz über mehrere Spuren oder Zylinder erstrecken, so ist zwischenzeitlich ein Spur- oder Zylinderwechsel erforderlich. Um hierbei zu vermeiden, daß eine Rotationsverzögerung erneut anfällt, sind logisch benachbarte Blöcke über Spur- bzw. Zylindergrenzen hinweg um einen bestimmte Distanz versetzt angeordnet (engl. track skewing, cylinder skewing). Diese Distanz entspricht der maximalen Zeit, die benötigt wird, um den Spur- oder Zylinderwechsel, d.h. die Abstimmphase, durchzuführen, so daß anschließend der logisch nachfolgende Block sofort gelesen werden kann.

Da die äußeren Spuren im Vergleich zu den inneren Spuren einer Platte länger sind, besteht die Möglichkeit, hierauf eine größere Anzahl von Blöcken zu speichern. Bei modernen *Mehr-Zonen*-Platten (engl. multi zone disks) wird hiervon Gebrauch gemacht, um die Speicherkapazität zu erhöhen. Benachbarte Spuren, die die gleiche Anzahl von Blöcken aufweisen, werden in *Zonen* zusammengefaßt (siehe Abbildung 1.5). Die Anzahl der Zonen pro Platte schwankt zwischen $3$ und $20$.

Die Geschwindigkeit, mit der die Daten eines Plattenzugriffs gelesen bzw. geschrieben werden, ist durch die *Transferrate* der Platte als Quotient zwischen Datenmenge und Transferzeit des Plattenzugriffs definiert. Aufgrund der konstanten Winkelgeschwindigkeit und der steigenden Blockanzahl von der innersten zur äußersten Zone steigt bei einer Mehr-Zonen-Platte entsprechend die Transferrate von der innersten zur äußersten Zone. Bei *Ein-Zonen*-Platten (engl. single zone disks) ist die Anzahl der Blöcke, die durch die Kapazität der innersten Spur bestimmt wird, auf allen Spuren identisch und somit die Transferrate konstant.

Die Geschwindigkeit, mit der die Daten eines Plattenzugriffs gelesen bzw. geschrieben werden, ist durch die *Bedienrate* als Quotient zwischen Datenmenge und der Bedienzeit des Plattenzugriffs definiert. Die maximale Bedienrate wird erreicht, wenn ohne Rotationsverzögerungen und ohne anfallende Positionierungszeiten im optimalen Fall Daten gelesen werden können. Die Bedienrate ist in diesem Fall durch die Transferrate, d.h. durch die Rotationsgeschwindigkeit

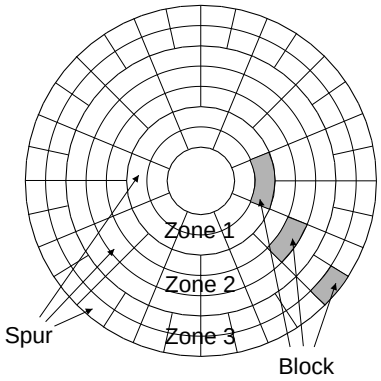



Abbildung 1.5: Gruppierung von Spuren mit gleicher Blockanzahl zu Zonen

der Platte und die Anzahl der Blöcke pro Spur, begrenzt. Die Bedienrate wird um so geringer, je weniger Blöcke bei einem Plattenzugriff transferiert werden und je häufiger der Plattenkopf über größere Positionierungsdistanzen bewegt werden muß. In beiden Fällen wächst der Anteil der Positionierungzeit und der Rotationsverzögerung an der Bedienzeit eines Plattenzugriffs und hierdurch sinkt die pro Zeiteinheit übertragbare Datenmenge.

#### Scheduling und Datenplazierung

In der Praxis muß ein Daten-Server mehrere kontinuierliche Datenströme gleichzeitig bedienen. Da die Bedienrate einer Magnetplatte größer ist als die Datenrate eines kontinuierlichen Objektes, können von einer Platte mehrere Datenströme gleichzeitig mit Daten versorgt werden. Hierbei muß darauf geachtet werden, daß durch ein geeignetes *Scheduling* der Plattenzugriffe, d.h. durch eine geeignete Koordination und Festlegung der Bedienungsreihenfolge, sichergestellt wird, daß die Bereitstellung der Fragmente im Server-Puffer rechtzeitig vor Ablauf der Frist erfolgen kann. Des weiteren bestimmt das Scheduling den Zeitpunkt der Ausführung von Zugriffen auf diskrete Datenobjekte und legt hierdurch die Antwortzeit fest.

Eng verbunden mit dem Scheduling ist die *Datenplazierung*. Die Datenplazierung umfaßt zum einen die Verteilung der Daten innerhalb einer Platte (intra disk placement) als auch die Verteilung auf verschiedene Platten des Disk-Arrays (inter disk placement). Letzteres erlaubt bei Verteilung eines Datenobjektes auf mehrere Platten durch parallelen Zugriff, eine akkumulierte Bedienrate zu nutzen.

## 1.3 Problemstellungen und Ziele

Neben technischen Aspekten spielen bei der Entwicklung eines Daten-Servers für multimediale Daten auch wirtschaftliche Aspekte eine Rolle. Dem Anwender können die Daten nur dann preisgünstig zur Verfügung gestellt werden, wenn ein wirtschaftlicher Betrieb gewährleistet werden kann. Hierfür ist es notwendig, daß der Daten-Server u.a. ein niedriges Preis-Leistungs-Verhältnis aufweist.

Letzteres läßt sich nur dann erreichen, wenn alle Komponenten effizient genutzt und möglichst vollständig ausgelastet werden. Es lassen sich insgesamt vier Bereiche identifizieren, in denen

zur Effizienzsteigerung optimiert werden kann:

1. Optimierung des Datenzugriffs
2. Verwendung stochastischer Service-Garantien
3. Gemeinsame Nutzung der Ressourcen
4. Sorgfältige Kapazitätsplanung

### 1.3.1 Optimierung des Datenzugriffs

Das Problem hoher Kosten bei multimedialen Daten-Servern entstehen durch die Performance-Unterschiede zwischen dem Hauptspeicher und dem angeschlossenen Speichersystem. Die maximale Bedienrate des Daten-Servers, d.h. die maximale Geschwindigkeit mit der die Daten vom Speichersystem in den Server-Puffer übertragen werden, wird durch die maximale Bedienrate des Disk-Array und damit durch die maximale Bedienrate jeder einzelnen Platte bestimmt. Die Verzögerungen, die durch die übrigen Systemkomponenten (E/A-Bus, E/A-Schnittstelle, Systembus) entlang des Datenpfades zum Server-Puffer verursacht werden, können im Vergleich zu den an der Platte auftretenden Verzögerungen (Positionierungszeit, Rotationsverzögerung, Transferzeit) bei ausreichender Dimensionierung der Hardware vernachlässigt werden. Es ist absehbar, daß die Lücke in der Leistungsfähigkeit zwischen dem Speichersystem und dem Hauptspeicher auch in Zukunft nicht geschlossen werden kann, sondern eher noch vergrößert wird, weil die technologische Entwicklung in der Vergangenheit nur geringe Steigerungen der Bedienrate bei Magnetplatten ermöglicht hat, die Leistungsfähigkeit der übrigen Komponenten aber ständig zugenommen hat. Da alternative kostengünstige Speichersysteme (z.B. holographische Speicher) zur Massenspeicherung multimedialer Daten auch in naher Zukunft nicht zur Verfügung stehen werden, wird bei der Lieferung multimedialer Daten das Speichersystem für unbestimmte Zeit der Engpaß bleiben.

Ausgangsbasis verschiedener Forschungsarbeiten ist es deshalb, durch eine Optimierung die Effizienz des Datenzugriffs zu steigern und somit die Anzahl benötigter Magnetplatten zu reduzieren. Hierbei werden u.a. die beiden folgenden Ansätze untersucht.

**Optimierung der Datenplazierung** Die Anordnung der Daten innerhalb einer Platte kann so geschehen, daß häufig referenzierte Daten möglichst nahe zusammen abgelegt werden [TCG98, TKKD96]. Hierdurch läßt sich der Positionierungsaufwand der Plattenzugriffe reduzieren und insgesamt die Bedienrate der Platte steigern. In [TCG98] wird ein Verfahren für verschiedene neuere Sekundärspeichertechnologien aufgezeigt, das eine optimale analytische Lösung der Datenallokation auf einer Platte berechnet. Voraussetzung hierfür ist, daß die Zugriffhäufigkeiten der einzelnen Datenblöcke im voraus bekannt und unabhängig voneinander sind. Dieses Verfahren läßt sich sowohl auf diskrete als auch auf kontinuierliche Datenobjekte anwenden. Daneben kann bei periodischen wiederkehrenden und damit bekannten Zugriffsmustern, wie sie bei kontinuierlichen Datenobjekten auftreten, die Anordnung der Daten so in eine sequentielle Reihenfolge gebracht werden, daß ebenfalls Positionierungszeiten verkürzt werden [VR93, RV93, GKS96].

Ein weiterer Beitrag zur Reduzierung von Positionierungszeiten und Rotationsverzögerungen ist die Optimierung der Zugriffsgröße. Hierbei werden Daten, von denen bekannt ist, daß sie stets als Ganzes angefordert werden, zusammenhängend auf der Platte abgelegt. Verschiedene Arbeiten beschäftigen sich damit, wie groß die Zugriffsgröße gewählt werden muß, damit unter bestimmten Randbedingungen, z.B. Puffer-Größe, die größtmögliche Bedienrate der Platte erreicht werden kann [VRG95, GKS96, ORS96].

In weiteren Arbeiten wird untersucht, wie die Plazierung der Daten auf mehreren Magnetplatten eines Disk-Arrays optimiert werden kann. Eine Herausforderung hierbei ist die Skalierbarkeit der Lösung. Die Ansätze [TPBG93, BGMJ94, ORS96] verwenden Striping-Verfahren, die ein kontinuierliches Datenobjekt in kleinere Einheiten zerlegen und anschließend diese kleineren Einheiten auf verschiedene Platten verteilen. Die Zugriffe können hierdurch von mehreren Platten parallel durchgeführt werden. Dasselbe Verfahren findet auch für diskrete Daten Verwendung [CLG+94, Zab94, CL95, SWZ98].

**Optimierung der Bedienungsreihenfolge** Wenn die Positionen der Plattenzugriffe eine kurze Zeitspanne im voraus bekannt sind, kann die Bedienungsreihenfolge der Zugriffe durch das Scheduling so umgestellt werden, daß die Positionierungsdistanzen zwischen den Fragmenten und damit die Positionierungszeiten verringert werden. Da diese Methoden den Plattenarm möglichst lange nur in eine Richtung bewegen, d.h. entweder von außen nach innen oder umgekehrt, werden sie auch als *Fahrstuhl-* oder *SCAN-Algorithmen* [TP72, CKY93, SG94a, WGP94, CKY93] bezeichnet. Im Vergleich hierzu wird bei *FCFS-Algorithmen* keine Umstellung der Bedienungsreihenfolge durchgeführt. Die Zugriffe werden ohne Optimierung in der Reihenfolge ihres Eintreffens ausgeführt.

### 1.3.2 Verwendung stochastischer Service-Garantien

Damit der Daten-Server dem Client eine deterministische Service-Garantie [TPBG93, GHBC94, ORS96, CZ94b, GKS96] zusichern kann, muß sichergestellt sein, daß alle Fristen hinsichtlich der Bereitstellung der Fragmente eingehalten werden können. Hierfür muß das zeitliche *worst-case* Verhalten aller Komponenten betrachtet werden. Bezogen auf das Speichersystem wird durch Annahme der größtmöglichen Verzögerungen (Positionierungszeit, Rotationsverzögerung, Transferzeit) die kleinstmögliche Bedienrate ermittelt und hieraus die maximale Anzahl bedienbarer Datenströme berechnet. Bei dieser konservativen Abschätzung bleibt die stochastische Natur dieser Verzögerungen (variable Transferraten bei Mehr-Zonen-Platten, zufällige Rotationsverzögerung, variable Fragmentgrößen) unberücksichtigt. Dies führt zwangsläufig zu einer sehr geringen Anzahl an Datenströmen, die von der Zulassungkontrolle gleichzeitig akzeptiert werden können. Als Preis für die deterministische Garantie sind die Platten nur schwach ausgelastet.

Die Verwendung stochastischer Service-Garantien läßt sich fast mit allen Anwendungen vereinbaren, da die meisten Multimedia-Anwendungen gegenüber einer temporären Verringerung der Service-Qualität tolerant sind (ausgenommen z.B. medizinische und sicherheitskritische Anwendungen). Mit einer dynamischen Anpassung der Wiedergabequalität lassen sich zum Beispiel kurze Störungen vom Benutzer unbemerkt überbrücken [TKC+97, HKR97].

Bei diesen Anwendungen läßt sich durch Verwendung stochastischer Service-Garantien die *Auslastung* der Platten, d.h. der Zeitanteil, in der der Server mit der Ausführung von Aufträgen beschäftigt ist, steigern. Ausschlaggebend hierfür ist, daß das stochastische Verhalten der Magnetplatten und der Datenströme in die Berechnungen einfließen kann [MSB98, VGGG94, CZ94a] und somit bei einer festgelegten tolerierbaren Häufigkeit der Störungen eine größere Anzahl an Datenströmen zugelassen werden kann. So läßt sich mit stochastischen Service-Garantien, die Anzahl der Datenströme, die gleichzeitig bedient werden, u.U. verdoppeln, was zu einem besseren Preis-Leistungs-Verhältnis führt.

### 1.3.3 Gemeinsame Nutzung der Ressourcen

Der größte Teil der bisher durchgeführten Arbeiten hat ein weiteres Optimierungspotential, nämlich die gemeinsame Nutzung der Magnetplatten durch kontinuierliche *und* diskrete Daten, außer acht gelassen. Sie betrachten größtenteils entweder Daten-Server, die ausschließlich kontinuierliche Daten liefern (z.B. Video-Server) oder diskrete Daten (z.B. File-Server) zur Verfügung stellen. Um aber die zur Verfügung stehenden Hardware-Ressourcen optimal auszunutzen, ist es zwingend notwendig, alle Daten auf gemeinsam genutzten Magnetplatten abzulegen und keine Trennung nach Datentyp vorzunehmen. Durch diese gemeinsame Nutzung der Hardware-Ressourcen kann vermieden werden, daß bei einer temporären Verschiebung der Zugriffshäufigkeit auf kontinuierliche und diskrete Daten eine zu geringe Auslastung bzw. eine Überlastung der Ressourcen hervorgerufen wird. Dieser Fall tritt z.B. dann ein, wenn zeitweise ein häufigerer Zugriff auf Geschäftsdokumente erfolgt und in anderen Zeiträumen, z.B. während Schulungszeiten, ein verstärkter Zugriff auf audio-visuelles Unterrichtsmaterial benötigt wird. Lastschwankungen beim Zugriff auf beide Datentypen können somit ausgeglichen werden.

In der Literatur ist dieser Ansatz der gemeinsamen Nutzung der Magnetplatten bisher nur unzureichend betrachtet worden. In den ersten Veröffentlichungen werden lediglich Methoden aufgezeigt [MNO+96], die zeigen wie bei einer *gemischten Arbeitslast*, d.h. bei Zugriffen auf beide Datentypen, das Scheduling der Plattenzugriffe durchgeführt werden kann. Weitergehende Arbeiten [GLdG98, CPRV98] beschreiben, wie eine Optimierung des Schedulings durchgeführt werden kann. In allen Fällen unterbleibt allerdings eine Untersuchung wie stochastische Service-Garantien für kontinuierliche und diskrete Daten sichergestellt werden können.

### 1.3.4 Sorgfältige Kapazitätsplanung

Durch die Kapazitätsplanung wird vor Inbetriebnahme des Daten-Servers der notwendige Ressourcenbedarf in Abhängigkeit von der zu erwartenden Last bestimmt. Hierzu gehört z.B. die Festlegung der Magnetplattenanzahl im Disk-Array. Eine Überschätzung des tatsächlichen Bedarfs führt zu hohen Investitionskosten und einer zu geringen Auslastung der Platten. Die Unterschätzung führt zur Nichteinhaltung der Service-Garantien und bei wiederholter Aufstockung der Ressourcen zu Betriebsunterbrechungen.

Damit sorgfältige Kapazitätplanung möglich ist, muß durch ein Verhaltensmodell des Daten-Servers eine Vorhersage der Server-Performance bei vorgegebener Last durchgeführt werden können.

## 1.4 Beitrag und Aufbau der Arbeit

Die vorliegende Arbeit liefert die Grundlagen, damit die im vorangegangenen Abschnitt beschriebenen vier notwendigen Bedingungen zur Realisierung eines multimedialen Daten-Servers mit günstigem Preis-Leistungs-Verhältnis erfüllt werden können. Der Beitrag besteht im einzelnen aus:

1. der Untersuchung verschiedener Scheduling-Algorithmen zur Optimierung des Datenzugriffs bei gemischten Arbeitslasten sowie
2. der Entwicklung eines analytisches Modells zur stochastischen Berechnung des Performance-Verhaltens.

### 1.4.1 Scheduling-Algorithmen für gemischte Arbeitslasten

Das Scheduling für Datenzugriffe bei gemischten Arbeitslasten ist bisher nur unzureichend in Ansätzen untersucht worden. Erst neuere Arbeiten [MNO+96, GLdG98, CPRV98] widmen sich diesem Thema. Ursache für Probleme, die sich aus der gemeinsamen Nutzung des Speichersystems ergeben, sind die unterschiedlichen anwendungsorientierten Performance-Anforderungen (geringe Störungsrate / kurze Antwortzeit) für beide Datentypen. Beide Anforderungen müssen bei der Durchführung der Plattenzugriffe berücksichtigt werden. Darüber hinaus muß den systemorientierten Performance-Anforderungen Genüge geleistet werden, indem durch geeignete Methoden, z.B. durch eine Optimierung des Datenzugriffs, die Effizienz der Platten gesteigert und hierdurch eine Erhöhung des Durchsatz ermöglicht wird. In der Arbeit werden folgende Punkte betrachtet:

- Es wird eine Taxonomie vorgestellt, mit der verschiedene Scheduling-Strategien anhand weniger Parameter beschrieben werden können.
- Es werden mehrere Scheduling-Strategien, die eine Optimierung des Datenzugriffs durchführen, hinsichtlich ihrer Auswirkung auf die Service-Qualität (Antwortzeit, Störungsrate) untersucht. Experimentell wird die beste Scheduling-Strategie ermittelt.
- Es wird im Detail die Implementierung der zu den Scheduling-Strategien gehörigen Algorithmen beschrieben und ein Prototypsystem vorstellt, in das ein Teil der Scheduling-Algorithmen integriert worden ist.

### 1.4.2 Stochastische Modellierung des Performance-Verhaltens

Um Aussagen über die Performance des Daten-Servers zu erhalten, ohne hierzu Messungen in einer Simulation oder einer realen Implementierung durchführen zu müssen, kann auf Basis mathematischer Verfahren ein analytisches Modell des Server-Verhaltens erstellt werden. In der vorliegenden Arbeit wird ein analytische Modell, bestehend aus zwei Teilmodellen, entwickelt, das die stochastische Service-Qualität für kontinuierliche und diskrete Daten berechnen kann. Die Berechnungen erfolgen für eine festgelegte *Last*, d.h. für eine festgelegte Häufigkeit von

Zugriffen auf diskrete Daten und eine festgelegte Anzahl von Datenströmen, die parallel bedient werden.

Grundlage beider Teilmodelle ist eine detaillierte stochastische Modellierung des Plattenzugriffs, in der zum ersten Mal auch Mehr-Zonen-Platten und die Verwendung von SCAN-Algorithmen Berücksichtigung finden. Bisherige Arbeiten [BDEM+94, VGG95, ORS96] sind bei der Modellierung stets von Ein-Zonen-Platten ausgegangen und haben konservative worst-case Annahmen für Rotationsverzögerung und Positionierungszeiten vorausgesetzt. Sofern eine stochastische Modellierung vorgenommen wurde [CZ94b, VGGG94, CL96], ist eine Berücksichtigung von SCAN-Algorithmen für den effizienten Plattenzugriff unterblieben.

Das erste Teilmodell berechnet für kontinuierliche Daten eine obere Schranke für die Häufigkeit, mit der eine festgelegte Störungsrate überschritten wird. Hierbei wird eine stochastische Größenverteilung der Fragmente vorausgesetzt, wie sie bei kontinuierlichen Datenobjekten mit variabler Datenrate auftreten. Bisherige Arbeiten haben hier größtenteils konstante Fragmentgrößen angenommen.

Das zweite Teilmodell ermittelt auf Basis eines Warteschlangenmodells eine obere Schranke für die Wahrscheinlichkeit, daß eine festgelegte Antwortzeitschranke überschritten wird. Auch hier wird eine stochastische Größenverteilung der diskreten Datenobjekte angenommen und zudem ein stochastischer Poisson-Ankunftsprozeß der Aufträge, die auf diskrete Daten zugreifen, berücksichtigt. Dies ist die erste Arbeit, die sich diesem Problem im Kontext mit multimedialen Daten-Servern widmet und dabei eine praktisch umsetzbare Lösung erarbeitet.

Einsetzbar sind diese analytischen Vorhersagen erstens bei der Konfiguration des Daten-Servers, bei der man mit möglichst geringem Ressourceneinsatz, d.h. mit einer möglichst geringen Anzahl Magnetplatten, die Performance-Anforderungen der Anwendungen erfüllen möchte. Zweitens bildet die analytische Vorhersage der Server-Performance die Grundlage für die Zulassungskontrolle kontinuierlicher Datenobjekte unter Verwendung stochastischer Service-Garantien.

Zusammenfassend werden in der Arbeit folgende Punkte betrachtet:

- Es erfolgt eine stochastische Modellierung des Plattenzugriffs für Ein- und Mehr-Zonen-Platten bei Verwendung von SCAN- und FCFS-Algorithmen.
- Es wird ein stochastisches Modell entwickelt, daß Service-Garantien sowohl für kontinuierliche als auch diskrete Datenobjekte berechnen kann.
- Die Vorhersagen des analytischen Modells werden mit Meßergebnissen aus Experimenten verglichen, in denen die entwickelten Scheduling-Algorithmen für gemischte Arbeitslasten simuliert werden.
- Basierend auf dem analytischen Modell wird ein Konfigurationsalgorithmus vorgestellt, der bei Vorgabe der gewünschten Service-Garantien und der erwarteten Last die minimale Anzahl benötigter Magnetplatten berechnet.

### 1.4.3 Aufbau der Arbeit

Die Gliederung der Arbeit erfolgt grob in fünf Teile. Nach dieser Einleitung folgt der zweite Teil, die sich mit Scheduling-Algorithmen für gemischte Arbeitslasten befaßt.

Hierin wird in Kapitel 2 zunächst eine Übersicht über verschiedene Möglichkeiten der Datenpartitionierung und Datenplazierung für kontinuierliche und diskrete Datenobjekte gegeben sowie deren Vor- und Nachteile abgewägt.

In Kapitel 3 folgt eine Übersicht über verschiedene Scheduling-Strategien, die ausschließlich zur Lieferung kontinuierlicher Datenobjekte entwickelt worden sind. Auf Basis dieser Strategien werden verschiedene Möglichkeiten aufgezeigt, wie ein Zugriff auf diskrete Daten integriert werden kann. Die Freiheitsgrade werden in einer Taxonomie anhand weniger Parameter beschrieben und es wird gezeigt, wie die Scheduling-Strategien konkret in Algorithmen umgesetzt werden können und welche Datenstrukturen hierbei Verwendung finden.

Die Evaluation der Scheduling-Algorithmen wird in Kapitel 4 durchgeführt. Die Untersuchung beschränkt sich auf diejenigen Algorithmen, die eine hohe Kapazität hinsichtlich der Bearbeitung von Plattenzugriffen erwarten lassen. Es wird die Modellierung der Hardwarekomponenten, der Datenobjekte und der Last beschrieben und die Simulationsumgebung, in der die Messungen durchgeführt werden, näher erläutert. Zum Schluß wird als Fazit eine Empfehlung für die Auswahl des geeignetsten Scheduling-Algorithmus gegeben.

Der dritte Teil der Arbeit beschäftigt sich mit der Entwicklung der beiden analytischen Teilmodelle zur Vorhersage der Server-Performance. Kapitel 5 bereitet hierfür die Grundlagen vor und beschreibt die stochastische Modellierung der Magnetplatte und u.a. die Größenverteilung für kontinuierliche und diskreten Datenobjekte.

Kapitel 6 entwickelt das erste analytische Teilmodell. Mit ihm ist es möglich, die Restwahrscheinlichkeit abzuschätzen, mit der eine festgelegte Störungsrate überschritten wird. Als erstes wird die Zeit stochastisch charakterisiert, die benötigt wird, eine festgelegte Anzahl von Datenströmen zu bedienen. Hierzu werden zwei Ansätze zur Berechnung konkreter Werte vorgestellt, auf die aufbauend im Anschluß die stochastische Abschätzung der Störungsrate erfolgt. Als Abschluß werden die Vorhersagen des Modells mit Meßergebnissen aus Simulationen verglichen, um die Güte des Modells zu dokumentieren.

Das zweite analytische Teilmodell wird in Kapitel 7 vorgestellt. Es ermöglicht, die Restwahrscheinlichkeit der Antwortzeit abzuschätzen. Analog zum vorangegangenen Kapitel erfolgt auch hier zum Abschluß eine Evaluierung des Modells, indem ein Vergleich der Vorhersageergebnisse mit Meßergebnissen durchgeführt wird.

Das Kapitel 8 integriert die beiden analytischen Modelle aus Kapitel 6 und 7 in einem Konfigurationsalgorithmus. Der Algorithmus liefert auf Basis der analytischen Modelle die minimale Anzahl benötigter Platten, damit der Daten-Server vorgegebene Performance-Anforderungen erfüllen kann.

Der vierte Teil der Arbeit beschäftigt sich mit der Realisierung des Prototypsystems, in das die Ideen und Ansätze der vorangegangenen Kapitel eingeflossen sind. In Kapitel 9 erfolgt eine Übersicht über Funktionalität und Architektur.

Den Abschluß der Arbeit bildet der fünfte Teil mit einem Ausblick auf Erweiterungsmöglichkeiten in Kapitel 10 und einer Zusammenfassung in Kapitel 11 und 12.

Im Anhang der Arbeit ist ein Glossar und eine Liste der verwendeten Symbole zu finden.

# Teil II

# Scheduling-Algorithmen

# Kapitel 2

# Plazierung der Datenobjekte

Dieses Kapitel gibt zunächst eine Übersicht über verschiedene in der Literatur vorgestellte Verfahren zur Datenplazierung von diskreten und kontinuierlichen Datenobjekten. Weiterhin wird unter Abwägung von Vor- und Nachteilen eine Empfehlung zum Systementwurf ausgesprochen.

Dateisysteme, wie sie bisher zur Ablage von diskreten Daten in konventionellen Systemen Verwendung finden, sind primär darauf ausgerichtet, durch die Plazierung der Daten eine effiziente Ausnutzung der Speicherplatzkapazität zu gewährleisten [SG94b] und insbesondere eine Fragmentierung des Speicherplatzes zu verhindern. Hierbei bleiben die spezifischen Anforderungen kontinuierlicher Datenobjekte sowie deren spezifische Zugriffsmuster unberücksichtigt. Durch spezielle Techniken kann deren Plazierung die rechtzeitige und effiziente Ausführung der kontinuierlichen Datenzugriffe unterstützen und den meist sequentiellen Zugriff auf diese Objekte ausnutzen.

In diesem Kapitel wird die Plazierung der Daten beider Objekttypen in einem Disk-Array beschrieben. Ausgangspunkt für die Plazierung ist deren Partitionierung, d.h. die Zerlegung der Datenobjekte in kleinere Teile, und die Verteilung auf mehrere Platten des Disk-Arrays. Dieses Verfahren ist als *Disk-Striping* oder *Data-Striping* [Kim86, SGM86, PGK88, LKB87, CLG+94] bekannt und kann sowohl auf diskrete [Zab94, SWZ98] als auch auf kontinuierliche Datenobjekte [BGMJ94, GK95b, ORS96] angewandt werden. Es bietet für beide Datentypen mehrere Vorteile [GVK+95, TPBG93]:

- Steigerung der Bedienrate:
  Dadurch, daß mehrere Platten beim Datenzugriff gleichzeitig zusammenarbeiten können, steht pro Datenzugriff eine höhere Bedienrate zur Verfügung. Hierdurch kann unter bestimmten Bedingungen der Durchsatz erhöht werden.

- Balancierung der Last:
  Durch die Verteilung der Zugriffe auf mehrere Platten ergibt sich eine Lastverteilung unter der Voraussetzung, daß die Datenobjekte gleichmäßig über alle Platten verteilt werden und daß die Datenzugriffe gleichmäßig alle Datenobjekte betreffen.

- Ausfallsicherheit:
  Durch redundante Datenhaltung ist es möglich, Lese- und Schreibfehler oder den Ausfall einzelner Platten zu maskieren.

- Verringerung der Speicherfragmentierung:
  Bei sehr großen Datenobjekten, besonders bei kontinuierlichen Datenobjekten kann die Speicherplatzkapazität einer Platte nicht vollständig durch eine ganze Anzahl dieser Objekte ausgenutzt werden. Werden hingegen kleinere Teile des kontinuierlichen Datenobjektes verteilt, werden alle beteiligten Platten gleichmäßiger aufgefüllt und damit hinsichtlich der Speicherplatzkapazität gleichmäßiger ausgelastet.

Die Übersicht beginnt zunächst mit Plazierungsverfahren für diskrete Datenobjekte. Daraufhin werden verschiedene Plazierungsvarianten für kontinuierliche Datenobjekte beschrieben, bevor eine Empfehlung zum Systementwurf bezüglich beider Datentypen abgegeben wird.

## 2.1 Plazierung diskreter Datenobjekte

Die Partitionierung, d.h. die Zerlegung diskreter Datenobjekte erfolgt auf Blockebene. Das Objekt wird hierzu in *Segmente* zerlegt, die logisch benachbarte Daten enthalten und jeweils zusammenhängend in aufeinanderfolgenden logischen Blöcken einer Platte gespeichert werden. Die Größe der Segmente, die als *Striping-Tiefe* bezeichnet wird, hat einen konstanten Wert, der als Vielfaches der Plattenblockgröße festgelegt wird. Die Striping-Tiefe kann für jedes Datenobjekt individuell festgelegt werden. Die Verteilung der Segmente erfolgt objektweise und kann das gesamte Disk-Array oder nur einen Teil hiervon betreffen. Nach dem *Round-Robin-Prinzip* werden benachbarte Segmente eines Datenobjektes benachbarten, ggf. zyklisch benachbarten, Platten des Disk-Arrays zugeordnet. Neben dieser Form der *regelmäßigen* Plazierung, die einem geordneten Muster folgt, erfolgt bei einer *randomisierten* Verteilung die Zuordnung auf die Platten zufällig. Die Anzahl der Platten, die bei der Verteilung verwendet werden, bezeichnet man als *Striping-Breite* des Datenobjektes. Die Abbildung 2.1 illustriert beide Parameter an einem Disk-Array mit fünfzehn Platten auf dem vier verschiedene diskrete Datenobjekte abgelegt sind.

Die Datenobjekte A und B besitzen beide eine Striping-Breite von fünf Platten. Die Segmentgröße, d.h. die Striping-Tiefe von Objekt A beträgt vier Blöcke, die von Objekt B hat einen Wert von zwei Blöcken. Die Auswirkung der unterschiedlichen Striping-Tiefe wird bei den Datenzugriffen deutlich, die auf einem Ausschnitt des Datenobjektes ausgeführt werden. Soll etwa ein acht Block großer Ausschnitt von Objekt A gelesen werden, so verteilen sich dessen Blöcke auf mindestens zwei und höchstens drei verschiedene Platten. Beim Zugriff auf den Ausschnitt können so mindestens zwei Platten gleichzeitig den Auftrag bearbeiten. Das Lesen eines Ausschnitts mit einer Größe von acht Blöcken von Objekt B hat im Vergleich hierzu einen *Parallelitätsgrad* von mindestens vier und höchstens fünf. Der Parallelitätsgrad bezeichnet die Anzahl der an einem Datenzugriff beteiligten Platten. Er ist nach oben durch die Striping-Breite beschränkt.

Durch den parallelen Datenzugriff steht das Vielfache der Transferrate einer Platte zur Verfügung und die Transferzeit verkürzt sich im Vergleich zu einer Datenplazierung, bei der alle Blöcke ausschließlich von einer Platte gelesen werden müssen. Im Gegensatz zur Transferzeit werden die Positionierungszeiten und die Rotationsverzögerungen an denen am Datenzugriff beteiligten Platten nicht verringert, sondern bleiben in unveränderter Größe bestehen. Für kleine Datenzugriffe, d.h. für Zugriffe mit einer geringen Anzahl von Blöcken, die wesentlich von der

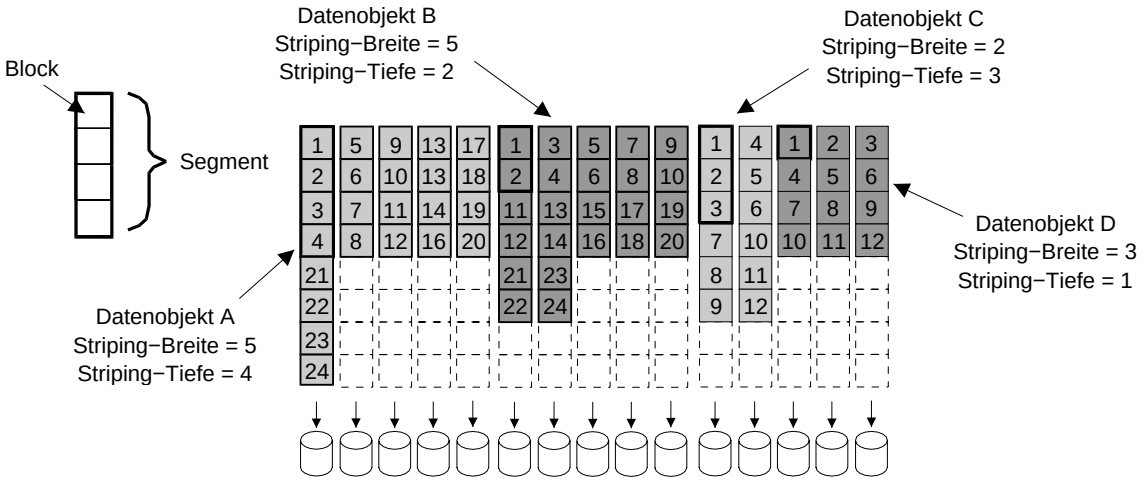



Abbildung 2.1: Partitionierung und regelmäßige Plazierung mit verschiedenen Striping-Tiefen und -Breiten.

Positionierungszeit und der Rotationsverzögerung abhängen, kann folglich eine Parallelisierung zu einer Erhöhung der Bedienzeit führen.

Die Parallelisierung hat weiterhin Einfluß auf den Durchsatz der Aufträge. Da Positionierungszeiten und die Rotationsverzögerungen in unveränderter Größe anfallen, steigt die *Plattenbelegungszeit* an. Sie ist als Summe der Einzelbedienzeiten bei einem parallelen Datenzugriff definiert (siehe Abbildung 2.2) und charakterisiert den Ressourcenverbrauch eines Zugriffs. Je mehr Ressourcen pro Zugriff benötigt werden, desto geringer ist die Anzahl der Zugriffe, die pro Zeiteinheit ausgeführt werden können. Dies bedeutet das mit wachsendem Parallelitätsgrad der Durchsatz sinkt.

Der sinkende Durchsatz bewirkt, daß Aufträge ggf. für einen längeren Zeitraum auf das Ende vorangegangener Aufträge warten müssen, bevor ihre Ausführung stattfinden kann. Dies bewirkt eine verlängerte Wartezeit, die die verkürzte Transferzeit kompensieren kann.

In verschiedenen Arbeiten [CP90, GHW90, CT93, Zab94, CL95] wird gezeigt, wie man durch analytische Berechnungen bei einer festgelegten Last einen optimalen Parallelitätsgrad erreicht, der die Antwortzeit, d.h. die Summe aus Warte- und Bedienzeit, minimiert. Weiterhin wird erläutert, wie bei Kenntnis der Auftragsgrößen die Striping-Breite und -Tiefe gewählt werden muß, um diesen optimalen Parallelitätsgrad beim Plattenzugriff zu gewährleisten.

## 2.2 Plazierung kontinuierlicher Datenobjekte

Im folgenden werden die bereits aus dem vorangegangenen Abschnitt bekannten Verfahren auf kontinuierliche Datenobjekte angewandt. Hierbei sind aufgrund der Charakteristik der verwendeten Daten verschiedene Freiheitsgrade möglich.

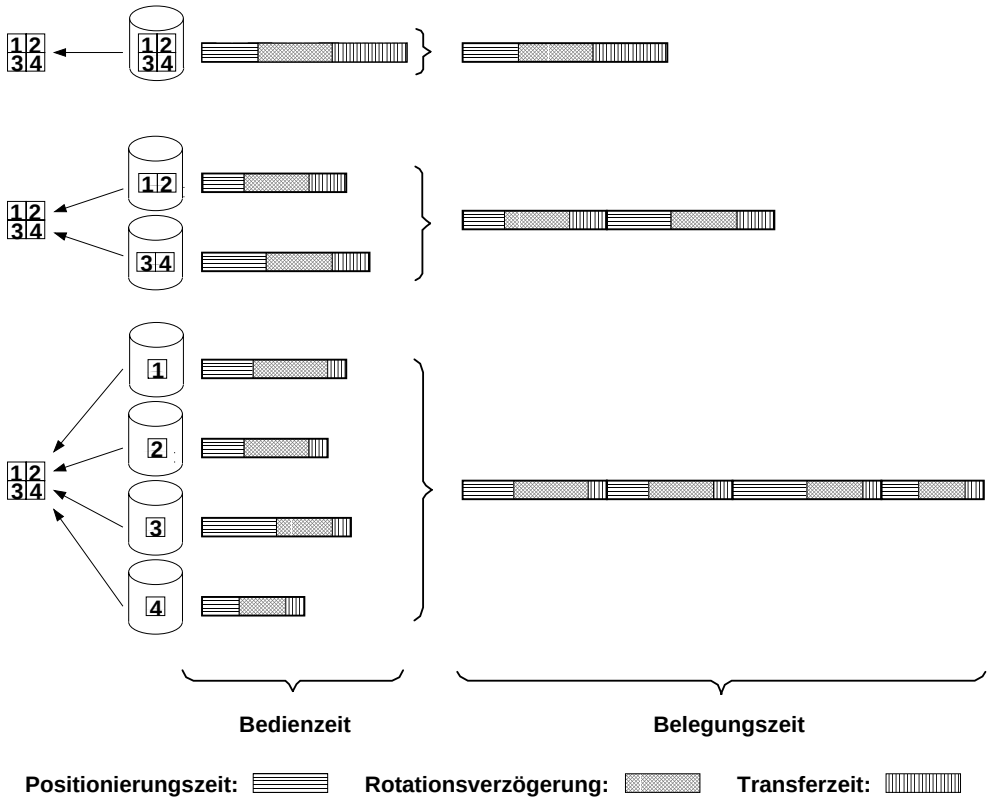



Abbildung 2.2: Plattenbelegungszeit für verschiedene Parallelitätsgrade [Zab94]

### 2.2.1 Datenpartitionierung

Die Verteilung der kontinuierlichen Datenobjekte auf das Disk-Array des Daten-Servers erfordert, wie es auch bei den diskreten Datenobjekten der Fall war, zunächst deren Zerlegung (*Partitionierung*) in Teilobjekte. Dieses geschieht zunächst auf einer anwendungsorientierten Ebene, da die Teilobjekte, die im folgenden als *Fragmente* bezeichnet werden, die kleinste logische Dateneinheit darstellen, auf die der Daten-Server mit Hilfe seines Dateisystems zugreifen kann. Eine weitergehende blockorientierte Zerlegung der Fragmente in Segmente, ist darüber hinaus möglich. Es lassen sich zwei Arten dieser anwendungsorientierten Partitionierung unterscheiden.

**KDL-Partitionierung:** Die *KDL-Partitionierung* (KDL = konstante Datenlänge, engl. CDL = constant data length) erzeugt Fragmente mit konstanter Datenlänge, d.h. alle Fragmente beinhalten die gleiche Datenmenge in Bytes.

**KWL-Partitionierung:** Die *KWL-Partitionierung* (KWL = konstante Wiedergabelänge, engl. CTL = constant time length) erzeugt Fragmente mit konstanter Wiedergabelänge [CZ94a, CZ96, CBR95]. Bei konstanter Wiedergabelänge hat die Wiedergabedauer der in einem Fragment gespeicherten Informationen einen konstanten Wert. Daraus resultiert dann in der Regel, d.h. bei einer Kodierung mit variabler Datenrate, eine unterschiedliche Datenlänge der Fragmente.

Die kleinste sinnvolle Wiedergabelänge eines KWL-Fragments ist die Wiedergabedauer eines Einzelbildes ggf. mit den dazugehörigen Audiodaten, da ein Einzelbild für die Präsentation die kleinste einzeln darstellbare Einheit bildet. Üblicherweise werden mehrere Bilder in einem KWL-Fragment zusammengefaßt (siehe Abbildung 2.3). Ihre Anzahl ist abhängig von

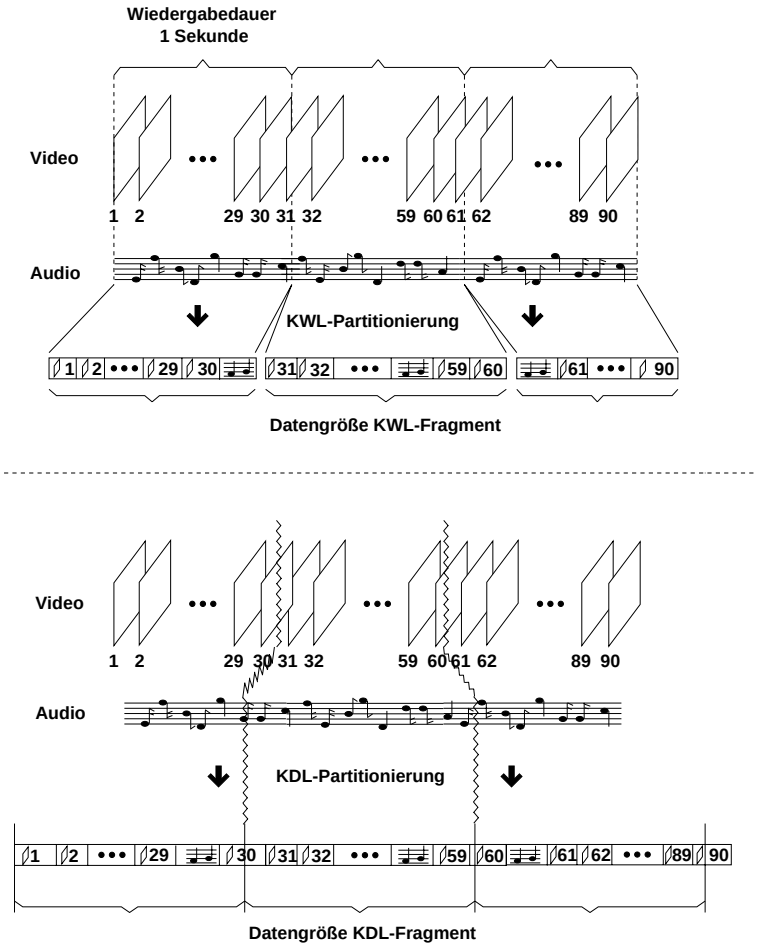



Abbildung 2.3: KWL-Partitionierung und KDL-Partitionierung im Vergleich

der Bildrate des kontinuierlichen Datenobjektes. Bei Verwendung üblicher digitaler Aufzeichnungsnormen [ITU90] werden 30 Einzelbilder pro Sekunde wiedergegeben, die z.B. zusammengefaßt in einem KWL-Fragment mit einer Wiedergabelänge von einer Sekunde enthalten sind.

Der Unterschied zwischen diesen beiden Partitionierungsverfahren wirkt sich nur bei VDR-Datenobjekten aus. Bei ihnen schwankt aufgrund der variablen Datenrate die Menge der pro Zeiteinheit abgespeicherten Daten, und die Größe eines KWL-Fragments ist davon abhängig, wie stark deren Inhalt komprimiert ist. Bei Objekten mit konstanter Datenrate sind KWL-Fragmente und KDL-Fragmente gleich groß und beide Partitionierungsverfahren liefern identische Ergebnisse.

### 2.2.2 Regelmäßige Plazierung

Die Plazierung der Fragmente auf dem Disk-Array des Daten-Servers kann, wie bei den Segmenten diskreter Datenobjekte, entweder *regelmäßig* oder *randomisiert* erfolgen. Eine regelmäßige Plazierung bedeutet, daß die Auswahl der Platten, auf dem die Fragmente abgelegt werden, nach einem spezifischen Muster erfolgt. Die randomisierte Plazierung wählt hingegen die Platten nach dem Zufallsprinzip aus. Im folgenden wird zur Vereinfachung der Darstel-

lung davon ausgegangen, daß die Verteilung stets alle Platten betrifft. Dies bedeutet, daß mit einer für alle Datenobjekte gleichen Striping-Breite gearbeitet wird, deren Wert der Anzahl der Platten im Disk-Array entspricht. Sofern das Disk-Array in mehrere disjunkte Plattenmengen aufgeteilt wird, denen jeweils einzelne kontinuierliche Datenobjekte zugewiesen werden [CBR95, GK95b], können die im folgenden beschrieben Verfahren ebenso auf jede dieser Teilmengen angewandt werden.

Bei der regelmäßigen Verteilung der Daten lassen sich zwei Verfahren in Abhängigkeit vom Parallelitätsgrad beim Datenzugriff unterscheiden.

**Feinkörniges regelmäßiges Striping (FRS):** Das *feinkörnige regelmäßige Striping* (engl. fine grained striping) [TPBG93, ORS96] (siehe Abbildung 2.4) führt eine weitere blockorientierte Zerlegung der Fragmente in Segmente durch und verteilt diese Segmente gleichmäßig über alle Platten. Konzeptionell kann damit das Disk-Array als eine einzige virtuelle Platte betrachtet werden.

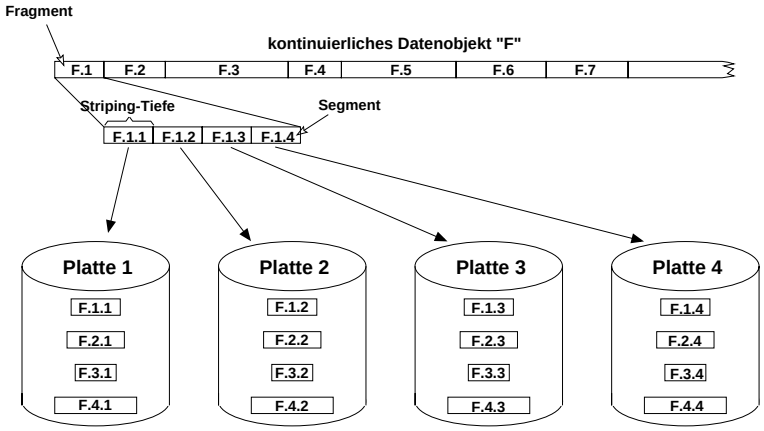



Abbildung 2.4: Feinkörniges regelmäßiges Striping: jedes Fragment wird gleichmäßig zerlegt über alle Platten verteilt

**Grobkörniges regelmäßiges Striping (GRS):** Das grobkörnige regelmäßige Striping (engl. coarse grained striping) [ORS96, GK95b], plaziert zyklisch aufeinanderfolgende Fragmente vollständig auf benachbarten Platten des Disk-Arrays (siehe Abbildung 2.5). Eine weitergehende blockorientierte Zerlegung findet nicht statt.

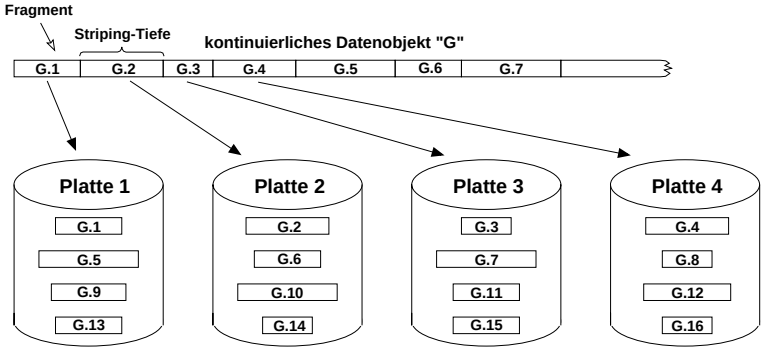



Abbildung 2.5: Grobkörniges regelmäßiges Striping: jedes Fragment liegt auf einer einzigen Platte

In beiden Verfahren werden Segmente bzw. Fragmente jeweils zusammenhängend auf den Platten abgelegt, so daß beim Zugriff hierauf eine Repositionierung des Schreib-Lese-Kopfes nur auf benachbarte Spuren der Platte notwendig ist. Da das Fragment die kleinste Zugriffeinheit bildet, wird angenommen, daß es entweder vollständig geschrieben oder vollständig gelesen wird. Beim feinkörnigen Striping entspricht somit der Parallelitätsgrad der Anzahl der Platten, da alle Platten beim Fragmentzugriff beteiligt sind. Das grobkörnige Striping hat einen Parallelitätsgrad von eins, da das Fragment lediglich von einer Platte geliefert wird.

Beim grobkörnigen Striping entspricht die Striping-Tiefe der Größe eines Fragments. Bei variablen Fragmentgrößen ergibt sich somit auch eine variable Striping-Tiefe. Beim feinkörnigen Striping ist die Striping-Tiefe durch die Größe eines Segments bestimmt. Auch hier ist die Striping-Tiefe ein variabler Parameter, der von der Datenkomprimierung und der Partitionierungsart bestimmt wird.

Beide Striping-Schemata können sowohl auf KWL- als auch auf KDL-Fragmente angewandt werden. Bei der Verwendung von KDL-Fragmenten der Größe $s$ und grobkörnigem Striping auf $P_{disks}$ Platten ergibt sich dieselbe Datenplazierung, als wenn KDL-Fragmente der Größe $P_{disks} * s$ und feinkörniges Striping einsetzt würde. Dieses gilt aufgrund der variablen Komprimierungsrate nicht, wenn KWL-Fragmente mit einer Wiedergabelänge von $T$ bzw. $P_{disks} * T$ benutzt werden.

### 2.2.3 Randomisierte Plazierung

Bei der randomisierten Plazierung werden Fragmente den zur Verfügung stehenden Platten zufällig zugeordnet. Hierbei gibt es wiederum zwei unterschiedliche Verfahren. Das *feinkörnige zufällige Striping (FZS)* [BMW96, KGK96] ist vergleichbar mit der regelmäßigen Variante, die im vorigen Abschnitt beschrieben wurde. Der Unterschied liegt darin, daß für jedes Fragment eine zufällige Teilmenge der Platten bestimmt wird, auf der die Segmente des Fragments plaziert werden. Die Größe der Plattenmenge ist für alle Fragmente konstant. Hat sie den Wert $i$, so ergeben sich bei insgesamt $P_{disks}$ Platten $\binom{i}{P_{disks}}$ Möglichkeiten für die Plazierung eines Fragments. Wie beim feinkörnigen regelmäßigen Striping erfordert der Zugriff auf ein Fragment den gleichzeitigen Zugriff auf alle Segmente, d.h, es sind $i$ Platten beim Zugriff beteiligt.

Das *grobkörnige zufällige Striping (GZS)* [Kor97] plaziert im Vergleich zum regelmäßigen Verfahren Fragmente nicht mehr zyklisch, sondern wählt die Platte, auf der jeweils ein Fragment vollständig abgelegt wird, zufällig aus.

## 2.3 Empfehlungen zum Systementwurf

Die in den vorangegangenen Abschnitten vorgestellten Partitionierungs- und Plazierungsverfahren, lassen sich beliebig miteinander kombinieren. Unter bestimmten Bedingungen sind aber nur weniger Kombinationen in Verbindung mit dem im nächsten Kapitel vorgestellten Scheduling-Algorithmen sinnvoll. Im voraus lassen sich bereits folgende Vor- und Nachteile erkennen.

### 2.3.1 Empfehlung für diskrete Datenobjekte

Die Partitionierung der diskreten Daten kann auf Basis der in der Literatur [CP90, GHW90, CT93, Zab94, CL95] vorgestellten analytischen Modelle erfolgen. Da die Modelle aber lediglich eine Optimierung der Datenplazierung vornehmen und ansonsten die Plattenzugriffe in FCFS-Reihenfolge voraussetzen, erscheint die Partitionierung nur dann sinnvoll, wenn der Daten-Server gewährleisten kann, daß die Zugriffe auf diskrete Datenobjekte entsprechend den Modellen auch in FCFS-Reihenfolge durchgeführt werden.

Sollte der Daten-Server SCAN-Algorithmen zur Optimierung der Bedienungsreihenfolge verwenden, so ist unklar, in welchem Ausmaß sich ein größerer Parallelitätsgrad überhaupt auf eine Senkung der Antwortzeit auswirken kann. Bevor diese offene Frage in zukünftigen Arbeiten nicht geklärt wird, sollte in diesem Fall auf eine Partitionierung diskreter Datenobjekte verzichtet und der Parallelitätsgrad 1 verwendet werden.

### 2.3.2 Empfehlung für kontinuierliche Datenobjekte

#### konstante Datenlänge vs. konstante Wiedergabelänge

Wird eine KDL-Partitionierung verwendet, so vereinfacht sich das Speichermanagement sowohl auf den Platten als auch im Server- und Client-Puffer aufgrund der konstanten Fragmentgröße und es kann eine Fragmentierung des Speichers auf einfache Weise vermieden werden. Diese Fragmentierungsprobleme treten besonders in Systemumgebungen auf, in denen häufige Einfüge- und Löschoperationen auf den Daten (z.B. Video-/Audio-Editing), durchgeführt werden.

Spielen Fragmentierungsprobleme keine Rolle, da die Datenobjekte nach dem initialen Laden auf das Disk-Array selten oder überhaupt nicht mehr verändert werden (z.B. bei Video-On-Demand) oder fragmentierter Plattenspeicher durch kleinere diskrete Datenobjekte genutzt werden kann (z.B. bei News-On-Demand oder Tele-Teaching), so hat die KWL-Partitionierung insbesonders bei VDR-Datenobjekten große Vorteile. Die KWL-Partitionierung ist besonders für Server-Push-Architekturen geeignet, da sie die Datenflußanpassung stark vereinfacht. Hat die Wiedergabelänge aller Fragmente eines kontinuierlichen Datenobjektes einen konstanten Wert von z.B. einer Sekunde, so hat der Daten-Server dem Client in jeder Sekunde genau ein Fragment zu schicken, damit die Datenrate eingehalten werden kann. Zusätzliche Mechanismen zur Überwachung der Bedienrate sind somit nicht notwendig, und eine Realisierung ist relativ einfach möglich.

#### feinkörniges vs. grobkörniges Striping

Ein erhöhter Parallelitätsgrad von mehr als 1, wie es durch das feinkörnige Striping erreicht wird, hat für kontinuierliche Datenobjekte nur dann Vorteile, wenn eine Minimierung der Antwortzeit für den Zugriff auf ein Fragment notwendig ist. Dies ist z.B. bei einer Client-Pull-Architektur relevant, bei der der Client die Fristen der Bereitstellungszeitpunkte kurzfristig festsetzt.

Bei einer Server-Push-Architektur, bei der der Daten-Server die Bereitstellungszeitpunkte selbst bestimmt, spielt die Antwortzeit eine untergeordnete Rolle. Entscheidend ist hierbei der Durchsatz, der bei einem Parallelitätsgrad von $1$ maximal wird.

**regelmäßige vs. randomisierte Plazierung**

Die Vorteile einer regelmäßigen Plazierung ergeben sich beim Datenzugriff durch die extrem einfache Berechnung, auf welcher Platte ein Fragment abgelegt worden ist. Für das grobkörnige Striping ergibt sich die Plattennummer für das $x$-te Fragment nach folgender Berechnungsvorschrift: $((k+x-1) \bmod P_{disks})$. Hierbei wird angenommen, daß das Disk-Array aus $P_{disks}$ Platten besteht, die von $1$ bis $P_{disks}$ durchnumeriert sind und sich das erste Fragment mit der Nummer $1$ auf der Platte $k$ befindet. Außerdem ergeben sich regelmäßige vorhersehbare Zugriffsmuster, die es erlauben, durch die Zulassungskontrolle die Last auf den Platten gleichmäßig zu verteilen.

Im Vergleich hierzu wird für die randomisierte Plazierung zusätzliche Kapazität für Verwaltungsdaten benötigt, da eine algorithmische Berechnung der Plattennummer bei gegebener Fragmentnummer nicht mehr möglich ist.

Dagegen werden in [MSB98] die beiden wichtigsten Vorteile einer randomisierten Plazierung aufgeführt. Hierzu zählt eine implizite Lastbalancierung, die sich durch die zufällige Verteilung der Daten langfristig ergibt, ohne daß durch eine explizite Zulassungkontrolle regulierend eingegriffen werden muß. Des weiteren werden unvorhersehbare Zugriffsmuster unterstützt, die z.B. durch einige Anwendungen (Video-/Audio-Editing) erzeugen werden und die durch häufige Unterbrechungen, Vor- und Zurückspulen charakterisiert sind. Da bei der zufälligen Plazierung keine geordneten Zugriffmuster vorausgesetzt werden, können diese Anwendungen ohne zusätzliche Maßnahmen einfach unterstützt werden.

# Kapitel 3

# Scheduling der Datenzugriffe

In diesem Kapitel wird erläutert, wie der Zugriff auf kontinuierliche und diskrete Datenobjekte erfolgt. Zunächst werden dabei nur kontinuierliche Datenobjekte betrachtet und einige Lösungsmöglichkeiten aus der Literatur vorgestellt, die es erlauben, daß der Daten-Server gleichzeitig mehrere Benutzer mit Daten versorgen kann. Die Verfahren, die den konkurrierenden Datenzugriff auf die Systemressourcen kontrollieren und optimieren, werden in den folgenden Abschnitten als *Scheduling-Strategien* bezeichnet. In weiteren Abschnitten wird gezeigt, wie die Einbettung von Zugriffen auf diskrete Datenobjekte durchgeführt werden kann, ohne daß es hierbei zu einer wechselseitigen Störung der beiden Zugriffstypen kommt. Die hierbei möglichen Freiheitsgrade werden in Form einer Taxonomie anhand weniger Parameter beschrieben. Weiterhin wird gezeigt, wie die Scheduling-Strategien konkret in *Scheduling-Algorithmen* umgesetzt werden können und welche Datenstrukturen hierfür notwendig sind.

## 3.1 Scheduling-Strategien für kontinuierliche Datenzugriffe

Der Daten-Server liest die Daten eines kontinuierlichen Datenobjektes als Folge von Fragmenten von den Platten und speichert sie jeweils temporär zur Übertragung über das Netz im Server-Puffer. Ein Auftrag zum Lesen eines Fragments wird im folgenden als *K-Auftrag* bezeichnet. Ein K-Auftrag kann, falls das Fragment auf mehreren Platten plaziert worden ist (siehe feinkörniges Striping in Abschnitt 2.2.2), in weitere *K-Teilaufträge* zerlegt werden. Der Daten-Server muß durch eine intelligente Strategie sicherstellen, daß durch die rechtzeitige Bedienung eines Datenstromes, d.h. durch die rechtzeitige Ausführung der dazugehörigen K-Aufträge, keine zeitlichen Verzögerungen innerhalb des Datenstromes und damit Unterbrechungen in der stetigen Wiedergabe auftreten. Analog gilt dasselbe, wenn der Client bei der Aufnahme von Audio oder Video als Datenquelle fungiert und Daten auf dem Server in Quasi-Echtzeit abzulegen sind. In diesem Fall muß der Daten-Server durch die rechtzeitige Abnahme und das Speichern der Fragmente einen Überlauf im Client-Puffer verhindern. Im folgenden ist die Darstellung auf den Fall der Wiedergabe eines kontinuierlichen Datenobjektes beschränkt.

Es lassen sich *periodische* und *aperiodische* Scheduling-Strategien unterscheiden. Periodische Scheduling-Strategien sind dadurch gekennzeichnet, daß die Datenzugriffe in einem regelmäßigen zeitlichen Muster erzeugt werden, das bei den aperiodischen Verfahren nicht vorhanden ist.

### 3.1.1 Periodische Scheduling-Strategien

Das periodische Scheduling bedient jeden Datenstrom in einem periodischen Muster. Dabei erfolgt bei den meisten Scheduling-Strategien die Bedienung der Datenströme *rundenbasiert*. Hierzu wird die Zeit in Runden eingeteilt und in jeder Runde pro aktivem Datenstrom ein K-Auftrag ausgeführt, d.h. es wird ein Fragment gelesen und für den Netztransport im Server-Puffer zur Verfügung gestellt. Durch dieses rundenbasierte Vorgehen ist eine einfache Zulassungskontrolle möglich, indem pro Runde die Anzahl der gelesenen Fragmente auf einen maximalen Wert festgelegt wird.

Die Länge einer Runde hat den konstanten Wert $T$ und bestimmt die maximale Zeitspanne, die zwischen der Bereitstellung zweier aufeinanderfolgender Fragmente im Server-Puffer liegen soll. Um die Zwischenpufferung zu minimieren, muß die pro Runde gelesene Datenmenge der Datenmenge entsprechen, die durch den Client in einer Runde wiedergegeben wird. Dies ist mit KWL-partitionierten Datenobjekten einfach möglich, da deren Fragmente eine Wiedergabelänge besitzen, die der Rundendauer $T$ entspricht. Die Verwendung von KDL-partitionierten Daten bei Datenobjekten mit variabler Datenrate macht ein zusätzliches Prefetching von Fragmenten und damit zusätzliche Pufferung auf dem Client erforderlich, da in diesem Fall periodische Scheduling-Strategien die Daten mit einer konstanten Rate liefern, d.h. ein Fragment konstanter Größe pro Runde gelesen wird, die Wiedergabe aber mit variabler Rate erfolgt [CZ96]. Im weiteren wird von einer KWL-Partitionierung der kontinuierlichen Daten ausgegangen. Weiterhin soll der Client bereits nach dem Eintreffen der ersten Daten mit der Wiedergabe beginnen, ohne daß ein Zwischenspeichern eines vollständigen Fragments bzw. mehrerer Fragmente im Client-Puffer für das Starten der Wiedergabe notwendig ist.

Im folgenden sollen drei periodische, rundenbasierte Scheduling-Strategien [TPBG93, YCK93, GHBC94] näher erläutert werden, die sich hinsichtlich der Anordnung der Datenzugriffe innerhalb einer Runde unterscheiden:

- Scheduling mit fester Reihenfolge der K-Aufträge
- Scheduling mit variabler Reihenfolge der K-Aufträge
- Scheduling mit mehreren Zugriffsgruppen

Zur Vereinfachung der Darstellung wird zunächst nur eine einzige Magnetplatte betrachtet. Im Anschluß erfolgt eine Verallgemeinerung der Scheduling-Strategien für die Verwendung in einem Disk-Array.

#### Scheduling mit fester Reihenfolge der K-Aufträge

Das Scheduling mit festgelegter Reihenfolge der K-Aufträge teilt jede Runde in eine feste Anzahl von Zeitintervallen auf, die im folgenden als *Slots* bezeichnet werden. Jeder aktive Datenstrom wird vor dem Lesen des ersten Fragments einem freien Slot zugewiesen. In diesem Slot erfolgt die Ausführung der K-Aufträge in den nachfolgenden Runden. Das in einem Slot gelesene Fragment wird im Server-Puffer abgelegt, von wo es zu Beginn des nächsten Slots über das Netzwerk in den Client-Puffer übertragen und sofort wiedergegeben wird. Sollte in einer

Runde das Lesen eines Fragments zeitlich nicht vollständig innerhalb des Slots möglich sein, so wird die Frist zur Bereitstellung des zu lesenden Fragments überschritten. Der Client könnte diese Verzögerung nur durch zusätzlich zwischengespeicherte Daten überbrücken. Da überdies auch die fristgerechte Ausführung der folgenden K-Aufträge gefährdet wird, soll die Bearbeitung des K-Auftrages am Slotende abgebrochen und ein teilweise bereits eingelesenes Fragment verworfen werden. Durch die fehlenden Daten dieses Fragments entsteht eine Unterbrechung im Datenstrom, die eine Störung bei der Wiedergabe hervorruft.

Die zeitliche Länge eines Slots sollte folglich derart gewählt sein, daß die Ausführung eines K-Auftrages zeitlich innerhalb des durch den Slot definierten Zeitintervalls durchgeführt werden kann. Dies bedeutet, daß die Bedienzeit $T_{svc,K}$ des K-Auftrages $K$, die sich als Summe aus der Positionierungszeit $T_{pos,K}$, der Rotationsverzögerung $T_{rot,K}$ und der Transferzeit $T_{trans,K}$ zusammensetzt, kleiner als die Slotlänge $T_S$ sein muß. Folgendes Theorem beschreibt diese Bedingung.

**Theorem 3.1.1.** *Notwendige Bedingung für die fristgerechte Bereitstellung der Fragmente beim Scheduling mit festgelegter Reihenfolge: der K-Auftrag $K$ muß innerhalb seines zugewiesenen Slots mit der Slotlänge $T_S$ zeitlich ausführbar sein:*

$$T_{svc,K} = T_{pos,K} + T_{rot,K} + T_{trans,K} \leq T_S$$

Die maximale Anzahl der pro Runde bedienbaren Datenströme ergibt sich aus der Slotlänge $T_S$ sowie der Rundendauer $T$ zu $\lfloor T/T_S \rfloor$.

In Abbildung 3.1 ist das Scheduling mit festgelegter Reihenfolge der K-Aufträge veranschaulicht. Die Zeitachse erstreckt sich von links nach rechts und ist in drei Runden aufgeteilt. Das Ende einer Runde ist durch eine lange vertikale Linie markiert, das Ende eines Slots ist durch eine kurze vertikale Linie gekennzeichnet. Die *Plattenaktivität* bei der Bearbeitung der K-Aufträge wird durch Rechtecke symbolisiert, in denen die Nummer des zu lesenden Fragments eingetragen ist. Die Lage der Rechtecke gibt an, wann die Aufträge bearbeitet werden. Die Breite eines Rechtecks gibt die Zeitdauer an, die für die Ausführung des K-Auftrages benötigt wird. In den freien Bereichen auf der Zeitachse findet keine Plattenaktivität statt.

Da stets vollständige Slots für die Bearbeitung von K-Aufträgen zur Verfügung stehen sollen, erfolgt die Zuweisung neuer Datenströme an freie Slots nur bei deren Beginn. Hierdurch ergibt sich die maximale Startverzögerung dann, wenn ein Datenstrom just zu dem Zeitpunkt zugelassen wird, wenn der einzig freie Slot gerade aktiv geworden ist. Die maximale Startverzögerung ist dann die Summe aus Rundendauer und Slotlänge, d.h. $T + T_S$.

Bei einer einfachen Verwaltung des Server-Puffers wird für jeden Datenstrom ein Abschnitt im Server-Puffer reserviert, dessen Größe dem des größten Fragments des zugehörigen kontinuierlichen Datenobjektes entspricht. Da sich, wie man in Abbildung 3.1 sieht, die Zeitintervalle für die Wiedergabe eines Fragments und das Lesen des nachfolgendes Fragments potentiell überschneiden, ist ein zusätzlicher Speicherbereich für diesen Zeitraum erforderlich. Ist $S_K^{max}$ die maximale Größe eines Fragments, so ergibt sich bei maximal $N_{max}$ Datenströmen pro Runde eine maximal benötigte Größe des Server-Puffers von $(N_{max} + 1) * S_K^{max}$.

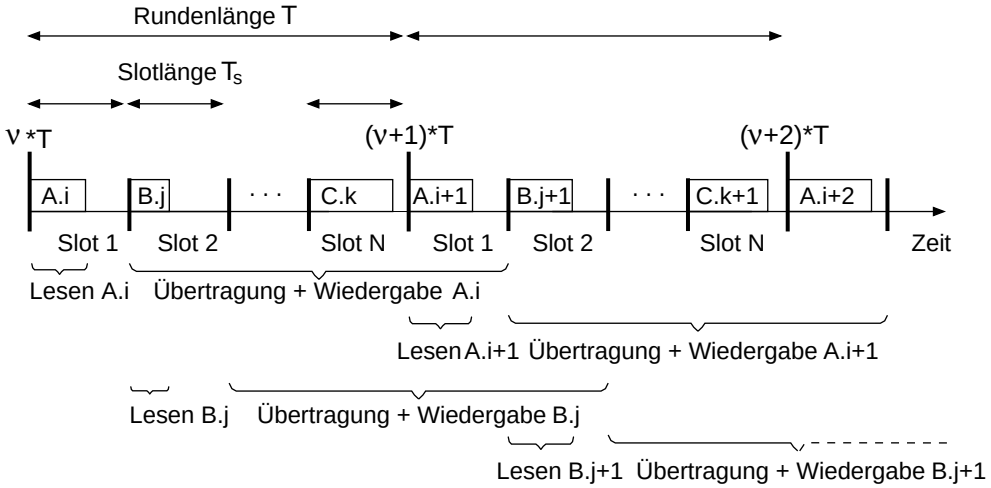



Abbildung 3.1: Scheduling mit fester Reihenfolge der K-Aufträge

**Scheduling mit variabler Reihenfolge der K-Aufträge**

Das Scheduling mit variabler Reihenfolge der K-Aufträge bedient alle Datenströme innerhalb eines einzigen Slots, der sich über die gesamte Runde erstreckt. Hierdurch ist es möglich, innerhalb einer Runde die Reihenfolge, in der die Datenströme bedient werden, frei zu wählen. Durch die Verwendung eines *SCAN-Algorithmus* [TP72, WGP94, RW94b], der die K-Aufträge entsprechend ihrer Position auf der Platte ordnet, werden Positionierungszeiten verkürzt. Weiterhin werden beim Scheduling mit variabler Reihenfolge die Varianzen der Bedienzeiten ausgeglichen, da die gesamte Rundendauer für alle K-Aufträge zur Verfügung steht. Abbildung 3.2 zeigt diese Variante des Schedulings.

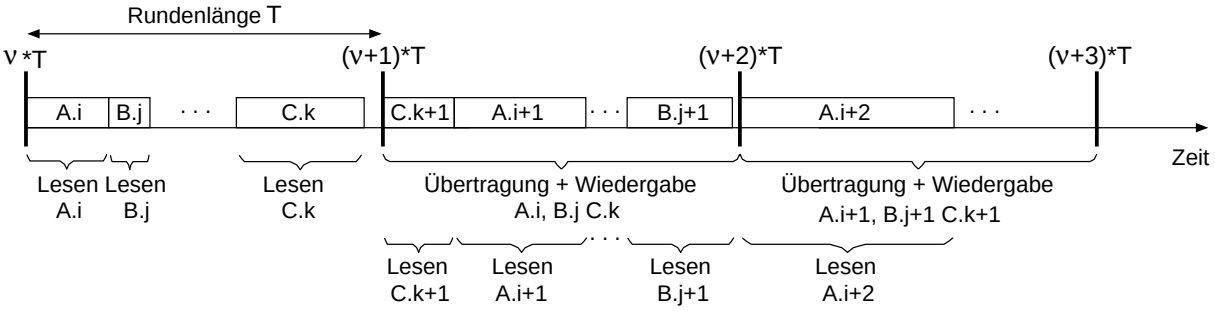



Abbildung 3.2: Scheduling mit variabler Reihenfolge der K-Aufträge

Rundenbeginn und -ende sind durch vertikale Linien wie in Abbildung 3.1 dargestellt.

Die K-Aufträge werden innerhalb der verschiedenen Runden in unterschiedlicher Reihenfolge bearbeitet. Alle Fragmente müssen jeweils am Ende der Runde für die Übertragung aus dem Server-Puffer zur Verfügung stehen, wenn keine Störung in der Wiedergabe eintreten soll. Auch hier wird die Bearbeitung eines K-Auftrages zum Rundenende aufgrund der bereits aufgeführten Argumente abgebrochen. Die Bedingung für eine fristgerechte Bereitstellung der Fragmente liefert mit den Bezeichnern wie in Theorem 3.1.1 das folgende Theorem:

**Theorem 3.1.2.** *Notwendige Bedingung für die fristgerechte Bereitstellung aller Fragmente beim Scheduling mit variabler Reihenfolge: alle $N$ K-Aufträge ($K_1, \ldots, K_N$) müssen innerhalb einer Runde mit der Rundendauer $T$ zeitlich ausführbar sein.*

$$T_{svc}(N) = \sum_{i=1}^{N} T_{svc,K_i} = \sum_{i=1}^{N} T_{pos,K_i} + T_{rot,K_i} + T_{trans,K_i} \leq T$$

Die *Gesamtbedienzeit*, d.h die Summe aller Bedienzeiten in der Runde, ist durch $T_{svc}$ gegeben. Die *Auslastung* ist der Zeitanteil der für die Ausführung der Aufträge benötigt wird. Die Auslastung des Servers ergibt sich gemittelt aus der Auslastung der einzelnen Runden, die durch $T_{svc}(N)/T$ berechnet wird.

Die maximale Startverzögerung bei diesem Schema beträgt $2 * T$, also das Zweifache der Rundendauer, wenn die Einplanung eines neuen Datenstromes nach der Zulassung nur vor Rundenbeginn durchgeführt werden kann. Die maximale Startverzögerung tritt dann ein, wenn die Zulassung kurz nach Rundenbeginn erfolgt und die Bereitstellung des ersten Fragments erst zum Ende der nachfolgenden Runde stattfindet. Das Speichern der Fragmente im Server-Puffer erfolgt der Einfachheit halber gemäß dem *Doppelpufferprinzip* [GVK+95] (siehe auch Abschnitt 1.2.3). Dieses Doppelpufferprinzip ist notwendig, da die Ausführung eines K-Auftrages zu einem Zeitpunkt innerhalb der Runde durchgeführt werden kann, zu dem das vorangegangene Fragment noch nicht vollständig an den Client übertragen wurde. Hierdurch ergibt sich ein maximaler Pufferbedarf auf Seiten des Daten-Servers von $2 * N_{max} * S_K^{max}$ [YCK93, NY94]. $N_{max}$ ist dabei die maximale Anzahl von K-Aufträgen pro Runde, $S_K^{max}$ ist die maximale Fragmentgröße.

#### Scheduling mit mehreren Zugriffsgruppen

Ein hybrider Ansatz, der beide zuvor beschriebenen Schedulingansätze miteinander kombiniert, wurde in [YCK93] unter dem Namen *group sweep scheduling (GSS)* vorgestellt. Hierbei wird die Runde in Slots unterteilt und jedem Slot eine Gruppe bestehend aus mehreren Datenströmen zugeordnet. Es werden die Vorteile einer kürzeren Startverzögerung und eines geringeren Pufferbedarfs, wie sie beim Scheduling mit fester Zugriffsreihenfolge vorhanden sind, mit dem Vorteil einer möglichen Optimierung der Positionierungszeit, den das Scheduling mit variabler Zugriffsreihenfolge aufweist, kombiniert. Die Gruppengröße $G$, d.h. die maximale Anzahl der Datenströme pro Slot, ist in diesem Ansatz ein Parameter, der die Gewichtung zwischen Zugriffsoptimierung und Optimierung des Pufferbedarfs bestimmt. Die beiden zuvor beschriebenen Scheduling-Strategien erhält man als Sonderfälle. Der Fall $G = 1$ entspricht dem Scheduling mit variabler Reihenfolge, der Fall $G = N$ entspricht dem Fall mit fester Reihenfolge der K-Aufträge.

### 3.1.2 Aperiodische Scheduling-Strategien

Für jedes Fragment läßt sich der Zeitpunkt bestimmen, an dem es spätestens im Server-Puffer zur Verfügung stehen muß, damit es trotz einer Netzwerkverzögerung rechtzeitig den Client erreicht. Die Nichteinhaltung dieser Frist verzögert den kontinuierlichen Datenstrom und ruft eine

Störung in der Wiedergabe hervor. Werden KDL-partitionierte kontinuierliche Datenobjekte mit variabler Datenrate verwendet, so ergeben sich für die Fragmente eines einzelnen Datenstroms und damit für die Ausführung der zugehörigen K-Aufträge unregelmäßige aperiodische Fristen. Aperiodische Scheduling-Strategien berücksichtigen bei der Aufstellung der Ausführungsreihenfolge der K-Aufträge diese Fristen. So führt ein EDF-Algorithmus (EDF=earliest deadline first) [SZKT97] den K-Auftrag mit der frühesten Frist als nächstes aus. Hierdurch wird allerdings keine Optimierung des Plattenzugriffs durchgeführt und durch lange Positionierungszeiten pro Fragmentzugriff sinkt die Bedienrate der Platte. Falls die Frist eines K-Auftrages bereits vor dessen Ausführung überschritten ist, wird der Auftrag nicht ausgeführt.

Ein erweiterter SCAN-EDF-Algorithmus [RW94a] wendet einen SCAN-Algorithmus auf Aufträge mit derselben Frist an. Eine Optimierung kann hierbei aber nur dann stattfinden, wenn die Fristen der K-Aufträge, z.B. mit KWL-Fragmenten, synchronisiert werden. Dieses führt allerdings zu einem periodischen Scheduling mit variabler Reihenfolge der K-Aufträge, wie es im vorangegangenen Abschnitt beschrieben worden ist.

### 3.1.3 Scheduling bei mehreren Platten

Die im vorangegangenen Abschnitt beschriebenen Scheduling-Strategien lassen sich auch bei mehreren Platten anwenden. Die Zugriffsmuster auf die Fragmente ergeben sich dabei durch die Plazierung des kontinuierlichen Datenobjektes. Im folgenden wird lediglich das Scheduling mit variabler Reihenfolge der K-Aufträge betrachtet, da hier eine Optimierung des Plattenzugriffs durch einen SCAN-Algorithmus möglich ist. Als Plazierungsgrundlage wird das feinkörnige und das grobkörnige regelmäßige Striping auf mehrere Platten verwendet.

#### Scheduling bei feinkörnigem Striping über mehrere Platten

Das Scheduling mit variabler Reihenfolge unter Verwendung mehrerer Platten erfordert generell, daß die Rundendauer $T$ für alle Platten denselben Wert hat und daß der Rundenbeginn für alle Platten zum selben Zeitpunkt stattfindet. Feinkörniges Striping erfordert den Zugriff auf alle Platten des Disk-Arrays, um innerhalb einer Runde ein Fragment zu lesen. Ein K-Auftrag wird in mehrere Teilaufträge zerlegt, die jeweils von einer Platte ausgeführt werden. In Abbildung 3.3 ist zu sehen, daß die Ausführung der Teilaufträge selbst nicht synchronisiert erfolgen kann. Auf die Teilfragmente F.1.1, F.1.2 sowie F.1.3 wird zu unterschiedlichen Zeitpunkten innerhalb der Runde zugegriffen.

Die maximale Startverzögerung ändert sich durch den Einsatz mehrerer Platten bei diesem Plazierungsverfahren nicht und beträgt unter den bereits getroffenen Annahmen das Zweifache der Rundendauer.

**Theorem 3.1.3.** *Die maximale Startverzögerung beim regelmäßigen feinkörnigen Striping und Scheduling mit variabler Reihenfolge hat einen konstanten Wert von $2 * T$ und ist somit unabhängig von der Anzahl der verwendeten Platten.*

Auf Basis von Theorem 3.1.2 läßt sich sich die Gesamtzahl $N_{tot}^{wc}$ der Datenströme bestimmen, die beim feinkörnigen Striping von einem Disk-Array mit $P_{disks}$ Platten maximal bedient werden können. Der Einfachheit halber soll eine deterministische Service-Garantie gewährleistet

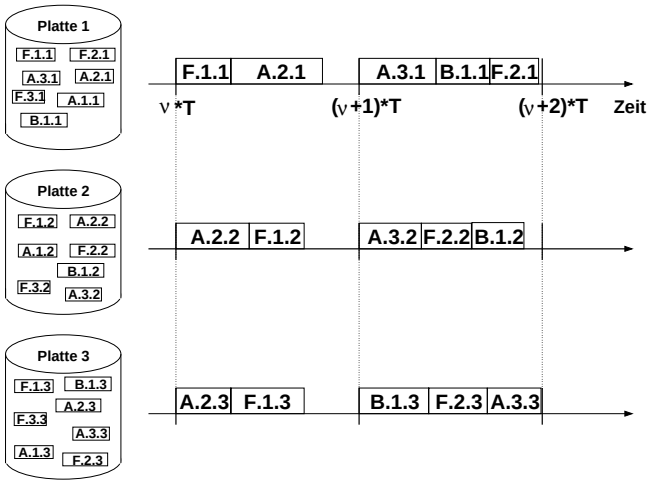



Abbildung 3.3: Scheduling mit variabler Reihenfolge der K-Aufträge unter Verwendung des feinkörnigen Stripings

werden. Des weiteren haben Rotationsverzögerungen sowie Positionierungszeiten für alle Zugriffe einen konstanten Wert, alle Fragmente haben dieselbe konstante Größe und alle Platten besitzen dieselbe konstante Transferrate. Mit $S_K^{max}$ sei im folgenden die maximale Größe eines einzelnen Fragments, mit $P_{rate}^{min}$ die minimale Transferrate jeweils einer Platte und mit $P_{rot}$ bzw. $P_{pos}^{max}$ die maximale Rotationsverzögerung bzw. Positionierungszeit eines einzelnen Fragmentzugriffs bezeichnet. Man erhält (vgl. [ORS96]):

$$
\begin{aligned}
N_{tot}^{wc} &= \max\left\{N \;:\; N * \left(\frac{S_K^{max}}{P_{disks} * P_{rate}^{min}} + P_{rot} + P_{pos}^{max}\right) \leq T\right\} \\
&= \left\lfloor \frac{T}{\frac{S_K^{max}}{P_{disks} * P_{rate}^{min}} + P_{rot} + P_{pos}^{max}} \right\rfloor
\end{aligned}
$$

Obige Gleichung zeigt, daß bei gleichzeitigem Zugriff auf alle Platten zwar die Transferzeit linear mit der Anzahl der Platten abnimmt, Rotationsverzögerung und Positionierungszeit aber unverändert bestehen bleiben. Der Grenzwert für $P_{disks} \rightarrow \infty$ ist somit $T/(P_{rot} + P_{pos}^{max})$. Da dieses Verhalten auch bei einer stochastischen Betrachtung der Fragmentgrößen, Transferraten, Rotationsverzögerungen und Positionierungszeiten gilt, läßt sich hinsichtlich der Skalierbarkeit folgendes ableiten.

**Theorem 3.1.4.** *Die Anzahl der Datenströme, die von einem Disk-Array unterstützt werden, auf dem die kontinuierlichen Datenobjekte mit feinkörnigem Striping plaziert werden, wächst nicht linear mit der Anzahl der Magnetplatten. Diese Plazierung der Datenobjekte ist hinsichtlich des Durchsatzes folglich nicht linear skalierbar.*

### Scheduling bei grobkörnigem Striping über mehrere Platten

Durch das grobkörnige Striping erfolgt, da keine Zerlegung der K-Aufträge stattfindet, pro Fragment und Runde nur ein einziger Plattenzugriff. Pro Runde werden von einer Platte mehrere

K-Aufträge ausgeführt. Die Platten arbeiten bis auf die Synchronisation des Rundenbeginns unabhängig voneinander.

Das Datenzugriffsmuster wird in Abbildung 3.4 sichtbar. Die Fragmente, mit denen über mehrere Runden ein Datenstrom beliefert wird, sind einheitlich schraffiert. Die Datenströme, die in Runde $\nu * T$ von der Platte $i$ bedient wurden, werden in der nachfolgenden Runde $(\nu+1)*T$ von der Platte $(i+1) \bmod P_{disks}$ bedient. $P_{disks}$ ist hierbei die Anzahl der Platten im Disk-Array.

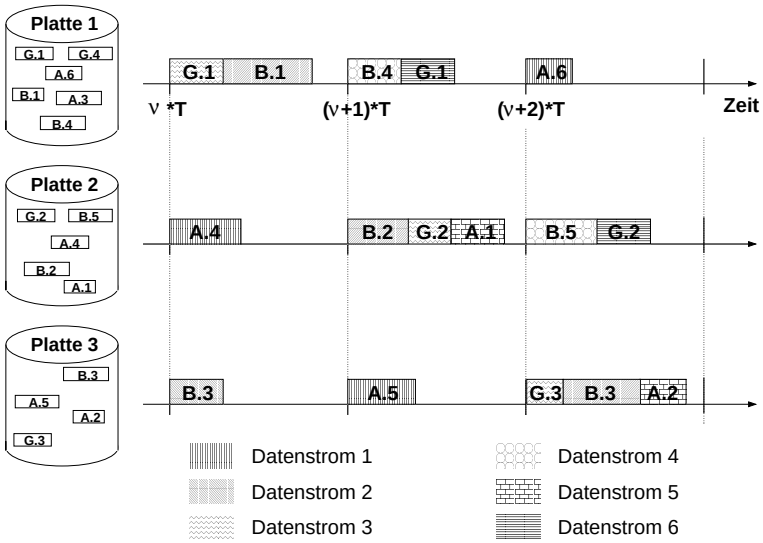



Abbildung 3.4: Scheduling mit variabler Reihenfolge der K-Aufträge unter Verwendung des grobkörnigen Stripings

Die maximale Startverzögerung hat einen Wert von $(P_{disks} + 1) * T$. Vorausgesetzt wird, daß die Einplanung eines neues Datenstromes nur vor Rundenbeginn durchgeführt werden kann. Können insgesamt pro Platte $N_{max}$ Datenströme zugelassen werden, so tritt der Fall mit maximaler Startverzögerung dann ein, wenn bereits $P_{disks} * N_{max} - 1$ Datenströme aktiv sind und die Zulassung zu Beginn der Runde fällt, in der die Platte, die den ersten K-Auftrag ausführen soll, lediglich mit $N_{max} - 1$ K-Aufträgen beschäftigt ist. Es muß $P_{disks}$ Runden gewartet werden, bis diese Platte erneut $N_{max} - 1$ Aufträge ausführt. Zusammen mit der maximalen Verzögerung von einer Runde bis das Fragment im Server-Puffer zur Verfügung steht, ergibt sich der Wert von $(P_{disks} + 1) * T$.

**Theorem 3.1.5.** *Die maximale Startverzögerung beim regelmäßigen grobkörnigen Striping und Scheduling mit variabler Reihenfolge ist linear abhängig von der Anzahl der verwendeten Platten. Bei $P_{disks}$ Platten hat sie den Wert $(P_{disks} + 1) * T$.*

Möchte man die Gesamtzahl $N_{tot}^{wc}$ der Datenströme bestimmen, die beim grobkörnigen Striping von einem Disk-Array mit $P_{disks}$ Platten maximal bedient werden kann, so erhält man unter denselben Annahmen, die bereits für das feinkörnige Striping getroffen wurden, und mit Theorem 3.1.2 folgende Gleichung:

$$
\begin{aligned}
N_{max}^{wc} &= P_{disks} * \max\left\{N \ : \ N * \left(\frac{S_K^{max}}{P_{rate}^{min}} + P_{rot} + P_{pos}^{max}\right) \leq T\right\} \\
&= P_{disks} * \left\lfloor \frac{T}{\frac{S_K^{max}}{P_{rate}^{min}} + P_{rot} + P_{pos}^{max}} \right\rfloor
\end{aligned}
$$

Obige Gleichung zeigt, daß die Anzahl der Datenströme linear mit der Anzahl der Platten wächst. Hieraus folgt das Theorem 3.1.6.

**Theorem 3.1.6.** *Die Anzahl der Datenströme, die von einem Disk-Array unterstützt werden, auf dem die kontinuierlichen Datenobjekte mit grobkörnigem Striping plaziert werden, wächst linear mit der Anzahl der Magnetplatten. Diese Plazierung der Datenobjekte ist hinsichtlich des Durchsatzes linear skalierbar.*

### 3.1.4 Empfehlungen zum Systementwurf

#### Periodische vs. aperiodische Scheduling-Strategien

Das periodische Scheduling erlaubt eine einfache Zulassungskontrolle, da die Anzahl der Plattenzugriffe und damit die von ihnen benötigte Zeit für ein festgelegtes Zeitintervall, d.h. für eine Runde, festgelegt werden kann. Die Einhaltung von Fristen kann auf diese Weise entweder deterministisch oder stochastisch garantiert werden. Beim aperiodischen Scheduling kann zwar die Anzahl der zugelassenen Datenströme kontrolliert werden, hierdurch ist aber nicht sichergestellt, daß zu jedem Zeitpunkt die Ausführung aller K-Aufträge fristgerecht möglich ist. Letzteres ist nur möglich, wenn im voraus die Fristen aller Fragmente, insbesondere bei KDL-Fragmenten, aufwendig berechnet werden und unter Berücksichtigung der Bedienzeit geprüft wird, ob diese Fristen eingehalten werden können.

#### Variable vs. feste Reihenfolge der Bedienung

Das Scheduling mit variabler Bedienungsreihenfolge der K-Aufträge ist gegenüber dem Scheduling mit festgelegter Bedienungsreihenfolge vorzuziehen, da hier eine SCAN-Optimierung der Plattenzugriffe möglich ist. Der höhere Bedarf an Server-Puffer kann vernachlässigt werden, da in Kombination mit weiteren Verfahren z.B. durch Puffer-Sharing [NY94] eine Verringerung des Pufferbedarfs möglich ist.

In heutigen kommerziellen Systemen wird überwiegend Scheduling mit festgelegter Reihenfolge bzw. aperiodisches Scheduling verwendet, was durch fehlende Optimierung der Zugriffe zu einer Verschwendung von Plattenressourcen führt.

#### Scheduling mit feinkörnigem vs. Scheduling mit grobkörnigem Striping

Das grobkörnige Striping bietet durch die lineare Skalierbarkeit hinsichtlich des Durchsatzes Vorteile bei Verwendung großer Disk-Arrays. Der Preis hierfür ist eine potentiell längere Startverzögerung, die mit der Anzahl zugelassener Datenströme steigt und potentiell um so größer

wird, je mehr Platten die maximale Anzahl K-Aufträge pro Runde bearbeiten. Die Varianz für die Wahrscheinlichkeit einer langen Startverzögerung kann insgesamt verringert werden, wenn die Zulassungkontrolle, z.B. durch die vereinzelte Verzögerung der Zulassung, die K-Aufträge gleichmäßig über alle Platten verteilt. Implizit wird dies auch erreicht, wenn das erste Fragment eines kontinuierlichen Datenobjektes zufällig auf eine beliebige Platte plaziert wird und durch eine große Anzahl Benutzer der Zulassungzeitpunkt zufällig liegt.

Das feinkörnige Striping eignet sich besonders für Anwendungen, die durch eine hohe Interaktivität bei der Wiedergabe kontinuierlicher Daten (Start/Stop, Zurück-/Vorspulen) gekennzeichnet sind und hierbei eine deterministische Service-Garantie für die Startverzögerung fordern. Ist hingegen ein hoher Durchsatz bei geringen Kosten gefordert, so ist das grobkörnige Striping vorzuziehen.

#### Wahl der Rundendauer

Die Wahl der Rundendauer bestimmt neben der Startverzögerung auch den Pufferbedarf des Daten-Servers. Je größer die Rundendauer desto größer sind die zu puffernden Fragmente. Die Festsetzung der Rundendauer auf eine Sekunde stellt in der Praxis einen guten Kompromiß zwischen Pufferbedarf und maximaler Startverzögerung dar und wird deshalb im folgenden benutzt.

## 3.2 Scheduling-Strategien für kontinuierliche und diskrete Datenzugriffe

Als Ausgangspunkt und Rahmen für die Bedienung von Zugriffen auf diskrete Datenobjekte, sogenannte *diskrete Datenzugriffe*, dient das in den vorangegangenen Abschnitten beschriebene periodische Scheduling mit variabler Reihenfolge der K-Aufträge. Diese Scheduling-Strategie ist in vielerlei Hinsicht der bessere Ansatz zur Bedienung der kontinuierlichen Datenströme und bietet gleichzeitig mehrere Freiheitsgrade für die Bedienung diskreter Datenzugriffe. Analog zu den kontinuierlichen Datenzugriffen wird der Auftrag zum Lesen eines diskreten Datenobjektes als *D-Auftrag* bezeichnet.

Im folgenden Abschnitt werden zunächst im Rahmen einer Taxonomie verschiedene Teilstrategien erläutert, die die Komponenten einer umfassenden Scheduling-Strategie bilden (siehe auch [NMP+98, NMP+99]). Die Kombination dieser Teilstrategien führt theoretisch zu einer Vielzahl verschiedener Scheduling-Strategien, von denen aber nicht alle praktikabel oder effizient sind. Aus diesem Grund wird am Ende dieses Abschnitts eine Auswahl der Scheduling-Strategien getroffen, die im folgenden Kapitel weiter untersucht werden. Die Darstellung erfolgt am Beispiel *einer* Platte. Sollten mehrere Platten zur Datenplazierung verwendet werden, so können unter Benutzung des grobkörnigen Stripings über mehrere Platten (siehe Abschnitt 3.1.3) die Ergebnisse direkt auf ein Disk-Array übertragen werden.

### 3.2.1 Komponenten einer Scheduling-Strategie

Das Scheduling und die gemeinsame Ausführung von K- und D-Aufträgen läßt sich anhand der folgenden fünf Teilstrategien beschreiben.

1. Einteilungsstrategie
   Die *Einteilungsstrategie* legt fest, ob die K- und D-Aufträge gemeinsam oder in getrennten *Bedienabschnitten* innerhalb der Runde oder Subrunde (siehe 2.) ausgeführt werden.

2. Subrundenstrategie
   Mit der *Subrundenstrategie* wird festgelegt, ob und in wie viele *Subrunden* eine Runde weiter unterteilt wird.

3. Begrenzungsstrategie
   Da die Rundendauer als feste Größe vorgegeben ist, kann nur eine beschränkte Anzahl von K- und D-Aufträgen pro (Sub)Runde bearbeitet werden. Diese kann *anzahllimitiert* oder durch eine obere Zeitschranke *zeitlimitiert* werden und sich auf die gesamte (Sub)Runde oder auch nur auf einen Bedienabschnitt beziehen.

4. Auswahlstrategie
   Innerhalb jeder (Sub)Runde bzw. innerhalb jedes Bedienabschnitts muß eine Auswahl der auszuführenden Aufträge getroffen werden. Diese Auswahl wird durch die *Auswahlstrategie* festgelegt.

5. Anordnungsstrategie
   Die *Anordnungsstrategie* legt die Bearbeitungsreihenfolge fest, in der die Aufträge innerhalb des Bedienabschnitts ausgeführt werden.

Im folgenden sollen die obigen fünf Dimensionen zur Beschreibung des Schedulings näher präzisiert werden.

#### Einteilungsstrategie

Die Ausführung der K- und D-Aufträge kann entweder *getrennt* oder *gemischt* erfolgen. Die gemischte Ausführung nimmt keine Unterscheidung zwischen den beiden Auftragstypen vor. Bei der getrennten Einteilungsstrategie erfolgt die Ausführung der K-Aufträge und der D-Aufträge separat in voneinander getrennten Bedienabschnitten. Im *K-Abschnitt* werden ausschließlich K-Aufträge bearbeitet und im *D-Abschnitt* werden ausschließlich D-Aufträge ausgeführt. Durch eine solche Zweiteilung der Runde in einen K-Abschnitt und einen D-Abschnitt wechseln sich beide Abschnitte über mehrere Runden gesehen gegenseitig ab. Abhängig davon, welchem Auftragstyp eine höhere Priorität zugewiesen wird, erfolgt die Festlegung, welcher Abschnitt in der Runde als erster beginnt. Liegt der K-Abschnitt an erster Stelle, so besteht die Möglichkeit, die Dauer dieses Abschnitts dynamisch zu vergrößern, falls nicht alle K-Aufträge innerhalb des Zeitrahmens ausgeführt werden können. Hierdurch kann die Häufigkeit von Störungen verringert werden. Liegt dagegen der K-Abschnitt an zweiter Stelle, so ist dessen Ende durch das Rundenende fest vorgegeben.

Abbildung 3.5 illustriert die getrennte Einteilungsstrategie, in der eine Runde in einen K- und D-Abschnitt eingeteilt wird und der K-Abschnitt dem D-Abschnitt vorausgeht. Der Ausführungszeitpunkt und die Ausführungsdauer von K-Aufträgen sind durch dunkel hinterlegte Rechtecke angegeben. Hell hinterlegte Rechtecke beziehen sich auf D-Aufträge. Während jedes K-Abschnitts werden drei Datenströme in unterschiedlicher Reihenfolge in Abhängigkeit von der Plattenposition der jeweils benötigten Daten gemäß dem SCAN-Algorithmus ausgeführt. Durch Pfeile oberhalb der Zeitachse wird der *Ankunftszeitpunkt* von D-Aufträgen gekennzeichnet. Nach der Ausführung gibt ein Pfeil unterhalb der Zeitachse den *Abgangszeitpunkt* an, an dem der Auftrag gelöscht wird. Ankunfts- und Abgangszeitpunkt sind mit den Nummern der D-Aufträge gekennzeichnet. Die Zeitspanne zwischen Ankunfts- und Abgangszeitpunkt ist die *Antwortzeit* des Auftrages. Die Abbildung zeigt einige Fälle, in denen mit der Ausführung eines D-Auftrages sofort nach dem Eintreffen im D-Abschnitt begonnen werden kann (Auftrag 1 und 6). Andere Aufträge, die während des K-Abschnitts eintreffen, werden in den nachfolgenden D-Abschnitt (Aufträge 4 und 5) bzw. in die nächste Runde (Aufträge 9 und 10) verzögert. Der größte Nachteil dieser Einteilungsstrategie mit zwei Bedienabschnitten ist durch die Verzögerung der D-Aufträge gegeben, die innerhalb des K-Abschnitts ankommen. Der ungünstigste Fall tritt dann auf, wenn der Ankunftszeitpunkt eines Auftrages am Anfang des K-Abschnitts liegt (Auftrag 4) und trotz geringer Auslastung der Auftrag bis zum Beginn des nächsten D-Abschnitts warten muß.

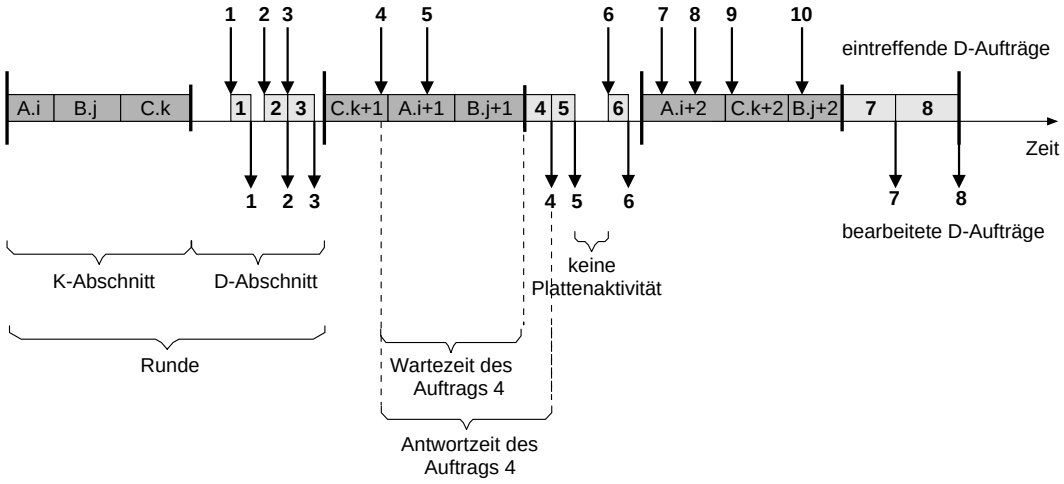



Abbildung 3.5: Einteilungsstrategie mit getrennten Bedienabschnitten

Die Alternative zur getrennten Abarbeitung beider Auftragstypen ist die *gemischte* Ausführung. Hierbei müssen D-Aufträge nicht das Ende eines K-Abschnitts abwarten, sondern können direkt nach Beendigung des zum Ankunftszeitpunkt aktiven Auftrages berücksichtigt werden. Ein möglicher Ausführungsplan ist in Abbildung 3.6 zu sehen. Die Aufträge 7 und 8 werden sofort nach Abarbeitung des ersten K-Auftrages ausgeführt. Hierdurch ist eine weitere Verkürzung der Antwortzeit möglich.

### Subrundenstrategie

Die Subrundenstrategie legt fest, ob und in wie viele Subrunden eine Runde weiter unterteilt wird. Diese Subrunden können gemäß der Einteilungsstrategie in weitere Bedienabschnitte ge-

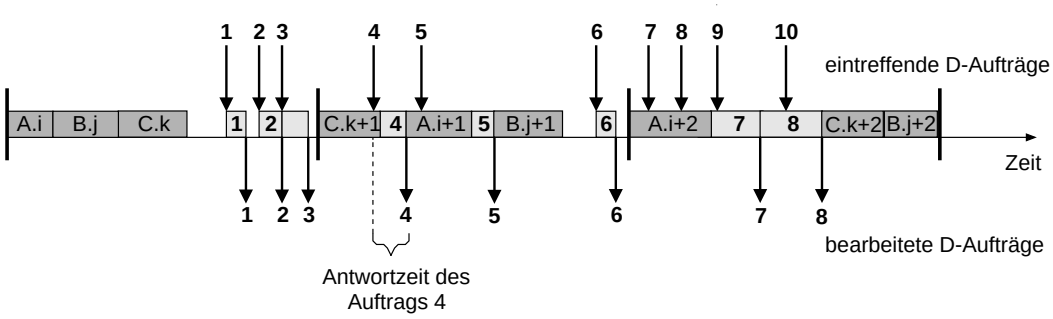



Abbildung 3.6: Einteilungsstrategie mit gemischter Ausführung der Aufträge

teilt werden. Der Vorteil von Subrunden unter Verwendung einer getrennten Einteilungsstrategie ist, daß die Bedienabschnitte kürzer werden und die *Wartezeit*, d.h. der Zeitraum zwischen dem Eintreffen des Auftrages und dessen Ausführung verringert wird. K-Aufträge einer Runde werden gleichmäßig auf die K-Abschnitte der Subrunden verteilt. Die Anzahl der Subrunden ist ein frei wählbarer Parameter, der die Antwortzeit der D-Aufträge bestimmt.

In Abbildung 3.7 ist ein Subrundenmodell mit zwei Subrunden pro Runde und einer getrennten Einteilungsstrategie dargestellt. In diesem abgebildeten Fall müssen die Aufträge 4 und 5 auf ihre Ausführung anstatt bis zum Ende des K-Abschnitts einer Runde, wie in Abbildung 3.5 zu sehen, nur bis zum Ende des K-Abschnitts der ersten Subrunde warten. Ihre Wartezeit verkürzt sich hierdurch.

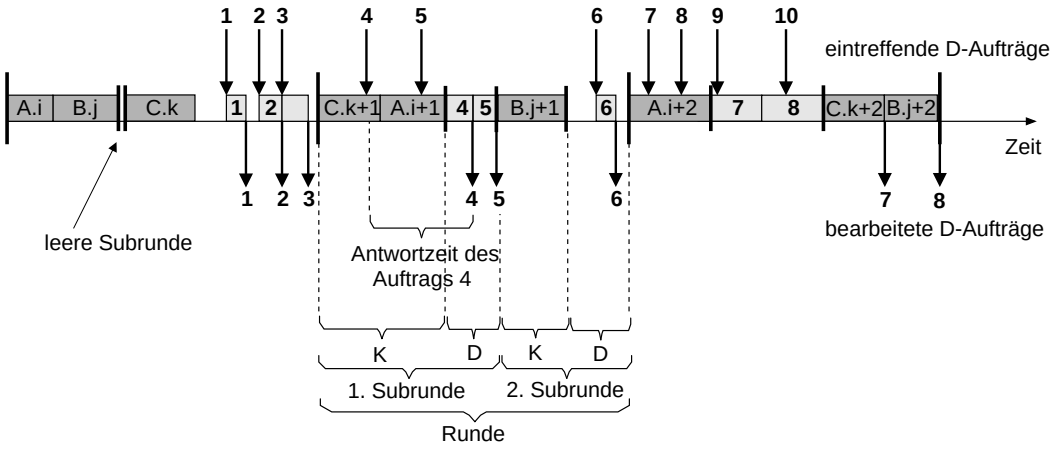



Abbildung 3.7: Subrundenstrategie mit 2 Subrunden pro Runde und getrennter Einteilungsstrategie

Die gemischte Einteilungsstrategie kann als Sonderfall der Subrundenstrategie interpretiert werden, den man erhält, wenn in jeder Subrunde die getrennte Einteilungsstrategie Verwendung findet und die Anzahl der Subrunden der Anzahl der K-Aufträge entspricht. Werden zwei K-Aufträge in der gemischten Einteilungsstrategie direkt nacheinander in den K-Abschnitten aufeinanderfolgender Subrunden ausgeführt, so ist bei dieser Sichtweise der dazwischen liegende D-Abschnitt leer.

**Begrenzungsstrategie**

Aufgrund der festen Länge kann innerhalb einer Runde nur eine begrenzte Anzahl von K- und D-Aufträgen ausgeführt werden. Durch die *Begrenzungsstrategie* wird die maximale Anzahl von Aufträgen bestimmt, die in einer (Sub)Runde bearbeitet werden können. Es lassen sich zwei Begrenzungsstrategien unterscheiden.

Bei der *Anzahllimitierung* wird die maximale Anzahl der pro (Sub)Runde ausgeführten D- und K-Aufträge durch zwei konkrete Werte festgelegt. Beide Werte müssen so gewählt werden, daß die Gesamtbedienzeit die Rundendauer nicht überschreitet. Es gilt analog zu Theorem 3.1.2:

**Theorem 3.2.1.** *Wenn $N$ D-Aufträge ($D_1, \ldots, D_N$) und $i$ K-Aufträge ($K_1, \ldots, K_i$) in einer Runde ausgeführt werden sollen, so muß folgende Ungleichung eingehalten werden, damit die Gesamtbedienzeit die Rundendauer nicht überschreitet:*

$$T_{svc}(N,i) = \sum_{l=1}^{N} T_{pos,K_l} + T_{rot,K_l} + T_{trans,K_l} + \sum_{j=1}^{i} T_{pos,D_j} + T_{rot,D_j} + T_{trans,D_j} \leq T$$

Zum einen soll hierbei die Anzahl der pro Runde ausgeführten K-Aufträge möglichst groß und andererseits die Wartezeit für D-Aufträge hinreichend klein werden. Die geeignete Festlegung des Wertes von $N$ und $i$ kann im voraus mit Hilfe mathematischer Modelle (siehe Kapitel 6 und 7) erfolgen, die die Gesamtbedienzeit, d.h. die Summe aller Bedienzeiten der in der Runde vorhandenen Aufträge, schätzen. Hierdurch wird eine *statische* Festlegung erreicht, die für alle Runden gilt, solange die Lastparameter (z.B. die Ankunftsrate der D-Aufträge) sich nicht verändern. Wird die Gesamtbedienzeit zu Anfang oder während einer Runde anhand aktueller Daten berechnet, z.B. aktuelle Auftragsgrößen und Auftragsposition sowie die Anzahl der K-Aufträge, so läßt sich die maximal mögliche Anzahl bedienbarer D-Aufträge auch *dynamisch* festlegen.

Anstatt die Anzahl der Aufträge zu beschränken, legt die *Zeitlimitierung* die maximal pro Auftragstyp und pro (Sub)Runde bzw. Bedienabschnitt zur Verfügung stehende Zeit fest. Auch hierbei kann die Limitierung statisch oder dynamisch erfolgen. Eine statische Einteilung für eine getrennte Scheduling-Strategie mit einer einzigen Subrunde könnte z.B. bei einer Rundendauer von $1$ Sekunde dem K-Abschnitt maximal $0.7$ Sekunden und dem darauffolgenden D-Abschnitt die verbleibende Zeit, d.h. minimal $0.3$ Sekunden, zuordnen. Hierdurch wird zur Ausführung der D-Aufträge in jeder Runde eine minimal zur Verfügung stehende Zeitspanne garantiert. Die dynamische Zeitlimitierung berechnet hingegen die innerhalb der (Sub)Runde noch für weitere Aufträge zur Verfügung stehende Zeit unter Berücksichtigung der Bedienzeit vorangegangener Aufträge. Hierbei handelt es sich um die duale Sichtweise der dynamischen Anzahllimitierung, da durch beide Verfahren die Anzahl der innerhalb einer (Sub)Runde ausgeführten Aufträge auf dieselbe Weise beschränkt wird. Im folgenden wird deshalb nur noch der Begriff der *dynamischen Begrenzungsstrategie* verwendet.

**Auswahl- und Anordnungsstrategie**

Eingetroffene D-Aufträge werden vor ihrer Ausführung in einer Warteschlange gesammelt. Gemäß der *Auswahlstrategie* werden hieraus zu Beginn und auch während einer (Sub)Runde

Aufträge selektiert. Die maximale Anzahl ist durch die Begrenzungsstrategie festgesetzt. K-Aufträge unterliegen keiner Auswahlstrategie, da die von der Zulassungskontrolle festgelegte Anzahl stets in der jeweiligen Runde ausgeführt werden muß. Um zu vermeiden, daß D-Aufträge niemals aus der Warteschlange entnommen werden und somit theoretisch unendliche Antwortzeiten entstehen können, wird im folgenden davon ausgegangen, daß die Warteschlange nach den Ankunftszeitpunkten der Aufträge sortiert wird und Aufträge gemäß ihrer Ankunftsreihenfolge hieraus entnommen (*FCFS-Prinzip*) werden. Hierdurch wird auf einfache Art und Weise eine faire Ausführung unter den D-Aufträgen garantiert.

Wenn zu Beginn einer (Sub)Runde eine Anzahl von D-Aufträgen aus der Warteschlange ausgewählt worden ist, können diese D-Aufträge für die Ausführung zusammen mit den K-Aufträgen dieser (Sub)Runde in eine Ausführungsreihenfolge gebracht werden. Die *Anordnungsstrategie* bestimmt deren Ordnung. Es bieten sich zwei Alternativen an [WGP94, SG94a]. Die *FCFS-Anordnungsstrategie* sortiert die Aufträge gemäß ihrem Ankunftszeitpunkt. Motivation für diese Ausführungsreihenfolge ist die faire Ausführung der Aufträge. Die *SCAN-Anordnungsstrategie* sortiert die Aufträge hinsichtlich ihrer Plattenposition relativ zum innersten oder äußersten Zylinder. Hierdurch ist bei der Abarbeitung der Aufträge eine Reduzierung der Positionierungszeiten möglich.

Auswahlstrategien lassen sich weiterhin danach klassifizieren, wann während einer (Sub)Runde eintreffende D-Aufträge in der Ausführungsreihenfolge Berücksichtigung finden, sofern die von der Begrenzungsstrategie festgelegte Obergrenze noch nicht überschritten ist. Eine *nicht inkrementelle* (engl. gated service) Auswahlstrategie legt die Ausführungsreihenfolge nur zu Beginn der (Sub)Runde in einem *Ausführungsplan* fest. Eintreffende D-Aufträge können somit erst zum Beginn der nächsten (Sub)Runde im folgenden Ausführungsplan berücksichtigt werden. Nachteil dieser Strategie ist, daß die Ausführung von Aufträgen immer in die nächste Runde verzögert wird, obwohl die Bearbeitung in manchen Fällen in der derzeitigen (Sub)Runde möglich wäre. Hierdurch erhöht sich die Antwortzeit. Vorteil des nicht inkrementellen Verfahrens ist, daß der Ausführungsplan nur einmal zu Beginn der Runde erstellt werden muß und somit nur geringer Berechnungsaufwand entsteht.

In Abbildung 3.8 ist ein Beispiel unter Verwendung einer nicht inkrementellen Auswahlstrategie in Kombination mit der SCAN-Anordnungsstrategie und gemischter Auftragsabarbeitung gegeben. K-Aufträge sind hierin dunkel, D-Aufträge hell hinterlegt. Zusätzlich ist unterhalb der Auftragsnummer in eckigen Klammern die Auftragsposition (Zylindernummer) angegeben, nach der der SCAN-Algorithmus die Sortierung der Aufträge vornimmt. Zu Beginn der ersten abgebildeten Runde wird der Ausführungsplan für die K-Aufträge A.i und B.j erstellt. Unter der Annahme, daß der Plattenkopf sich auf dem Zylinder $412$ befindet, sortiert der SCAN-Algorithmus die Aufträge nach aufsteigender Zylindernummer. Nach Abarbeitung von B.j wird auf den Beginn der nächsten Runde gewartet. Zu Beginn der zweiten Runde sind die K-Aufträge A.i+1, B.j+1 und die D-Aufträge 1, 2, 3 einzuplanen. Da der Plattenkopf auf Zylinder $3854$ steht, erfolgt in dieser Runde die Ausführung mit absteigender Zylindernummer. Auftrag 3 kann zum einen nicht eingeplant werden, da die Gesamtbedienzeit überschritten wird, wenn mehr als zwei D-Aufträge in dieser Runde bearbeitet werden, und zum anderen die Auswahlstrategie auf dem FCFS-Prinzip basiert. Dies bedeutet, daß die Ausführung von Auftrag 3 anstelle von Auftrag 2 oder 1 nicht vorgezogen werden kann. Die Ausführung kann erst in der dritten Runde erfolgen, in der der Ausführungsplan mit aufsteigender Zylindernummer sortiert wird.

Eine *beschränkt inkrementelle* (engl. incremental gated service) Auswahlstrategie erstellt im-

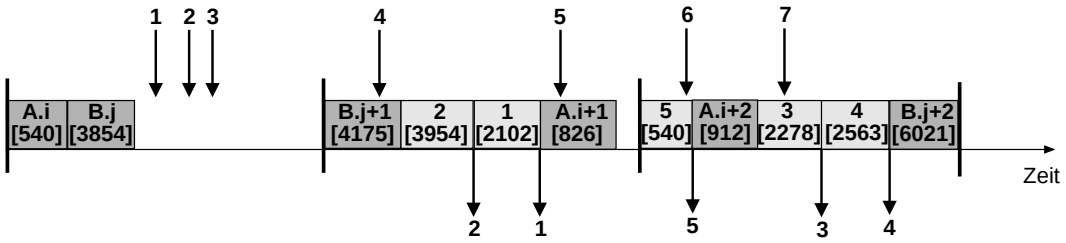



Abbildung 3.8: Nicht inkrementelle Auswahlstrategie in Kombination mit einer SCAN-Anordnungsstrategie und gemischter Ausführung der Aufträge

mer dann einen neuen Ausführungsplan, wenn der zuvor erstellte vollständig abgearbeitet wurde und die Begrenzungsstrategie die Ausführung weiterer D-Aufträge erlaubt. Dies bedeutet, daß D-Aufträge innerhalb der Runde, in der sie eintreffen, ausgeführt werden können. In Abbildung 3.9 ist die Vorgehensweise dieser Strategie illustriert. Zu Beginn der ersten Runde wird der Ausführungsplan für die K-Aufträge A.i und B.j erstellt, der bei Eintreffen des D-Auftrages 1 bereits abgearbeitet ist. Da die Runde noch nicht beendet ist, wird folglich ein neuer Ausführungsplan mit dem einzig zur Verfügung stehenden D-Auftrag 1 erstellt. Ist dieser ausgeführt, wird der dritte Ausführungsplan innerhalb der ersten Runde erstellt, da das Rundeende noch nicht erreicht ist. Aufgrund der zeitlichen Beschränkung kann hierbei nur Auftrag 2 Berücksichtigung finden. Auftrag 3 wird in die nächste Runde verschoben. Der Ausführungsplan zu Beginn dieser nächsten Runde enthält die Aufträge A.i+1, B.j+1 und 3, die nach absteigender Zylindersortierung ausgeführt werden. Da auch in dieser Runde das Rundenende noch nicht erreicht ist, ist die Ausführung von Auftrag 4 möglich. Vergleichbares gilt in der dritten Runde für den Auftrag 6.

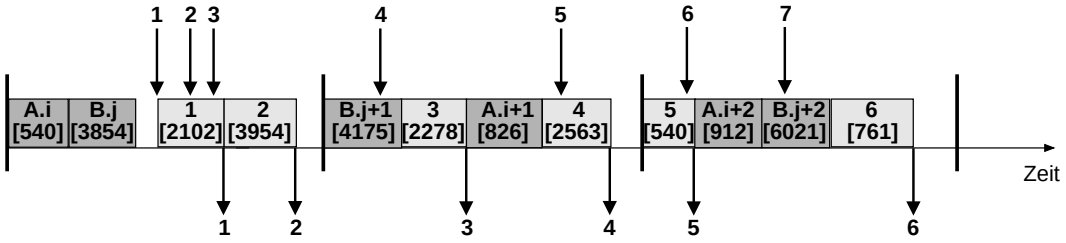



Abbildung 3.9: Beschränkt inkrementelle Auswahlstrategie in Kombination mit einer SCAN-Anordnungsstrategie und gemischter Ausführung der Aufträge

Eine *voll inkrementelle* Auswahlstrategie kann D-Aufträge in bereits bestehende Ausführungspläne gemäß der Anordnungsstrategie einfügen, solange die von der Begrenzungsstrategie festgelegte Obergrenze noch nicht erreicht worden ist. Dieses führt häufig zu einer veränderten Ausführungsreihenfolge der Aufträge. Bei einer voll inkrementellen Auswahlstrategie in Kombination mit einer SCAN-Anordnungsstrategie wird bei leerer Warteschlange ein eintreffender D-Auftrag sofort in den gerade aktuellen Ausführungsplan eingefügt, wenn dessen Position sich vor der aktuellen Position des Plattenkopfes befindet. Ansonsten wird der Auftrag in die Warteschlange eingefügt und seine Ausführung auf den nächsten SCAN innerhalb derselben (Sub)Runde oder der nächsten (Sub)Runde verschoben. Sollten alle D- und K-Aufträge des Ausführungsplanes, inklusive der inkrementell eingefügten, bearbeitet worden sein und sollte die Beschränkungsstrategie die Ausführung weiterer D-Aufträge erlauben, so kann ein weiterer Plan mit Aufträgen aus der Warteschlange, wie zu (Sub)Rundenbeginn, erstellt werden.

Eine beschränkt inkrementelle Auswahlstrategie mit FCFS-Anordnungsstrategie zeigt im Vergleich zu einer voll inkrementellen Auswahlstrategie mit FCFS-Anordnungsstrategie keine veränderte Ausführungsreihenfolge der D-Aufträge, da diese stets in der Ankunftsreihenfolge abgearbeitet werden. Abbildung 3.10 zeigt ein Beispielszenario, an dem die Vorteile einer voll inkrementellen Auswahlstrategie deutlich werden. In der zweiten Runde wird, anders als bei einer beschränkt inkrementellen Auswahlstrategie mit SCAN-Anordnungsstrategie, der Auftrag 4 in den Ausführungsplan aufgenommen. Dies geschieht in der dritten Runde ebenfalls für Auftrag 6. Hierdurch wird beim Übergang von Auftrag 5 zu Auftrag 6 weniger Zeit für die Positionierung benötigt ($|761 - 540| = 221$ Zylinder) als beim beschränkt inkrementellen Verfahren von K-Auftrag B.j+2 zu Auftrag 6 ($|6021 - 761| = 5260$ Zylinder) in Abbildung 3.9. Durch diese in der dritten Runde hinzugewonnende Zeit ist es möglich, den Auftrag 7 zusätzlich in derselben Runde auszuführen.

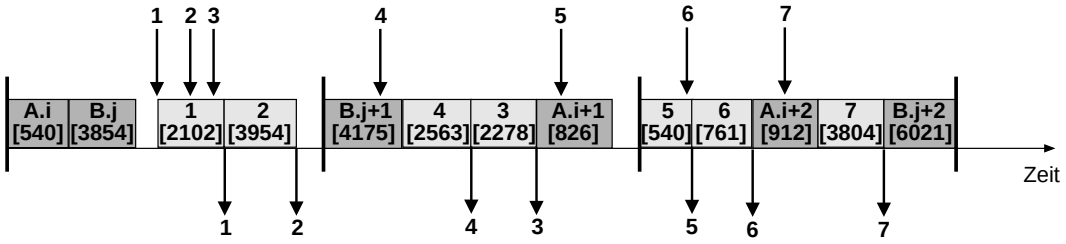



Abbildung 3.10: Voll inkrementelle Auswahlstrategie mit SCAN-Anordnungsstrategie und gemischter Ausführung der Aufträge

### 3.2.2 Bewertung und Auswahl geeigneter Scheduling-Strategien

Anhand der im vorangegangenen Kapitel aufgeführten Teilstrategien lassen sich mit fünf Parametern verschiedene Scheduling-Strategien für die gemeinsame Ausführung von K- und D-Aufträgen beschreiben. Die fünf Parameter sind:

1. Festlegung der Einteilungsstrategie. Mögliche Alternativen sind:
   a) getrennte Ausführung mit vorangehendem K-Abschnitt und nachfolgendem D-Abschnitt
   b) getrennte Ausführung mit vorangehendem D-Abschnitt und nachfolgendem K-Abschnitt
   c) gemischte Ausführung
2. Anzahl der Subrunden.
3. Festlegung der Begrenzungsstrategie. Mögliche Alternativen sind:
   a) statische Anzahllimitierung
   b) statische Zeitlimitierung
   c) dynamische Begrenzung

4. Festlegung der Auswahlstrategie
   - FCFS nicht inkrementell
   - FCFS beschränkt inkrementell
   - FCFS voll inkrementell
5. Festlegung der Anordnungsstrategie. Mögliche Alternativen sind:
   a) SCAN-Anordnung für K-Aufträge und FCFS-Anordnung für D-Aufträge
   b) SCAN-Anordnung für K-Aufträge und SCAN-Anordnung für D-Aufträge

Nicht alle Varianten, die sich hieraus kombinatorisch ermitteln lassen, sind sinnvoll. Es lassen sich unmittelbar einige besonders vorteilhafte Kombinationen erkennen, so daß die Menge der geeigneten Scheduling-Strategien im Vorfeld bereits eingegrenzt werden kann. Diese sollen mit Hilfe einer experimentellen Evaluation in Kapitel 4 hinsichtlich ihrer Leistungsfähigkeit untersucht werden.

**Bewertung der Einteilungsstrategien: Vorteile durch vorangehenden K-Abschnitt / durch gemischte Ausführung**

Favorisiert man K-Aufträge gegenüber den D-Aufträgen, so erscheint es sinnvoll bei einer getrennten Einteilungsstrategie K-Aufträge innerhalb der Runde als erste auszuführen. Somit können zu Beginn einer Runde zunächst alle K-Aufträge bearbeitet und die verbleibende Zeit für D-Aufträge genutzt werden. Hierdurch erhöht sich die Wahrscheinlichkeit, daß alle K-Aufträge in der für sie vorgesehenen Runde ausgeführt werden können.

Durch eine gemischte Ausführung besteht bei Verwendung einer SCAN-Anordnung zusätzliches Optimierungspotential, da die Optimierung des Plattenzugriffs über alle Aufträge der Runde durchgeführt werden kann.

**Bewertung der Begrenzungsstrategien: Vorteile durch dynamische Begrenzung**

Weiterhin läßt sich feststellen, daß eine dynamische Begrenzungsstrategie für D-Aufträge gegenüber einer statischen Strategie mit Anzahl- oder Zeitlimitierung von Vorteil ist, da die dynamische Strategie die aktuelle Lastsituation innerhalb einer Runde bewerten kann. Sie kann hierdurch die maximal zulässige Anzahl von D-Aufträgen für die aktuelle Runde ausführen. Bei den statischen Strategien hingegen wird einerseits die Rundendauer nicht in jeder Runde vollständig zur Ausführung von Aufträgen genutzt und es kann andererseits nicht ausgeschlossen werden, daß die Gesamtbedienzeit der Aufträge die Rundendauer überschreitet.

Ein Nachteil der dynamischen Begrenzung ist, daß im voraus die Gesamtbedienzeit berechnet werden muß, obwohl einige Parameter, wie z.B. die Rotationsverzögerung, im voraus nicht bekannt sind. In diesen Fällen muß mit einer Abschätzung gearbeitet werden.

**Bewertung der Auswahlstrategien: Vorteile durch inkrementelle Auswahl**

Für die Auswahlstrategie kann festgestellt werden, daß die inkrementellen Verfahren Vorteile gegenüber den nicht inkrementellen aufweisen, da bei geringer Plattenauslastung eintreffende D-Aufträge im günstigsten Fall sofort ausgeführt werden können. Weiterhin wird eine voll inkrementelle Auswahlstrategie in Kombination mit einer SCAN-Anordnungsstrategie einer beschränkt inkrementellen Auswahlstrategie in Kombination mit einer SCAN-Anordnungsstrategie überlegen sein, da durch die Möglichkeit des Einfügens von D-Aufträgen in einen bereits erstellten Ausführungsplan der Anteil der Positionierungszeiten pro Auftrag weiter reduziert werden kann.

Der Nachteil einer inkrementellen Auswahlstrategie ist ein erhöhter Verwaltungsaufwand. Dieser ist zwingend notwendig, damit entschieden werden kann, ob das Einfügen eines Auftrages hinsichtlich der Plattenposition und hinsichtlich der Begrenzungsstrategie während der aktuellen Runde möglich ist.

**Bewertung der Anordnungsstrategien: Vorteile durch SCAN-Anordnung**

Generell kann im voraus festgestellt werden, daß bei Verwendung der SCAN-Anordnung im Vergleich zur FCFS-Anordnung ein höherer Durchsatz durch die Optimierung des Plattenzugriffs erzielt werden kann. Da für K-Aufträge die Ausführungsreihenfolge keine Bedeutung hat, solange alle K-Aufträge innerhalb der Runde ausgeführt werden können, sollte hier generell die SCAN-Anordnung verwendet werden. Die Verwendung einer FCFS-Anordnungsstrategie ist nur für D-Aufträge potentiell relevant, da eventuell eine Verkürzung der Antwortzeit möglich ist.

**Wechselseitige Abhängigkeiten**

Ein weiteres Ausschlußkriterium bestimmter Strategien sind wechselseitige Abhängigkeiten. Werden z.B. verschiedene Anordnungsstrategien für D- und K-Aufträge gewählt, so ist eine gemischte Einteilungsstrategie nicht möglich, da mindestens zwei getrennte Bedienabschnitte benötigt werden.

**Auswahl der geeignetsten Algorithmen**

Für eine nähere algorithmische Betrachtung und eine experimentelle Evaluation in den nachfolgenden Abschnitten werden folgende Scheduling-Strategien ausgewählt:

1. *getrennte, dynamische, voll inkrementelle FCFS-Strategie*:
   getrennte Bedienabschnitte mit vorangehendem K-Abschnitt, in der die Aufträge gemäß SCAN-Anordnung ausgeführt werden; im D-Abschnitt erfolgt die Ausführung gemäß FCFS-Anordnung; dynamische Begrenzung der D-Auftragsanzahl sowie voll inkrementelle Auswahl innerhalb des D-Abschnitts.

2. *getrennte, dynamische, voll inkrementelle SCAN-Strategie*:
   getrennte Bedienabschnitte mit vorangehendem K-Abschnitt; Aufträge werden im K- und D-Abschnitt jeweils gemäß SCAN-Anordnung ausgeführt; dynamische Begrenzung der D-Auftragsanzahl sowie voll inkrementelle Auswahl innerhalb des D-Abschnitts.

3. *gemischte, dynamische, nicht inkrementelle SCAN-Strategie*:
   gemischte Ausführung beider Auftragstypen gemäß SCAN-Anordnung; dynamische Begrenzung der D-Auftragsanzahl sowie nicht inkrementelle Auswahl der D-Aufträge zu Rundenbeginn.

4. *gemischte, dynamische, beschränkt inkrementelle SCAN-Strategie*:
   gemischte Ausführung beider Auftragstypen gemäß SCAN-Anordnung; dynamische Begrenzung der D-Auftragsanzahl sowie beschränkt inkrementelle Auswahl der D-Aufträge.

5. *gemischte, dynamische, voll inkrementelle SCAN-Strategie*:
   gemischte Ausführung beider Auftragstypen gemäß SCAN-Anordnung; dynamische Begrenzung der D-Auftragsanzahl sowie beschränkt inkrementelle Auswahl der D-Aufträge.

Innerhalb der Gruppe der gemischten Strategien steigt von Strategie 3 über Strategie 4 bis zu Strategie 5 der Aufwand für die Erstellung und Aktualisierung des Ausführungsplanes. In der Evaluation ist deshalb die Fragestellung interessant, welche dieser drei Strategien mit akzeptablem Aufwand eine wesentliche Verbesserung der Performance (Antwortzeit, Durchsatz) ermöglicht.

## 3.3 Scheduling-Algorithmen für die Verarbeitung von gemischten Arbeitslasten

### 3.3.1 Algorithmen und Datenstrukturen

In diesem Abschnitt werden die im vorangegangenen Abschnitt ausgewählten Strategien algorithmisch mit den dazugehörigen Datenstrukturen näher präzisiert. Alle Algorithmen benutzen die folgenden beiden Datenstrukturen:

- eine FIFO-Warteschlange (*fifoPuffer*), die alle eintreffenden D-Aufträge aufnimmt und der Aufträge gemäß ihrer Eingangsreihenfolge entnommen werden können,

- eine sortierte SCAN-Liste (*scanListe*), die Aufträge entsprechend der SCAN-Strategie geordnet nach aufsteigender Zylindernummer enthält und aus der die Aufträge für die Ausführung entnommen werden können.

Mit diesen beiden Datenstrukturen lassen sich die Algorithmen wie folgt beschreiben.

**Algorithmus I : getrennte, dynamische, voll inkrementelle FCFS-Strategie**

Zu Beginn einer neuen Runde wird die SCAN-Liste (*scanListe*) mit den K-Aufträgen gefüllt, die in dieser Runde ausgeführt werden müssen. Der Algorithmus bestimmt die SCAN-Richtung, indem er die Position des ersten und des letzten in der SCAN-Liste stehenden Auftrages mit der aktuellen Position des Plattenkopfes vergleicht und die Position des Auftrages anfährt, die am nächsten liegt. Bei einem *Vorwärts-SCAN* liegt die Position des ersten Auftrages in der SCAN-Liste näher, bei einem *Rückwärts-SCAN* die Position des letzten Auftrages. Im weiteren Verlauf wird dann die SCAN-Liste von vorne bzw. hinten beginnend sequentiell abgearbeitet und die ausgeführten Aufträge werden aus der Liste entfernt. Falls die Ausführung eines Auftrages das Ende einer Runde überschreitet, so werden die noch in der Liste vorhandenen Aufträge nicht mehr ausgeführt und aus der SCAN-Liste entfernt. Wenn die Liste komplett abgearbeitet wurde und das Ende der Runde noch nicht erreicht ist, startet die Ausführung von D-Aufträgen. Diese werden der FIFO-Warteschlange (*fifoPuffer*) entnommen und ausgeführt bis die aktuelle Runde beendet ist. Ist die FIFO-Warteschlange leer, so wird auf das Eintreffen von D-Aufträgen bis zum Rundenende gewartet. Da ein in Ausführung befindlicher D-Auftrag zu Rundenbeginn nicht unterbrochen wird, kann es passieren, daß die Bearbeitung eines D-Auftrages sich in die nächste Runde erstreckt.

**Algorithmus II: getrennte, dynamische, voll inkrementelle SCAN-Strategie**

Die Ausführung der K-Aufträge erfolgt wie in Algorithmus I beschrieben. Ist die SCAN-Liste mit K-Aufträgen vollständig abgearbeitet und das Ende der Runde noch nicht erreicht, so wird die SCAN-Liste mit D-Aufträgen aus der Warteschlange so lange gefüllt, bis die vorab berechnete *Restbedienzeit*, d.h. die voraussichtlich benötigte Bedienzeit für alle in der Liste enthaltenen D-Aufträge, die verbleibende Rundendauer überschreitet oder die Warteschlange vollständig geleert wurde. Damit eine Runde vollständig ausgenutzt werden kann, ist es erlaubt, daß sich die Ausführung des letzten D-Auftrages in die nächste Runde erstreckt. Die SCAN-Richtung wird wie in Algorithmus I bestimmt. Danach kann die Ausführung der D-Aufträge beginnen. Vor der Entnahme eines Auftrages aus der SCAN-Liste wird versucht ein oder mehrere D-Aufträge aus der FIFO-Warteschlange in die SCAN-Liste einzufügen. Hierzu wird so lange ein D-Auftrag aus der Warteschlange entnommen und in die Liste eingefügt, bis die berechnete Restbedienzeit die verbleibende Rundendauer überschreitet oder die Warteschlange leer wird oder kein D-Auftrag in die Liste eingefügt werden kann. Voraussetzung für das Einfügen ist, daß die Position eines einzufügenden Auftrages in SCAN-Richtung vor der aktuellen Position des Plattenkopfes liegt. Ist die SCAN-Liste abgearbeitet und die Warteschlange nicht leer und hat die nächste Runde noch nicht begonnen, so wird das initiale Auffüllen der SCAN-Liste erneut durchgeführt. Ist die Warteschlange leer, so wird auf das Eintreffen neuer D-Aufträge gewartet, das das initiale Auffüllen erneut startet, oder es wird auf das Rundenende gewartet, das die Ausführung der K-Aufträge, wie in Algorithmus I beschrieben, erneut initiiert.

**Algorithmus III: gemischte, dynamische, nicht inkrementelle SCAN-Strategie**

Zu Beginn einer neuen Runde werden zunächst alle K-Aufträge, die für diese Runde vorgesehen sind, in die SCAN-Liste eingefügt. Daraufhin werden so lange D-Aufträge aus der Warteschlange entfernt und in die SCAN-Liste eingefügt, bis die berechnete Gesamtbedienzeit aller

Aufträge in der SCAN-Liste die verbleibende Rundendauer überschreitet oder die Warteschlange vollständig geleert ist. Hierbei ist es wie in Algorithmus II erlaubt, daß die Ausführung des letzten Auftrages, sofern es sich um einen D-Auftrag handelt, sich zeitlich in die nächste Runde erstreckt. Analog zu Algorithmus I erfolgt die Bestimmung der SCAN-Richtung und die sequentielle Ausführung aller Aufträge in der SCAN-Liste. Aufgrund der nicht inkrementellen Auswahlstrategie wird die SCAN-Liste nur einmal zu Beginn einer Runde mit Aufträgen gefüllt.

**Algorithmus IV: gemischte, dynamische, beschränkt inkrementelle SCAN-Strategie**

Dieser Algorithmus verhält sich zu Rundenbeginn wie in Algorithmus III beschrieben, indem er die SCAN-Liste zunächst mit K- und D-Aufträgen auffüllt, die SCAN-Richtung bestimmt und die Bearbeitung der SCAN-Liste beginnt. Im weiteren Ablauf unterscheidet er sich von Algorithmus III insofern, daß, wenn die SCAN-Liste vollständig abgearbeitet wurde und das Ende der Runde noch nicht erreicht ist, jedesmal versucht wird, wie zu Rundenbeginn, die SCAN-Liste erneut mit Aufträgen aus der Warteschlange aufzufüllen. Ist die Warteschlange vor Rundenende leer, so wird entweder gewartet bis ein neuer D-Auftrag eintrifft oder bis eine neue Runde beginnt.

**Algorithmus V: gemischte, dynamische, voll inkrementelle SCAN-Strategie**

Zu Rundenbeginn wird die SCAN-Liste, wie in Algorithmus III beschrieben, mit K- und D-Aufträgen gefüllt, die SCAN-Richtung bestimmt und die Bearbeitung der SCAN-Liste begonnen. Wie schon zuvor in Algorithmus II wird bei den voll inkrementellen Algorithmen versucht, vor jeder Entnahme eines Auftrages aus der SCAN-Liste einen oder mehrere D-Aufträge aus der Warteschlange in die SCAN-Liste einzufügen. Es wird so lange ein D-Auftrag aus der Warteschlange entnommen und in die SCAN-Liste eingefügt, bis die berechnete Restbedienzeit die verbleibende Rundendauer überschreitet. Im Gegensatz zu einer getrennten Einteilungsstrategie muß hierbei berücksichtigt werden, daß kein K-Auftrag am Ende bzw. am Anfang der SCAN-Liste auftritt, wenn die Restbedienzeit die verbleibende Rundendauer überschreitet, da in diesem Fall der K-Auftrag unter Umständen nicht mehr rechtzeitig vor Rundenende ausgeführt werden kann. Das Einfügen in die SCAN-Liste wird ebenfalls unterbrochen, wenn die Warteschlange leer wird oder kein D-Auftrag mehr eingefügt werden kann. Dies ist aufgrund der FCFS-Auswahl dann der Fall, wenn die Position des ersten D-Auftrages in der Warteschlange in SCAN-Richtung gesehen bereits hinter der aktuellen Position des Plattenkopfes der Platte liegt. Ist die SCAN-Liste abgearbeitet und die Warteschlange nicht leer und hat die nächste Runde noch nicht begonnen, so wird das Auffüllen der SCAN-Liste ausschließlich mit D-Aufträgen erneut durchgeführt. Hierbei können die neu angekommenen D-Aufträge Berücksichtigung finden, deren Position zum Ankunftszeitpunkt hinter der des Plattenkopfes lag. Ist die Warteschlange leer, so wird auf das Eintreffen neuer D-Aufträge gewartet, und das Auffüllen mit D-Aufträgen startet erneut. Wird das Rundenende vorzeitig erreicht, so beginnt das initiale Auffüllen mit K- und D-Aufträgen wieder von vorne.

### 3.3.2 Pseudocode

Alle fünf Algorithmen lassen sich zur Evaluation in einer einzigen Prozedur *bearbeiteAufträge* kombinieren. Der dazugehörige Pseudocode ist in Abbildung 3.11 aufgeführt. Die Prozedur definiert in Zeile 3 die globalen Variablen, die von den übrigen Prozeduren ebenfalls verwendet werden. Innerhalb einer Endlosschleife, die sich von Zeile 4 bis Zeile 51 erstreckt, erfolgt zunächst im Falle einer getrennten Einteilungsstrategie in den Zeilen 7 bis 14 die Ausführung der K-Aufträge. Bei den gemischten Strategien erfolgt die Ausführung aller Aufträge in den Zeilen 18 bis 50. Die Abfrage nach der jeweiligen Auswahl- und Anordnungsstrategie erfolgt in den Zeilen 21, 27, 36, 44 und 7. Immer, wenn sich die SCAN-Richtung ändern kann, wird die Variable *scanRichtung* auf den Wert *undefiniert* gesetzt. Dies ist dann der Fall, wenn bei einer getrennten Strategie ein neuer Bedienabschnitt beginnt oder wenn bei einer gemischten Strategie die *scanListe* leer wird.

Die tatsächliche Ausführung der Aufträge erfolgt in der Prozedur *bearbeiteAuftrag* (siehe Abbildung 3.12), die entsprechend der SCAN-Richtung genau einen Auftrag aus *scanListe* entfernt und ausführt.

Die Prozedur *fuelleScanListe* dient dem Auffüllen der SCAN-Liste und wird von der Prozedur *bearbeiteAufträge* aufgerufen. Sie ist in Abbildung 3.13 skizziert. Sie liest in Zeile 9 den jeweiligen D-Auftrag in die Variable *auftrag*, der am längsten in *fifoPuffer* steht. Er wird in die SCAN-Liste (Zeile 13) eingefügt. Danach erfolgt die Überprüfung (Zeile 17), ob alle K-Aufträge trotz dieses zusätzlichen D-Auftrages vollständig innerhalb der Runde ausgeführt werden können. Falls ja, wird der D-Auftrag endgültig (Zeile 24) aus dem *fifoPuffer* entfernt. Falls nein, muß der Auftrag aus der *scanListe* gelöscht werden (Zeile 21) und bleibt im *fifoPuffer* erhalten. Zum Schluß (Zeile 32) erfolgt noch die Bestimmung der SCAN-Richtung, wenn, im Falle eines initialen Auffüllens zu Rundenbeginn oder nachdem *scanListe* vollständig geleert wurde, diese noch nicht bestimmt ist. Da die Prozedur versucht, so viele D-Aufträge wie möglich in die SCAN-Liste aufzunehmen, und dabei gleichzeitig garantieren muß, daß alle K-Aufträge innerhalb der Runde ausgeführt werden, ist eine Berechnung der Bedienzeit aller Aufträge in der SCAN-Liste notwendig. Diese Berechnung wird in der Prozedur *berechneBedienzeit* (siehe Abbildung 3.14) durchgeführt und basiert auf der Kenntnis aller Auftragspositionen, eines hinreichend exakten Modells zur Berechnung der Positionierungszeiten, aller Auftragsgrößen sowie der Transferrate der Platte (Zeile 16-20). Der einzige Parameter, der im voraus unbekannt sein kann, ist die Rotationsverzögerung, die bei jedem einzelnen Auftrag in der SCAN-Liste entsteht. Sie wird gegebenenfalls konservativ abgeschätzt, indem für jeden Auftrag die maximale Rotationsverzögerung angenommen wird (Zeile 19). Als Rückgabewert (Zeile 26) liefert die Prozedur die berechnete Bedienzeit. Ist die SCAN-Richtung undefiniert, so muß, damit eine minimale Bedienzeit berechnet werden kann, zuvor die SCAN-Richtung (Zeile 8) bestimmt werden. Dies geschieht in der Prozedur *bestimmeScanRichtung* in Abbildung 3.15. Sie liefert als Rückgabewert die ermittelte Richtung für eine mit Aufträgen gefüllte SCAN-Liste.

## 3.4 Verkürzung der Rotationsverzögerungen

Da die Bedienzeit eines Auftrages besonders bei kleinen Auftragsgrößen mit kurzer Transferzeit maßgeblich von der Rotationsverzögerung bestimmt wird, gibt es Ansätze, mit der Optimierung

der Ausführungsreihenfolge nicht nur die Positionierungszeit, sondern gleichzeitig auch die Rotationsverzögerung pro Auftrag zu verkürzen [RNP+98].

Hierbei erfolgt die Abbildung auf ein TSP-Problem (Traveling Salesman Problem) unter Verwendung eines vollständigen Graphen: jeder Knoten entspricht einem Auftrag zu Beginn der aktuellen Runde, der letzte Auftrag der vorangegangenen Runde ist ebenfalls als Knoten im Graphen vorhanden und markiert die aktuelle Position des Plattenkopfes. Die Kosten, die beim Übergang zwischen Auftrag $A$ zu Auftrag $B$ entstehen, entsprechen der Bedienzeit von Auftrag $B$, wenn zuvor Auftrag $A$ ausgeführt wurde. Ausgehend von der aktuellen Position wird ein Weg im Graphen gesucht, der alle Aufträge miteinander verbindet und dabei die Summe der Übergangskosten, d.h. die Gesamtbedienzeit, minimiert.

Das Aufstellen des Ausführungsplanes zu Rundenbeginn kann wie beim SCAN-Algorithmus erfolgen, indem D-Aufträge so lange eingefügt werden, bis die gefundene Gesamtbedienzeit die Rundendauer überschreitet. Des weiteren kann bei einer inkrementellen Auswahlstrategie diese Prozedur während der Runde wiederholt werden, falls die Gesamtbedienzeit die Rundendauer zu Rundenbeginn unterschreitet.

Nachteile dieses Optimierungsansatzes sind praktische Probleme bei der Umsetzung. Zum einen ist die Berechnung zur Lösung des TSP-Problems bei einer realistischen Anzahl von mehreren hundert Aufträgen nicht in kurzer Zeit (wenige Millisekunden) möglich. Hierdurch ist die zur Berechnung des Ausführungsplanes benötigte Zeit, die von der Rundendauer abzuziehen ist, größer als die durch Optimierung erzielte Verringerung der Gesamtbedienzeit.

Außerdem setzt die Berechnung ein Modell der Magnetplatte voraus, daß die Positionierungszeiten und Rotationsverzögerungen exakt berechnen kann. Gibt es nur geringe Abweichungen in der Rotationsverzögerung, z.B. durch thermische oder mechanische Einflüsse, so kann die durchführte vermeintliche Optimierung die Gesamtbedienzeit verlängern. Dies geschieht z.B. dann, wenn zugunsten einer großen Verringerung der Rotationsverzögerung eine geringfügig größere Positionierungszeit in Kauf genommen wird.

Eine Implementierung muß, um auf alle aktuellen Parameter der Magnetplatte zugreifen zu können, auf der Ebene der Plattensteuerung auf der Platte selbst durchgeführt werden. Dieses schließt die Verwendung konventioneller, handelsüblicher und deshalb preisgünstiger Magnetplatten aus. Gegen eine Implementierung auf der Ebene des Betriebssystems und eine Verwendung konventioneller Platten spricht auch die Tatsache, daß moderne Magnetplatten z.T. bereits ähnliche, aber undokumentierte Mechanismen zur Verringerung der Rotationsverzögerung (ROSE: 3-Dimensional Rotationally Optimized Seek Execution) [Qua99a] neben weiteren Caching-Strategien [Qua99b] realisieren, die von außen durch das Betriebssystems nicht kontrollierbar sind. Eine gegenseitige negative Beeinflussung zwischen dem Betriebssystem und der Steuerungssoftware auf der Platte wäre hierdurch wahrscheinlich.

Um eine einfache praktische Realisierbarkeit gewährleisten zu können, wird in der vorliegenden Arbeit der einfachere Ansatz gewählt, der lediglich eine Verkürzung der Positionierungszeiten erreicht und dabei die Rotationsverzögerungen unberücksichtigt läßt.

```
1   procedure bearbeiteAufträge()
2   begin
3     var global scanRichtung; var global scanListe; var global fifoPuffer;
4     forever do
5       /* Beginn einer neuen Runde */
6       füge alle K-Aufträge der aktuellen Runde in scanListe ein;
7       if ( getrennte Einteilungsstrategie ) then
8         /* führe alle K-Aufträge im ersten Bedienabschnitt der Runde aus */
9         scanRichtung := bestimmeScanRichtung();
10        while ( (aktuelle Runde noch nicht beendet) and (scanListe nicht leer) ) do
11          verarbeiteAuftrag();
12        od
13        scanRichtung := undefiniert;
14      fi
15      /* initiales Auffüllen der SCAN-Liste */
16      fuelleScanListe();
17      /* Schleife bis die Runde beendet ist */
18      while ( aktuelle Runde noch nicht beendet ) do
19        /* wähle Auswahl- und Anordnungsstrategie */
20        switch ( Auswahl- und Anordnungsstrategie ) do
21          case ( nicht inkrementelle SCAN-Strategie ) do
22            if ( scanListe ist leer ) then
23              scanRichtung := undefiniert;
24              warte bis Runde beendet;
25            else verarbeiteAuftrag(); fi
26          od
27          case ( beschränkt inkrementelle SCAN-Strategie ) do
28            if ( scanListe ist leer ) then fuelleScanListe(); fi
29            if ( scanListe nicht leer ) then
30              verarbeiteAuftrag();
31            else
32              scanRichtung := undefiniert;
33              warte bis Runde beendet oder D-Auftrag eintrifft;
34            fi
35          od
36          case ( voll inkrementelle SCAN-Strategie ) do
37            fuelleScanListe();
38            if ( scanListe nicht leer ) then verarbeiteAuftrag();
39            else
40              scanRichtung := undefiniert;
41              warte bis Runde beendet oder D-Auftrag eintrifft;
42            fi
43          od
44          case ( voll inkrementelle FCFS-Strategie ) do
45            if ( fifoPuffer nicht leer ) then
46              entferne einen Auftrag aus fifoPuffer und verarbeite diesen;
47            else warte bis Runde beendet oder D-Auftrag eintrifft; fi
48          od
49        od /* switch */
50      od /* while */
51    od /* forever */
52  end
```

Abbildung 3.11: Die Hauptprozedur zur Auswahl des verwendeten Scheduling-Algorithmus.

```
1   procedure bearbeiteAuftrag()
2   begin
3     if ( scanRichtung == vorwärts ) then
4       entferne einen Auftrag von vorne aus scanListe und führe diesen Auftrag aus;
5     else
6       entferne einen Auftrag von hinten aus scanListe und führe diesen Auftrag aus;
7     fi
8   end
```

Abbildung 3.12: Die Prozedur zur Ausführung eines einzelnen K- oder D-Auftrages.

```
1   procedure fuelleScanListe()
2   begin
3     var local auftrag; var local iterator;
4     iterator := erste Position in fifoPuffer;
5     while ( fifoPuffer nicht vollständig durchlaufen ) do
6       if ( berechneBedienzeit() > verbleibende Rundendauer ) then
7         break; /* verlassen while-Schleife */
8       fi
9       auftrag := Auftrag an Position von iterator;
10      /* SCAN-Richtung ist nur im Falle des initialen Auffüllens undefiniert */
11      if ( ((scanRichtung != undefiniert) and (auftrag liegt vor aktueller Kopfposition))
12                                          or (scanRichtung == undefiniert) ) then
13        füge auftrag in scanListe ein;
14        /* überprüfe, ob alle K-Aufträge innerhalb der Runde */
15        /* ausgeführt werden können */
16        if ( (berechneBedienzeit() ≤ verbleibende Rundendauer) or
17             (alle K-Aufträge können innerhalb der Runde ausgeführt werden) ) then
18          if ( scanRichtung == vorwärts )
19            iterator := Position hinter auftrag;
20          else iterator := Position vor auftrag; fi
21          lösche auftrag aus fifoPuffer;
22          /* füge ggf. weitere Aufträge ein */
23        else
24          lösche auftrag aus scanListe;
25          break; /* verlasse while-Schleife */
26        fi
27      else
28        break; /* verlasse while-Schleife */
29      fi
30    od
31    if ( scanRichtung == undefiniert ) then
32      scanRichtung := bestimmeScanRichtung();
33    fi
34  end
```

Abbildung 3.13: Die Prozedur zum Einfügen von D-Aufträgen in die SCAN-Liste.

```
1   procedure berechneBedienzeit() : real
2   begin
3     var local bedienzeit; var local tempzylinder; var local tempScanRichtung;
4     var local auftrag; var local iterator;
5     bedienzeit := 0;
6     tempZylinder := Zylinderposition des Plattenkopfes;
7       if ( scanRichtung == undefiniert) then
8         tempScanRichtung := bestimmeScanRichtung();
9       else tempScanRichtung := scanRichtung; fi
10    if ( tempScanRichtung == vorwärts ) then
11      iterator := Position des ersten Auftrages in scanListe;
12    else iterator := Position des letzten Auftrages in scanListe; fi
13    while ( scanListe noch nicht vollständig durchlaufen ) do
14      auftrag := Auftrag an der Position von iterator in scanListe;
15      bedienzeit += Positionierungszeit zwischen tempZylinder und
16                    Zylinderposition von auftrag;
17      if ( Rotationsverzögerung von auftrag bekannt ) then
18        bedienzeit += Rotationsverzögerung von auftrag;
19      else bedienzeit += maximale Rotationsverzögerung; fi
20      bedienzeit += (Größe von Auftrag auftrag) / Plattentransferrate;
21      tempZylinder := Zylinderposition von Auftrag auftrag;
22      if ( tempScanRichtung == vorwärts ) then
23        iterator := nachfolgende Position in scanListe;
24      else iterator := vorangehende Position in scanListe; fi
25    od
26    return bedienzeit;
27  end
```

Abbildung 3.14: Die Prozedur zur Berechnung der Bedienzeit.

```
1   procedure bestimmeScanRichtung: { vorwärts, rückwärts }
2   begin
3     var local äußererZylinder; var local innererZylinder; var local aktuellerZylinder;
4     äußererZylinder := Zylinderposition des Auftrages am Beginn von scanListe;
5     innererZylinder := Zylinderposition des Auftrages am Ende von scanListe;
6     aktuellerZylinder := aktuelle Zylinderposition des Plattenkopfes;
7     /* wähle den kürzesten Weg zum nächsten Auftrag */
8     if ( (|äußererZylinder - aktuellerZylinder| < |innererZylinder - aktuellerZylinder|)
9          or (( scanListe enthält nur einen Auftrag)
10           and (äußererZylinder > aktuellerZylinder)) ) then
11      return vorwärts;
12    else return rückwärts; fi
13  end
```

Abbildung 3.15: Die Prozedur zur Berechnung der SCAN-Richtung.

# Kapitel 4

# Evaluation ausgewählter Scheduling-Algorithmen

## 4.1 Modellierung der Hardwarekomponenten, Datenobjekte und der Last

Die Evaluation der Algorithmen I bis V aus Abschnitt 3.3 erfolgt unter Verwendung von Modellen, die die Hardware, Datenobjekte und Last simulieren. Durch die systematische Veränderung der Modellparameter ist eine breite Untersuchung unabhängig von den Charakteristika einer gegebenen technischen Konfiguration auf einfache Art und Weise möglich. Im folgenden werden die in der Simulation benutzten Modelle vorgestellt. Als erstes erfolgt die Beschreibung, wie die Simulation des Disk-Arrays durchgeführt wird. Es wird angenommen, daß die im Disk-Array zusammengefaßten $P_{disks}$ Platten unabhängig voneinander, d.h. unsynchronisiert, arbeiten können, so daß im folgenden die Betrachtung des Modells an einer einzigen Platte vorgenommen werden kann.

### 4.1.1 Das Plattenmodell

Das Plattenmodell simuliert das zeitliche Verhalten einer Platte beim Datenzugriff und berechnet die Bedienzeit für einen Auftrag. Im folgenden wird die Modellierung der einzelnen Bestandteile der Bedienzeit pro Platte näher erläutert. Hierbei wird davon ausgegangen, daß die Daten von der Platte gelesen werden müssen, ohne daß ein eventuell vorhandener Zwischenspeicher auf der Platte (*Platten-Cache*) tätig wird, der u.U. Teile der zu lesenden Daten aufgrund eines vorangegangenen Zugriffs noch enthält. Diese Einschränkung in der Modellierung ist gerechtfertigt, da in der Realität aufgrund der Größe der zu lesenden Fragmente und der großen Restlast, d.h. Anzahl und Häufigkeit verschiedener Zugriffe auf diskrete Daten, der Platten-Cache keine Wirkung zeigt.

#### Modell für die Positionierungszeit

Die Positionierungszeit ist abhängig von der Distanz $d$, die der Plattenkopf bis zur Position des Auftrages zu überqueren hat, und den physikalischen Eigenschaften der Platte. Im Modell wird

die Positionierungszeit durch die Funktion $t_{pos}$ berechnet, die hinreichend genau das Verhalten der Platte beschreibt [RW94b]:

$$t_{pos}(d) = \begin{cases} 1.868 * 10^{-3} + 1.316 * 10^{-4}\sqrt{d}\ \text{[sec]} & : \quad 0 \leq d < 1344 \\ 3.865 * 10^{-3} + 2.104 * 10^{-6} d\ \text{[sec]} & : \quad 1344 \leq d < P_{cyls} - 1 \end{cases}$$

$$\text{mit} \quad P_{cyls} = 6720$$

Die physikalischen Eigenschaften fließen in diese Funktion als Positionierungsparameter ein. Ihre Berechnung erfolgt auf Basis von Herstellerangaben in Abschnitt 5.2. Aus der Funktion erkennt man, daß bei einer kurzen Distanz $d$ die Positionierungszeit durch einen nichtlinearen Anteil bestimmt wird, da der Arm den größten Zeitanteil in der Beschleunigungsphase verbringt. Ab einer bestimmten Positionierungsdistanz, im Modell ist dies $1344$ Zylinder, überwiegt der linear wachsende Zeitanteil der Hochgeschwindigkeitsphase. Da die Zylinderanzahl der Platte durch $P_{cyls}$ gegeben ist, beträgt die maximale Positionierungsdistanz $P_{cyls} - 1$. Die Positionierungszeit in Abhängigkeit der Positionierungsdistanz ist in Abbildung 4.1 zu sehen.


Abbildung 4.1: Positionierungszeit in Abhängigkeit von der Positionierungsdistanz

In der Simulation werden Positionierungszeiten, die innerhalb eines Datenzugriffes anfallen können, nicht berücksichtigt. Der zeitliche Anteil, der für eine Repositionierung auf eine benachbarte Spur mit einem damit verbundenen Zylinderwechsel oder Kopfwechsel benötigt wird, ist an der Positionierungszeit gemessen derart gering, daß er ignoriert werden kann.

### Modell für die Rotationsverzögerung

Um die Rotationsverzögerung bei jedem Zugriff exakt bestimmen zu können, müßte man zu jedem Zeitpunkt die exakte Stellung der Scheiben und damit den Winkel zwischen Plattenkopf und dem anzufahrenden Block bestimmen können. Dies hätte einen hohen Berechnungsaufwand und damit eine unakzeptabel hohe Simulationsdauer zur Folge. Aus diesem Grund wird der Wert der Rotationsverzögerung bei jedem Zugriff durch das Ziehen einer Zufallszahl für

eine gleichverteilte Zufallsvariable $T_{rot}$ ermittelt. Dies ist unter der Annahme möglich, daß die Datenzugriffe und damit die Auswahl der anzufahrenden Blöcke unabhängig voneinander erfolgen. Die Zufallsvariable $T_{rot}$ kann Werte von $0$ bis zu der Zeit $P_{rot}$ für eine volle Plattenumdrehung annehmen. $P_{rot}$ läßt sich als Kehrwert der Rotationsgeschwindigkeit aus den Herstellerdaten berechnen. Die Dichtefunktion $f_{rot}$ von $T_{rot}$ lautet wie folgt:

$$f_{rot}(t) = \begin{cases} \frac{1}{P_{rot}} & : \quad 0 \leq t \leq P_{rot} \\ 0 & : \quad \text{sonst} \end{cases}$$

$$\text{mit} \quad P_{rot} = 8.34 \text{ msec}$$

**Modell für die Transferzeit**

Die Transferzeit, die für das Lesen der Blöcke des Datenobjektes notwendig ist, ist durch die Transferrate und die Größe des Datenobjektes festgelegt. Die Simulation verwendet ein Modell für eine Ein-Zonen-Platte, die eine konstante Transferrate $P_{rate}$ besitzt. Die Verwendung eines Modells für eine Mehr-Zonen-Platte würde bei der gleichmäßigen Verteilung der Daten die Simulationsergebnisse qualitativ nicht verändern, sondern lediglich aufgrund zusätzlicher Berechnungen im Modell dessen Simulationsdauer verlängern. Die Transferzeit wird durch die Funktion $t_{trans}$ berechnet und hat für die Auftragsgröße $s$ den Wert:

$$t_{trans}(s) = \frac{s}{P_{rate}} \qquad \text{mit} \qquad P_{rate} = 8.79\text{MBytes/sec}$$

### 4.1.2 Das Datenmodell

Das Datenmodell beschreibt die Verteilung und die Größe der in der Simulation verwendeten Daten. Die Allokation der Datenobjekte erfolgt in der Simulation lediglich *virtuell*, d.h., es wird keine Verteilung der Daten mit dazugehöriger Buchführung über Position und Größe aller gespeicherten Datenobjekte durchgeführt. Stattdessen wird bei jedem Zugriff ein *virtuelles Datenobjekt* bzw. ein *virtuelles Fragment* mit zufälliger Position und zufälliger Größe bestimmt.

**Datenplazierung**

Das Modell der Datenplazierung geht davon aus, daß ein *virtuelles* diskretes Datenobjekt zusammenhängend auf einer einzigen Platte abgelegt wird. Ein kontinuierliches Datenobjekt wird entsprechend dem in Abschnitt 2.2.1 vorgestellten KWL-Partitionierungsverfahren in Fragmente zerlegt und auf mehrere Platten mittels grobkörnigen Stripings verteilt (siehe Abschnitt 2.2.2). Jedes Fragment wird in zusammenhängenden Blöcken auf einer Platte plaziert. Die Auswahl der Platten erfolgt derart, daß eine gleichmäßige Verteilung der Daten gewährleistet ist. Die bedeutet, daß beim Datenzugriff die Platte mit einer gleichverteilten Zufallsvariablen bestimmt wird. Da ein Disk-Array mit $P_{disks}$ Platten simuliert wird, hat die Wahrscheinlichkeit,

daß ein diskretes Datenobjekt bzw. das erste Fragment eines kontinuierlichen Datenobjektes auf einer bestimmten Platte liegt, den Wert $1/P_{disks}$. Innerhalb der Platte erfolgt die Plazierung der Daten ebenfalls gleichmäßig über alle Zylinder mit einer gleichverteilten Zufallsvariablen. Die Wahrscheinlichkeit, einen bestimmten Zylinder der simulierten Platte auszuwählen, hat den Wert $1/P_{cyls} \approx 1.448 * 10^{-4}$.

#### Charakterisierung diskreter Datenobjekte

Die Größe eines diskreten Datenobjektes wird durch eine Zufallsvariable $S_D$ bestimmt. In den Experimenten kommen drei verschiedene Charakterisierungen dieser Zufallsvariable mit unterschiedlicher Dichtefunktion, unterschiedlichem Erwartungswert $E[S_D]$ und Varianz *Var*$[S_D]$ zum Einsatz. Eine Untersuchung mit *Datenparameter I* betrachtet große diskrete Datenobjekte mit einer *normalverteilten* mittleren Größe von $50000$ Bytes. Durch die *Datenparameter II* sind kleinere Objekte mit einer mittleren Größe von $10000$ Bytes charakterisiert. Der Variationskoeffizient (engl. coefficient of variation), d.h. das Verhältnis zwischen Standardabweichung und Erwartungswert ($\sqrt{Var[S_D]}/E[S_D]$), ist in beiden Fällen identisch und hat den Wert $0.5$. Die Verwendung von *Datenparameter III* nimmt *gammaverteilte* Objektgrößen an und verwendet denselben Erwartungwert und dieselbe Varianz wie für Datenparameter I. Dichtefunktionen und Parameter sind in Tabelle 4.1 zusammengefaßt.

#### Charakterisierung kontinuierlicher Datenobjekte

Die Größe der Fragmente kann ungefähr durch eine *gammaverteilte* Zufallsvariable $S_K$ modelliert werden [Ros95, KH95]. Die in der Simulation verwendete mittlere Größe der Fragmente beträgt $800000$ Bytes, die Varianz $(200000 \text{ Bytes})^2$. Dies entspricht der Größe und der Charakteristik von Fragmenten eines MPEG-2 Datenobjektes, die eine Wiedergabelänge von einer Sekunde besitzen. Parameter und Dichtefunktion der Zufallsvariablen $S_K$ sind ebenfalls in Tabelle 4.1 zusammengefaßt. Die Länge der kontinuierlichen Datenobjekte, d.h. deren Anzahl an Fragmenten, wird nicht modelliert, da dieser Wert im Lastmodell keine Berücksichtigung findet.

### 4.1.3 Das Lastmodell

Das Lastmodell simuliert Benutzer, die Anfragen an das System stellen und hierdurch Aufträge erzeugen. Die *Last* wird bestimmt durch die Häufigkeit der Datenzugriffe, die Zugriffsverteilung innerhalb der Datenobjekte und der Größe der angeforderten Datenobjekte. Für beide Datentypen wird im folgenden die Last getrennt spezifiziert.

#### D-Lastmodell

Die Häufigkeit der Zugriffe auf diskrete Datenobjekte wird durch einen stochastischen Poisson-Prozeß beschrieben [All90, Nel95]. Dieser Prozeß simuliert das Eintreffen von Benutzeranfragen, die D-Aufträge erzeugen und somit Datenzugriffe auslösen. Durch den Poisson-Prozeß

**Größe diskreter Datenobjekte / Auftragsgröße D-Aufträge**

| **Datenparameter I:** | **Datenparameter II:** |
|---|---|
| $E[S_D] = 50000$ Bytes | $E[S_D] = 10000$ Bytes |
| $\mathit{Var}[S_D] = (25000 \text{ Bytes})^2$ | $\mathit{Var}[S_D] = (5000 \text{ Bytes})^2$ |

normalverteilte Zufallsvariable $S_D$ mit Dichtefunktion $f_{S_D}$

$$f_{S_D}(s) = \begin{cases} \frac{1}{\sqrt{2\pi \mathit{Var}[S_D]}} \, e^{-\frac{1}{2}\frac{(s - E[S_D])^2}{\mathit{Var}[S_D]}} & : \quad s > 0 \\ 0 & : \quad \text{sonst} \end{cases}$$

**Datenparameter III:**

$$\alpha = \frac{E[S_D]}{\mathit{Var}[S_D]} = \frac{50000 \text{ Bytes}}{(25000 \text{ Bytes})^2} \qquad \beta = \frac{(E[S_D])^2}{\mathit{Var}[S_D]} = \left(\frac{50000 \text{ Bytes}}{25000 \text{ Bytes}}\right)^2$$

gammaverteilte Zufallsvariable $S_D$ mit Dichtefunktion $f_{S_D}$

$$f_{S_D}(s) = \begin{cases} \frac{\alpha(\alpha s)^{\beta-1} * e^{-\alpha s}}{\Gamma(\beta)} & : \quad s > 0 \\ 0 & : \quad \text{sonst} \end{cases}$$

**Größe kontinuierlicher Datenfragmente / Auftragsgröße K-Aufträge**

$$\alpha = \frac{E[S_K]}{\mathit{Var}[S_K]} = \frac{800000 \text{ Bytes}}{(200000 \text{ Bytes})^2} \qquad \beta = \frac{(E[S_K])^2}{\mathit{Var}[S_K]} = \left(\frac{800000 \text{ Bytes}}{200000 \text{ Bytes}}\right)^2$$

gammaverteilte Zufallsvariable $S_K$ mit Dichtefunktion $f_{S_K}$

$$f_{S_K}(s) = \begin{cases} \frac{\alpha(\alpha s)^{\beta-1} * e^{-\alpha s}}{\Gamma(\beta)} & : \quad s > 0 \\ 0 & : \quad \text{sonst} \end{cases}$$

Tabelle 4.1: Datenparameter für die Evaluation der Scheduling-Algorithmen

sind die *Zwischenankunftszeiten*, d.h. die Zeiten zwischen zwei direkt aufeinander eintreffenden Benutzeranfragen und damit auch zwischen zwei D-Aufträgen, exponential verteilt. Die Häufigkeit der eintreffenden Anfragen wird durch die Ankunftsrate $\lambda_{tot}$ beschrieben. Dieser Parameter, der für das Gesamtsystem gilt, wird in den Experimenten systematisch variiert und bestimmt die durch diskrete Datenzugriffe hervorgerufene Last, die im folgenden als *D-Last* bezeichnet wird.

Jede Anfrage bewirkt, daß der resultierende D-Auftrag stets das virtuelle diskrete Datenobjekt vollständig liest (Schreiboperationen werden im Modell vernachlässigt, da die betrachteten typischen Anwendungen zum größten Teil nur lesend auf die Daten zugreifen). Die Verteilung der Auftragsgrößen von D-Aufträgen ist somit identisch mit der Größenverteilung der diskreten Datenobjekte im Datenmodell.

Die Wahrscheinlichkeit für die Auswahl eines bestimmten Datenobjektes ist im Lastmodell über alle Platten gleichverteilt.

#### K-Lastmodell

Die *K-Last* wird durch eine konstante Anzahl $N$ der pro Runde und Platte auszuführenden K-Aufträge beschrieben. In den Experimenten wird dieser Parameter für eine niedrige K-Last ($N = 3$) und eine hohe K-Last ($N = 7$) variiert, wodurch eine Auslastung der Platten von ungefähr $30\%$ bzw. $70\%$ hervorgerufen wird. Insgesamt werden mit $P_{disks}$ Platten folglich vom Gesamtsystem $N_{tot} = P_{disks} * N$, d.h. $P_{disks} * 3$ bzw. $P_{disks} * 7$ Datenströme permanent bedient. Die Rundendauer hat einen konstanten Wert $T$.

## 4.2 Simulationsumgebung

Die Implementierung der Umgebung ist in C++ [Str97, MSS96] unter Verwendung der Simulationsbibliothek CSIM [Sch98] vorgenommen worden. Diese Bibliothek erlaubt durch die Verwendung des Konzepts *diskreter Ereignisse* und die Bereitstellung von Konstrukten zur Definition paralleler Prozesse [Kre86] das dynamische Verhalten der Scheduling-Algorithmen exakt nachzubilden. Weiterhin können die zu untersuchenden Leistungsmerkmale mit Hilfe geeigneter CSIM-Datenstrukturen gesammelt und statistisch aufbereitet werden. Im folgenden soll zunächst die Architektur des Simulators beschrieben werden:

### 4.2.1 Architektur

Die Architektur der Simulationsumgebung, in der die experimentelle Evaluation stattfindet, ist in Abbildung 4.2 skizziert und setzt sich aus folgenden Komponenten zusammen:

- Last-Module
- Zeitgeber-Modul
- Steuerungs-Module

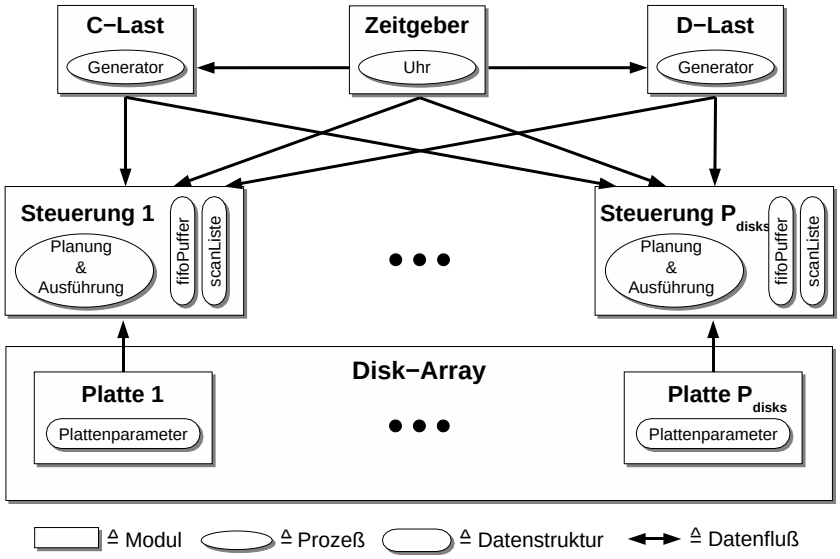



Abbildung 4.2: Architektur der Simulationsumgebung

- Disk-Array-Modul

### Last-Module

Es existieren für D- und K-Last zwei getrennte Lastgeneratoren, die in parallel arbeitenden Prozessen D-Aufträge bzw. K-Aufträge erzeugen und an die Module zur Plattensteuerung (Steuerungs-Module) weiterleiten. Der D-Lastprozeß fügt die gemäß eines Poisson-Ankunftstroms auftretenden D-Aufträge in die Warteschlange des jeweiligen Steuerungs-Moduls ein. K-Aufträge werden nur zu Beginn einer Runde erzeugt und dem Steuerungs-Modul zum direkten Einfügen in die SCAN-Liste zur Verfügung gestellt. Die hierfür benötigten zeitlichen Informationen erhalten beide Prozesse über ein Zeitgeber-Modul.

### Zeitgeber-Modul

Der Zeitgeber synchronisiert über eine für die gesamte Simulationsumgebung gültige Uhr die einzelnen Komponenten. Die Realisierung erfolgt als eigenständiger Prozeß, der am Ende einer Runde über einen Ereignismechanismus den Beginn der neuen Runde den einzelnen Steuerungs-Modulen signalisiert. Des weiteren wird den Steuerungs-Modulen zur Bestimmung der noch verbleibenden Rundendauer die aktuelle Simulationszeit des Zeitgebers zur Verfügung gestellt.

### Steuerungs-Module

Innerhalb der Steuerungs-Module wird der Scheduling-Algorithmus als selbständiger Prozeß ausgeführt, der die Einplanung und die Ausführung der Aufträge vornimmt. Je nach Konfiguration ist jeweils einer der zu evaluierenden Algorithmen aktiv. Innerhalb des Moduls befinden sich die beiden Datenstrukturen *fifoPuffer* und *scanListe*, die zum Aufbau des Ausführungsplanes benötigt werden. Informationen über die aktuelle Zeit sowie über das Rundenende wird dem Steuerungs-Modul über das Zeitgeber-Modul zur Verfügung gestellt. Für jede Platte existiert ein eigenes Steuerungs-Modul, so daß in der Simulation die Einplanung und die Ausführung der Aufträge wie in einem realen System parallel durchgeführt werden können.

**Disk-Array-Modul**

Das Disk-Array-Modul übernimmt die Simulation der einzelnen Platten. Hierbei stellt es im wesentlichen Routinen zur Verfügung, die die Bedienzeit eines Auftrages, wie im vorangegangenen Kapitel beschrieben, gemäß dem verwendeten Plattenmodell berechnen. Die Simulationsumgebung bietet die Möglichkeit, unterschiedliche Plattentypen gleichzeitig in einem somit heterogenen Disk-Array zu verwenden. Hiervon wird jedoch in den vorgestellten Experimenten kein Gebrauch gemacht.

### 4.2.2 Untersuchte Leistungsmetriken

Für den Vergleich der Scheduling-Algorithmen sind zwei Metriken von Interesse, die die Leistungsfähigkeit der Algorithmen dokumentieren. Die eine Metrik ist die *Leistungskapazität* des Servers hinsichtlich D-Aufträge, d.h. der maximale *Durchsatz* an D-Aufträgen für eine festgelegte Anzahl von K-Aufträgen pro Runde. Als Durchsatz ist die Anzahl der pro Zeiteinheit ausgeführten Aufträge definiert. Die andere Leistungsmetrik ist die *mittlere Antwortzeit* für D-Aufträge. Hinsichtlich der K-Aufträge ist keine Leistungsmetrik notwendig, da alle Algorithmen dafür sorgen, daß Störungen bei K-Aufträgen vermieden werden. Dies gilt allerdings nur, sofern die Störungen dadurch entstehen, daß zu viele D-Aufträge in der Runde ausgeführt werden. Störungen können in dem Fall auftreten, in dem die alleinige Bearbeitung der K-Aufträge bereits die Rundendauer überschreitet. Aufgrund der gewählten Parameter (Abschnitte 4.1.1, 4.1.2) wird dies in den Experimenten aber nie beobachtet.

### 4.2.3 Simulationsdauer und Signifikanz

Mit steigender Simulationsdauer steigt die Anzahl der insgesamt bearbeiteten Aufträge und damit die Anzahl der gemessenen Antwortzeiten, die zur Berechnung des Mittelwertes verwendet werden. Um sicherzugehen, daß der aus einer endlichen Anzahl $\{a_1, a_2, a_3, \dots\}$ von Messungen berechnete Mittelwert $\bar{a}$ nur geringfügig vom tatsächlichen Erwartungswert abweicht, wird die Messung so lange durchgeführt bis der Mittelwert der Antwortzeit eine vorgegebene *Konfidenz* erreicht [Jai91]. In den Messungen wird das *Konfidenzniveau* auf $95\%$ und der relative Fehler auf $1\%$ festgelegt. Dies bedeutet, daß der tatsächliche Erwartungswert $\mu_a$ mit $95\%$ Wahrscheinlichkeit innerhalb des Konfidenzintervalls $[c_1, c_2]$ liegt, d.h.

$$P[c_1 \leq \mu_a \leq c_2] = 0.95 \quad ,$$

und daß die Größe des Intervalls $[c_1, c_2]$ durch den relativen Fehlers $\rho_a$, so wie ihn die Simulationsbibliothek definiert, begrenzt wird:

$$\rho_a = \frac{c_2 - c_1}{2c_1} \leq 0.01 \quad .$$

## 4.3 Experimentelle Evaluation

Ziel der experimentellen Evaluation ist es, die Leistungsfähigkeit der vorgestellten Scheduling-Strategien anhand der Algorithmen I bis V zu untersuchen und den zur gemeinsamen Aus-

führung von D- und K-Aufträgen geeignetsten Algorithmus zu ermitteln. Die experimentelle Evaluation untersucht die Vor- und Nachteile dieser Algorithmen.

Hierbei erfolgt die Gliederung anhand der verwendeten Scheduling-Strategien. Es werden zunächst die Datenparameter I (große diskrete Datenobjekte) aus Tabelle 4.1 verwendet. Das Disk-Array besteht in allen Messungen aus fünf Platten ($P_{disks} = 5$). Die konstante Rundendauer $T$ ist bei allen Experimenten auf den Wert von einer Sekunde festgesetzt und wird nicht variiert. Alle Algorithmen benutzen bei ihren Berechnungen exakte Rotationsverzögerungen und sind so in der Lage, beim Einfügen von D-Aufträgen in den Ausführungsplan im voraus die Gesamtbedienzeit genau zu berechnen. Hierdurch kann zu Rundenbeginn, wenn hinreichend D-Aufträge in der Warteschlange zur Verfügung stehen, der Ausführungsplan optimal mit D-Aufträgen aufgefüllt werden. Die Auswirkungen veränderter Datenparameter und die Abschätzung der Rotationsverzögerung werden in den letzten beiden Abschnitten untersucht.

### 4.3.1 Auswirkungen der Anordnungsstrategie

Zunächst sollen die Auswirkungen der Anordnungsstrategie anhand der Algorithmen I und II aus Abschnitt 3.3.1 untersucht werden. Beide Algorithmen verwenden eine getrennte Einteilungsstrategie. Unterschiede gibt es hinsichtlich der Anordnungsstrategie: Algorithmus I verwendet eine FCFS-Strategie, Algorithmus II verwendet dagegen eine SCAN-Strategie. Die Ergebnisse der Messungen sind in Abbildung 4.3 zu sehen. Der Graph zeigt in Abhängigkeit von der Ankunftsrate die mittlere Antwortzeit der D-Aufträge bei hoher und niedriger K-Last, d.h. bei insgesamt $5 * 7 = 35$ und $5 * 3 = 15$ Aufträgen pro Runde. Der maximale Durchsatz ist dort zu erkennen, wo die mittlere Antwortzeit stark ansteigt und gegen einen (theoretisch) unendlichen Wert strebt. Mit ungefähr $115$ D-Aufträgen pro Sekunde bei $7$ K-Aufträgen pro Runde und Platte ist die Leistungskapazität von Algorithmus II um ca. $35\%$ größer als bei Algorithmus I mit FCFS-Anordnungsstrategie, der lediglich eine Leistungskapazität von ca. $85$ D-Aufträgen pro Sekunde aufweist. Ursache hierfür ist die Verwendung der SCAN-Anordnungsstrategie im D-Abschnitt der Bearbeitungsrunde, die für Algorithmus II die benötigten Positionierungszeiten verringert. Die hierdurch eingesparte Zeit kann für die Bearbeitung zusätzlicher D-Aufträge verwendet werden. Je geringer die K-Last desto mehr Zeit steht für die Ausführung der D-Aufträge zur Verfügung und desto häufiger kann die SCAN-Anordnungsstrategie im Vergleich zur FCFS-Strategie optimieren. Aus diesem Grund wird bei niedriger K-Last die Differenz im Durchsatz zwischen beiden Algorithmen zugunsten von Algorithmus II größer. Bei einer K-Last von $3$ K-Aufträgen pro Runde steigert Algorithmus II den Durchsatz um ca. $42\%$ von ca. $190$ auf ca. $270$ Aufträge pro Sekunde.

### 4.3.2 Auswirkungen der Auswahlstrategie

In einer weiteren Testreihe wird die Auswirkung der Auswahlstrategie anhand der Algorithmen III, IV und V aus Abschnitt 3.3.1, die K- und D-Aufträge gemischt ausführen, untersucht. Abbildung 4.4 stellt die mittlere Antwortzeit der D-Aufträge bei hoher und niedriger K-Last in Abhängigkeit von der Ankunftsrate dar. Es zeigt sich, daß der Algorithmus V mit seiner voll dynamischen Auswahlstrategie für alle Ankunftsraten die kürzeren Antwortzeiten für beide K-Lasten im Vergleich zu den beiden anderen Algorithmen aufweist. Aufgrund der nicht

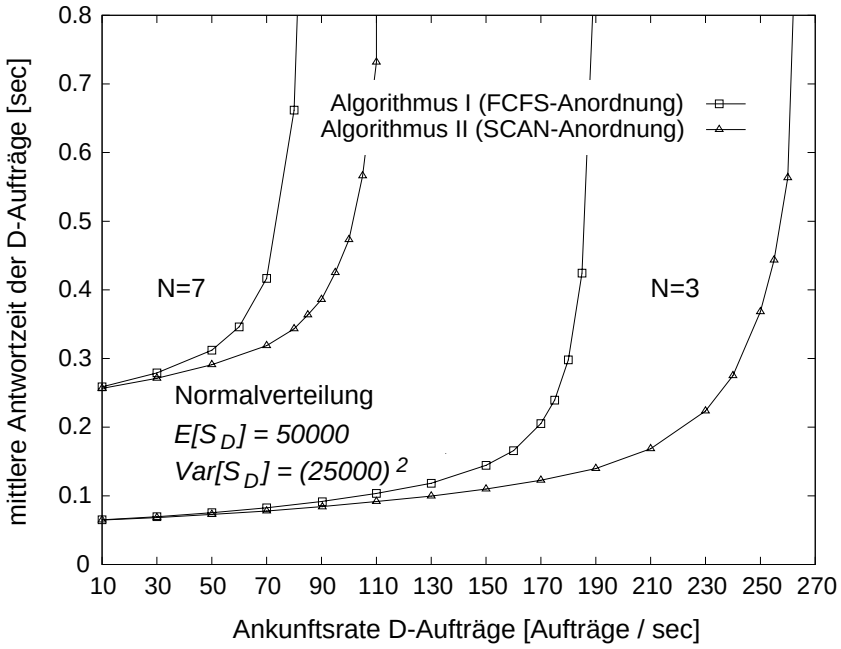



Abbildung 4.3: Auswirkung der Anordnungsstrategie: Antwortzeitvergleich der Algorithmen I und II bei großen diskreten Datenobjekten

inkrementellen Auswahl müssen alle eintreffenden D-Aufträge bei Verwendung von Algorithmus III mindestens das Ende der Ankunftsrunde abwarten, was die im Vergleich hohe Antwortzeit von mindestens $0.67$ bzw. $0.86$ Sekunden erklärt. Algorithmus IV weist ebenfalls im Vergleich zu Algorithmus V eine höhere Antwortzeit auf. Die Zeitdifferenz fällt bei geringer K-Last niedriger aus. Dieser Effekt läßt sich dadurch erklären, daß zu Beginn der Runde nur wenige D-Aufträge in der Warteschlange vorhanden sind, die in der vorangegangenen Runde nicht mehr ausgeführt werden konnten. Deshalb findet Algorithmus IV zu Beginn einer neuen Runde bei der Erstellung des Ausführungsplanes fast ausschließlich K-Aufträge vor, die dann als erstes bearbeitet werden. Zwischenzeitlich eintreffende D-Aufträge können erst nach Abarbeitung der K-Aufträge ausgeführt werden und erfahren hierdurch eine Verzögerung, die bei steigender K-Last größer wird. Algorithmus V versucht, D-Aufträge nach jeder Bearbeitung in den Ausführungsplan zu übernehmen, und kann so eintreffende D-Aufträge zeitlich den K-Aufträgen vorziehen. Hinsichtlich der Leistungskapazität erweisen sich alle drei Algorithmen als gleichwertig. Die Leistungskapazität der drei Algorithmen ist nahezu identisch und liegt bei den verwendeten Daten- und Lastparametern bei ca. $280$ ($N = 3$) und ca. $125$ ($N = 7$) D-Aufträgen pro Sekunde. Der Grund hierfür ist, daß bei voller Auslastung zu Beginn einer Runde bereits so viele D-Aufträge in den Ausführungsplan übernommen werden, wie maximal in einer Runde bearbeitet werden können. Aus diesem Grund können bei maximaler Auslastung zwischenzeitlich eintreffende Aufträge ebensowenig von einem inkrementellen Algorithmus wie von einem nicht inkrementellen Algorithmus vor Rundenende ausgeführt werden.

### 4.3.3 Auswirkungen der Einteilungsstrategie

Der Effekt der verwendeten Einteilungsstrategie auf die Antwortzeit ist in Abbildung 4.5 für die Algorithmen II, IV und V zu sehen. Algorithmus V mit der gemischten Einteilungsstrategie

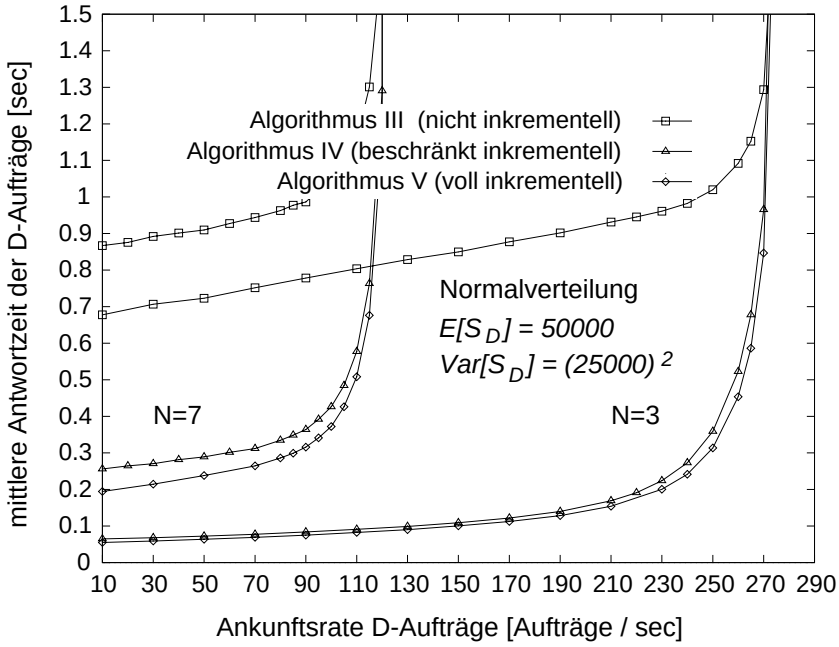



Abbildung 4.4: Auswirkungen der Auswahlstrategie: Antwortzeitvergleich der Algorithmen III, IV und V bei großen diskreten Datenobjekten

zeigt die niedrigsten Antwortzeiten. Dies liegt daran, daß dieser Algorithmus eintreffende D-Aufträge während der Bearbeitung von K-Aufträgen in die SCAN-Liste aufnehmen kann. Hingegen erzwingt Algorithmus II mit der getrennten Einteilungsstrategie, daß alle D-Aufträge, die während des K-Abschnitts eintreffen, bis mindestens zum Ende des K-Abschnitts warten müssen. Hierdurch entsteht die Differenz in der mittleren Antwortzeit von ungefähr $0.05$ Sekunden bei hoher K-Last. Bei geringer K-Last ist der K-Abschnitt kürzer und damit auch die Differenz von ca. $0.01$ Sekunden geringer. Algorithmus IV zeigt trotz gemischter Einteilungsstrategie mit Algorithmus II vergleichbare Antwortzeiten. Die Ursache hierfür ist darin zu sehen, daß bei geringer D-Last zu Anfang der Runde nur wenige D-Aufträge in der Warteschlange vorhanden sind und somit, vergleichbar mit Algorithmus II, fast ausschließlich K-Aufträge in die SCAN-Liste eingefügt und bearbeitet werden. Eintreffende D-Aufträge können durch die beschränkt inkrementelle Auswahl erst wieder nach deren Bearbeitung berücksichtigt werden.

### 4.3.4 Einfluß der Datenparameter

Um die Stabilität der Algorithmen hinsichtlich verschiedener Datenparameter zu überprüfen, wird in weiteren Messungen die Größe der diskreten Datenobjekte verringert. Die verwendeten Parameter sind als Datenparameter II in Tabelle 4.1 angegeben. Abbildung 4.6 zeigt, daß die Sättigung bei verwendeter FCFS-Strategie im Vergleich zur SCAN-Strategie früher, also bei niedrigerer Ankunftsrate einsetzt. Durch die geringere Objektgröße sinkt die Transferzeit und damit auch die Bedienzeit pro Auftrag. Hierdurch kann bei einer größeren Anzahl bedienbarer D-Aufträge pro Runde die SCAN-Strategie häufiger optimieren. Dadurch steigt die Differenz im Durchsatz zwischen FCFS- und SCAN-Strategie an. Aus dem Graphen läßt sich bei einer K-Last von $7$ eine Steigerung des Durchsatzes von ca. $52\%$ (von $115$ auf $175$) und bei einer K-Last von $3$ ein Steigerung um ca. $116\%$ (von $190$ auf $410$) zugunsten der SCAN-Strategie ablesen.

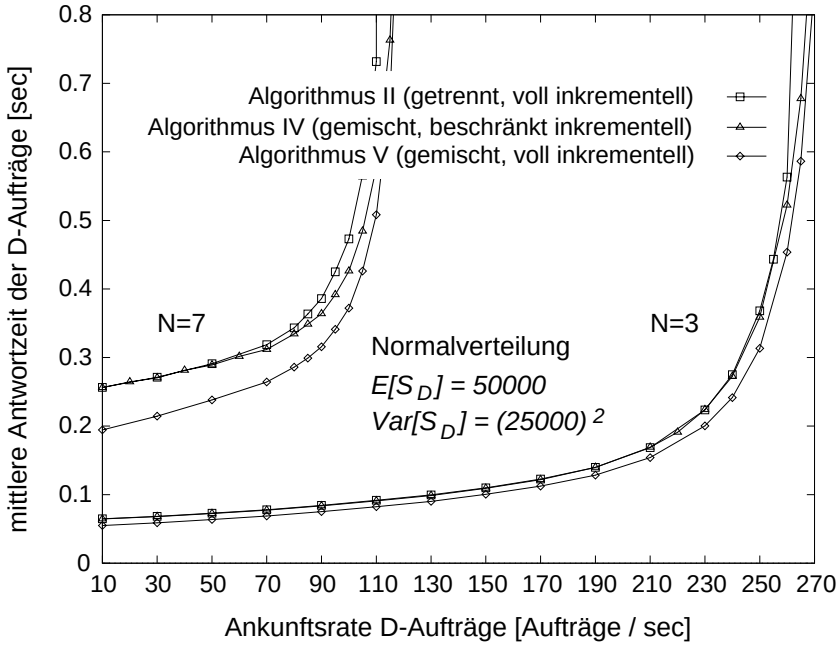



Abbildung 4.5: Auswirkung der Einteilungsstrategie: Antwortzeitvergleich der Algorithmen II, IV und V bei großen diskreten Datenobjekten

Weiterhin ist im Graphen erkennbar, daß eine geringere Objektgröße auf den Kurvenverlauf bei den Algorithmen IV und V keinen signifikanten Einfluß hat. Die Differenz in der mittleren Antwortzeit zwischen Algorithmus IV und V bei geringer D-Last ändert ihren Wert nicht, da die Fragmentgröße unverändert bleibt.

In weiteren Experimenten wird die Größe der diskreten Datenobjekte durch eine gammaverteilte Zufallsvariable bestimmt (Datenparameter III in Tabelle 4.1). Die Ergebnisse zeigen praktisch keinerlei Unterschiede zu denjenigen, die bei gleichem Erwartungswert und gleicher Varianz mit einer normalverteilten Größenverteilung (Datenparameter I in Tabelle 4.1) gemessen werden.

### 4.3.5 Einfluß der Bedienzeitabschätzung

In den vorangegangenen Experimenten haben alle Algorithmen bei der Einplanung der Aufträge und der Aufstellung des Ausführungsplanes die exakte Bedienzeit eines Datenzugriffs a priori, also vor dessen Ausführung, exakt berechnen können. Unter Umständen ist diese Berechnung bei unvollständiger Kenntnis des genauen Verhaltens der verwendeten Geräte oder bei Verwendung komplexer Übertragungs- und Netzwerkprotokolle nicht möglich. Die Vorausberechnung der Rotationsverzögerung setzt z.B. ein exaktes Verhaltensmodell voraus, daß sowohl die genaue Kenntnis der Steuerungsalgorithmen auf der Platte als auch präzise Zustandsinformationen von der Plattensteuerung während des Betriebs benötigt. Diese Informationen sind nur dann zugänglich, wenn eine Implementierung der Algorithmen auf Ebene der Plattensteuerung auf der Platte durchgeführt wird. Ist dies nicht realisierbar oder gewünscht, so muß eine konservative Abschätzung der Rotationsverzögerung vorgenommen werden. Im folgenden soll untersucht werden, inwieweit eine solche Abschätzung die bisher ermittelten Ergebnisse beeinflußt. Abbil-

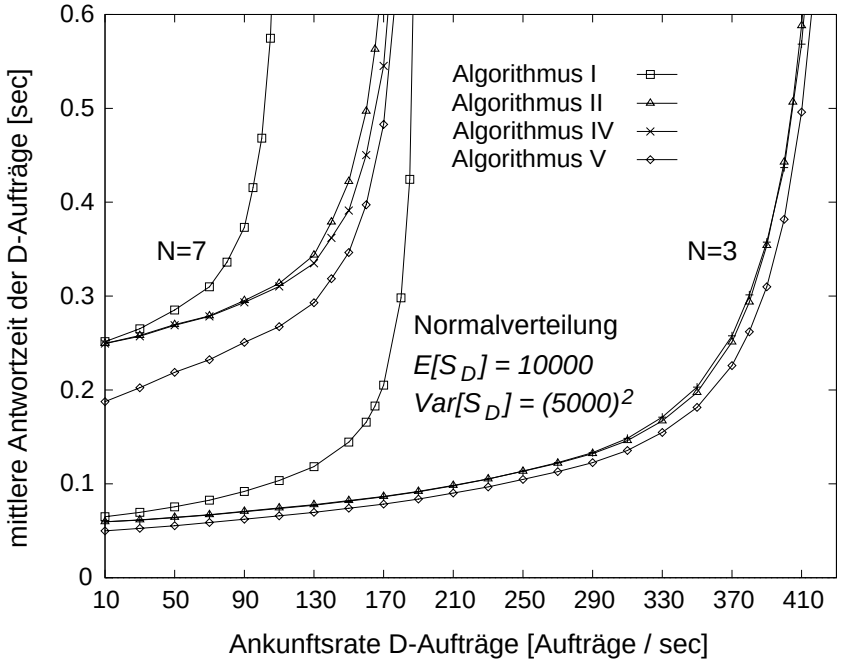



Abbildung 4.6: Einfluß geringerer Objektgrößen: Antwortzeitvergleich der Algorithmen I, II, IV und V bei kleinen diskreten Datenobjekten

dung 4.7 zeigt die mittlere Antwortzeit der Algorithmen III bis V bei exakter Berechnung und bei Abschätzung der Rotationsverzögerung, für die der maximal mögliche Wert, d.h. die Zeit für eine vollständige Drehung, angenommen wird.

Die geringere Leistungskapazität von Algorithmus III mit ca. 85 D-Aufträgen pro Sekunde läßt sich im Fall der konservativen Abschätzung dadurch erklären, daß zu Rundenbeginn weniger D-Aufträge ausgewählt und in der Runde ausgeführt werden als tatsächlich möglich sind. Algorithmus IV füllt in diesem Fall zu Beginn der Runde die SCAN-Liste ebenfalls mit zu wenigen D-Aufträgen auf. Dieser Abschätzungfehler wird innerhalb der Runde aber dadurch korrigiert, daß nach Abarbeitung der SCAN-Liste die tatsächlich zur Bearbeitung verwendete Zeit gemessen wird und gegebenfalls die SCAN-Liste bei verbleibender Rundendauer erneut mit D-Aufträgen aufgefüllt wird. Algorithmus V berücksichtigt die tatsächlich benötigte Bedienzeit vorausgegangener D-Aufträge aufgrund der inkrementellen Auswahl zusätzlich nach jedem D-Auftrag. Der Abschätzungfehler und die dadurch hervorgerufene geringere Leistungskapazität von Algorithmus III verstärkt sich mit abnehmender K-Last (siehe Abbildung 4.8) und kleinerer D-Auftragsgröße (siehe Abbildung 4.9). Die Algorithmen IV und V weisen bei Verwendung der konservativen Abschätzung eine geringfügig kleinere Leistungskapazität als bei exakter Berechnung auf, obwohl die Auswahl inkrementell stattfindet und der Abschätzungsfehler zumindest bei Algorithmus V mit jeder Abarbeitung eines D-Auftrages korrigiert wird. Diese Abweichung verstärkt sich, wie in den Graphen zu sehen, bei geringer werdender K-Last und kleineren Objektgrößen. Ursache hierfür ist, daß Situationen auftreten können, in denen der Abschätzungfehler erst dann derart klein wird, wenn ein D-Auftrag nicht mehr in den Ausführungsplan aufgenommen werden kann, weil der Plattenkopf die Auftragsposition bereits überfahren hat. Bei exakter Berechnung hätte das Einfügen in den Ausführungsplan rechtzeitig geschehen können, bevor der Plattenkopf die Auftragsposition erreicht.

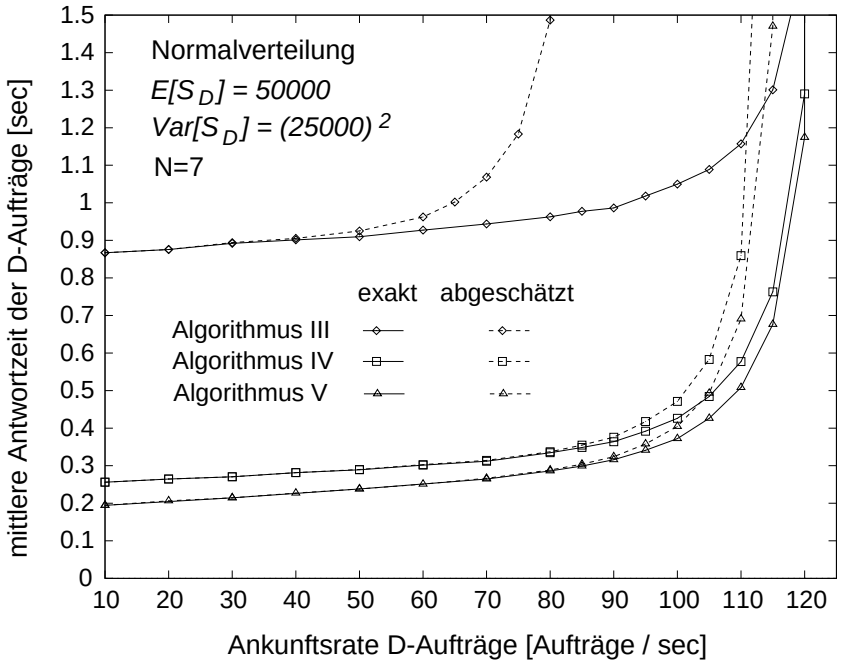



Abbildung 4.7: Auswirkung der Bedienzeitabschätzung bei hoher K-Last und großen Datenobjekten

### 4.3.6 Berechnungsaufwand

Der größte Berechnungsaufwand tritt bei Verwendung von Algorithmus V auf, der die Neuberechnung der Restbedienzeit nach jedem Auftrag durchführt. In den folgenden Experimenten soll deshalb die Zeit gemessen werden, die dieser Algorithmus pro Runde und Platte für die Aufstellung des Ausführungsplanes, d.h. für das Einfügen der Aufträge in die SCAN-Liste und für die Berechnung der Restbedienzeit benötigt. Die Messungen wurden auf einer Sun Ultra $1$ Workstation mit einem $167$Mhz UltraSPARC Prozessor durchgeführt, dabei sind zuvor keine speziellen Codeoptimierungen an den Routinen aus Abschnitt 3.3.2 vorgenommen worden. Es wurden die Datenparameter I aus Tabelle 4.1 verwendet, die K-Last wurde mit $3$ und $7$ K-Aufträgen pro Runde und Platte variiert. Die Rotationsverzögerung wurde wie im vorangegangenen Abschnitt konservativ abgeschätzt. Die gemessenen Zeiten liegen zwischen $0.5$ und $9.5$ Millisekunden (siehe Tabelle 4.2).

Das Ansteigen des Aufwands mit wachsender Ankunftsrate hat zwei Gründe. Zum einen steigt die Anzahl der in der SCAN-Liste eingefügten Aufträge und damit die Häufigkeit der Neuberechnung der Restbedienzeit, zum anderen wächst die Anzahl der Aufträge und damit die Länge der SCAN-Liste. Letzteres bedeutet, daß der Aufwand für eine Neuberechnung ebenfalls wächst, da die Bedienzeit jedes Auftrages in der SCAN-Liste bei der Berechnung der Restbedienzeit bestimmt werden muß. Dies ist ein Grund, weshalb kein linearer Zusammenhang zwischen Berechnungsaufwand und Ankunftsrate besteht. Unter hoher Auslastung mit $3$ K-Aufträgen und einer Ankunftsrate von $260$ beträgt der Anteil des Berechnungsaufwandes an der Rundendauer $1\%$. Dieser geringe Anteil ist vernachlässigbar, wenn man davon ausgeht, daß durch Codeoptimierung und schnellere Prozessoren die Berechnungen nochmals um eine Größenordnung beschleunigt werden können. Darüber hinaus können moderne E/A Schnittstellen den Datentransfer ohne CPU-Kontrolle durchführen. Hierdurch ist es möglich, die Be-

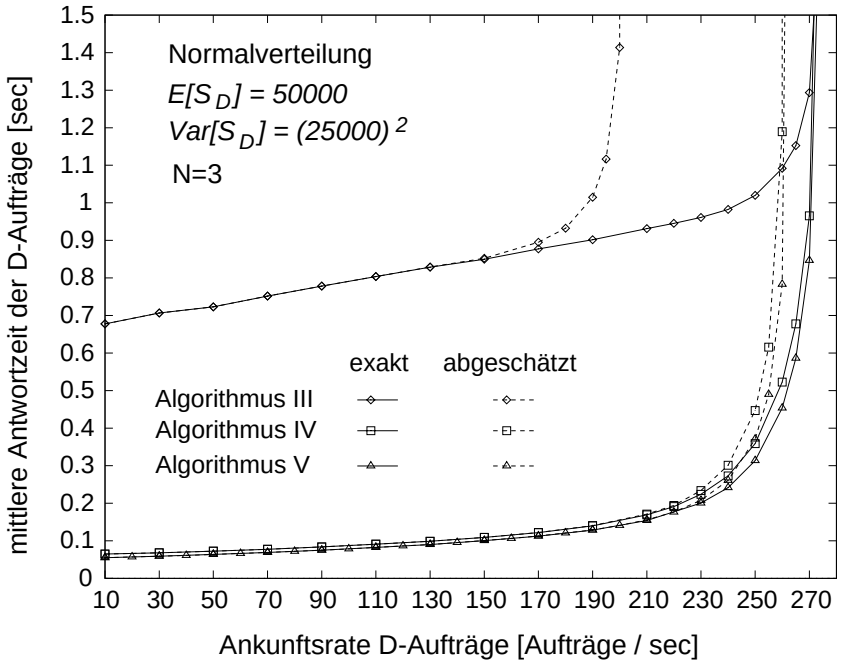



Abbildung 4.8: Auswirkung der Bedienzeitabschätzung bei niedriger K-Last und großen Datenobjekten

rechnung der Ausführungspläne parallel zum Datentransfer stattfinden zu lassen. Durch diese Parallelisierung wird die Zeit, die in einer Runde für Datenzugriffe zur Verfügung steht, nicht reduziert.

| Anzahl an K-Aufträgen pro Runde | $N = 7$ | | | | |
|---|---|---|---|---|---|
| Ankunftsrate $\lambda_{tot}$ [D-Aufträge / sec] | 30 | 60 | 90 | 100 | 110 |
| Berechnungsaufwand pro Runde und Platte [msec] | 0.5 | 1.1 | 1.8 | 2.2 | 3.1 |

| Anzahl an K-Aufträgen pro Runde | $N = 3$ | | | | |
|---|---|---|---|---|---|
| Ankunftsrate $\lambda_{tot}$ [D-Aufträge / sec] | 90 | 110 | 150 | 250 | 260 |
| Berechnungsaufwand pro Runde und Platte [msec] | 0.6 | 1.1 | 1.8 | 6.5 | 9.5 |

Tabelle 4.2: Berechnungsaufwand pro Runde und Platte bei Verwendung von Algorithmus V

## 4.4 Fazit und Empfehlung zum Systementwurf

Zusammenfassend läßt sich feststellen, daß Algorithmus V die beste Wahl für das Platten-Scheduling in einem Daten-Server mit gemischter Datenhaltung ist.

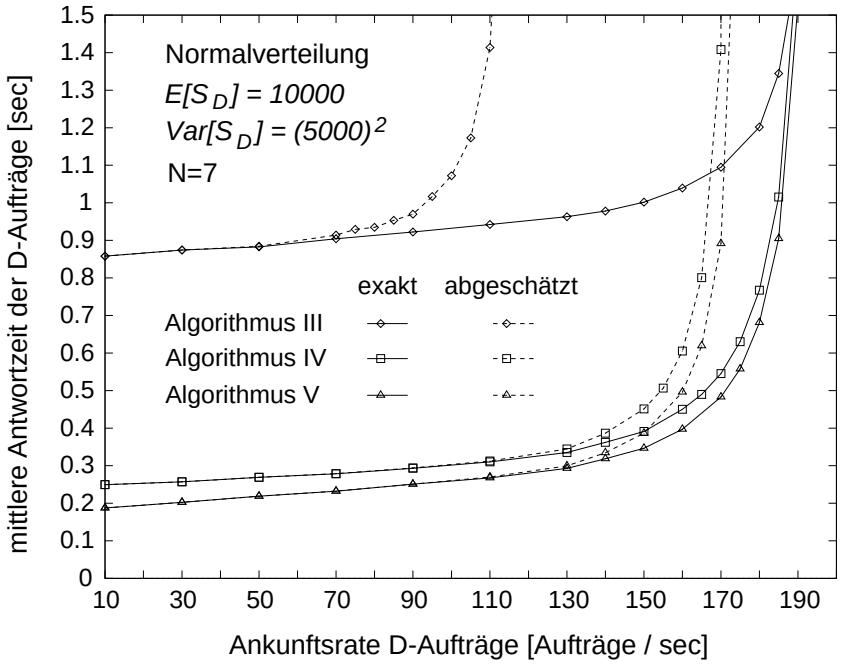



Abbildung 4.9: Auswirkung der Bedienzeitabschätzung bei hoher K-Last und kleinen Datenobjekten

Dieser Algorithmus erzielt die niedrigste mittlere Antwortzeit für D-Aufträge und kann gleichzeitig die höchste Leistungskapazität aufweisen. Die Vorteile kommen um so mehr zum Tragen, je höher die K-Last ist. In diesem Fall läßt sich durch das voll inkrementelle Einfügen von D-Aufträgen die mittlere Antwortzeit im Vergleich zu den anderen Algorithmen deutlich senken. Des weiteren schöpft der Algorithmus besonders bei sehr kleinen D-Aufträgen durch die SCAN-Strategie die Ressourcen der Platte in stärkerem Maße aus als andere Algorithmen (z.B. FCFS) und garantiert selbst am Rande der Leistungskapazität, daß die Ausführung von D-Aufträgen keinen Einfluß auf die Störungshäufigkeit der kontinuierlichen Datenströme hat.

Algorithmus V verhält sich weiterhin robust, wenn aus praktischen Erwägungen bei der Einplanung der Aufträge eine Abschätzung der Rotationsverzögerung durchgeführt werden muß. Da weiterhin nur ein geringer Zeitaufwand für die Aufstellung der Ausführungspläne benötigt wird, wird die Implementierung des Algorithmus auch unter Verwendung realer Hardware vergleichbar gute Ergebnisse wie in der Simulation liefern können.

# Teil III

# Analyse stochastischer Verhaltensmodelle

# Kapitel 5

# Stochastische Modellierung der System- und Lastkomponenten

Um Vorhersagen über das Verhalten des Daten-Servers unabhängig von Messungen oder Simulationsexperimenten treffen zu können, ist es notwendig, den Daten-Server durch ein analytisches Modell zu beschreiben. Parameter dieses analytischen Modells sind die Charakteristika der Systemkomponenten, z.B. Plattenparameter, sowie die Beschreibung der Last. Hierauf aufbauend lassen sich in den nachfolgenden Kapiteln analytische Modelle zur Vorhersage der Server-Performance erstellen, die es ermöglichen, die Service-Qualität des Daten-Servers in Form von Störungshäufigkeit und Antwortzeit quantitativ festzulegen.

Die stochastische Analyse charakterisiert System- und Lastparameter anhand ihrer Wahrscheinlichkeitsverteilung, z.B. durch deren Dichte- oder Verteilungsfunktion, mit dem Ziel, Aussagen über die Wahrscheinlichkeitsverteilung von Störungsrate und Antwortzeit zu erhalten. In den folgenden Abschnitten wird deshalb als Grundlage zunächst das Verhalten der im Modell berücksichtigten Systemkomponenten, d.h. der Magnetplatten sowie der Last, durch geeignete Dichtefunktionen modelliert.

Im Gegensatz zu einem Simulationsmodell, das im vorangegangenen Abschnitt verwendet worden ist, um die Antwortzeit der Scheduling-Strategien experimentell zu bestimmen, liegt der Vorteil der analytischen Vorgehensweise darin, daß die Bestimmung der Antwortzeit bzw. der Störungshäufigkeit in einem einzigen Berechnungsvorgang in kurzer Zeit erfolgen kann. Bei der Durchführung von Simulationexperimenten ist es hingegen notwendig, die Messungen so lange zu wiederholen, bis eine hinreichende statistische Konfidenz in den beobachteten Meßwerten erzielt worden ist. Der Aufwand hierfür kann je nach Fragestellung und Leistungsfähigkeit der verwendeten Simulationsumgebung extrem hoch sein.

In Abschnitt 5.1 wird im folgenden das Lastmodell beschrieben. Daraufhin werden die Verzögerungen, die beim Plattenzugriff entstehen, modelliert: Positionierungszeit in Abschnitt 5.2, Rotationsverzögerung in Abschnitt 5.3 sowie Transferzeit in Abschnitt 5.4.

Als Notation für die Dichtefunktion der Zufallsvariablen $X$ bzw. $V_W$ werden im folgenden die Bezeichner $f_X$ bzw. $f_W$ verwendet. Eine Verzeichnis aller verwendeten Symbole befindet sich in Anhang B.

## 5.1 Modellierung der Last

### 5.1.1 Auftragsgrößen

In den analytischen Berechnungen der nachfolgenden Kapitel wird davon ausgegangen, daß die Auftragsgröße, d.h., die Datenmenge, die pro Auftrag gelesen bzw. geschrieben wird, für D-Aufträge durch eine normalverteilte Zufallsvariable $S_D$ und für K-Aufträge durch eine gammaverteilte Zufallsvariable $S_K$ modelliert werden kann (siehe Abschnitt 4.1.2 bzw. [Ros95, KH95]). Die Dichtefunktion von $S_D$ wird durch die Parameter $\mu_S$ und $\sigma_S$ bestimmt, die dem Erwartungswert und der Standardabweichung entsprechen. Da die Auftragsgrößen nur positive Werte annehmen können, lautet die Dichtefunktion $f_{S_D}$ der Zufallsvariablen $S_D$ somit:

$$f_{S_D}(s) = \begin{cases} \frac{1}{\sigma_S\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{s-\mu_S}{\sigma_S}\right)^2} & : \quad s > 0 \\ 0 & : \quad \text{sonst} \end{cases} \tag{5.1}$$

$$\text{mit} \quad \mu_S = E[S_D] \quad \text{und} \quad (\sigma_S)^2 = Var[S_D] \tag{5.2}$$

Für typische Parameter ist der Fehler, der durch die Verwendung dieser ”abgeschnittenen“ Normalverteilung in den weiteren Berechnungen entsteht, zu vernachlässigen.

Die Dichtefunktion der gammaverteilten K-Auftragsgrößenverteilung wird durch die Parameter $\alpha_S$ und $\beta_S$ bestimmt. Die Dichtefunktion $f_{S_K}$ der Zufallsvariablen $S_K$ ist wie folgt definiert:

$$f_{S_K}(s) = \begin{cases} \frac{\alpha_S(\alpha_S s)^{\beta_S - 1} * e^{-\alpha_S s}}{\Gamma(\beta_S)} & : \quad s > 0 \\ 0 & : \quad \text{sonst} \end{cases} \tag{5.3}$$

$$\text{mit} \quad \alpha_S = \frac{E[S_K]}{Var[S_K]} \quad \text{und} \quad \beta_S = \frac{(E[S_K])^2}{Var[S_K]} \tag{5.4}$$

$\Gamma(x)$ bezeichnet in Gleichung 5.3 die Gamma-Funktion [BSM98].

### 5.1.2 Ankunftsprozesse

Die Ankünfte der K- und D-Aufträge werden durch zwei Ankunftsprozesse beschrieben. Die Ankünfte aller K-Aufträge einer Runde fallen auf den Beginn dieser Runde. Die Anzahl der Ankünfte pro Runde ist im folgenden durch eine Konstante $N$ festgelegt.

Die Ankünfte der D-Aufträge werden durch einen Poisson-Prozeß [All90, Lan92] modelliert. Die Wahrscheinlichkeit von $k$ Ankünften in einem Zeitintervall der Länge $t$ ist bei einem Poisson-Prozeß:

$$P[\text{ Anzahl der Ankünfte im Intervall } [0,t] = k\,] = e^{-\lambda t}\frac{(\lambda t)^k}{k!} \tag{5.5}$$

Der Parameter $\lambda$ ist die Rate des Prozesses und gibt die mittlere Anzahl der Ankünfte pro Zeiteinheit an.

## 5.2 Modellierung der Positionierungszeiten

Die Zufallsvariable $T_{pos}$ gibt die Positionierungszeit an, die der Plattenkopf benötigt wird, um die durch den Datenzugriff adressierte Spur anzufahren. Um die Verteilungsfunktion dieser Zufallsvariablen bestimmen zu können, wird zunächst eine Funktion aufgestellt, die für eine konkrete Positionierungsdistanz die Positionierungszeit berechnet. Daraufhin wird unter Berücksichtigung der Wahrscheinlichkeit, mit der eine bestimmte Spur bei einem Datenzugriff adressiert wird, die Verteilung der Positionierungsdistanz für die FCFS- und die SCAN-Anordnungsstrategie charakterisiert. Aus der Verteilung der Positionierungsdistanz und der Funktion zur Berechnung der konkreten Positionierungszeit läßt sich dann die Verteilung der Positionierungszeit bestimmen. Im letzten Abschnitt wird durch eine obere Schranke die Positionierungszeit für die SCAN-Anordnungsstrategie abgeschätzt.

### 5.2.1 Positionierungszeit pro Plattenzugriff

Die bereits im Simulationsmodell in Abschnitt 4.1.1 verwendete Zeitfunktion $t_{pos}$ für eine Positionierungsdistanz $d$ findet ebenfalls im analytischen Modell zur Berechnung der Positionierungzeit eines einzelnen Datenzugriffes Verwendung:

$$t_{pos}(d) = \begin{cases} P^1_{pos} + P^3_{pos} * \sqrt{d} & : \quad 0 \leq d < P^5_{pos} \\ P^2_{pos} + P^4_{pos} * d & : \quad P^5_{pos} \leq d < P_{cyls} \end{cases} \tag{5.6}$$

Die Werte der Zeitfunktion in Gleichung 5.6 werden durch die fünf *Positionierungsparameter* $P^1_{pos}, \ldots, P^5_{pos}$ bestimmt, die durch die physikalischen Eigenschaften der Platte festgelegt sind. Im folgenden soll gezeigt werden, wie diese Parameter berechnet werden können.

Bei einer kurzen Distanz wird die Positionierungszeit durch einen nichtlinearen Anteil bestimmt, da der Arm den größten Zeitanteil in der Beschleunigungsphase verbringt (siehe Abschnitt 1.2.5). Die Parameter $P^1_{pos}$ und $P^2_{pos}$ bestimmen den konstanten Zeitanteil der Abstimmphase. Der Parameter $P^5_{pos}$ beschreibt, ab welcher Distanz der linear wachsende Zeitanteil der Hochgeschwindigkeitsphase die Positionierungszeit dominiert.

Die Schätzung der Positionierungsparameter erfolgt durch Messungen wie in [WGPW95] oder durch eine Berechnung anhand von Plattenparametern [CBR94], die den Datenblättern der Magnetplatten entnommen werden können. Die benötigten Plattenparameter sind im einzelnen:

- $P_{cyls}$, die Zylinderanzahl der Platte,
- $P^{min}_{pos}$, die minimale Positionierungszeit, die bei einem Wechsel auf einen benachbarten Zylinder auftritt, d.h. $t_{pos}(1) = P^{min}_{pos}$,
- $P^{max}_{pos}$, die maximale Positionierungszeit bei der maximal möglichen Positionierungsdistanz von $P_{cyls} - 1$ Zylindern, d.h. $t_{pos}(P_{cyls} - 1) = P^{max}_{pos}$, und
- $P^{avg}_{pos}$, die durchschnittliche Positionierungszeit, die für das Anfahren einer zufällig ausgewählten Spur benötigt wird.

Der Wert von $P^5_{pos}$ beträgt aufgrund von Erfahrungswerten ungefähr ein Fünftel der maximal möglichen Positionierungsdistanz und wird deshalb auf den Wert $P^5_{pos} = \lfloor (P_{cyls}-1)/5 \rfloor$ gesetzt. Die übrigen Positionierungsparameter $P^1_{pos}, \ldots, P^4_{pos}$ lassen sich mit obigen Plattenparametern berechnen. Hierzu wird ein System mit vier Gleichungen aufgestellt:

$$\left|\begin{array}{lll} P^{min}_{pos} = & P^1_{pos} + P^3_{pos} & (a) \\ P^{max}_{pos} = & P^2_{pos} + P^4_{pos} * (P_{cyls} - 1) & (b) \\ & P^1_{pos} + P^3_{pos} * \sqrt{P^5_{pos}} = P^2_{pos} + P^4_{pos} * P^5_{pos} & (c) \\ P^{avg}_{pos} = & \sum_{d=0}^{P^5_{pos}} P[\,D = d\,] * (P^1_{pos} + P^3_{pos} * \sqrt{d}) + & \\ & \sum_{d=P^5_{pos}+1}^{P_{cyls}-1} P[\,D = d\,] * (P^2_{pos} + P^4_{pos} * d) & (d) \end{array}\right| \tag{5.7}$$

Die Gleichung (c) des Systems 5.7 garantiert die Stetigkeit der Funktion $t_{pos}$ an der Stelle $P^5_{pos}$. In der Gleichung (d) bezeichnet $P[D = d]$ die Wahrscheinlichkeit, mit der bei der Messung der durchschnittlichen Positionierungszeit $P^{avg}_{pos}$ eine Positionierungsdistanz von $d$ Zylindern aufgetreten ist. Diese Wahrscheinlichkeit kann während der Messung protokolliert oder unter Annahme eines gleichverteilten Zugriffs auf alle Zylinder der Platte berechnet werden. Die Berechnung von $P[D = d]$ wird im nachfolgenden Abschnitt 5.2.2 näher erläutert.

Das Lösen des Gleichungssystems 5.7 unter Verwendung der Herstellerdaten aus Tabelle 5.1 führt bei einer Magnetplatte des Typs Quantum Viking zu folgender Zeitfunktion $t_{pos}$, die in Abhängigkeit der Positionierungsdistanz $d$ die Positionierungszeit berechnet. Es gilt:

$$t_{pos}(d) = \left\{\begin{array}{rcl} 1.868 * 10^{-3} + 1.316 * 10^{-4}\sqrt{d}\ \text{[sec]} & : & 0 \leq d < 1344 \\ 3.865 * 10^{-3} + 2.104 * 10^{-6} d\ \text{[sec]} & : & 1344 \leq d < 6720 \end{array}\right. \tag{5.8}$$

Die zu $t_{pos}$ gehörige inverse Funktion $t^{-1}_{pos}$, die einen Zeitwert $t$ in eine Positionierungsdistanz umrechnet, ergibt sich mit obigen Werten zu:

$$t^{-1}_{pos}(t) = \left\{\begin{array}{rcl} \left(\frac{t - 1.868 * 10^{-3}}{1.316 * 10^{-4}}\right)^2 & : & t < 6.69 * 10^{-3} \\ \frac{t - 3.865 * 10^{-3}}{2.104 * 10^{-6}} & : & t \geq 6.69 * 10^{-3} \end{array}\right. \tag{5.9}$$

Die erste Ableitung $(t^{-1}_{pos})'$ der inversen Zeitfunktion $t^{-1}_{pos}$ beschreibt die Geschwindigkeit mit der sich der Plattenkopf bewegt. Aus der Gleichung 5.9 erhält man durch Ableitung:

$$(t^{-1}_{pos})'(t) = \left\{\begin{array}{rcl} \frac{2(t - 1.868 * 10^{-3})}{1.316 * 10^{-4}} & : & t < 6.69 * 10^{-3} \\ \text{undefiniert} & : & t = 6.69 * 10^{-3} \\ \frac{1}{2.104 * 10^{-6}} & : & t > 6.69 * 10^{-3} \end{array}\right. \tag{5.10}$$

An der Stelle $t_{pos}(1344)$ ist die Funktion $t^{-1}_{pos}$ nicht differenzierbar und somit $(t^{-1}_{pos})'$ nicht definiert. Damit die Funktion $t_{pos}$ und damit auch $t^{-1}_{pos}$ an allen Stellen differenzierbar wird, ließe

| **Konfigurationsdaten** | | |
|---|---|---|
| Anzahl der Scheiben | $P_{plat}$ | $2$ |
| Anzahl der Köpfe | $P_{heads}$ | $4$ |
| Bytes pro Block | | $512$ |
| Zylinderanzahl | $P_{cyls}$ | $6720$ |
| Anzahl der Zonen | $P_{zones}$ | $15$ |
| Speicherkapazität der innersten Spur | $P_{cap}^{min}$ | $114$ Blöcke = $58368$ Bytes |
| Speicherkapazität der äußersten Spur | $P_{cap}^{max}$ | $187$ Blöcke = $95744$ Bytes |
| Speicherkapazität einer Spur der Zone $i$ | $P_{cap}^{i}$ | Annahme: linearer Zuwachs |
| Anzahl der Spuren in Zone $i$ | $P_{cpz}^{i}$ | Annahme: $P_{cpz}^{i} = P_{cyls}/P_{zones}$ |
| Gesamtspeicherkapazität | $P_{cap}^{tot}$ | $2.03$ GBytes |
| **Leistungsdaten** | | |
| Positionierungszeit über einen Zylinder | $P_{pos}^{min}$ | $2$ msec |
| durchschnittliche Positionierungzeit | $P_{pos}^{avg}$ | $8.5$ msec |
| Positionierungszeit über alle Zylinder | $P_{pos}^{max}$ | $18$ msec |
| Rotationsgeschwindigkeit | | $7200$ U/min |
| maximale Rotationsverzögerung | $P_{rot}$ | $8.33$ msec |
| maximale Transferrate | $P_{rate}^{max}$ | $P_{cap}^{max}/P_{rot} \approx 10.96$ MBytes/sec |
| minimale Transferrate | $P_{rate}^{min}$ | $P_{cap}^{min}/P_{rot} \approx 6.68$ MBytes/sec |

Tabelle 5.1: Herstellerangaben über Plattenparameter einer Quantum Viking 2.1 Magnetplatte

sich theoretisch durch das Hinzufügen einer weiteren Gleichung zum Gleichungssystem in 5.7 diese Bedingung berücksichtigen:

$$\left.\frac{d(P_{pos}^1 + P_{pos}^3 * \sqrt{x})}{dx}\right|_{x=P_{pos}^5} = \frac{P_{pos}^3}{2 * \sqrt{P_{pos}^5}} = P_{pos}^4 = \left.\frac{d(P_{pos}^2 + P_{pos}^4 * x)}{dx}\right|_{x=P_{pos}^5}$$

Der Parameter $P_{pos}^5$ müßte in diesem Fall als weiterer Parameter durch das erweiterte Gleichungssystem bestimmt werden. Für die betrachteten Parameter der Magnetplatte aus Tabelle 5.1, existiert für das erweiterte Gleichungssystem allerdings keine Lösung, so daß im folgenden die undefinierte Stelle der Funktion $(t_{pos}^{-1})'$ vernachlässigt werden muß.

### 5.2.2 Verteilung der Positionierungsdistanz bei der FCFS-Anordnung

Als zweiter Schritt auf dem Weg zur Charakterisierung der Positionierungszeit erfolgt die Berechnung der Verteilung der Positionierungsdistanz. Hierzu wird zunächst die Wahrscheinlichkeit berechnet, daß ein Zylinder bei einem Datenzugriff betroffen ist. Im folgenden wird stets angenommen, daß die Daten gleichmäßig über alle Zylinder verteilt sind und daß der Datenzugriff alle Daten gleich wahrscheinlich betrifft.

**Auswahlwahrscheinlichkeit eines Zylinders**

Die Wahrscheinlichkeit, einen Zylinder auszuwählen, hängt von der Speicherkapazität dieses Zylinders ab. Je größer die Speicherkapazität desto größer die Anzahl der Daten auf diesem Zylinder und desto größer die Wahrscheinlichkeit, daß der Zugriff auf ein Datum dieses Zylinders stattfindet. Die Wahrscheinlichkeit, daß der Zylinder $z$ in Zone $i$ einer Mehr-Zonen-Platte ausgewählt wird, läßt sich mit der Kapazität einer Spur $P^i_{cap}$ dieser Zone, der Anzahl der Plattenköpfe $P_{heads}$ (=Anzahl der Oberflächen), der Anzahl der Zylinder $P^i_{cpz}$ einer Zone $i$ sowie der Anzahl der Zonen $P_{zones}$ der Platte beschreiben. Vorausgesetzt wird, daß die Spuren eines Zylinders die gleiche Speicherkapazität aufweisen. Die Zufallsvariable der Zylinderauswahl sei mit $Z$ und dessen Dichtefunktion sei mit $f_Z$ bezeichnet. Es gilt:

$$\begin{aligned} f_Z(z) &= P[\,Z = z\,] = P[\,\text{Auswahl von Zylinder } z\,] \\ &= \frac{\text{Speicherkapazität Zylinder } z \text{ in Zone } i}{\text{Gesamtspeicherkapazität der Platte}} = \frac{P_{heads} * P^i_{cap}}{P_{heads} * \sum_{k=1}^{P_{zones}} P^k_{cap} * P^k_{cpz}} \\ &= \frac{P^i_{cap}}{\sum_{k=1}^{P_{zones}} P^k_{cap} * P^k_{cpz}} \quad \text{mit} \quad \sum_{j=1}^{i-1} P^j_{cpz} < z \leq \sum_{j=1}^{i} P^j_{cpz} \end{aligned} \tag{5.11}$$

Die Bedingung in Gleichung 5.11 stellt sicher, daß der betrachtete Zylinder $z$ sich in der $i$-ten Zone befindet.

Die Dichtefunktion einer Ein-Zonen-Platte ergibt sich durch die Betrachtung eines Sonderfalls von Gleichung 5.11. Da bei Ein-Zonen-Platten die Speicherkapazität einer Spur konstant ist und nur eine Zone existiert, d.h $P_{zones} = 1$ und $P^1_{cpz} = P_{cyls}$, ist die Auswahlverteilung der Zylinder gleichverteilt mit der Dichtefunktion $f_{Z,SZ}$:

$$f_{Z,SZ}(z) = \frac{1}{P_{cyls}} \quad \text{mit} \quad 1 \leq z \leq P_{cyls} \tag{5.12}$$

Um eine konkrete Dichtefunktion $f_Z$ der Auswahlverteilung einer Mehr-Zonen-Platte zu bestimmen, ist es zunächst notwendig, die Speicherkapazität einer Spur $P^i_{cap}$ auf Basis der Herstellerdaten zu bestimmen. Im folgenden wird eine Mehr-Zonen-Platte mit einer minimalen und einer maximalen Speicherkapazität von $P^{min}_{cap}$ bzw. $P^{max}_{cap}$ pro Spur angenommen. Unter der Annahme, daß die Speicherkapazität von Zone zu Zone linear wächst, läßt sich die Speicherkapazität $P^i_{cap}$ einer Spur $z$ in der $i$-ten Zone wie folgt berechnen:

$$P^i_{cap} = P^{min}_{cap} + \frac{P^{max}_{cap} - P^{min}_{cap}}{P_{zones} - 1} * (i - 1) \qquad \text{für } 1 \leq i \leq P_{zones} \tag{5.13}$$

In obiger Gleichung wurde auf das Runden auf einen ganzzahligen Bytewert aufgrund einer besseren mathematischen Handhabbarkeit bei nur geringen Unterschieden verzichtet. Die Gesamtspeicherkapazität der Platte $P^{tot}_{cap}$ ergibt bei dieser Vereinfachung und der Annahme, daß in allen Zonen die gleiche Anzahl von Spuren zusammengefaßt ist, mit $P^i_{cpz} = P_{cyls}/P_{zones}$:

$$P^{tot}_{cap} = P_{heads} * \sum_{i=1}^{P_{zones}} P^i_{cap} * \frac{P_{cyls}}{P_{zones}} = P_{heads} * P_{cyls} * \frac{P^{min}_{cap} + P^{max}_{cap}}{2} \tag{5.14}$$

Für die Daten aus Tabelle 5.1 errechnet sich unter Verwendung von Gleichung 5.14 eine Gesamtspeicherkapazität von $1.93$ GBytes. Der vom Hersteller angegebene Wert von $2.03$ GBytes ist geringfügig größer, was darauf schließen läßt, daß die Kapazität der äußeren Zonen größer ist als angenommen. Dies kann z.B. durch eine höhere Zylinderanzahl $P^i_{cpz}$ oder durch eine größere Speicherkapazität der Spuren $P^i_{cap}$ in den äußeren Zonen hervorgerufen werden. Für die weiteren Berechnungen wird die geringe Abweichung ignoriert.

Werden die Gleichungen 5.14 und 5.13 in Gleichung 5.11 eingesetzt, so erhält man unter den genannten Annahmen die Dichtefunktion $f_{Z,MZ}$ der Auswahlwahrscheinlichkeit einer Mehr-Zonen-Platte:

$$f_{Z,MZ}(z) = 2 * \frac{P^{min}_{cap} + \frac{P^{max}_{cap} - P^{min}_{cap}}{P_{zones}-1} * \left( \lceil z * \frac{P_{zones}}{P_{cyls}} \rceil - 1 \right)}{(P^{min}_{cap} + P^{max}_{cap}) P_{cyls}} \tag{5.15}$$

**Wahrscheinlichkeitsverteilung der Positionierungsdistanz**

Die Positionierungsdistanz ist der Abstand, gemessen in Zylindern, den der Plattenkopf zwischen zwei Aufträgen zurücklegen muß. Ihre Wahrscheinlichkeitsverteilung ist abhängig von der Auswahlwahrscheinlichkeit der Zylinder und der Anordnung der Aufträge. In diesem Abschnitt soll gezeigt werden, wie die Dichtefunktion sowohl für die FCFS- als auch für die SCAN-Anordnungstrategie berechnet werden kann.

Die Kopfbewegung eines Datenzugriff bei der FCFS-Anordnungstrategie der Aufträge läßt sich durch ein Zweiertupel $(Z_1, Z_2)$ beschreiben. Die Zufallsvariable $Z_1$ stellt die Ausgangsposition des Plattenkopfes und die Zufallsvariable $Z_2$ die Zielposition dar. Beide Zufallsvariablen werden als voneinander unabhängig postuliert und sind somit durch die Dichtefunktionen $f_{Z,SZ}$ bzw. $f_{Z,MZ}$ der Auswahlverteilung in Gleichung 5.12 bzw. 5.15 charakterisiert. Die Wahrscheinlichkeit für eine bestimmte Positionierungsdistanz ungleich Null ergibt sich durch das Aufsummieren der Wahrscheinlichkeiten, mit denen die verschiedenen Zweiertupel gebildet werden können:

$$\begin{aligned} & P[\text{ Positionierungsdistanz = } d \text{ Zylinder }] \qquad (5.16) \\ &= \sum_{i=1}^{P_{cyls}-d} P[\, Z_2 = i + d \,|\, Z_1 = i \,] * P[\, Z_1 = i \,] \\ &\qquad + \sum_{i=1+d}^{P_{cyls}} P[\, Z_2 = i - d \,|\, Z_2 = i \,] * P[\, Z_1 = i \,] \\ &= \sum_{i=1}^{P_{cyls}-d} P[\, Z_2 = i + d \,] * P[\, Z_1 = i \,] + \sum_{i=1+d}^{P_{cyls}} P[\, Z_2 = i - d \,] * P[\, Z_1 = i \,] \end{aligned}$$

Für den Fall einer Positionierungsdistanz gleich Null gilt:

$$\begin{aligned} & P[\text{ Positionierungsdistanz = } 0 \text{ Zylinder }] \qquad (5.17) \\ &= \sum_{i=1}^{P_{cyls}} P[\, Z_2 = i \,|\, Z_1 = i \,] * P[\, Z_1 = i \,] = \sum_{i=1}^{P_{cyls}} P[\, Z_2 = i \,] * P[\, Z_1 = i \,] \end{aligned}$$

Faßt man die beiden obigen Gleichungen zusammen, so erhält man die Dichtefunktion $f_D^{fcfs}$ für die Verteilung der Positionierungsdistanz bei einer FCFS-Anordnung der Aufträge. $D$ sei hierbei die Zufallsvariable der Positionierungsdistanz:

$$\begin{aligned} f_D^{fcfs}(d) &= P[\,D = d\,] = P[\,\text{Positionierungsdistanz} = d\,] \\ &= \begin{cases} \sum_{i=1}^{P_{cyls}} [f_Z(i)]^2 & : \quad d = 0 \\ 2 * \sum_{i=1}^{P_{cyls}-d} f_Z(i+d) * f_Z(i) & : \quad 0 < d < P_{cyls} \\ 0 & : \quad \text{sonst} \end{cases} \end{aligned} \tag{5.18}$$

Für eine Ein-Zonen-Platte vereinfacht sich durch Einsetzen der Dichtefunktion $f_{Z,SZ}$ aus Gleichung 5.12 die Gleichung 5.18 wie folgt:

$$f_{D,SZ}^{fcfs}(d) = \begin{cases} \frac{1}{P_{cyls}} & : \quad d = 0 \\ 2 * \frac{P_{cyls}-d}{P_{cyls}^2} & : \quad 0 < d < P_{cyls} \\ 0 & : \quad \text{sonst} \end{cases} \tag{5.19}$$

Für eine Mehr-Zonen-Platte erfolgt die Berechnung der Dichtefunktion $f_{D,MZ}^{fcfs}$ unter Verwendung der Gleichung 5.15 analog. Mit Hilfe eines Programms zur symbolischen Berechnung und Vereinfachung von Gleichungen, wie z.B. Maple [Kof96] oder Mathematica [Wol96], ergibt sich mit $P_{zones} = P_{cyls}$:

- für $(d \neq 0)$

$$\begin{aligned} f_{D,MZ}^{fcfs}(d) = 4/3 * \ ( \ & (P_{cap}^{min})^2 * P_{cyls} + (P_{cap}^{max})^2 * P_{cyls} - 3 * (P_{cap}^{min})^2 * (P_{cyls})^2 \\ & - 3 * (P_{cap}^{max})^2 * (P_{cyls})^2 + 2 * (P_{cap}^{max})^2 * (P_{cyls})^3 + 2 * (P_{cap}^{min})^2 * (P_{cyls})^3 \\ & - (P_{cap}^{max})^2 * d - (P_{cap}^{min})^2 * d + (P_{cap}^{max})^2 * d^3 + (P_{cap}^{min})^2 * d^3 \\ & + 3 * (P_{cap}^{min})^2 * d * P_{cyls} + 3 * (P_{cap}^{max})^2 * d * P_{cyls} \\ & - 3 * (P_{cap}^{max})^2 * d * (P_{cyls})^2 - 3 * (P_{cap}^{min})^2 * d * (P_{cyls})^2 \\ & - 4 * P_{cap}^{min} * P_{cap}^{max} * d - 2 * P_{cap}^{min} * P_{cap}^{max} * d^3 \\ & + 4 * P_{cap}^{min} * P_{cap}^{max} * P_{cyls} - 6 * P_{cap}^{min} * P_{cap}^{max} * (P_{cyls})^2 \\ & + 2 * P_{cap}^{min} * P_{cap}^{max} * (P_{cyls})^3 + 6 * P_{cap}^{min} * P_{cap}^{max} * d * P_{cyls} \ ) \ / \\ & ((P_{cap}^{min} + P_{cap}^{max})^2 * P_{cyls}^2 * (P_{cyls} - 1)^2) \end{aligned} \tag{5.20}$$

- für $(d = 0)$

$$\begin{aligned} f_{D,MZ}^{fcfs}(0) = 2/3 * \ ( \ & -(P_{cap}^{min})^2 - (P_{cap}^{max})^2 + 2 * (P_{cap}^{max})^2 * P_{cyls} \\ & + 2 * (P_{cap}^{min})^2 * P_{cyls} - 4 * P_{cap}^{min} * P_{cap}^{max} 2 * P_{cap}^{min} * P_{cap}^{max} * P_{cyls} \ ) \ / \\ & (P_{cyls} * (P_{cap}^{min} + P_{cap}^{max})^2 * (P_{cyls} - 1)) \end{aligned} \tag{5.21}$$

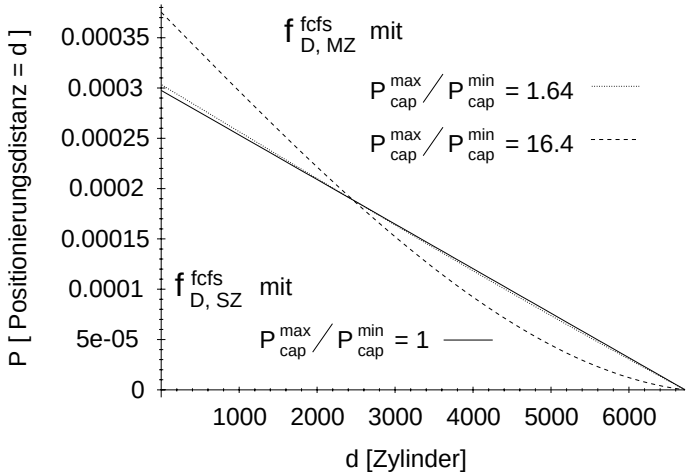



Abbildung 5.1: Vergleich der Dichtefunktion $f^{fcfs}_{D,MZ}$ und $f^{fcfs}_{D,SZ}$ bei unterschiedlichen Plattenparametern

In Abbildung 5.1 sind die Dichtefunktionen der Positionierungsdistanz für Ein- und Mehr-Zonen-Platten aufgetragen. Werden die Parameter aus Tabelle 5.1 mit einem Verhältnis von $P^{max}_{cap}/P^{min}_{cap} = 1.64$ für die Mehr-Zonen-Platte und einem Verhältnis von $P^{max}_{cap}/P^{min}_{cap} = 1$ für die Ein-Zonen-Platte verwendet, so unterscheiden sich die beiden Dichtefunktionen $f^{fcfs}_{D,MZ}$ und $f^{fcfs}_{D,SZ}$ nur geringfügig voneinander. Erst die Vergrößerung des Kapazitätsverhältnisses zwischen der äußeren und der inneren Spur der Mehr-Zonen-Platte um den Faktor $10$ mit $P^{max}_{cap}/P^{min}_{cap} = 16.4$, läßt Unterschiede in der Verteilung der Positionierungsdistanz zwischen beiden Plattentypen erkennbar werden. Für Mehr-Zonen-Platten nimmt die Wahrscheinlichkeit kürzerer Positionierungsdistanzen zu und die Wahrscheinlichkeit länger Positionierungsdistanzen nimmt ab. Aufgrund der größeren Speicherkapazität konzentrieren sich die Datenzugriffe bei Mehr-Zonen-Platten auf die äußeren Zylinder, und es ist wahrscheinlicher, daß der Kopf für zwei aufeinanderfolgende Datenzugriffe eine kurze Distanz auf dem äußeren Rand einer solchen Platte zurücklegen muß. Hieraus kann man ableiten, daß unter den gegebenen Plattenparametern die Modellierung einer Ein-Zonen-Platte ausreicht, da hiermit die Verteilung der Positionierungsdistanz einer Mehr-Zonen-Platte hinreichend genau approximiert wird.

### 5.2.3 Verteilung der Positionierungsdistanz bei der SCAN-Anordnung

Da bei der SCAN-Anordnungsstrategie vor der Ausführung der Aufträge deren Ausführungsreihenfolge neu bestimmt wird, ist die Dichtefunktion der Positionierungsdistanz in den Gleichungen 5.16 bzw. 5.17 nicht anwendbar. So ist z.B. die Dichte von der Anzahl der Aufträge abhängig, die durch die SCAN-Anordnungstrategie sortiert werden. Je größer die Anzahl der Aufträge, desto wahrscheinlicher ist es, kürzere Distanzen zwischen den sortierten Aufträgen zu erhalten. Im folgenden wird davon ausgegangen, daß die Daten, wie bei einer Ein-Zonen-Platte, gleichmäßig über alle Zylinder verteilt werden und daß durch einen gleichverteilten Zugriff auf diese Daten die Wahrscheinlichkeit zur Auswahl eines Zylinders für alle Zylinder den gleichen Wert hat. Im Gegensatz zum SCAN-Algorithmus, der im ersten Teil dieser Arbeit vorgestellt worden ist, wird im folgenden zwecks einfacherer Modellierung davon ausgegangen, daß der SCAN immer vollständig ausgeführt wird, so daß nach Ausführung des letzten Auftrages der Plattenkopf auf den ersten oder letzten Zylinder positioniert wird. Der nachfolgende

SCAN startet folglich immer vom innersten bzw. äußersten Zylinder, anstatt, wie bei den im vorangegangenen Kapitel beschriebenen Algorithmen, ausgehend von der Position des letzten Auftrages den SCAN zu beginnen.

Bei der Berechnung der Dichtefunktion für die Positionierungsdistanz wird im folgenden unterschieden, ob die Distanz zwischen der Startposition des Plattenkopfes und der Position des ersten Auftrages oder ob die Distanz zwischen zwei Aufträgen betrachtet werden soll. Bei ersterem liegt die Startposition des Plattenkopfes mit Zylinder $1$ fest, bei letzterem variiert die Startposition ebenso wie die Zielposition. Aus diesem Grund werden die Dichtefunktionen $f_{D,SZ}^{scan_1}$ mit Zylinder $1$ als Startposition und die Dichtefunktion $f_{D,SZ}^{scan_X}$ mit der vorangegangen Auftragsposition als Startposition betrachtet.

Als erstes soll die Dichtefunktion $f_{D,SZ}^{scan_1}$ analysiert werden. Hierzu betrachtet man die Positionierungsdistanz zwischen dem ersten Zylinder, auf dem sich zu Beginn der Plattenkopf befindet, und dem ersten Auftrag. Es wird angenommen, daß insgesamt $n$ Aufträge in einer Runde durch den SCAN-Algorithmus bearbeitet werden. Sei durch $(r_1, r_2, \ldots, r_n)$ ein Tupel mit den aufsteigend geordneten, zufällig gleichverteilt ermittelten Zylindernummern der Aufträge $r_i \in \{1, \ldots, P_{cyls}\}$ gegeben.

Die Anzahl der möglichen Tupel, bei denen die Distanz zwischen der Startposition des SCANs auf dem ersten Zylinder und dem Zylinder $r_1$ des ersten zu bearbeitenden Auftrages $d$ Zylinder beträgt, läßt sich kombinatorisch berechnen. Da die Position des ersten Auftrags durch die Zylindernummer $(d+1)$ festgelegt ist, gibt es für die verbleibenden $(n-1)$ Aufträge

$$\binom{P_{cyls} - d - 1}{n-1} \tag{5.22}$$

Möglichkeiten, die Aufträge den Zylindern zuzuweisen und hierbei ein Tupel in der Form $(d+1, r_2, \ldots, r_n)$ zu bilden. Voraussetzung hierfür ist, daß alle Aufträge jeweils verschiedenen Zylindern zugeordnet werden. Diese Annahme hat in der Gesamtrechnung nur geringe Ungenauigkeiten zur Folge, da die Wahrscheinlichkeit, daß zwei Aufträge den selben Zylinder betreffen, für typische Werte, z.B. für $P_{cyls} = 6720$ mit $1/P_{cyls} = 1.488 * 10^{-4}$, relativ gering ist.

Betrachtet man das unsortierte Tupel $(r_{i_1}, \ldots, r_{i_n})$, wie es vor der Sortierung des SCAN-Algorithmus durch die zufällige Auswahl der Positionen auf der Platte erzeugt wird, so gibt es für jede der in 5.22 angegebenen sortierten Möglichkeiten $n!$ Permutationen, dieses unsortierte Tupel zu erzeugen.

Die Gesamtzahl der Möglichkeiten, $n$ Aufträge auf $P_{cyls}$ Zylinder zu verteilen und ein Tupel $(r_{i_1}, \ldots, r_{i_n})$ zu bilden, ohne daß zwei Aufträge den selben Zylinder betreffen, entspricht der Anzahl der Variationen von $P_{cyls}$ Zylindern zur $n$-ten Klasse ohne Wiederholung:

$$P_{cyls} * (P_{cyls} - 1) * (P_{cyls} - 2) * \ldots * (P_{cyls} - n + 1) = \frac{P_{cyls}!}{(P_{cyls} - n)!} \tag{5.23}$$

Unter Verwendung der Gleichungen 5.22 und 5.23 läßt sich die Dichtefunktion $f_{D,SZ}^{scan_1}$ der Positionierungsdistanz für die erste Plattenkopfbewegung ausgehend von der Startposition auf dem ersten Zylinder bestimmen. Es wird der Quotient aus der Anzahl der günstigen Fälle und der Anzahl aller möglichen Fälle gebildet. Die günstigen Fälle umfassen alle Tupel $(r_{i_1}, \ldots, r_{i_n})$,

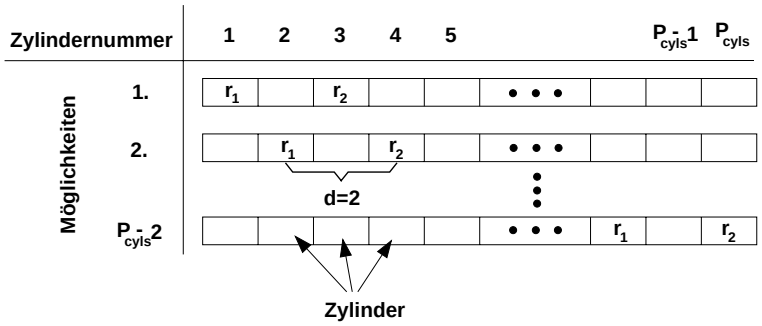



Abbildung 5.2: Anzahl der Plazierungsmöglichkeiten bei 2 Aufträgen im SCAN und einer Positionierungsdistanz von 2 Zylindern

aus denen nach der Sortierung der Aufträge ein Tupel der Form $(d+1, r_2, \ldots, r_n)$ entsteht. Die möglichen Fälle umfassen alle Tupel der Form $(r_{i_1}, \ldots, r_{i_n})$. Somit gilt:

$$
\begin{aligned}
f^{scan_1}_{D,SZ}(n,d) &= P[\,\text{Distanz für erste Positionierung = } d \text{ Zylinder} \mid n \text{ Aufträge in Bedienung}\,] \\
&= \binom{P_{cyls}-d-1}{n-1} * n! \Big/ \frac{P_{cyls}!}{(P_{cyls}-n)!} \qquad (5.24) \\
&= \frac{n * (P_{cyls}-d-1)! * (P_{cyls}-n)!}{(P_{cyls}-n-d)! * P_{cyls}!} \qquad \text{für } n > 1 \qquad (5.25)
\end{aligned}
$$

Die Gleichung 5.25 ist das Ergebnis einer Vereinfachung von Gleichung 5.24 durch Maple, die sich aufgrund der Eigenschaft der Gamma-Funktion für ganzzahlige Argumente ergibt.

Die Gleichung 5.25 gilt auch dann, wenn die Startposition der Kopfes sich auf dem letzten Zylinder $P_{cyls}$ befindet und somit die entgegengesetzte Kopfbewegung in Richtung von Zylinder 1 durchgeführt wird. Theoretisch könnte die obige Betrachtung auch für Mehr-Zonen-Platten durchgeführt werden. Hierbei müßte die Auswahlwahrscheinlichkeit eines Zylinders berücksichtigt werden, was eine aufwendige Fallunterscheidung in der obigen Methode nach sich ziehen würde.

Die Verteilungsfunktion für die Positionierungsdistanz zwischen zwei beliebigen in der Anordnungsreihenfolge benachbarten Aufträgen läßt sich basierend auf dem oben beschriebenen Sonderfall in analoger Weise herleiten. Hierzu werden die Positionen $r_1$ und $r_2$ des nach Zylindernummern sortierten Tupels $(r_1, r_2, \ldots, r_n)$ betrachtet. Bei einem Abstand von $d$ Zylindern und $n$ Aufträgen gibt es $(P_{cyls} - (n+d) + 2)$ Möglichkeiten die Positionen $r_1$ und $r_2$ auszuwählen, vorausgesetzt, daß wiederum alle Aufträge verschiedene Zylinder betreffen.

Die verbleibenden $n-2$ Positionen $r_3$ bis $r_n$ müssen Werte größer als $r_2$ annehmen. Es gibt hierfür

$$\binom{P_{cyls}-d-i}{n-2} \qquad (5.26)$$

Kombinationen für ein sortiertes Tupel, wenn $r_1$ die Position $i$ annimmt. Betrachtet man analog zum obigen Sonderfall das unsortierte Tupel $(r_{i_1}, \ldots, r_{i_n})$, wie es vor der Sortierung durch den SCAN-Algorithmus gegeben ist, so gibt es für jedes sortierte Tupel $n!$ Permutationen, ein solches unsortiertes Tupel zu bilden.

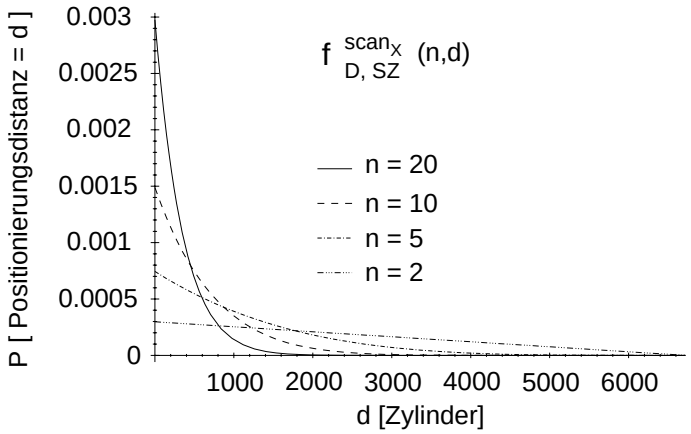



Abbildung 5.3: Dichtefunktion $f_{D,SZ}^{scan_X}$ der Positionierungsdistanz beim SCAN-Algorithmus

Die Gesamtzahl der Möglichkeiten, $n$ Aufträge auf $P_{cyls}$ Zylinder zu plazieren, unter der Bedingung, daß die Distanz zwischen $r_1$ und $r_2$ $d$ Zylinder beträgt, ist somit die Summe aller oben beschriebenen Möglichkeiten

$$\sum_{i=1}^{P_{cyls}-(n+d)+2} \binom{P_{cyls}-d-i}{n-2} * n! \tag{5.27}$$

Die Gesamtzahl der Möglichkeiten $n$ Aufträge auf $P_{cyls}$ zu verteilen, ist bereits in Gleichung 5.23 angegeben worden. Die Dichtefunktion ergibt sich wiederum durch eine Betrachtung des Verhältnisses zwischen günstigen Fällen und den insgesamt möglichen Fällen. Da die Wahrscheinlichkeit einer Positionierungsdistanz von $d$ zwischen zwei benachbarten Aufträgen für alle Paare benachbarter Aufträge identisch sein muß, gelten die obigen Betrachtungen ebenso für $r_2$ und $r_3$ oder auch für $r_{n-1}$ und $r_n$. Aus diesem Überlegungen ergibt sich somit die Dichtefunktion $f_{D,SZ}^{scan_X}$ der Positionierungsdistanz zwischen zwei Aufträgen bei insgesamt $n$ Aufträgen zu:

$$\begin{aligned} f_{D,SZ}^{scan_X}(n,d) &= P[\text{Distanz zwischen zwei Aufträgen=}d\text{ Zylinder} \mid n\text{ Aufträge in Bedienung}] \\ &= \sum_{i=1}^{P_{cyls}-(n+d)+2} \binom{P_{cyls}-d-i}{n-2} * n! \Big/ \frac{P_{cyls}!}{(P_{cyls}-n)!} && (5.28) \\ &= \frac{n*(P_{cyls}-d)!*(P_{cyls}-n)!}{(P_{cyls}-n-d+1)!*P_{cyls}!} \qquad \text{für } n>1 && (5.29) \end{aligned}$$

Die Gleichung 5.29 geht aus einer Vereinfachung von Gleichung 5.28 durch Maple hervor.

In Abbildung 5.3 ist der Graph der Funktion $f_{D,SZ}^{scan_X}$ für eine unterschiedliche Anzahl $n$ von Aufträgen im SCAN zu sehen. Wie bereits vermutet, nimmt bei wachsender Anzahl der Aufträge pro SCAN die Wahrscheinlichkeit kurzer Positionierungsdistanzen zu und die Wahrscheinlichkeit langer Distanzen ab.

Ein Vergleich zwischen den Dichtefunktionen $f_{D,SZ}^{scan_X}$ und $f_{D,SZ}^{scan_1}$ ist in Abbildung 5.4 zu sehen. Mit steigender Positionierungsdistanz und mit steigender Auftragsanzahl $n$ erhöht sich die relative Abweichung zwischen beiden Dichtefunktionen. Die absolute Abweichung sinkt

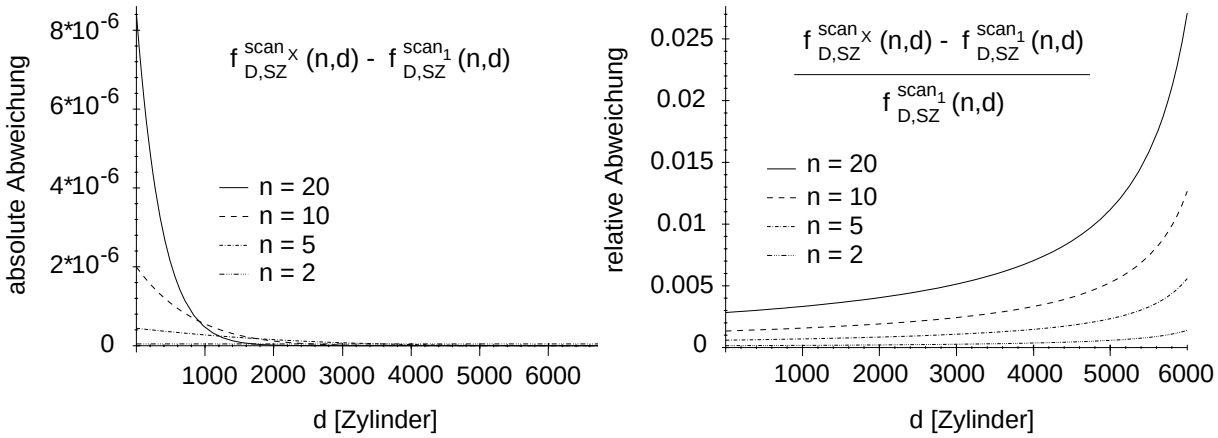



Abbildung 5.4: absolute und relative Abweichung zwischen den beiden Dichtefunktionen $f_{D,SZ}^{scan_1}$ und $f_{D,SZ}^{scan_X}$

bei steigender Positionierungsdistanz auf sehr geringe Werte. Aus diesem Grund kann in den folgenden Abschnitten mit nur einer Dichtefunktion gearbeitet und die Dichtefunktion $f_{D,SZ}^{scan_1}$ durch die Dichtefunktion $f_{D,SZ}^{scan_X}$ approximiert werden.

### 5.2.4 Berechnung der Dichte der Positionierungszeit

Die Dichtefunktion der Positionierungszeit läßt sich mit Hilfe der inversen Zeitfunktion $t_{pos}^{-1}$ aus Gleichung 5.9 und der allgemeinen Dichtefunktion $f_D$ der Positionierungsdistanz bestimmen. Hierzu betrachtet man zunächst die Verteilungsfunktion $F_D$ der Positionierungsdistanz $D$. Diese Funktion ist als Summe über der Dichtefunktion $f_D$ definiert und wird im folgenden durch eine Integration approximiert:

$$F_D(d) = P[\, D \leq d \,] = \sum_{i=0}^{d} f_D(i) \approx \int_0^d f_D(x)\, dx \quad \text{für} \quad 0 \leq d < P_{cyls} \tag{5.30}$$

Durch Anwenden der Substitutionsregel für Integrale [BSM98] auf die Gleichung 5.30 mit $x = t_{pos}^{-1}(y)$ und das Ersetzen von $d$ durch $t_{pos}^{-1}(t)$ läßt sich die Verteilungsfunktion $F_{pos}$ der Positionierungszeit $T_{pos}$ bestimmen:

$$\begin{aligned} F_D(t_{pos}^{-1}(t)) = P[\, D \leq t_{pos}^{-1}(t) \,] &= \int_{t_{pos}(0)}^{t} f_D(t_{pos}^{-1}(y)) * (t_{pos}^{-1})'(y)\, dy \\ &= P[T_{pos} \leq t] = F_{pos}(t) \end{aligned} \tag{5.31}$$

In Gleichung 5.31 läßt sich der Integrand des Integrals als Dichtefunktion $f_{pos}$ der Positionierungszeit $T_{pos}$ interpretieren. $(t_{pos}^{-1})'$ ist die erste Ableitung der inversen Zeitfunktion aus Gleichung 5.10. $f_{pos}$ ist somit gegeben durch:

$$f_{pos}(t) = f_D(t_{pos}^{-1}(t)) * (t_{pos}^{-1})'(t) \tag{5.32}$$

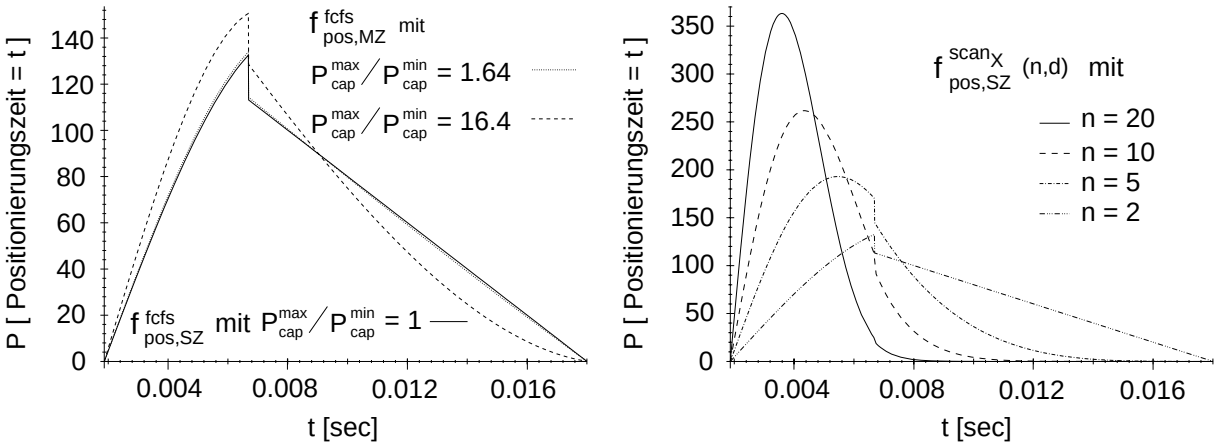



Abbildung 5.5: Dichtefunktionen der Positionierungszeitverteilung bei der FCFS- und der SCAN-Anordnungsstrategie der Aufträge

Um für die FCFS- bzw. SCAN-Anordnungsstrategie die Dichtefunktionen $f^{fcfs}_{pos,SZ}$, $f^{fcfs}_{pos,MZ}$, $f^{scan_1}_{pos,SZ}$ und $f^{scan_X}_{pos,SZ}$ der Positionierungszeitverteilung zu erhalten, werden die Dichtefunktionen $f^{fcfs}_{D,SZ}$, $f^{fcfs}_{D,MZ}$, $f^{scan_1}_{D,SZ}$ und $f^{scan_X}_{D,SZ}$ aus den Gleichungen 5.19, 5.20, 5.21 5.25 und 5.29 in Gleichung 5.32 eingesetzt.

Die Abbildung 5.5 zeigt links die Dichtefunktion $f^{fcfs}_{pos,SZ}$ und $f^{fcfs}_{pos,MZ}$ für Ein- und Mehr-Zonen-Platten unter Verwendung der FCFS-Anordnungstrategie. Die Unstetigkeit des Graphen an der Stelle $0.006686$ wird durch die Unstetigkeit der Funktion $(t^{-1}_{pos})'$ an dieser Stelle hervorgerufen. Im rechten Graphen ist die Dichtefunktion $f^{scan_X}_{pos,SZ}$ für eine unterschiedliche Anzahl von Aufträgen bei SCAN-Anordnungstrategie dargestellt.

### 5.2.5 Abschätzung der Positionierungszeit

Sind die Voraussetzungen für die obigen Berechnungen nicht erfüllt, weil z.B. die Annahmen der gleichmäßig verteilten Daten oder der gleichverteilten Zylinderauswahl nicht gelten, so läßt sich die Positionierungszeit durch ein oberes Maximum abschätzen. Im Falle der FCFS-Anordnungstrategie der Aufträge wird dieses Maximum der Positionierungzeit eines Auftrages durch die maximal mögliche Positionierungsdistanz $P_{cyls} - 1$ bestimmt. Das Maximum hat für einen Auftrag somit den Wert $t_{pos}(P_{cyls} - 1)$.

Bei der FCFS-Anordnung der Aufträge kann für jeden Auftrag die maximal mögliche Verzögerung auftreten. Aus diesem Grund ist die maximal mögliche *Gesamtpositionierungszeit,* d.h. die Summe der Positionierungszeiten aller Aufträge, bei insgesamt $n$ Aufträgen gegeben durch $n * t_{pos}(P_{cyls} - 1)$.

Bei Verwendung der SCAN-Anordnungstrategie kann die Gesamtpositionierungszeit für $n$ Aufträge durch eine obere Schranke abgeschätzt werden. Aufgrund der Konvexität der Funktion $t_{pos}$ in Gleichung 5.8 ergibt sich das Maximum genau dann, wenn die $n$ Aufträge äquidistant über alle Zylinder verteilt werden [Oya95]. Die Position des $i$-ten Auftrages befindet sich hierbei auf Zylinder $i * P_{cyls}/(n+1)$. Die obere Schranke der Gesamtpositionierungszeit $t^{max}_{pos}(n)$ hat für

$n$ Aufträge folgenden Wert:

$$t_{pos}^{max}(n) = (n+1) * t_{pos} \left( \frac{P_{cyls}}{n+1} \right) \tag{5.33}$$

## 5.3 Modellierung der Rotationsverzögerung

Die Modellierung der Rotationsverzögerung in den analytischen Modellen erfolgt, wie auch im Simulationsmodell des Abschnitts 4.1.1, durch eine gleichverteilte Zufallsvariable $T_{rot}$, die Werte zwischen 0 und $P_{rot}$ (siehe Tabelle 5.1) annehmen kann. Die Dichtefunktion $f_{rot}$ ist dann wie folgt definiert:

$$f_{rot}(t) = \begin{cases} \frac{1}{P_{rot}} & : \quad 0 \leq t \leq P_{rot} \\ 0 & : \quad \text{sonst} \end{cases} \tag{5.34}$$

Hierbei liegt die Annahme zugrunde, daß die Plattensteuerung auf der Magnetplatte keine Veränderung der Auftragsreihenfolge durchführt, um eventuell den Plattenzugriff durch eine Verringerung der Rotationsverzögerung zu optimieren. Ist letzteres nicht erfüllt, so läßt sich nur eine Abschätzung der Rotationsverzögerung durch den maximalen Wert $P_{rot}$ durchführen.

## 5.4 Modellierung der Transferzeiten

Die Transferzeit für die Durchführung eines Datenzugriffs wird durch die Transferrate der Platte und die Menge der zu transferierenden Daten bestimmt. Die Verteilung der Transferzeit ist damit abhängig von der Verteilung der Daten- und Zugriffsgrößen sowie von der Transferrate, die für Mehr-Zonen-Platten durch eine Verteilungsfunktion charakterisiert werden kann. Im folgenden soll zunächst diese Transferratenverteilung der Magnetplatte bestimmt werden.

### 5.4.1 Verteilung der Transferrate

Die Winkelgeschwindigkeit der in dieser Arbeit betrachteten Magnetplatten ist im Gegensatz zu den in [TCG98] betrachteten Sekundärspeichern konstant. Hierdurch kann die Transferrate $P_{rate}^{i}$ der $i$-ten Zone durch die Kapazität $P_{cap}^{i}$ einer Spur dieser Zone und der Zeitdauer für eine Umdrehung $P_{rot}$ berechnet werden. Es gilt $P_{rate}^{i} = P_{cap}^{i} / P_{rot}$. Unter der Annahme einer linear steigenden Speicherkapazität der Spuren steigt hiermit auch die Transferrate linear an. Es gilt analog zu Gleichung 5.13 für die Transferrate $P_{rate}^{i}$:

$$P_{rate}^{i} = \left( P_{cap}^{min} + \frac{P_{cap}^{max} - P_{cap}^{min}}{P_{zones} - 1} * (i-1) \right) * \frac{1}{P_{rot}} \qquad \text{für } 1 \leq i \leq P_{zones} \tag{5.35}$$

Die Verteilung der Transferrate ist abhängig von der Wahrscheinlichkeit, mit der auf eine Zone zugegriffen wird. Für deren Berechnung werden ebenfalls die Annahmen aus Abschnitt 5.2.1, d.h. gleichmäßige Verteilung der Daten und der Zugriffe auf diese Daten, übernommen. Die

Wahrscheinlichkeit, daß ein Auftrag mit der Transferrate $P^i_{rate}$ bearbeitet wird, ist gleich der Wahrscheinlichkeit, daß der Auftrag auf einer Spur der Zone $i$ liegt. Unter der Annahme, daß alle Zonen die gleiche Anzahl an Zylindern besitzen, gilt für die Wahrscheinlichkeit der Transferrate $V$:

$$P[\,V = P^i_{rate}\,] = \frac{\text{Speicherkapazität der Zone } i}{\text{Gesamtkapazität der Platte}} = \frac{P_{heads} * P^i_{cap} * P^i_{cpz}}{P_{heads} * \sum_{l=1}^{P_{zones}} P^l_{cap} * P^i_{cpz}} = \frac{P^i_{cap}}{\sum_{l=1}^{P_{zones}} P^l_{cap}} \tag{5.36}$$

Da in den nachfolgenden Berechnungen die Transferrate $V$ zur Vereinfachung als kontinuierliche Zufallsvariable betrachtet werden soll, wird zunächst deren Dichtefunktion bestimmt. Hierzu ist es notwendig, die Verteilungsfunktion der diskreten Transferratenverteilung zu untersuchen. Aus Gleichung 5.36 und Gleichung 5.13 ergibt sich:

$$P[\,V \leq P^i_{rate}\,] = \frac{\sum_{l=1}^{i} P^l_{cap}}{\sum_{l=1}^{P_{zones}} P^l_{cap}} = \left( i * P^{min}_{cap} + \frac{P^{max}_{cap} - P^{min}_{cap}}{P_{zones} - 1} * \frac{i * (i-1)}{2} \right) * \frac{1}{\sum_{l=1}^{P_{zones}} P^l_{cap}} \tag{5.37}$$

Setzt man $v = P^i_{rate}$ und löst Gleichung 5.35 für $i$, ergibt sich:

$$i = \frac{v * P_{rot} * P_{cyls} - v * P_{rot} - P^{min}_{cap} * P_{cyls} + P^{max}_{cap}}{P^{max}_{cap} - P^{min}_{cap}} \tag{5.38}$$

Das Ergebnis aus Gleichung 5.38 wird in 5.37 eingesetzt, und man erhält die Verteilungsfunktion der kontinuierlichen Zufallsvariablen $V$:

$$P[\mathrm{V} \leq \mathrm{v}] = \frac{1}{2} \frac{(v * P_{rot} * P_{cyls} - v * P_{rot} - P^{min}_{cap} * P_{cyls} + P^{max}_{cap})(P^{min}_{cap} + v * P_{rot})}{(P^{max}_{cap} - P^{min}_{cap}) * P_{cyls} * (P^{max}_{cap} + P^{min}_{cap})} \tag{5.39}$$

Durch die Ableitung von Gleichung 5.39 nach $v$ erhält man schließlich die gesuchte Dichtefunktion $f_{V,MZ}$ der Transferratenverteilung einer Mehr-Zonen-Platte:

$$f_{V,MZ}(v) = \begin{cases} \frac{P_{rot}(2*v*P_{rot}*P_{cyls}+P^{max}_{cap}-2*v*P_{rot}-P^{min}_{cap})}{(P^{max}_{cap}-P^{min}_{cap})*P_{cyls}*(P^{max}_{cap}+P^{min}_{cap})} & : \frac{P^{min}_{cap}}{P_{rot}} \leq v \leq \frac{P^{max}_{cap}}{P_{rot}} \\ 0 & : \text{sonst} \end{cases} \tag{5.40}$$

### 5.4.2 Verteilung der Transferzeit

Bei einer stochastischen Charakterisierung der Datengröße durch die Zufallsvariable $S$ und/oder der stochastischen Charakterisierung der Transferrate durch die Zufallsvariable $V$ ist die Transferzeit eines Datenzugriffs ebenfalls eine stochastische Größe. Die Zufallsvariable der Transferzeit $T_{trans}$ ist als Quotient $S/V$ definiert, deren Dichte gemäß [Fel71] durch das nachfolgende

faltungsähnliche Integral berechnet werden kann. Ist die Dichtefunktion $f_V$ der Transferrate und die Dichtefunktion $f_S$ der Auftragsgrößenverteilung gegeben, so ergibt sich die Dichtefunktion der Transferzeitverteilung wie folgt:

$$f_{trans}(t) = \int_{P_{cap}^{min}/P_{rot}}^{P_{cap}^{max}/P_{rot}} f_V(v) * v * f_S(v * t)\, dv \tag{5.41}$$

Die Integrationsgrenzen werden durch die minimale und maximale Transferrate bestimmt.

**Ein-Zonen-Platte**

Im Fall einer Ein-Zonen-Platte hat die Transferrate $P_{rate}$ den konstanten Wert $P_{cap}/P_{rot}$. Hiermit vereinfacht sich Gleichung 5.41 zu:

$$f_{trans,SZ}(t) = P_{rate} * f_S(P_{rate} * t)\, dr \tag{5.42}$$

Ist die Auftragsgröße, wie bei K-Aufträgen, gammaverteilt mit der Dichtefunktion $f_{S_K}$ aus Gleichung 5.3 und den Parametern $\alpha_S$ und $\beta_S$ aus Gleichung 5.4, so erhält man durch das Einsetzen in Gleichung 5.42 die Transferzeitverteilung für Ein-Zonen-Platten. Sie ist ebenfalls gammaverteilt mit den Parametern $\alpha_{trans} = \alpha_S * P_{rate}$ und $\beta_{trans} = \beta_S$:

$$\begin{aligned} f_{trans,SZ}^{S_K}(t) &= \frac{P_{rate} * \alpha_S(\alpha_S * P_{rate} * t)^{(\beta_S - 1)} * e^{\alpha_S * P_{rate} * t}}{\Gamma(\beta_S)} \\ &= \frac{\alpha_{trans}(\alpha_{trans} * t)^{(\beta_{trans} - 1)} * e^{\alpha_{trans} * t}}{\Gamma(\beta_{trans})} \end{aligned} \tag{5.43}$$

Für eine normalverteilte Auftragsgröße mit der Dichtefunktion $f_{S_D}$ aus Gleichung 5.1 und den Parametern $\mu_S$ und $\sigma_S$ (siehe Gleichung 5.2) gilt ähnliches. Die Dichtefunktion $f_{trans,SZ}$ aus Gleichung 5.42 ist hierfür normalverteilt mit den Parametern $\mu_{trans} = \mu_S/P_{rate}$ und $\sigma_{trans} = \sigma_S/P_{rate}$. Es gilt:

$$f_{trans,SZ}^{S_D}(t) = \frac{P_{rate}}{\sigma_S\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{P_{rate} * t - \mu_S}{\sigma_S}\right)^2} = \frac{1}{\sigma_{trans}\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{t - \mu_{trans}}{\sigma_{trans}}\right)^2} \tag{5.44}$$

**Mehr-Zonen-Platte**

Die Dichtefunktion $f_{trans,MZ}^{S_K}$ der Transferratenverteilung einer Mehr-Zonen-Platte bei gammaverteilter Auftragsgröße erhält man durch das Einsetzen von Gleichung 5.3 und Gleichung 5.40 in Gleichung 5.41. Die Vereinfachung des dadurch entstehenden Ausdrucks mit Maple [Kof96] liefert folgendes Ergebnis:

$$f_{trans,MZ}^{S_K}(t) = \sum_{i=0}^{5} t^{i-3} * (c_i * e^{f_0 t} - d_i * e^{f_1 t}) \tag{5.45}$$

Die positiven Parameter $c_0, \ldots, c_5$,$d_0, \ldots, d_5$ sowie $f_0$ und $f_1$ hängen von den Plattenparametern $P_{cap}^{min}$, $P_{cap}^{max}$ und $P_{cyls}$ sowie den Lastparametern $\alpha_S$ und $\beta_S$ ab.

Wird eine normalverteilte Auftragsgröße angenommen, so erhält man mit Gleichung 5.1 und Gleichung 5.40 nach Vereinfachung des Ausdrucks folgende Dichtefunktion:

$$f_{trans,MZ}^{S_D}(t) = \sum_{i=0}^{1} t^{i-3} * ( a_i * e^{(f_0 * (f_2 * t - f_1)^2)} + b_i * \text{erf}(f_4 * t - f_5) + c_i * e^{(f_0 * (f_3 * t - f_1)^2)} - b_i * \text{erf}(f_6 * t - f_5) ) \quad (5.46)$$

$$\text{mit} \quad \text{erf}(x) = \frac{2}{\sqrt{\pi}} * \int_{t=0}^{x} e^{-t^2}$$

Die positiven Parameter $f_0, \ldots, f_6$ sowie $a_1, a_2$,$b_1, b_2$ und $c_1, c_2$ hängen auch hier von Plattenparametern und den Lastparametern $\mu_S$ und $\sigma_S$ ab.

### Abschätzung der Transferzeit

Sind die Voraussetzungen für die vorausgegangenen Berechnungen nicht erfüllt, wie z.B. fehlende Informationen über die Verteilung der Auftragsgrößen oder über die Zonen-Parameter der Platte, so ist u.U. nur eine Abschätzung der maximalen Transferzeit möglich. Diese Abschätzung ergibt sich bei Verwendung der kleinstmöglichen Transferrate $P_{cap}^{min}/P_{rot}$ und der größtmöglichen Auftragsgröße $s$. Bei einer kontinuierlichen Verteilung der Auftragsgröße muß hierbei der Wert von $s$ auf ein hohes Quantil, z.B. auf das $99\%$-Quantil mit $P[S \leq s] = 99\%$, gesetzt werden.

# Kapitel 6

# Analytische Berechnung der Störungsrate

Die *Störungsrate* wird als Metrik für die Service-Qualität des Daten-Servers in der Bedienung kontinuierlicher Datenströme verwendet. Diese Metrik betrachtet einen einzelnen Datenstrom und beschreibt die Häufigkeit, mit der einzelne Störungen während der gesamten Zeitdauer, in der dieser Datenstrom vom Server beliefert wird, auftreten. Das Modell der nachfolgenden analytischen Betrachtung nimmt an, daß eine periodische rundenbasierte Scheduling-Strategie mit variabler Reihenfolge der K-Aufträge, wie sie in den Abschnitten 3.1 und 3.2 benutzt wird, verwendet wird. Hiermit tritt immer dann für einen kontinuierlichen Datenstrom eine einzelne Störung auf, wenn die Ausführung des zugehörigen K-Auftrages zeitlich nicht mehr innerhalb der Runde möglich ist. Dies geschieht immer dann, wenn die zuvor in der Runde bearbeiteten Aufträge so viel Zeit beansprucht haben, daß die Ausführung des von der Störung betroffenen K-Auftrages in der nachfolgenden Runde stattfinden müßte oder die bereits begonnene Ausführung sich zeitlich in die nachfolgende Runde erstreckt. Bei ersterem verwirft die betrachtete Scheduling-Strategie den K-Auftrag und das erforderliche Fragment wird nicht an den Client geliefert. Hierdurch wird sowohl eine Datenlücke als auch eine zeitliche Lücke erzeugt. Die Datenlücke entsteht dadurch, daß das Fragment ausgelassen wird; die zeitliche Lücke entsteht dadurch, daß das nachfolgende Fragment erst in der nächsten Runde, womöglich erst am Ende, geliefert werden kann. Wird die Ausführung eines K-Auftrages zu Rundenende abgebrochen, so steht nur ein Teil des Fragments im Server-Puffer zur Verfügung, wodurch ebenfalls eine Lücke im Datenstrom hervorgerufen wird. In beiden Fällen wird durch fehlende Daten eine u.U. wahrnehmbare Beeinträchtigung der Wiedergabe hervorgerufen. Die Häufigkeit, d.h. die Störungsrate, mit der dieses passiert, hängt von der Wahrscheinlichkeit ab, daß die Bearbeitung aller K-Aufträge innerhalb einer Runde die Rundendauer überschreitet. D-Aufträge brauchen in dieser Betrachtung nicht berücksichtigt werden, da diese durch die in Abschnitt 3.2 vorgestellten Scheduling-Strategien nur dann bearbeitet werden, wenn hierdurch keine Beeinträchtigung der K-Aufträge gegeben ist.

Im folgenden soll zunächst in Abschnitt 6.1 die Gesamtbedienzeit stochastisch berechnet werden und dann hieraus in Abschnitt 6.2 die Störungshäufigkeit bestimmt werden. Die Berechnung basiert auf der Verwendung der in Abschnitt 5 festgelegten Dichtefunktionen für System- und Lastverhalten. Das Ergebnis in Abschnitt 6.1 ist eine stochastische Abschätzung der Gesamtbedienzeit. Das Ergebnis in Abschnitt 6.2 ist eine stochastische Abschätzung der Störungsrate, die es erlaubt, bei gegebenen System- und Lastparametern quantitative Aussagen über die Service-Qualität des Daten-Servers zu treffen. In Abschnitt 6.3 wird gezeigt, wie die von der Zulassungkontrolle benutzte maximale Anzahl Datenströme $N_{max}$, die in einer Runde unter

Einhaltung einer vorgegebenen Service-Qualität bedient werden können, berechnet wird. Danach folgt in Abschnitt 6.4 ein Vergleich zwischen den analytisch berechneten Vorhersagen des Modells und experimentell ermittelten Meßergebnissen einer Simulation. Abschnitt 6.5 schließt dieses Kapitel mit einem kurzen Fazit.

Ein Verzeichnis der in den folgenden Abschnitten verwendeten Bezeichner findet sich im Anhang B.

## 6.1 Berechnung der Gesamtbedienzeit

Seien $T_{pos,i}$ die Positionierungszeit, $T_{rot,i}$ die Rotationsverzögerung und $T_{trans,i}$ die Transferzeit des $i$-ten Auftrages einer Runde. Die Gesamtbedienzeit $T_{svc}$ für die Ausführung von insgesamt $N$ K-Aufträgen ergibt sich als Summe:

$$T_{svc}(N) := \sum_{i=1}^{N} T_{pos,i} + T_{rot,i} + T_{trans,i} \tag{6.1}$$

Die Komponenten der Gesamtbedienzeit $T_{svc}$ werden im folgenden als unabhängige Zufallsvariablen betrachtet. Ziel der Berechnungen ist es, für eine festgelegte Anzahl $N$ Datenströme die *Restwahrscheinlichkeit* (engl. tail probability) $p_{late}$ der Verteilung von $T_{svc}$ zu bestimmen, mit der die Gesamtbedienzeit die Rundendauer $T$ überschreitet:

$$p_{late}(N) := P[T_{svc}(N) > T] \tag{6.2}$$

Die Berechnung wird mit zwei alternativen Methoden durchgeführt, die im Abschluß hinsichtlich ihrer Berechnungsgenauigkeit durch einen Vergleich mit Simulationsergebnissen untersucht werden. Bei der ersten Methode wird eine Approximation der Verteilungsfunktion von $T_{svc}$ durch eine numerische Faltung berechnet, mit der die Restwahrscheinlichkeit durch Integration direkt bestimmt werden kann. Das zweite Verfahren schätzt die Restwahrscheinlichkeit durch eine Chernoff-Schranke ab, die die Laplace-Transformierte von $T_{svc}$ benutzt. Beide Berechnungsverfahren verwenden, damit die Berechnungen mit konkreten Werten durchführbar sind, konservative Abschätzungen. Dies hat zur Folge, daß in beiden Fällen eine obere Schranke der Restwahrscheinlichkeit ermittelt wird.

### 6.1.1 Stochastische Abschätzung durch numerische Faltung

Die erste Methode bestimmt eine z.T. konservativ approximierte Verteilungsfunktion $F^{\approx}_{svc}$ der Zufallsvariablen $T_{svc}$ und erlaubt so die Berechnung einer oberen Schranke $b^{NC}_{late}$ (NC = numerical convolution) für der Restwahrscheinlichkeit $p_{late}$. Hierfür gilt:

$$p_{late}(N) := P[\ T_{svc}(N) > T\ ] = 1 - P[\ T_{svc}(N) \leq T\ ] \approx 1 - F^{\approx}_{svc}(N) =: b^{NC}_{late}(N) \tag{6.3}$$

Um die Verteilungsfunktion einer Summe zweier unabhängiger Zufallsvariablen zu bestimmen, muß eine Faltungsoperation durchgeführt werden, die für die Zufallsvariablen $Y_1$ und $Y_2$ und deren Verteilungsfunktion $F_{Y_1}$ bzw. Dichtefunktion $f_{Y_2}$ wie folgt definiert ist:

$$P[\ Y_1 + Y_2 \leq y\ ] = \int_{-\infty}^{\infty} F_{Y_1}(y - x) * f_{Y_2}(x)dx \tag{6.4}$$

Übertragen auf Gleichung 6.1 bedeutet dies die Durchführung von insgesamt $3 * N - 1$ Faltungen, damit die Verteilungsfunktion $F^{\approx}_{svc}$ berechnet werden kann. Dieses führt aufgrund der durchzuführenden Integration weder symbolisch ausgewertet zu einer geschlossenen Lösung noch numerisch berechnet zu akzeptablen Berechnungszeiten.

Eine praktikable Lösung, damit die Gleichung 6.3 für konkrete Werte ausgewertet werden kann, ist nur dann möglich, wenn die Anzahl der durchzuführenden Integrationen zur Berechnung von $F^{\approx}_{svc}$ auf ein Minimum beschränkt wird. Hierfür kann man es sich zunutze machen, daß bei einigen Verteilungen, wie z.B. der Normal- und Gammaverteilung, deren Summenbildung wieder dieselbe Verteilung ergibt. Aus diesem Grund sollen deshalb zunächst die Verteilungen folgender Zufallsvariablen von links beginnend untersucht werden:

$$T_{\sum trans} := \sum_{i=1}^{N} T_{trans} \quad , \quad T_{\sum rot} := \sum_{i=1}^{N} T_{rot} \quad \text{und} \quad T_{\sum pos} := \sum_{i=1}^{N} T_{pos}$$

**Summe der Transferzeiten**

Wird eine Ein-Zonen-Platte mit konstanter Transferrate $P_{rate}$ und gammaverteilten Auftragsgrößen aus Gleichung 5.4 verwendet, so ist, wie in Gleichung 5.43 zu sehen, die Transferzeit $T_{trans}$ eines einzelnen Auftrages ebenfalls gammaverteilt. Hierdurch ist schließlich die Summe $T_{\sum trans}$ der $N$ unabhängigen Zufallsvariablen $T_{trans}$ auch gammaverteilt mit den Parametern $\alpha_{\sum trans}$ und $\beta_{\sum trans}$ [All90]:

$$\alpha_{\sum trans} = \alpha_{trans} = P_{rate} * \frac{E[S_K]}{Var[S_K]} \quad \text{und} \quad \beta_{\sum trans} = N * \beta_{trans} = N * \frac{(E[S_K])^2}{Var[S_K]} \qquad (6.5)$$

Für Mehr-Zonen-Platten wird die Transferzeit $T_{trans}$ ebenfalls durch eine Gammaverteilung modelliert. Bei dieser Approximation werden die Parameter $\alpha^{\approx}_{trans}$ und $\beta^{\approx}_{trans}$ so bestimmt, daß Erwartungswert und Varianz mit der ursprünglichen Verteilung übereinstimmen. Die Berechnung der Parameter $\alpha^{\approx}_{\sum trans}$ und $\beta^{\approx}_{\sum trans}$ für die Summe der $N$ Transferzeiten kann hiermit analog zu Gleichung 6.5 erfolgen.

Die Abbildung 6.1 zeigt links die Werte der Dichtefunktion $f^{S_K}_{trans,MZ}$ aus Gleichung 5.45 sowie die Dichtefunktion $f^{S_K \approx}_{trans,MZ}$ der Approximation für Mehr-Zonen-Platten. Der rechte Graph zeigt den relativen Fehler. Dieser liegt bei weniger als $6\%$ im Hauptbereich der Transferzeit zwischen 5 und 100 Millisekunden, d.h. zwischen einer halben und ca. 12 Plattenumdrehungen.

**Summe der Rotationsverzögerungen**

Die für $T_{\sum rot}$ notwendige Berechnung der Dichtefunktion einer Summe gleichverteilter Zufallsvariablen ist nur über die Integration in Formel 6.4 möglich. Der zur Berechnung konkreter Werte benötigte numerische Aufwand ist für eine große Anzahl Summanden nicht praktikabel. Aus diesem Grund wird die Zufallsvariable $T_{rot}$ durch eine "abgeschnittene" Normalverteilung approximiert. Deren Parameter $\mu^{\approx}_{rot}$ und $\sigma^{\approx}_{rot}$ werden so gewählt, daß sie den gleichen Erwartungswert und die gleiche Varianz wie die ursprünglich angenommene Gleichverteilung besitzt.

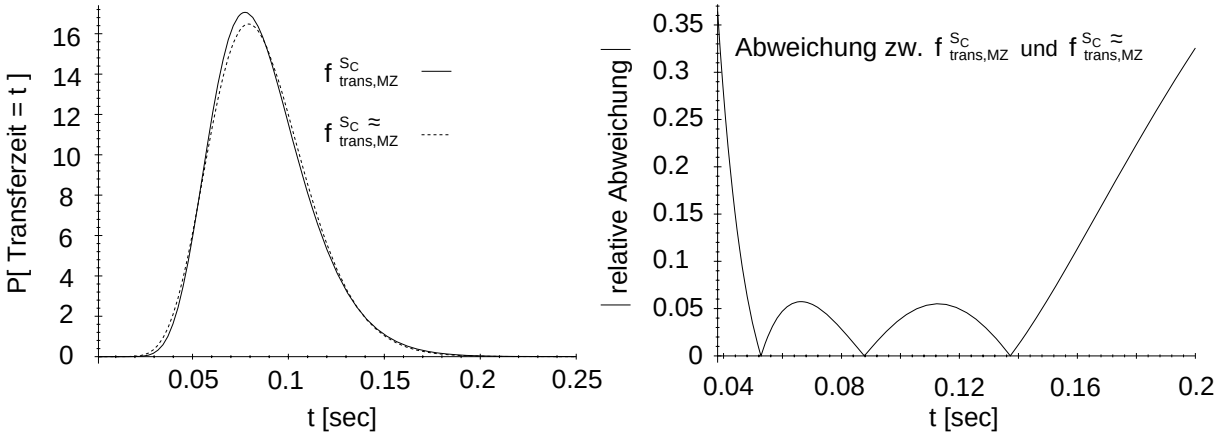



Abbildung 6.1: Approximation der Transferzeitverteilung bei Mehr-Zonen-Platten durch eine Gammaverteilung

Die Summe dieser normalverteilten Zufallsvariablen ist hiermit auch wieder normalverteilt mit den Parametern $\mu^{\approx}_{\sum rot}$ und $\sigma^{\approx}_{\sum rot}$ und der Dichtefunktion $f^{\approx}_{\sum rot}$:

$$f^{\approx}_{\sum rot}(t) = \begin{cases} \dfrac{1}{\sigma^{\approx}_{\sum rot}\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{t-\mu^{\approx}_{\sum rot}}{\sigma^{\approx}_{\sum rot}}\right)^2} & : \quad t > 0 \\ 0 & : \quad \text{sonst} \end{cases} \tag{6.6}$$

$$\text{mit} \quad \mu^{\approx}_{\sum rot} = N * \mu^{\approx}_{rot} = N * \frac{P_{rot}}{2} \quad \text{und} \quad (\sigma^{\approx}_{\sum rot})^2 = N * (\sigma^{\approx}_{rot})^2 = N * \frac{(P_{rot})^2}{12} \tag{6.7}$$

Abbildung 6.2 zeigt links Werte der Dichtefunktion $f_{\sum rot}$ der Variablen $T_{\sum rot}$ und deren Approximation durch die Dichtefunktion $f^{\approx}_{\sum rot}$ im Vergleich für verschiedene $N$. Die Werte von $f_{\sum rot}$ sind durch numerische Integration in Gleichung 6.4 mit der Dichtefunktion $f_{rot}$ aus Gleichung 5.34 berechnet worden. Die Werte von $f^{\approx}_{\sum rot}$ werden aus der Dichtefunktion in Gleichung 6.6 und den Parametern aus Gleichung 6.7 bestimmt. Die Plattenparameter entsprechen denen aus Tabelle 5.1. Zu erkennen ist in der Abbildung, daß mit wachsendem $N$ sich die approximierte Dichtefunktion der exakten Dichtefunktion annähert. So liegt bereits für $N = 4$, zu erkennen im rechten Graphen der Abbildung, der relative Fehler in weiten Bereichen unter $6\%$. Da in den folgenden Berechnungen typischerweise von einem drei- bis fünffach so großen Wert für $N$ ausgegangen wird, erscheinen die zu erwartenden Abweichungen vernachlässigbar.

**Summe der Positionierungszeiten**

Als weitere Approximation in der Berechnung wird die Summe der Positionierungszeiten für $N$ Aufträge $T_{\sum pos}$ in den nachfolgenden Berechnungen durch ihren Maximalwert $t^{max}_{pos}(N)$ aus Gleichung 5.33 ersetzt. Die Dichtefunktion $f^{\approx}_{\sum pos}$ für die Verteilung dieser Variablen lautet damit:

$$f^{\approx}_{\sum pos}(t) = \begin{cases} 1 & : \quad t = t^{max}_{pos}(N) \\ 0 & : \quad \text{sonst.} \end{cases} \tag{6.8}$$

Eine explizite $(N-1)$-fache Faltung der Dichtefunktion $f^{scan_X}_{pos,SZ}$ ist aufgrund zu großer Laufzeiten nicht möglich und wird durch diese Approximation vermieden.

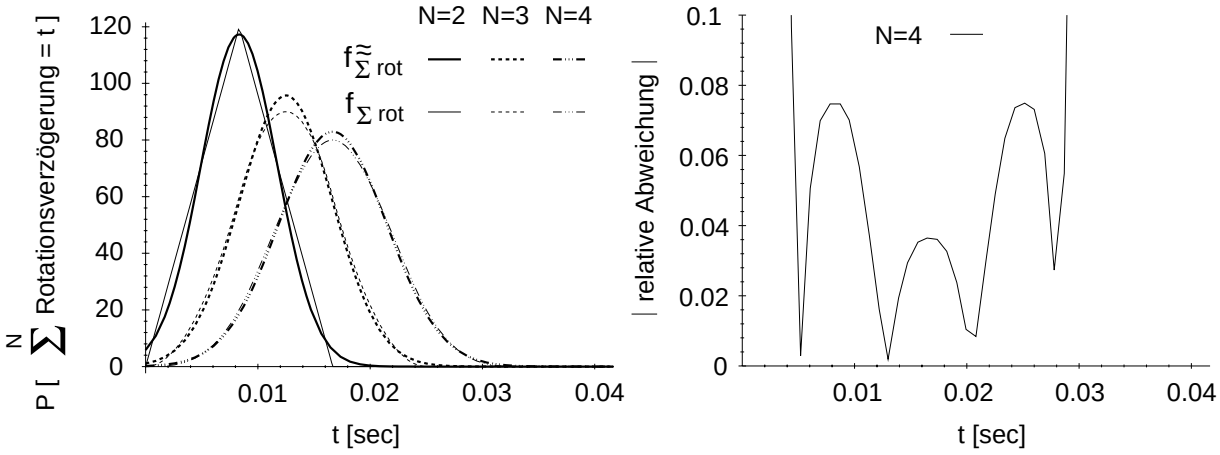



Abbildung 6.2: Vergleich der Dichtefunktionen zwischen approximierter und gleichverteilter Rotationsverzögerung

**Dichtefunktion der Gesamtbedienzeitverteilung**

Die Verteilungsfunktion $F^{\approx}_{svc}$ der Gesamtbedienzeit für eine Ein-Zonen-Platte erhält man mit Gleichung 6.4 wie folgt:

$$\begin{aligned} F^{\approx}_{svc}(N) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F^{\approx}_{\Sigma rot}(T-y-x) * f_{\Sigma trans}(x) * f^{\approx}_{\Sigma pos}(y)\,dx\,dy \\ &= \int_{0}^{T-t^{max}_{pos}(N)} F^{\approx}_{\Sigma rot}(T-t^{max}_{pos}(N)-x) * f_{\Sigma trans}(x)\,dx \qquad (6.9) \\ &= 1-b^{NC}_{late} \end{aligned}$$

Die Verteilungsfunktion $F^{\approx}_{\Sigma rot}$ wird durch Integration der Dichtefunktion $f^{\approx}_{\Sigma rot}$ berechnet. Zur Berechnung der Werte von $b^{NC}_{late}$ für unterschiedliche Werte von $N$ sind somit zwei numerische Integrationen, z.B. mit Maple [Kof96], durchzuführen. Für eine Mehr-Zonen-Platte wird in Gleichung 6.9 die zugehörige Dichtefunktion der Transferzeitverteilung eingesetzt.

### 6.1.2 Stochastische Abschätzung mit Laplace-Transformierten

Als zweites Verfahren soll die Abschätzung der Restwahrscheinlichkeit $p_{late}$ mit Hilfe der Laplace-Transformation vorgestellt werden. Die Laplace-Transformierte einer Zufallsvariablen $Y$ mit Dichte $f_Y$ ist definiert als [All90, Nel95]:

$$Y^*(s) := \int_0^{\infty} e^{-s*t} f_Y(t)\,dt \qquad (6.10)$$

Die Chernoff-Schranke erlaubt es, die Restwahrscheinlichkeit $P[\ Y \geq y\ ]$ einer Zufallsvariablen $Y$ mit Hilfe ihrer Laplace-Transformierten $Y^*$ abzuschätzen. Es gilt [Fel71, Nel95]

$$P[\ Y \geq y\ ] \leq \inf_{\theta \geq 0} \left\{ e^{-\theta y} * Y^*(-\theta) \right\} \qquad (6.11)$$

Um das optimale $\theta$ für das kleinstmögliche Maximum zu finden, kann der obige rechte Ausdruck nach $\theta$ abgeleitet und die Nullstelle bestimmt werden.

Eine weitere Eigenschaft der Laplace-Transformation vereinfacht die nachfolgenden Berechnungen, in denen die Summen von Zufallsvariablen betrachtet werden. Die Laplace-Transformierte einer Summe zweier unabhängiger Zufallsvariablen erhält man durch die Multiplikation der zugehörigen Laplace-Transformierten, im folgenden als *Faltungseigenschaft* bezeichnet:

$$Y^*(s) = Y_1^*(s) * Y_2^*(s) \quad \text{mit} \quad Y = Y_1 + Y_2 \tag{6.12}$$

Analog zum Vorgehen im vorangegangenen Abschnitt wird zunächst die Laplace-Transformierte der Transferzeit, daraufhin die Laplace-Transformierte der Rotationsverzögerung und zum Schluß die Laplace-Transformierte der Positionierungszeit betrachtet.

### Laplace-Transformierte der Transferzeit

Die Transferzeit ist bei Ein-Zonen-Platten gammaverteilt mit den Parametern aus Gleichung 6.5. Zusammen mit der Definition der Laplace-Transformierten aus Gleichung 6.10 und der Faltungseigenschaft aus Gleichung 6.12 erhält man die Laplace-Transformierte $T^*_{trans}$ der Variablen $T_{trans}$ bzw. die Laplace-Transformierte $T^*_{\sum trans}$ der Gesamttransferzeit $T_{\sum trans}$:

$$T^*_{trans}(s) = \left(\frac{\alpha_{trans}}{\alpha_{trans} + s}\right)^{\beta_{trans}} \quad \text{und} \quad T^*_{\sum trans}(s) = \left[T^*_{trans}(s)\right]^N \tag{6.13}$$

Für Mehr-Zonen-Platten erfolgt wie in Abschnitt 6.1.1 die Approximation der Transferratenverteilung durch eine Gammaverteilung. Die Parameter $\alpha^{\approx}_{trans}$ und $\beta^{\approx}_{trans}$ werden analog bestimmt.

### Laplace-Transformierte der Rotationsverzögerung

Für die Laplace-Transformierte $T^*_{rot}$ der gleichverteilten Zufallsvariablen $T_{rot}$ gilt:

$$T^*_{rot}(s) = \frac{1 - e^{-sP_{rot}}}{s * P_{rot}} \quad \text{und} \quad T^*_{\sum rot}(s) = \left[T^*_{rot}(s)\right]^N \tag{6.14}$$

Eine Approximation der Zufallsvariablen $T_{rot}$ durch eine normalverteilte Variable, wie sie im vorangegangenen Abschnitt durchgeführt wurde, ist hier nicht notwendig.

### Laplace-Transformierte der Positionierungszeit

Die Berechnung der Laplace-Transformierten für die Positionierungszeit kann auf zwei verschiedenen Wegen erfolgen.

- Methode LT I
  Die Summe der Positionierungszeiten wird wie im vorangegangenen Abschnitt durch das konstante Maximum $t^{max}_{pos}(N)$ abgeschätzt. Die Laplace-Transformierte $T^{\approx *}_{\sum pos}$ ergibt sich damit zu:

$$T^{LTI*}_{\sum pos}(s) = e^{-s*t^{max}_{pos}(N)} \tag{6.15}$$

- Methode LT II
  Die Laplace-Transformierte $T^{LTII*}_{\sum pos}$ wird für die Dichtefunktion $f^{scan_X}_{pos,SZ}$ aus Abschnitt 5.2.4 berechnet. Das Integral in der Definition 6.10 wird hierbei numerisch ausgewertet, da keine geschlossene Lösung existiert. Es gilt:

$$T^{LTII*}_{\sum pos} = \left[ \int_0^\infty e^{-st} f^{scan_X}_{pos,SZ}(N,t)\, dt \right]^N \tag{6.16}$$

  Diese Methode ist im Rahmen dieser Arbeit nur bei Ein-Zonen-Platten anwendbar, da die Herleitung einer Dichtefunktion für die Positionierungszeit bei Mehr-Zonen-Platten und SCAN-Algorithmus im vorangegangenen Kapitel nicht näher untersucht wurde.

**Laplace-Transformierte der Gesamtbedienzeit**

Aufgrund der Faltungseigenschaft aus Gleichung 6.12 ergibt sich aus den obigen Gleichungen 6.13, 6.14 und 6.15 die Laplace-Transformierte $T^{LTI*}_{svc}$ der Gesamtbedienzeit für Methode LT I bei $N$ Aufträgen und einer Ein-Zonen-Platte zu:

$$\begin{aligned} T^{LTI*}_{svc}(s) &= T^{LTI*}_{\sum pos}(s) * T^{*}_{\sum rot}(s) * T^{*}_{\sum trans}(s) \\ &= e^{-s*t^{max}_{pos}(N)} * \left( \frac{1 - e^{-sP_{rot}}}{s * P_{rot}} \right)^N * \left( \frac{\alpha_{trans}}{\alpha_{trans} + s} \right)^{\beta_{trans}*N} \end{aligned} \tag{6.17}$$

Für Mehr-Zonen-Platten werden die Parameter $\alpha^{\approx}_{trans}$ und $\beta^{\approx}_{trans}$ verwendet. Die Laplace-Transformierte $T^{LTII*}_{svc}$ ergibt sich analog bei Verwendung von Gleichung 6.16.

Durch Anwendung der Chernoff-Abschätzung aus Gleichung 6.11 läßt sich nun die obere Schranke $b^{LTI}_{late}$ bestimmen:

$$p_{late}(N) := P[\ T_{svc}(N) \geq T\ ] \leq \inf_{\theta \geq 0} \{h(\theta)\} =: b^{LTI}_{late}(N)$$

mit

$$h(\theta) = e^{-\theta T} e^{\theta * t^{max}_{pos}(N)} \left( \frac{e^{\theta P_{rot}} - 1}{\theta * P_{rot}} \right)^N * \left( \frac{\alpha_{trans}}{\alpha_{trans} - \theta} \right)^{\beta_{trans}*N} \tag{6.18}$$

Da die Gleichung $h'(\theta_{opt}) = 0$ zu keinem geschlossenen Ausdruck für $\theta_{opt}$ führt, wird das Minimum numerisch berechnet. Die Berechnung der oberen Schranke $b^{LTII}_{late}$ für die Laplace-Transformierte $T^{LTII*}_{svc}$ erfolgt analog.

### 6.1.3 Zusammenfassung der Berechnungsverfahren

Eine Zusammenfassung der in den vorangegangenen Abschnitten beschriebenen Methoden zur Abschätzung von $p_{late}$ ist in Tabelle 6.1 zu finden. Für jede Methode werden die verwendeten Verteilungsfunktionen bzw. Approximationen mit dem Verweis auf die dazugehörige Gleichung aufgeführt.

| Ein-Zonen-Platte | | | |
|---|---|---|---|
| Berechnungsmethoden | Transferzeit | Rotationsverzögerung | Positionierungszeit |
| numerische Faltung ( $b_{late}^{NC}$ ) | Gammaverteilung (6.5) | Normalverteilung (6.7) | Approx. durch Maximum (6.8) |
| Chernoff-Schranke ( $b_{late}^{LTI}$ ) | Gammaverteilung (6.13) | Gleichverteilung (6.14) | Approx. durch Maximum (6.15) |
| Chernoff-Schranke ( $b_{late}^{LTII}$ ) | Gammaverteilung (6.13) | Gleichverteilung (6.14) | numerische Transformation von $f_{pos,SZ}^{scan_X}$ (6.16) |

| Mehr-Zonen-Platte | | | |
|---|---|---|---|
| Berechnungsmethoden | Transferzeit | Rotationsverzögerung | Positionierungszeit |
| numerische Faltung ( $b_{late}^{NC}$ ) | Approx. durch Gammaverteilung | Normalverteilung(6.7) | Approx. durch Maximum (6.8) |
| Chernoff-Schranke ( $b_{late}^{LTI}$ ) | Approx. durch Gammaverteilung | Gleichverteilung (6.14) | Approx. durch Maximum (6.15) |

Tabelle 6.1: Überblick über die alternativen Verfahren zur Abschätzung von $p_{late}$

## 6.2 Berechnung der Störungshäufigkeit

Da die Restwahrscheinlichkeit $p_{late}$ der Gesamtbedienzeitverteilung eine abstrakte Metrik für die Service-Qualität des Servers darstellt, soll im folgenden Abschnitt ermittelt werden, wie häufig ein Datenstrom während der Gesamtdauer seiner Präsentation von einer Störung betroffen ist. Diese Metrik wird im folgenden als *Störungsrate* bezeichnet. Sie hat den Vorteil, daß sich hierfür die Toleranzschwelle der Benutzer in psychologischen Experimenten ermitteln läßt.

Zur Abschätzung der Störungsrate wird zunächst die Verteilung der Störungen innerhalb einer Runde berechnet. Hieraus wird die Anzahl der Störungen abgeleitet, die einen Datenstrom betreffen, dessen Lieferung sich über $C$ Runden erstreckt. Da ein einziger Datenstrom betrachtet wird, soll im folgenden angenommen werden, daß alle Datenströme gleich wahrscheinlich und innerhalb der Runden unabhängig voneinander von den Störungen betroffen sind. Letzteres läßt sich dadurch sicherstellen, daß die Plazierung der Fragmente zufällig stattfindet und dadurch die Bearbeitungsreihenfolge der Aufträge in aufeinanderfolgenden Runden unkorreliert ist. Hierdurch ist es möglich, das Auftreten von $k$ Störungen innerhalb einer Runde als ein Zufallsexperiment zu betrachten, bei dem aus den $N$ Datenströmen, die in der Runde bedient werden, $k$ ausgewählt werden. Die Wahrscheinlichkeit, daß ein bestimmter Datenstrom

von einer Störung innerhalb einer Runde betroffen ist, ergibt sich damit zu:

$$
\begin{aligned}
& P[\text{ Datenstrom } i \text{ hat eine Störung in einer Runde }] \\
&= \sum_{k=0}^{N} P\left[ \begin{array}{l|l} \text{Datenstrom } i & \text{Anzahl der nichtbedienten} \\ \text{nicht bedient} & \text{Datenströme in einer Runde} = k \end{array} \right] \\
&\qquad * P\left[ \begin{array}{l} \text{Anzahl der nichtbedienten} \\ \text{Datenströme in einer Runde} = k \end{array} \right] \\
&= \sum_{k=0}^{N} \frac{k}{N} * P\left[ \begin{array}{l} \text{Anzahl der nichtbedienten} \\ \text{Datenströme in einer Runde} = k \end{array} \right] \qquad (6.19)
\end{aligned}
$$

Sei $T_{svc}(i)$ die Gesamtbedienzeit für $i$ Aufträge innerhalb einer Runde, die die konstante Länge $T$ besitzt. Aus Gleichung 6.19 ergibt sich dann

$$
\begin{aligned}
& P[\text{ Datenstrom } i \text{ hat eine Störung in einer Runde }] \\
&= \frac{1}{N} \sum_{k=0}^{N} k * P\left[ \begin{array}{l} \text{Anzahl der bedienten} \\ \text{Datenströme in einer Runde} = N - k \end{array} \right] \\
&= \frac{1}{N} \sum_{i=0}^{N-1} (N-i) * P\left[ \begin{array}{l} \text{Anzahl der bedienten} \\ \text{Datenströme in einer Runde} = i \end{array} \right] \qquad (6.20)
\end{aligned}
$$

Es gilt $P\left[ \begin{array}{l} \text{Anzahl der bedienten} \\ \text{Datenströme in einer Runde} \geq i \end{array} \right] = P[\ T_{svc}(i) \leq T\ ]$. Hieraus folgt mit Gleichung 6.20:

$$
\begin{aligned}
& P[\text{ Datenstrom } i \text{ hat eine Störung in einer Runde }] \\
&= \frac{1}{N} \sum_{i=0}^{N-1} (N-i) * (P[\ T_{svc}(i) \leq T\ ] - P[\ T_{svc}(i+1) \leq T\ ]) \\
&= \frac{1}{N} \left( N * P[\ T_{svc}(0) \leq T\ ] - \sum_{i=1}^{N} P[\ T_{svc}(i) \leq T\ ] \right) \\
&= \frac{1}{N} \sum_{i=1}^{N} P[\ T_{svc}(i) > T\ ] \\
&= \frac{1}{N} \sum_{i=1}^{N} p_{late}(i) \qquad (6.21)
\end{aligned}
$$

Verwendet man eine der im vorangegangenen Abschnitt ermittelten Schranken $b^j_{late}$, so erhält man eine Abschätzung $b^j_{glitch},\ j \in \{_{NC,LTI,LTII}\}$ für die Wahrscheinlichkeit, daß ein Datenstrom in einer Runde eine Störung erzeugt:

$$
\begin{aligned}
p_{glitch}(N) &:= P[\text{ Datenstrom } i \text{ hat eine Störung in einer Runde }] \\
&= \frac{1}{N} \sum_{k=1}^{N} p^j_{late}(k) \\
&\leq \frac{1}{N} \sum_{k=1}^{N} b^j_{late}(k) \\
&=: b^j_{glitch}(N) \quad \text{mit } j \in \{_{NC,LTI,LTII}\} \qquad (6.22)
\end{aligned}
$$

Das Auftreten mehrerer Störungen, die sich über mehrere Runden verteilen, läßt sich aufgrund der gemachten Unabhängigkeitsannahmen durch eine Binomialverteilung modellieren. Ist $C$ die Anzahl der Runden und $g$ die Gesamtzahl der Störungen, so ist die Wahrscheinlichkeit, daß bei einem Datenstrom diese $g$ Störungen innerhalb von $C$ Runden auftreten, gegeben durch:

$$\begin{aligned} & P[\text{ Datenstrom } i \text{ hat } g \text{ Störungen während } C \text{ Runden }] \\ = \quad & \binom{C}{g} p_{glitch}(N)^g * [1 - p_{glitch}(N)]^{C-g} \end{aligned} \tag{6.23}$$

Angenommen wird hierbei, daß während der gesamten Zeitdauer, d.h. während der $C$ Runden, innerhalb jeder Runde stets $N$ K-Aufträge bearbeitet werden sollen. Die Berechnung der Restwahrscheinlichkeit $p_{error}$ für mehr als $g$ Störungen innerhalb der $C$ Runden kann durch explizite Berechnung der Werte aus Gleichung 6.23 oder mit weniger großem Berechnungsaufwand durch die Anwendung der Chernoff-Schranke aus [HR89] durchgeführt werden. Ersteres führt zur Abschätzung $b_{error}^{BNI,j}(N,C,g)$ in Gleichung 6.25, letzteres zur Abschätzung $b_{error}^{BNII,j}(N,C,g)$ in Gleichung 6.26. Der Parameter $j \in \{_{NC,LTI,LTII}\}$ gibt an, daß der Wert von $b_{error}^{BNI,j}$ und $b_{error}^{BNII,j}$ abhängig vom Abschätzungsverfahren für $p_{late}$ ist:

$$p_{error}(N,C,g) := P[\text{ Anzahl der Störungen für Datenstrom } i \text{ in } C \text{ Runden } \geq g\ ] \tag{6.24}$$

$$= 1 - \sum_{i=0}^{g-1} \binom{C}{i} p_{glitch}(N)^i * [1 - p_{glitch}(N)]^{C-i}$$

$$\leq 1 - \sum_{i=0}^{g-1} \binom{C}{i} b_{glitch}^j(N)^i * \left[1 - b_{glitch}^j(N)\right]^{C-i} \tag{6.25}$$

$$=: b_{error}^{BNI,j}(N,C,g) \quad \text{mit } j \in \{_{NC,LTI,LTII}\}$$

$$\leq \left(\frac{C b_{glitch}^j(N)}{g}\right)^g \left(\frac{C - C b_{glitch}^j(N)}{C-g}\right)^{C-g} \tag{6.26}$$

$$=: b_{error}^{BNII,j}(N,C,g) \quad \text{mit } j \in \{_{NC,LTI,LTII}\}$$

Voraussetzung für die Anwendbarkeit der Chernoff-Schranke in Gleichung 6.26 sind die Bedingungen $g/C \geq b_{glitch}^j$ und $0 \leq g/C \leq 1$. Die Abbildung 6.3 zeigt die Werte von $b_{error}^{BNII,j}$ und $b_{error}^{BNI,j}$ bei Variation von $b_{glitch}$. Der linke Graph betrachtet die Gesamtdauer von $3600$ Runden $(C = 3600)$ mit einer Anzahl von $36$ Störungen $(g = 36)$. Im rechten Graphen ist die Gesamtdauer auf $1200$ Runden $(C = 1200)$ mit $12$ Störungen $(g = 12)$ festgelegt.

Die Störungsrate eines Datenstromes wird definiert als der Quotient $g/C$ aus der Anzahl der Störungen $g$, die während der Wiedergabedauer des kontinuierlichen Datenobjektes über $C$ Runden auftreten. Wird eine Restwahrscheinlichkeit von höchstens $0.05$ für eine Störungsrate von mehr als $0.01$ durch den Benutzer toleriert, so muß bei einem kontinuierlichen Datenobjekt mit $3600$ Fragmenten die Wahrscheinlichkeit von mehr als $36$ Störungen, die über alle $3600$ Runden auftreten, unter $5\%$ liegen. Nimmt man eine Wiedergabelänge pro Fragment von einer Sekunde an, so beträgt die Wiedergabedauer dieses Datenobjektes $60$ Minuten. Folglich ist unter der Annahme einer gleichmäßigen Verteilung der Störungen die Wahrscheinlichkeit, daß innerhalb von $100$ Sekunden mehr als eine Störung auftritt, geringer als $5\%$. Dies bedeutet

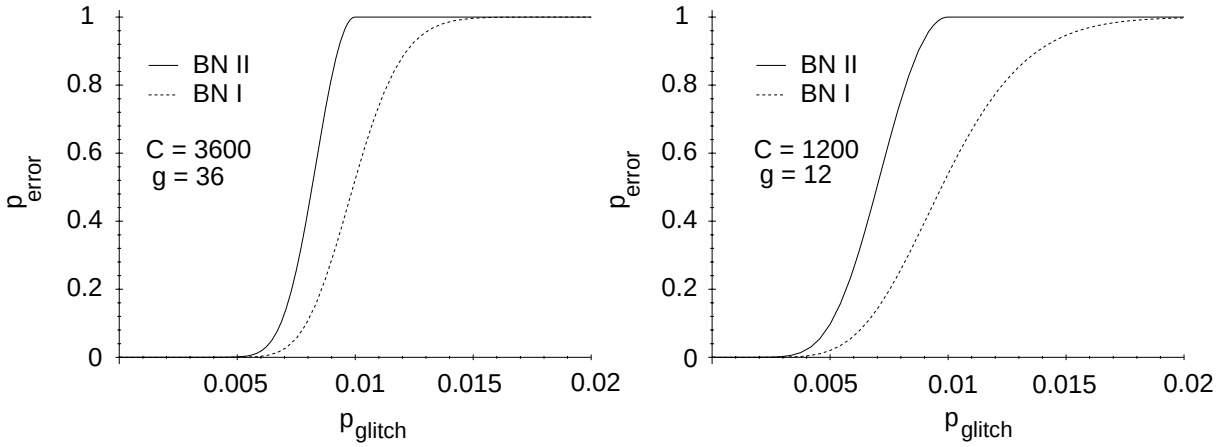



Abbildung 6.3: Approximation vs. explizite Berechnung der Restwahrscheinlichkeit für die Binomialverteilung

weiterhin, daß die Wahrscheinlichkeit, daß mehr als $100$ Sekunden zwischen zwei Störungen vergehen, größer ist als $95\%$.

Die Berechnung der Restwahrscheinlichkeit der Störungsrate erfolgt unter Verwendung von $p_{error}$ aus Gleichung 6.24. Es gilt:

$$\begin{aligned} p_{erate}(N,e) &:= P[\text{ Störungsrate für Datenstrom mit } C \text{ Runden } \geq e\,] && (6.27)\\ &= p_{error}(N,C,e*C)\\ &\leq b^{i,j}_{error}(N,C,e*C)\\ &=: b^{i,j}_{erate}(N,e) \quad \text{mit } j \in \{_{NC,LTI,LTII}\},\ i \in \{_{BNI,BNII}\} && (6.28) \end{aligned}$$

## 6.3 Maximale K-Last bei festgelegter Service-Qualität

In den bisherigen Betrachtungen des vorangegangenen Abschnitts 6.2 ist für eine gegebene Anzahl von Datenströmen $N$, deren Charakteristika ($E[S_K]$, $Var[S_K]$) sowie den Plattenparametern die Restwahrscheinlichkeit $p_{erate}$ einer gegebenen Störungsrate $e$ berechnet worden.

In umgekehrter Weise läßt sich bei einer vorgegebenen Schranke $\delta_{erate}$ der Restwahrscheinlichkeit und einer Störungsrate $\epsilon_{erate}$ die maximale Anzahl tolerierbarer Datenströme $N_{max}$ bestimmen. Diese Anzahl $N_{max}$ darf pro Runde und Magnetplatte von der Zulassungskontrolle maximal akzeptiert werden, damit die festgelegte Service-Qualität eingehalten werden kann.

Aufgrund der in den analytischen Berechnungen durchgeführten z.T. konservativen Approximationen kann nur der Wert $N^{\approx}_{max}$ berechnet werden, der eine untere Schranke des maximal möglichen Wertes $N_{max}$ darstellt. Es gilt:

$$\begin{aligned} N_{max} &:= \max\{\ N\ :\ p_{erate}(N,\epsilon_{erate}) \leq \delta_{erate}\ \} && (6.29)\\ &\leq \max\{\ N \mid b^{i,j}_{erate}(N,\epsilon_{erate}) \leq \delta_{erate}\ \} \quad \text{mit } j \in \{_{NC,LTI,LTII}\},\ i \in \{_{BNI,BNII}\}\\ &=: N^{\approx}_{max} && (6.30) \end{aligned}$$

Der Schwellwert $\delta_{erate}$ und die Störungsrate $\epsilon_{erate}$ beschreiben die Service-Qualität, die bei $N^{\approx}_{max}$ Datenströmen pro Runde mindestens garantiert werden kann. Da keine geschlossene Formel zur Berechnung von $N^{\approx}_{max}$ zur Verfügung steht, muß die Suche durch Einsetzen verschiedener Werte für $N$ so lange erfolgen, bis das Maximum gefunden ist. Dies kann effizient z.B. durch eine binäre Suche geschehen.

## 6.4 Evaluation des analytischen Modells

Um die Genauigkeit der in den vorangegangenen Abschnitten angegebenen Berechnungen zu bestimmen, werden exemplarisch für einige System- und Lastkonfigurationen konkrete Werte berechnet. Diese Werte werden mit Meßergebnissen verglichen, die durch Simulationen ermittelt werden. Es werden die Abschätzung der Gesamtbedienzeit und die Abschätzung der Störungsrate für K-Aufträge untersucht. Zuvor erfolgt eine Beschreibung des Simulationsmodells.

### 6.4.1 Parameter der Simulation und der Analyse

#### Die Plattenparameter

Das in diesem Abschnitt von Simulation und Analyse verwendete Plattenmodell basiert auf dem, das bereits in Abschnitt 4.1 beschrieben worden ist. Es benutzt dasselbe Modell für Positionierungszeit und Rotationsverzögerung. Das Modell für die Transferzeit wurde dahingehend erweitert, daß neben Ein-Zonen-Platten auch Mehr-Zonen-Platten mit einer variablen Transferrate betrachtet werden können. Die Parameter der Mehr-Zonen-Platte sind in Tabelle 5.1 zu finden. Es wird eine konstante Zylinderanzahl pro Zone von $448$ und eine linear steigende Spurkapazität sowie Transferrate angenommen, die sich nach den Gleichungen 5.13 und 5.35 berechnen. Die Transferzeit in der $i$-ten Zone ($i \in \{1, \ldots, P_{zones}\}$) bei der Auftragsgröße $s$ in Bytes beträgt somit in Sekunden:

$$t^i_{trans}(s) = \frac{2.278 * 10^{-4} * s}{1523 + 73 * i}$$

#### Die Datenparameter

Für die Evaluation ist lediglich der Zugriff auf kontinuierliche Daten notwendig. Wie in den Simulationen in Kapitel 4 werden die Fragmente nur virtuell gespeichert und bei jedem Zugriff Position und Auftragsgröße zufällig bestimmt. Die Position wird durch die gleichverteilte Auswahl eines Blocks bestimmt, für den sich die Zone und der Zylinder berechnen läßt.

Die Größe der Fragmente wird in zwei Szenarien variiert:

- Szenario $I_K$:
  Das erste Szenario geht von einer mittleren gammaverteilten Auftragsgröße von $E[S_K] = 200000$ Bytes und einer Varianz von $Var[S_K] = (100000 \text{ Bytes})^2$ aus. Dies entspricht der Charakteristik der Fragmente eines MPEG-1-kodierten Datenstromes.

- Szenario $II_K$:
  Das zweite Szenario betrachtet Auftragsgrößen, wie sie bei MPEG-2-kodierten Datenströmen auftreten, mit $E[S_K] = 800000$ Bytes und $Var[S_K] = (200000 \text{ Bytes})^2$. Diese Werte sind bereits in Kapitel 4 verwendet worden.

**Die Lastparameter**

Die K-Last wird in den Experimenten variiert, d.h., es werden mehrere Experimente für jeweils eine unterschiedliche Anzahl von K-Aufträgen pro Runde durchgeführt. Innerhalb eines Experiments bleibt die Anzahl der K-Aufträge $N$ in jeder Runde konstant. Die Rundendauer $T$ hat einen konstanten Wert von einer Sekunde. Eine D-Last wird in der Simulation nicht erzeugt, da sie die untersuchten Metriken bei Verwendung geeigneter Scheduling-Algorithmen nicht beeinflußt.

### 6.4.2 Evaluation des Modells bei Abschätzung der Gesamtbedienzeit

In den Simulationsläufen der ersten Untersuchung wird die in einer Runde benötigte Gesamtbedienzeit der K-Aufträge gemessen und die Häufigkeit bestimmt, mit der die Rundendauer überschritten wird. Hiermit wird die Restwahrscheinlichkeit $p_{late}$ geschätzt, die dann mit den analytisch berechneten Werten von $b_{late}^{LTI}$, $b_{late}^{LTII}$ und $b_{late}^{NC}$ verglichen werden kann.

In Tabelle 6.2 sind die Ergebnisse der Messungen und der analytischen Berechnungen für eine Ein- bzw. Mehr-Zonen-Platte mit der Auftragsgröße aus Szenario $I_K$ für die verschiedenen Abschätzungsmethoden aufgeführt. Der Vergleich mit den in der Simulation gemessenen Werten zeigt, daß bei der Durchführung der numerischen Faltung die genaueste Abschätzung für $p_{late}$ erzielt werden kann. Die geringe Abweichung der Approximation $b_{late}^{NC}$ vom gemessenen Wert $p_{late}$ wird durch die konservative Abschätzung der Positionierungszeit mit dem maximal möglichen Wert hervorgerufen.

Die mit Hilfe der Chernoff-Schranke berechneten Schranken $b_{late}^{LTI}$ und $b_{late}^{LTII}$ zeigen größere Abweichungen von den gemessenen Werten. Dieses ist durch die Abschätzung bedingt, die bei den gegebenen Parametern keine exaktere Berechnung zuläßt. Erkennbar wird dies, wenn durch Anwendung der Methode LT II eine exaktere Modellierung des tatsächlichen Positionierungszeitverhaltens des SCAN-Algorithmus, der in der Simulation benutzt wird, Verwendung findet. Die Meßergebnisse zeigen, daß darüber hinaus die Güte der Abschätzung nicht mehr wesentlich verbessert werden kann. Eine exaktere Modellierung kann in diesem Fall keine relevante Steigerung der Abschätzungsgenauigkeit bewirken.

Des weiteren kann beobachtet werden, daß bei Ein-Zonen-Platten die Intervalle für $N$, in denen Werte zwischen Null und eins berechnet werden, für die Chernoff-Schranken $b_{late}^{LTI}$ und $b_{late}^{LTII}$ mit $[24, 33]$ im Vergleich zu $p_{late}$ mit $[26, 40]$ oder $b_{late}^{NC}$ mit $[25, 39]$ kleiner sind. In den Ergebnissen für die Mehr-Zonen-Platten tritt diese Beobachtung aufgrund der variablen Transferrate und der damit verbundenen größeren Varianz in der Bedienzeit noch stärker hervor.

Tabelle 6.3 zeigt analoge Ergebnisse mit den Auftragsgrößen aus Szenario $II_K$. Aufgrund des geringeren Variationskoeffizienten ($\sqrt{Var[S_K]}/E[S_K]$) von $0.25$ im Vergleich zu $0.5$ für Szenario $I_K$ ist die Varianz der Restwahrscheinlichkeit $p_{late}$ geringer.

| Auftragsgröße: $E[S_K] = 200000$ Bytes, $Var[S_K] = (100000 \text{ Bytes})^2$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Ein-Zonen-Platte | | | | Mehr-Zonen-Platte | | |
| N | $p_{late}(N)$ | $b_{late}^{LTI}(N)$ | $b_{late}^{LTII}(N)$ | $b_{late}^{NC}(N)$ | $p_{late}(N)$ | $b_{late}^{LTI}(N)$ | $b_{late}^{NC}(N)$ |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0.00005 | 0.00003 | 0 | 0 | 0.00010 | 0.00001 |
| 25 | 0 | 0.00036 | 0.00021 | 0.00003 | 0.00003 | 0.00064 | 0.00006 |
| 26 | 0.00012 | 0.00210 | 0.00132 | 0.00020 | 0.00025 | 0.00325 | 0.00032 |
| 27 | 0.00060 | 0.00973 | 0.00641 | 0.00106 | 0.00121 | 0.01329 | 0.00150 |
| 28 | 0.00263 | 0.03589 | 0.02490 | 0.00454 | 0.00433 | 0.04428 | 0.00578 |
| 29 | 0.01033 | 0.10575 | 0.07763 | 0.01583 | 0.01485 | 0.12022 | 0.01848 |
| 30 | 0.03187 | 0.25116 | 0.19527 | 0.04545 | 0.04000 | 0.26830 | 0.04952 |
| 31 | 0.08217 | 0.48146 | 0.39816 | 0.10805 | 0.09233 | 0.49273 | 0.11188 |
| 32 | 0.17088 | 0.75051 | 0.66108 | 0.21620 | 0.18189 | 0.74968 | 0.21611 |
| 33 | 0.31073 | 0.95308 | 0.89764 | 0.36804 | 0.31587 | 0.94639 | 0.36026 |
| 34 | 0.47617 | 1 | 1 | 0.54268 | 0.47422 | 1 | 0.52643 |
| 35 | 0.64740 | 1 | 1 | 0.70896 | 0.63755 | 1 | 0.68774 |
| 36 | 0.79238 | 1 | 1 | 0.83892 | 0.77756 | 1 | 0.81843 |
| 37 | 0.89376 | 1 | 1 | 0.92311 | 0.87968 | 1 | 0.90760 |
| 38 | 0.95361 | 1 | 1 | 0.96872 | 0.94503 | 1 | 0.95931 |
| 39 | 0.98242 | 1 | 1 | 0.98904 | 0.97703 | 1 | 0.98435 |
| 40 | 0.99459 | 1 | 1 | 1 | 0.99227 | 1 | 0.99483 |
| 41 | 1 | 1 | 1 | 1 | 0.99755 | 1 | 0.99852 |
| 42 | 1 | 1 | 1 | 1 | 0.99940 | 1 | 1 |
| 43 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Tabelle 6.2: Vergleich von $p_{late}$ und $b_{late}^{m}$ bei Ein- bzw. Mehr-Zonen-Platten für Szenario $I_K$

| Auftragsgröße: $E[S_K] = 800000$ Bytes, $Var[S_K] = (200000 \text{ Bytes})^2$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Ein-Zonen-Platte | | | | Mehr-Zonen-Platte | | |
| N | $p_{late}(N)$ | $b_{late}^{LTI}(N)$ | $b_{late}^{LTII}(N)$ | $b_{late}^{NC}(N)$ | $p_{late}(N)$ | $b_{late}^{LTI}(N)$ | $b_{late}^{NC}(N)$ |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0.00018 | 0.00002 |
| 8 | 0.00023 | 0.00444 | 0.00318 | 0.00045 | 0.00200 | 0.01606 | 0.00180 |
| 9 | 0.02258 | 0.17527 | 0.14286 | 0.02916 | 0.04471 | 0.25807 | 0.04637 |
| 10 | 0.26422 | 0.88822 | 0.83524 | 0.30407 | 0.28992 | 0.90097 | 0.31178 |
| 11 | 0.76069 | 1 | 1 | 0.79585 | 0.71875 | 1 | 0.74687 |
| 12 | 0.97778 | 1 | 1 | 0.98342 | 0.95564 | 1 | 0.96323 |
| 13 | 0.99941 | 1 | 1 | 0.99966 | 0.99756 | 1 | 0.99805 |
| 14 | 1 | 1 | 1 | 1 | 0.9997 | 1 | 0.99996 |
| 15 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Tabelle 6.3: Vergleich von $p_{late}$ und $b_{late}^{m}$ bei Ein- bzw. Mehr-Zonen-Platten für Szenario $II_K$

### 6.4.3 Evaluation des Modells bei Abschätzung der Störungsrate

Die nachfolgende zweite Untersuchung soll wieder anhand einer exemplarischen System- und Lastkonfiguration aufzeigen, wie groß die Abweichungen zwischen den analytisch ermittelten Werten und den aus einer Simulation gewonnenen Meßwerten sind. Es wird die durch die Simulationen geschätzte Restwahrscheinlichkeit $p_{erate}$ mit den analytisch berechneten Schranken $b_{erate}^{i,j}$ verglichen.

Im voraus läßt sich hinsichtlich der Genauigkeit der Abschätzungen $b_{erate}^{i,j}$ unter Berücksichtigung der in Abschnitt 6.4.2 ermittelten Ergebnisse bereits die genaueste und die gröbste Abschätzung bestimmen. Die genaueste Abschätzung von $p_{erate}$ wird durch Verwendung der Werte von $b_{erate}^{BNI,NC}$ berechnet, da $b_{late}^{NC}$ die geringste Abweichung von $p_{late}$ aufweist und durch $b_{error}^{BNI,j}$ in Gleichung 6.26 die Restwahrscheinlichkeit der Binomialverteilung aus Gleichung 6.23 exakt berechnet wird. Die gröbste Abschätzung liefert $b_{erate}^{BNII,LTI}$, da zum einen die Werte von $b_{late}^{LTI}$ in den Meßergebnissen im Abschnitt 6.4.2 durch Anwendung der Chernoff-Schranke in Gleichung 6.18 die größte Abweichung besitzen und zum anderen die Anwendung der Chernoff-Schranke in Gleichung 6.26 zur Abschätzung der Restwahrscheinlichkeit der Binomialverteilung eine weitere Fehlerquelle ist.

Um die Anzahl der zu untersuchenden Fälle übersichtlich zu halten, werden im folgenden nur Werte für $b_{erate}^{BNI,NC}$ und $b_{erate}^{BNII,LTI}$ berechnet, wobei die in Abschnitt 6.4.2 analytisch ermittelten Werte als Grundlage dienen. Die Parameter des verwendeten Daten- und Plattenmodells entsprechen somit denen aus Abschnitt 6.4.2. Es wird eine Wiedergabedauer der Datenströme von $1200$ bzw. $3600$ Runden angenommen. Jede Runde hat eine Dauer von einer Sekunde. Die tolerierte Störungsrate $\epsilon_{erate}$ wird auf $1\%$ festgelegt. Es erfolgt eine exemplarische Untersuchung anhand einer Ein- und einer Mehr-Zonen-Platte.

Ausschlaggebend für die Bewertung soll im folgenden die Abweichung zwischen der K-Last sein, die das analytische Modell und damit auch die Zulassungskontrolle (siehe Abschnitt 6.3) bei festgelegter Schranke $\delta_{erate}$ und Störungsrate $e$ maximal erlaubt, und der K-Last, die in der Simulation unter Einhaltung der festgelegten Service-Qualität maximal möglich ist. Es wird also die Abweichung zwischen $N_{max}^{\approx}$ aus Gleichung 6.30 und $N_{max}$ aus Gleichung 6.30 betrachtet und damit der Grad festgestellt, inwieweit der Server hinsichtlich K-Aufträge unausgelastet bleiben würde.

In Tabelle 6.4 sind die Ergebnisse für eine Ein-Zonen-Platte mit MPEG-2-charakteristischen Fragmentgrößen aus Szenario $II_K$ aufgeführt. Zu erkennen ist, daß eine Erhöhung der Anzahl der Datenströme $N$ um den Wert $1$ bereits eine Erhöhung der Restwahrscheinlichkeit $p_{erate}(N, 1\%)$ von nahezu $0$ auf einen Wert von nahezu $1$ nach sich ziehen kann. Für die Festlegung von $N_{max}^{\approx}$ spielt hierdurch der Schwellwert $\delta_{erate}$ eine untergeordnete Größe. Dunkel hinterlegt sind die Werte von $b_{erate}^{BNI,NC}$, $b_{erate}^{BNII,LTI}$ und $p_{late}$, die den Schwellwert $\delta_{erate}$ von festgelegten $5\%$ bei maximalem $N$ unterschreiten.

Hiermit ergibt sich aus den Simulationsergebnissen die maximale Anzahl paralleler Datenströme zu $N_{max} = 9$, für sowohl $C = 1200$ als auch $C = 3600$ Runden Wiedergabedauer. Die analytische Berechnung liefert bei der gröbsten und der genauesten Abschätzung jeweils den Wert $N_{max} = 8$ und bewirkt somit einen absoluten Abschätzungsfehler von $1$. Der relative Fehler beträgt hierbei $11.1\%$. Die größere Berechnungsgenauigkeit verbunden mit dem größeren Berechnungsaufwand zur Bestimmung von $b_{erate}^{BNI,NC}$ im Vergleich zu $b_{erate}^{BNII,LTI}$ hat bei den gegebenen Parametern keinen positiven Einfluß auf das Abschätzungsergebnis.

| Ein-Zonen-Platte, Auftragsgröße: $E[S_K] = 800000$ Bytes, $Var[S_K] = (200000 \text{ Bytes})^2$ | | | | | | |
|---|---|---|---|---|---|---|
| | $C$ = 1200 Runden | | | $C$ = 3600 Runden | | |
| N | $p_{erate}(N, 1\%)$ | $b_{erate}^{BNII,LTI}$ | $b_{erate}^{BNI,NC}$ | $p_{erate}(N, 1\%)$ | $b_{erate}^{BNII,LTI}$ | $b_{erate}^{BNI,NC}$ |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0.04551 | 0 | 0 | 0.00009 | 0 |
| 9 | 0.00008 | 1 | 1 | 0 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 | 1 |

Tabelle 6.4: Ein-Zonen-Platte: Vergleich von Simulationsergebnissen und berechneter Abschätzung für $p_{erate}$ in Szenario $II_K$

| Ein-Zonen-Platte, Auftragsgröße: $E[S_K] = 200000$ Bytes, $Var[S_K] = (100000 \text{ Bytes})^2$ | | | | | | |
|---|---|---|---|---|---|---|
| | $C$ = 1200 Runden | | | $C$ = 3600 Runden | | |
| N | $p_{erate}(N, 1\%)$ | $b_{erate}^{BNII,LTI}$ | $b_{erate}^{BNI,NC}$ | $p_{erate}(N, 1\%)$ | $b_{erate}^{BNII,LTI}$ | $b_{erate}^{BNI,NC}$ |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 0 | 0 | 0 | 0 | 0.98659 | 0 |
| 28 | 0 | 0.05277 | 0.01010 | 0 | 1 | 0.00002 |
| 29 | 0 | 1 | 0.96630 | 0 | 1 | 0.99889 |
| 30 | 0 | 1 | 1 | 0 | 1 | 1 |
| 31 | 0.0015 | 1 | 1 | 0 | 1 | 1 |
| 32 | 0.3309 | 1 | 1 | 0.28364 | 1 | 1 |
| 33 | 0.9858 | 1 | 1 | 0.99980 | 1 | 1 |
| 34 | 1 | 1 | 1 | 1 | 1 | 1 |

Tabelle 6.5: Ein-Zonen-Platte: Vergleich von Simulationsergebnissen und berechneter Abschätzung für $p_{erate}$ in Szenario $I_K$

Dieses macht sich erst bei kleinerer, d.h. MPEG-1-charakteristischer Fragmentgröße bemerkbar. Aus Tabelle 6.5 ergibt sich bei der Festlegung von $\delta_{erate}$ auf $5\%$, $1200$ Runden Wiedergabedauer und genauer Abschätzung die maximal pro Runde bedienbare Anzahl an Datenströmen zu $N_{max}^{\approx} = 28$; dies entspricht einem relativen Fehler von ca. $10\%$ bei $31$ tatsächlich möglichen parallelen Datenströmen. Die grobe Abschätzung erhöht den relativen Fehler auf ca. $13\%$, der sich weiter auf ca. $16\%$ vergrößert, wenn die längere Wiedergabedauer von $3600$ Runden betrachtet wird.

In den Tabellen 6.6 und 6.7 zeigt sich, daß sich der Abschätzungsfehler bei den Berechnungen für die Mehr-Zonen-Platte vergrößert, wenn das grobe Abschätzungsverfahren verwendet wird. Dieses berechnet bei MPEG-2-spezifischer Fragmentgröße einen Wert von $N_{max}^{\approx} = 7$ bei praktisch möglichem $N_{max} = 9$. Im Vergleich zu den ca. $22\%$ Abweichung kann mit größerem Berechnungsaufwand der relative Fehler auf ca. $11\%$ gesenkt werden. Die größte Abweichung zwischen gemessenem und analytisch berechnetem Wert unter allen betrachteten Parametern tritt in Tabelle 6.7 bei MPEG-1-spezifischer Fragmentgröße, Mehr-Zonen-Platte und $1200$ Runden Wiedergabedauer auf. Die absolute Abweichung beträgt hierbei $5$, was einem relativen Abschätzungsfehler von ca. $16\%$ entspricht.

| Mehr-Zonen-Platte, Auftragsgröße: $E[S_K] = 800000$ Bytes, $Var[S_K] = (200000 \text{ Bytes})^2$ | | | | | | |
|---|---|---|---|---|---|---|
| | $C$ = 1200 Runden | | | $C$ = 3600 Runden | | |
| N | $p_{erate}(N, 1\%)$ | $b_{erate}^{BNII,LTI}$ | $b_{erate}^{BNI,NC}$ | $p_{erate}(N, 1\%)$ | $b_{erate}^{BNII,LTI}$ | $b_{erate}^{BNI,NC}$ |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 1 | 0 | 0 | 1 | 0 |
| 9 | 0.01169 | 1 | 1 | 0.00060 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 | 1 |

Tabelle 6.6: Mehr-Zonen-Platte: Vergleich von Simulationsergebnissen und berechneter Abschätzung für $p_{erate}$ in Szenario $II_K$

| Mehr-Zonen-Platte, Auftragsgröße: $E[S_K] = 200000$ Bytes, $Var[S_K] = (100000 \text{ Bytes})^2$ | | | | | | |
|---|---|---|---|---|---|---|
| | $C$ = 1200 Runden | | | $C$ = 3600 Runden | | |
| N | $p_{erate}(N, 1\%)$ | $b_{erate}^{BNII,LTI}$ | $b_{erate}^{BNI,NC}$ | $p_{erate}(N, 1\%)$ | $b_{erate}^{BNII,LTI}$ | $b_{erate}^{BNI,NC}$ |
| 25 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | 0 | 0.00445 | 0 | 0 | 0 | 0 |
| 27 | 0 | 1 | 0 | 0 | 1 | 0 |
| 28 | 0 | 1 | 0.05000 | 0 | 1 | 0.00151 |
| 29 | 0 | 1 | 0.99343 | 0 | 1 | 1 |
| 30 | 0 | 1 | 1 | 0 | 1 | 1 |
| 31 | 0.00874 | 1 | 1 | 0 | 1 | 1 |
| 32 | 0.46332 | 1 | 1 | 0.53704 | 1 | 1 |
| 33 | 0.99242 | 1 | 1 | 1 | 1 | 1 |
| 34 | 1 | 1 | 1 | 1 | 1 | 1 |

Tabelle 6.7: Mehr-Zonen-Platte: Vergleich von Simulationsergebnissen und berechneter Abschätzung für $p_{erate}$ in Szenario $I_K$

### 6.4.4 Vergleich mit deterministischen Modellen

Die Vorteile des stochastischen Modells zeigen sich, wenn die Berechnung der maximalen Anzahl paralleler Datenströme, wie in Abschnitt 6.3 beschrieben, auf Basis eines deterministischen (worst-case) Analysemodells durchführt wird. Dieses deterministische Modell garantiert zwar mit $100\%$ Sicherheit eine Störungsrate von $0\%$. Hierzu muß allerdings für die Aufträge die längstmögliche Gesamtbedienzeit angenommen werden. Die maximale Anzahl $N_{max}^{wc}$ an Datenströmen, die hierbei in einer Runde zugelassen werden können, ergibt sich zu:

$$N_{max}^{wc} = \max \left\{ N \ : \ t_{pos}^{max}(N) + N * P_{rot} + N * \frac{s}{v} \right\}$$

Verwendet man als Auftragsgröße für $s$ ein $99\%$-Quantil und für $v$ die kleinste Transferrate $P_{rate}^{min} = \frac{P_{cap}^{min}}{P_{rot}}$, so berechnet das deterministische Modell maximal $4$ mögliche MPEG-2 Datenströme oder $11$ mögliche MPEG-1 Datenströme (siehe Tabelle 6.8). Im Vergleich hierzu erlaubt das stochastische Modell mit den Parameter aus Abschnitt 6.4.3 mindestens $7$ bzw.

| $N_{max}^{wc}$ | $E[S_K] = 800000Bytes$<br>$Var[S_K] = (200000Bytes)^2$ | $E[S_K] = 200000Bytes$<br>$Var[S_K] = (100000Bytes)^2$ |
|---|---|---|
| $s = S_{max}^{99\%}$<br>$v = \frac{P_{cap}^{min}}{P_{rot}}$ | 4 | 11 |

| $N_{max}^{wc}$ | $E[S_K] = 800000Bytes$<br>$Var[S_K] = (200000Bytes)^2$ | $E[S_K] = 200000Bytes$<br>$Var[S_K] = (100000Bytes)^2$ |
|---|---|---|
| $s = S_{max}^{95\%}$<br>$v = \frac{P_{cap}^{min} + P_{cap}^{max}}{2 * P_{rot}}$ | 7 | 18 |

Tabelle 6.8: Anzahl möglicher Datenströme bei deterministischem Analysemodell

26 Datenströme (siehe Tabelle 6.6 und 6.7). Selbst wenn die Auftragsgröße $s$ auf ein $95\%$-Quantil gesenkt wird und eine optimistische Transferrate $v$ von $\frac{P_{cap}^{min}+P_{cap}^{max}}{2*P_{rot}}$ vorausgesetzt wird, würde durch das deterministische Analysemodell die Anzahl paralleler Datenströme auf 7 bzw. 18 beschränkt bleiben. Hierdurch zeigen sich bei diesem Beispiel klar die Vorteile eines stochastischen Analysemodells. Die Anzahl zugelassener Datenströme läßt sich bei stochastisch garantierter Service-Qualität in bestimmten Fällen mehr als verdoppeln.

## 6.5 Fazit

In diesem Kapitel ist gezeigt worden, wie die Service-Qualität für kontinuierliche Datenströme berechnet werden kann. Die dabei verwendeten Verfahren unterscheiden sich hinsichtlich ihres Ansatzes und den für die praktischen Berechnungen notwendigerweise durchzuführenden Approximationen.

In der praktischen Evaluation hat sich herausgestellt, daß mit sehr großem Rechenaufwand, d.h. mit der numerischen Berechnung der Faltung, letztendlich die Restwahrscheinlichkeit der Störungsrate auch am genauesten abgeschätzt werden kann. Der Genauigkeitsvorteil gegenüber dem weniger aufwendigen Verfahren mit Abschätzung der Laplace-Transformierten durch die Chernoff-Schranke ist aber gering und wirkt sich nur dann aus, wenn die Varianz der Gesamtbedienzeit innerhalb einer Runde genügend groß ist. Dieses kann z.B. durch eine Mehr-Zonen-Platte mit variabler Transferrate oder kleine Fragmente mit großer Varianz in der Größe hervorgerufen werden.

Im Vergleich zu einer deterministischen Modellierung überwiegen trotz der Abschätzungfehler die Vorteile des stochastischen Modells, da sich durch eine stochastische Zulassungskontrolle die Auslastung des Daten-Servers steigern läßt und somit die zur Verfügung stehende Kapazität eines Daten-Servers in weitaus stärkerem Maße ausgenutzt werden kann.

# Kapitel 7

# Analytische Berechnung der Antwortzeit

Die Antwortzeit beschreibt die Service-Qualität des Daten-Servers hinsichtlich diskreter Datenzugriffe. Um, wie im vorangegangenen Kapitel, diese Service-Qualität durch mathematische Methoden berechnen und somit vorhersagen zu können, wird ein Warteschlangenmodell verwendet, aus dessen stochastischer Analyse sich eine Aussage über die Verteilung der Antwortzeiten gewinnen läßt.

Um brauchbar genaue Vorhersagen zu erhalten, muß das Verhalten der verwendeten Warteschlangenmodelle möglichst genau dem Verhalten der im Daten-Server verwendeten Scheduling-Algorithmen nachempfunden werden. In Abschnitt 7.1 soll deshalb zunächst eine Reihe verschiedener aus der Literatur bekannter Warteschlangenmodelle betrachtet werden, die zur Modellierung der Scheduling-Algorithmen in Betracht gezogen werden können. In Abschnitt 7.2 erfolgt die Untersuchung eines konkreten Warteschlangenmodells mit der Herleitung der Antwortzeitverteilung in Form der Laplace-Transformierten. Diese Ergebnisse werden in Abschnitt 7.3 verwendet, um die maximale K-Last zu bestimmen, die bei gegebenen System- und Datenparametern möglich ist. In Abschnitt 7.4 werden die analytisch berechneten Vorhersagen des Modells mit den experimentellen Meßergebnissen eines Simulationsmodells verglichen. Der Abschnitt 7.5 schließt dieses Kapitel mit einem kurzen Fazit ab.

## 7.1 Modellierungsansätze

In diesem Abschnitt sollen Möglichkeiten aufgezeigt werden, wie die in Abschnitt 3.2 entwickelten Scheduling-Algorithmen auf in der Literatur bekannte Warteschlangenmodelle in ”erster Näherung“ abgebildet und welche Ansätze zur Anpassung der Modelle verfolgt werden können.

### 7.1.1 Warteschlangenmodelle mit Service-Unterbrechungen

Scheduling-Algorithmen mit getrennter Einteilungsstrategie (siehe Abschnitt 3.2), die die Bedienung der diskreten und kontinuierlichen Datenzugriffe in voneinander getrennten Bedienabschnitten durchführen, lassen sich näherungsweise auf M/G/1-Warteschlangenmodelle mit

Service-Unterbrechungen (engl. vacations) [Tak91] abbilden. Diese Modelle bilden eine erweiterte Klasse der üblichen M/G/1-Warteschlangenmodelle. Im Unterschied zum Standardfall unterbrechen sie die Bedienung der Aufträge zu bestimmten Zeitpunkten für eine festgelegte Zeitdauer. Die Zeitintervalle, in denen eine Service-Unterbrechung stattfindet, werden als *Urlaubsperioden* (engl. vacation periods); die Zeitintervalle, in denen die Bearbeitung der Aufträge stattfindet, werden als *Arbeitsperioden* (engl. busy periods) bezeichnet. Der Server des Modells befindet sich stets in einer dieser beiden Perioden, die sich zeitlich aufeinanderfolgend abwechseln.

Die Abbildung der zyklischen Arbeitsweise der Scheduling-Algorithmen auf ein solches M/G/1-Vacation-Modell ist naheliegend. Der Bedienabschnitt, in dem der Scheduling-Algorithmus die diskreten Datenzugriffe ausführt, entspricht der Arbeitsperiode des M/G/1-Modells. Analog entspricht der Bedienabschnitt, in dem der Scheduling-Algorithmus kontinuierliche Datenzugriffe ausführt, der Urlaubsperiode des Modells. Den veröffentlichten Resultaten für das M/G/1-Vacation-Modell, z.B. über Antwortzeitverteilungen, liegt eine FCFS-Bearbeitungsreihenfolge der Aufträge zugrunde. Dies entspricht den Scheduling-Algorithmen, die eine FCFS-Anordnung der D-Aufträge benutzen.

Die Bedingung, wann im M/G/1-Vacation-Modell eine Service-Unterbrechung stattfindet, d.h., wann von der Arbeitsperiode in die Urlaubsperiode gewechselt wird, hängt von der konkreten Bedienstrategie des Warteschlangenmodells ab. In der Literatur [FW88, Dos90, Tak91] werden diesbezüglich verschiedene Modelle unterschieden, die im folgenden hinsichtlich ihrer Anwendbarkeit auf Scheduling-Strategien mit getrennter Einteilungsstrategie und FCFS-Anordnung der Aufträge diskutiert werden sollen.

#### M/G/1-Vacation-Modell mit Exhaustive Service

Der Server eines M/G/1-Vacation-Modells mit *Exhaustive Service* [Tak91, Lee84] geht in die Urlaubsperiode, in der er für eine gewisse Zeit verweilt, wenn die Warteschlange mit Aufträgen komplett abgearbeitet worden ist. Nach Ablauf der Urlaubsperiode wird erneut geprüft, ob sich Aufträge in der Warteschlange befinden. Abhängig von der Bedingung, wann die Serviceunterbrechung beendet wird, d.h., wann der Server seine Arbeit wieder aufnehmen soll, wird gewartet, bis ein neuer Auftrag eintrifft (*single vacation*), oder es wird sofort eine weitere Urlaubsperiode (*multiple vacation*) begonnen. In [NMW97a] ist die erwartete Antwortzeit eines M/G/1-Vacation-Modells mit *multiple vacations* analytisch berechnet und mit den experimentell ermittelten mittleren Antwortzeiten eines Simulationmodells verglichen worden. Hierbei ist sichtbar geworden, daß ein M/G/1-Vacation-Modell mit *exhaustive service* nur ungenügend das periodische Verhalten des zu modellierenden Scheduling-Algorithmus wiedergibt.

Die Ursache hierfür ist, daß die Häufigkeit und die Dauer der Urlaubsperiode bei diesem Modell an die Ankunftsrate der D-Aufträge gekoppelt ist. Bei geringer Last, d.h. bei einer kleinen Ankunftsrate, ist die Warteschlange häufig leer und der Server befindet sich dementsprechend oft in einer Urlaubsperiode (*multiple vacation*). Der Scheduling-Algorithmus müßte vermeintlich mehrere K-Abschnitte pro Runde mit entsprechend schlechten Antwortzeiten für D-Aufträge einlegen. Tatsächlich geschieht dieses aber nur genau ein einziges Mal pro Runde. Die Abbildung 7.1 illustriert die Abfolge von Arbeits- und Urlaubsperioden im Modell bei niedriger Ankunftsrate unter der Annahme konstanter Bedienzeiten und einer konstanten Urlaubsperiode.

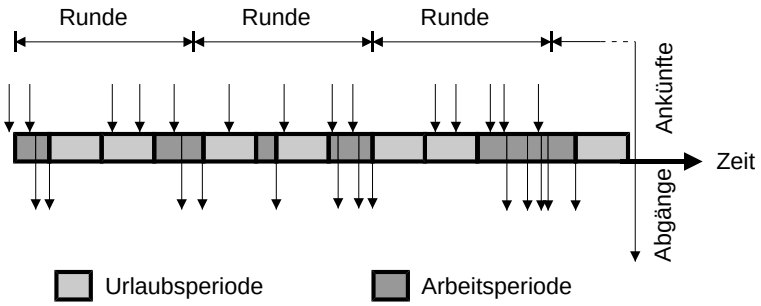



Abbildung 7.1: Periodeneinteilung beim M/G/1-Vacation-Modell mit *exhaustive service* bei niedriger Ankunftsrate

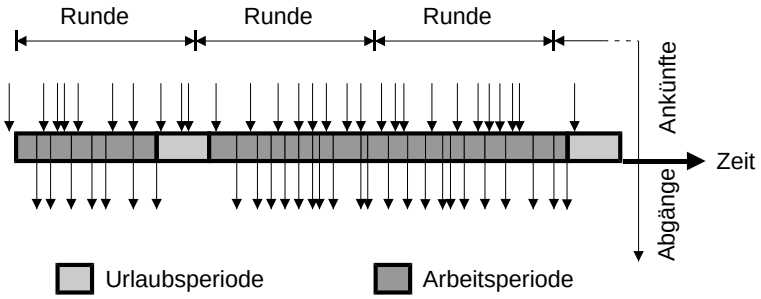



Abbildung 7.2: Periodeneinteilung beim M/G/1-Vacation-Modell mit *exhaustive service* bei hoher Ankunftsrate

Bei hoher Last, d.h. bei einer relativ zur Leistungskapazität großen Ankunftsrate, leert sich die Warteschlange nur selten und es werden nur wenige Urlaubsperioden eingelegt (siehe Abbildung 7.2). In diesem Fall kann es passieren, daß das Modell in einer Runde keine Urlaubsperiode vorsieht, obwohl der Scheduling-Algorithmus von genau einem K-Bedienabschnitt pro Runde ausgeht. Bei sehr hoher Auslastung in der Nähe der Leistungskapazität passiert es, daß der Server überhaupt keine Urlaubsperioden einlegt.

Insgesamt ergeben sich somit sowohl bei niedriger als auch bei hoher Last große Abweichungen zwischen den analytisch berechneten und den in der Simulation experimentell gemessenen Antwortzeiten. Lediglich in einem Bereich zwischen hoher und niedriger Last, der quantitativ nicht genauer spezifiziert werden kann, sind brauchbare Ergebnisse zu erwarten. Dieses Warteschlangenmodell ist für Vorhersagezwecke deshalb ungeeignet.

### M/G/1-Vacation-Modell mit Gated Service

Ein M/G/1-Vacation-Modell mit *Gated Service* bedient nur die Aufträge, die zu Beginn der Arbeitsperiode in der Warteschlange stehen. Hierdurch werden in einer Arbeitsperiode alle diejenigen Aufträge bearbeitet, die nach dem Beginn der letzten Arbeitsperiode und während der Urlaubsperiode den Server erreichen. Im Vergleich zum vorangegangenen Modell kann hierdurch auch bei hoher Last sichergestellt werden, daß der Server nach Abarbeitung endlich vieler Aufträge jeweils eine Urlaubsperiode einlegt.

Hinsichtlich der Vorhersagegenauigkeit ist dieses Modell nicht untersucht worden. Es weist aber vergleichbare Schwächen auf, wie das vorangegangene Warteschlangenmodell. Bei gerin-

ger Ankunftsrate befindet sich der Server häufig in einer Urlaubsperiode, und bei hoher Ankunftsrate wird die Zeitdauer der Arbeitsperiode durch die Anzahl der wartenden Aufträge zu Beginn der Arbeitsperiode bestimmt. Bei letzterem kann die zeitliche Länge der Arbeitsperiode die Länge einer Runde überschreiten. Hinsichtlich der Vorhersageergebnisse lassen sich somit ebenfalls große Abweichungen vermuten.

### M/G/1-Vacation-Modell mit Limited Service

M/G/1-Vacation-Modelle dieses Typs beschränken die Dauer der Arbeitsperiode und damit die Zeitspanne zwischen zwei aufeinanderfolgenden Urlaubsperioden. Dies geschieht direkt durch Angabe einer Zeitschranke (*time-limited service*) [LE91, LL94] oder indirekt durch Angabe der maximalen Anzahl an Aufträgen (*request-limited service*), die pro Arbeitsperiode [Lee89] bearbeitet werden dürfen. Bei letzterem geht der Server in die Urlaubsperiode, sobald eine festgelegte maximale Anzahl an Aufträgen bearbeitet worden ist bzw. wenn bereits vorher die Warteschlange vollständig geleert wurde. In der Literatur werden weitere Untertypen dieser Warteschlangenmodelle unterschieden [Tak91].

**Request-Limited Service** Während einer Arbeitsperiode eintreffende Aufträge werden bei einem Modell mit *gated request-limited service* in die Warteschlange eingefügt und nur zu Beginn der nächsten Arbeitsperiode berücksichtigt. Bei einem Modell mit *exhaustive request-limited service* können die Aufträge, sofern die maximale Anzahl $M_{max}$ noch nicht überschritten wurde, auch während der Arbeitsperiode entsprechend der FCFS-Bearbeitungsreihenfolge bearbeitet werden. Eine Schranke, die eine Zulassung der Aufträge nur zu bestimmten Zeitpunkten zuläßt, existiert hierbei nicht.

Bei allen in der Literatur [Tak91] beschriebenen Modellen dieses Typs ist die Länge der Urlaubsperiode unabhängig von der Länge der Arbeitsperiode. Um das periodische Verhalten der Algorithmen mit getrennter Einteilungsstrategie auf diese M/G/1-Modelle abbilden zu können, ist es deshalb notwendig, die zugrundeliegenden mathematischen Modelle dahingehend zu verändern, daß sich Arbeits- und Urlaubsperiode zu einer Runde von konstanter Zeitdauer ergänzen. Eine solche Modifikation des M/G/1-Vacation-Modells mit *exhaustive limited service* wird basierend auf [Lee89] in der Diplomarbeit [Jan98] vorgestellt. Die Dauer der Urlaubsperiode wird in dem modifizierten Modell so gewählt, daß sie sich zusammen mit der Arbeitsperiode zur konstanten Rundendauer $T$ ergänzt. Dieses Verhalten entspricht einem Scheduling-Algorithmus, dessen getrennte Bedienstrategie zunächst eine maximale Anzahl $M_{max}$ an D-Aufträgen ausführt und dann die verbleibende Rundendauer bis zum Rundenende zur Ausführung von K-Aufträgen nutzt. Sollten weniger als $M_{max}$ D-Aufträge bedient werden können, weil die Warteschlange leer ist, so wird mit der Bedienung der K-Aufträge vorzeitig begonnen und anschließend auf das Rundenende gewartet. Die Bedienung von D-Aufträgen findet erst wieder zu Beginn der nachfolgenden Runde statt.

Im Gegensatz hierzu beginnt der in Abschnitt 3.3 beschriebene Algorithmus I zu Beginn einer Runde mit der Ausführung von K-Aufträgen und nutzt die verbleibende Zeit für D-Aufträge. Die Anzahl der D-Aufträge, die bearbeitet werden, ist nicht statisch begrenzt, und bei leerer Warteschlange im D-Abschnitt wartet der Algorithmus so lange, bis ein D-Auftrag eintrifft oder die Runde abläuft.

Trotz dieser offensichtlichen Unterschiede in der Bedienstrategie zwischen Algorithmus und Warteschlangenmodell läßt sich das Modell zur Abschätzung der Antwortzeit von Algorithmus I nutzen. Hierzu wählt man die Schranke $M_{max}$ so, daß die Wahrscheinlichkeit, daß die Gesamtbedienzeit bei $M_{max}$ D-Aufträgen und $N$ K-Aufträgen die Rundendauer $T$ überschreitet, sehr gering ist, z.B. weniger als $0.1\%$. Mit dieser Parametereinstellung erhält man durch das Warteschlangenmodell eine konservative Abschätzung für die Antwortzeit der D-Aufträge bei Verwendung von Scheduling-Algorithmus I.

Ein Nachteil dieser konservativen Approximation ist, daß der Wert von $M_{max}$ hierbei kleiner gewählt werden muß als die Anzahl der D-Aufträge, die der Scheduling-Algorithmus im Mittel in einer Runde verarbeiten kann.

Die Leistungskapazität des Warteschlangenmodell $M_{max}/T$ ist somit kleiner als die Leistungskapazität des Scheduling-Algorithmus. Hierdurch werden Abweichungen in der Vorhersage unter hoher D-Last hervorgerufen. Um diese Abweichung zu verringern, kann die Anzahl der pro Runde ausführbaren D-Aufträge als Zufallsvariable $M$ modelliert werden. Die Verteilung von $M$ wird u.a. von der Gesamtbedienzeit der $N$ K-Aufträge bestimmt, die zu Beginn der Runde im K-Bedienabschnitt ausgeführt werden. Die Berechnung der Verteilung von $M$ ist auf Basis der in Abschnitt 6.1 erstellten Gleichungen möglich. Durch diese Berechnung entspricht der Erwartungswert von $M$ genau der Anzahl an D-Aufträgen, die der Scheduling-Algorithmus pro Runde im Mittel verarbeiten kann.

Dieses modifizierte Warteschlangenmodell mit variabler Schranke (*limit variation*), in dem sich wiederum Arbeitsperiode und Urlaubsperiode zu einer konstanten Rundendauer $T$ ergänzen müssen, stellt eine Kombination aus denen in [Jan98] und [LaM90, LaM92] (*M/G/1/N vacation model with varying e-limited service discipline*) vorgestellten Warteschlangenmodellen dar.

Die Ausführungen in [Jan98] zeigen allerdings, daß für realistische System- und Datenparameter die Berechnung von Werten nur für sehr kleine Lastparameter unter großem Aufwand durchführbar ist. Dieses läßt vermuten, daß für das erweiterte Modell mit probabilistischer Schranke $M$ die Durchführung der Berechnungen unter vertretbarem Aufwand nicht mehr möglich ist. Auf eine Untersuchung dieses erweiterten Modells wird aus diesem Grund verzichtet.

**Time-Limited Service** Die Arbeit [LL94] beschreibt zwei Warteschlangenmodelle, bei denen die Arbeitsperiode durch eine Zeitschranke begrenzt ist. Im ersten Modell wird eine konstante Zeitschranke gewählt (*constant time-limited service*). Das zweite Modell wählt die Zeitschranke in Abhängigkeit von der vorangegangenen Urlaubsperiode (*vacation-dependent time-limited service*), so daß sich Urlaubs- und Arbeitsperiode zu einer konstanten Rundendauer $T$ ergänzen. Letzteres entspricht exakt dem Verhalten eines Scheduling-Algorithmus mit getrennter Einteilungsstrategie.

In der Berechnung werden konstante Zeitabschnitte, z.B. die konstante Rundendauer, durch eine Folge aufeinanderfolgender Zeitschritte, die durch eine Erlang-Verteilung charakterisiert sind, approximiert. Man erreicht ein Konvergenzverhalten an das tatsächliche Modell durch die Erhöhung der Anzahl der Zeitschritte. Die in [LL94] durchgeführten Berechnungen zeigen, daß eine hinreichende Konvergenz bereits bei wenigen Zeitschritten erreicht werden kann. Allerdings werden in dem verwendeten Beispiel nur einfache Verteilungsfunktionen für Bedienzeit

(Exponentialverteilung) und Urlaubsperiode (konstanter Wert) angenommen, die die Berechnungen vereinfachen und den Berechnungsaufwand auf ein vertretbares Maß reduzieren. Die Verwendung allgemeinerer Verteilungsfunktionen, wie es für die Modellierung der der Magnetplatte und der Last notwendig ist, benötigt dagegen einen enormen Berechnungsaufwand, so daß dieser Ansatz für das verfolgte Ziel nicht praktikabel ist.

### 7.1.2 Modelle mit dynamischer Kapazitätsverteilung

Im Gegensatz zu den bisher betrachteten Warteschlangenmodellen, die versuchen das Verhalten der Scheduling-Algorithmen exakt nachzubilden, wird in [NMP+97] ein Modell entwickelt, das in abstrakter Form die Kapazität des Systems dynamisch zwischen der Ausführung von K-Aufträgen und der Ausführung von D-Aufträgen verteilt.

Zur Ausführung von D-Aufträgen wird zunächst eine Minimalkapazität reserviert. Die restliche Kapazität steht für die Ausführung von K-Aufträgen zur Verfügung. Sollte aufgrund einer geringen Anzahl von K-Aufträgen diese Kapazität nicht vollständig genutzt werden, so wird die ungenutzte Kapazität zur Ausführung von D-Aufträgen neu verteilt. Es wird ein $M/M/1/K$ Warteschlangenmodell mit FCFS-Bearbeitungsreihenfolge und, um die Berechnungen durchführen zu können, beschränkter Warteschlangenlänge verwendet. Die Anzahl der K-Aufträge, die pro Runde bedient werden müssen, schwankt über einen längeren Zeitraum, da hier, im Gegensatz zu den bisherigen Modellen, ein Ankunftsprozeß für Datenströme angenommen wird. Die Kapazitätsverteilung wird durch eine variable Servicerate, d.h. die Rate, mit der D-Aufträge bedient werden, erreicht. Diese Servicerate hängt von der Anzahl der K-Aufträge pro Runde ab, da die nicht von K-Aufträgen genutzte Kapazität auf die Ausführung von D-Aufträgen übertragen wird.

Der Vergleich mit Simulationsergebnissen in [Rom98] zeigt, daß die Vorhersagegenauigkeit stark von der Wahl der Minimalkapazität abhängt, die zur Ausführung von D-Aufträgen reserviert wird. Offen bleibt, wie diese Minimalkapazität bestimmt werden kann.

### 7.1.3 Modelle mit mehreren Ankunftsprozessen

[Ott87] untersucht ein Model, das aus zwei unabhängigen Ankunfts-/Bedienprozessen (GI/G, M/G), einer Warteschlange und einem Server gebildet wird. Die Bearbeitung der Aufträge erfolgt in FCFS-Reihenfolge, ohne daß dabei den Aufträgen einer der beiden Ankunftsprozesse eine höhere Priorität zugeordnet wird.

Um das Verhalten der Scheduling-Algorithmen näherungsweise auf dieses Modell abzubilden, könnte der Ankunftsstrom der D-Aufträge durch den M/G-Prozeß beschrieben werden. Der zweite Prozeß beschreibt das Auftreten der K-Aufträge. Dieses könnte als ein Spezialfall des GI/G-Prozesses durch eine konstante Zwischenankunftszeit modelliert werden. Hierzu werden alle K-Aufträge einer Runde als ein Gesamtauftrag aufgefaßt, der alle $T$ Zeiteinheiten beim Server eintrifft. Unterschiede zwischen Modell und Scheduling-Algorithmus treten dann hervor, wenn sich zu Rundenbeginn eine große Anzahl von D-Aufträgen in der Warteschlange befindet. Aufgrund der FCFS-Auswahl werden diese zunächst abgearbeitet, bevor der Gesamtauftrag mit den einzelnen K-Aufträgen berücksichtigt werden kann. Dieser Modellierungsfehler würde

vermieden werden, wenn dem periodisch eintreffenden Gesamtauftrag eine höhere Priorität gegeben werden könnte. Hierdurch würde, falls notwendig, die Bearbeitung der D-Aufträge mit niedriger Priorität unterbrochen.

Die Untersuchung dieses Modells mit Prioritäten in der Bearbeitung der Aufträge wird in [Sch74] durchgeführt. In der Arbeit werden aber weder Angaben über den Berechnungsaufwand gemacht noch exemplarische Berechnungen anhand konkreter Verteilungsfunktionen durchgeführt. Die praktische Anwendbarkeit des Modells auf den in dieser Arbeit geschilderten Problemfall erscheint deshalb fraglich.

### 7.1.4 Modelle mit mehreren Warteschlangen

Warteschlangenmodelle zur Leistungsvorhersage von SCAN-Algorithmen werden in [CH82, CH90, One75] vorgestellt. In der Analyse wird das System durch ein Modell mit einem Server und je einer Warteschlange pro Zylinder nachgebildet [Eis72]. In einer Warteschlange werden die den jeweiligen Zylinder betreffenden Aufträge gesammelt und vom Server des Modells in FCFS-Reihenfolge entnommen. Der Wechsel des Servers zur benachbarten Warteschlange geschieht dann, wenn eine Warteschlange komplett geleert wurde. Hierbei wird die aktuelle SCAN-Richtung eingehalten. Diese Richtung wird dann geändert, wenn die Warteschlange des äußersten bzw. des innersten Zylinders erreicht ist bzw. wenn in der aktuellen Richtung die nachfolgenden Warteschlangen keine Aufträge enthalten. Der Ankunftsprozeß wird durch einen Poisson-Prozeß modelliert, der die Aufträge gleichmäßig über alle Zylinder verteilt.

Da das Modell lediglich von einem Poisson-Ankunftsprozeß ausgeht, kann das Eintreffen der K-Aufträge zu konstanten periodisch wiederkehrenden Zeitpunkten nicht berücksichtigt werden. Des weiteren gibt es keine einfache Möglichkeit, das Modell dahingehend zu verändern, daß die Ausführung der K-Aufträge mit höherer Priorität geschieht und zudem sichergestellt wird, daß die Rundendauer bei der Ausführung der Aufträge nicht überschritten wird.

## 7.2 Ein Modell zur Berechnung der Antwortzeitverteilung für SCAN-Algorithmen

Da die im vorangegangenen Abschnitt vorgestellten Warteschlangenmodelle nur unzureichend genaue Ergebnisse liefern bzw. aufgrund ihrer Komplexität keine praktischen Berechnungen zulassen, soll im folgenden ein neues Warteschlangemodell entwickelt werden, das beide Anforderungen, d.h. hinreichende Genauigkeit und Durchführbarkeit der Berechnungen, unter Verwendung realistischer System- und Lastparameter erfüllt.

Die Evaluation in Abschnitt 4.3 hat gezeigt, daß die Klasse der SCAN-Algorithmen mit gemischter Einteilungsstrategie im Vergleich zu der Klasse der Algorithmen mit getrennter Einteilungsstrategie die besseren Performance-Ergebnisse liefert. Das zu entwickelnde Warteschlangenmodell soll aus diesem Grund zur Vorhersage der Antwortzeit bei einem SCAN-Algorithmus mit gemischter Einteilungsstrategie genutzt werden können.

Das Problem besteht darin, das Verhalten der Algorithmen auf die Bedienstrategie des Warteschlangenmodells abzubilden. Aufgrund der Komplexität des dynamischen Verhaltens der Algorithmen mit voll inkrementeller Auswahlstrategie (siehe Abschnitt 3.3) beschränkt sich die

nachfolgende Betrachtung deshalb auf einen Algorithmus mit *gemischter, dynamischer, nicht-inkrementeller* SCAN-Strategie. Die Antwortzeit eines D-Auftrages bei Verwendung dieser Strategie stellt, wie in der Evaluation in Kapitel 4 gezeigt, eine obere Schranke für die mit den inkrementellen SCAN-Strategien erreichbaren mittleren Antwortzeiten dar.

Das hier entwickelte Warteschlangenmodell gilt deshalb als konservative Approximation und liefert hinsichtlich der Antwortzeitvorhersage eine obere Schranke für die tatsächlich mit dem Algorithmus V (Abschnitt 3.2) erzielbaren Werte.

Als Grundlage der folgenden Untersuchung dient ein M/G/1/K-Warteschlangenmodell, das die Auswahl der D-Aufträge zu festgelegten Zeitpunkten trifft und bei dem die Anzahl der ausführbaren Aufträge durch eine probabilistische Schranke begrenzt wird (*gated request-limited service with variable limit*). Ein ähnliches Modell ist in [LaM91] zu finden, wo ein M/G/1/K-Vacation-Modell mit dieser Bedienstrategie untersucht wird.

Die Bedienstrategie des Modells wählt, wie durch die nicht-inkrementelle Strategie des Algorithmus vorgegeben, zu Beginn einer Runde in FCFS-Reihenfolge eine Menge von D-Aufträgen aus der Warteschlange. Die Anzahl der maximal auswählbaren D-Aufträge wird durch eine Zufallsvariable $M$ bestimmt. Durch diese variable Schranke wird berücksichtigt, daß der Algorithmus nur so viele D-Aufträge in die Runde hineinnimmt, daß insgesamt alle D- und K-Aufträge innerhalb der Runde bedient werden können. Die Zufallsvariable $M$ hängt von der Zeitdauer der Bedienung der K-Aufträge ab.

Betrachtet man die Zeitdauer, die ein Auftrag im System verweilt, so lassen sich aus der Sichtweise eines D-Auftrages drei verschiedene Rundentypen unterscheiden: ein D-Auftrag betritt das Modell in der *Ankunftsrunde*, danach hat er eine oder mehrere *Zwischenrunden* zu warten, bevor seine Bedienung in der *Abgangsrunde* durchgeführt wird und er daraufhin das System verläßt. Hiermit setzt sich die Antwortzeit eines D-Auftrages $a$ aus den folgenden drei Komponenten (siehe Abbildung 7.3) zusammen:

1. Nach der Ankunft hat der Auftrag $a$ das Ende der Ankunftsrunde abzuwarten. Diese Zeitspanne wird im folgenden durch die Zufallsvariable $A$ angegeben.

2. In den Zwischenrunden werden vor $a$ eingetroffene D-Aufträge sowie K-Aufträge bedient. Die gesamte Zeitdauer aller Zwischenrunden, die Auftrag $a$ warten muß, wird durch die Zufallsvariable $I$ beschrieben. Aufgrund der FCFS-Auswahlstrategie zu Rundenbeginn beeinflussen nach $a$ eintreffende D-Aufträge die Anzahl der Zwischenrunden nicht.

3. In der Abgangsrunde wird gemäß der SCAN-Anordnungsstrategie der Auftrag ausgeführt. Die Verweilzeit des Auftrages in der Abgangsrunde setzt sich zusammen aus der Gesamtbedienzeit aller Aufträge, die $a$ in der SCAN-Anordnung vorausgehen, und die Bedienzeit von $a$ selbst. Die Verweilzeit in der Abgangsrunde wird im folgenden durch die Zufallsvariable $B$ angegeben.

Die Zufallsvariablen $I$ und $B$ hängen von der Anzahl der Aufträge in der Warteschlange zum Ankunftszeitpunkt und von der Anzahl der K-Aufträge ab, die in jeder Runde bedient werden.

Um die Durchführbarkeit der Berechnungen zu gewährleisten, wird von einer begrenzten maximalen Warteschlangenlänge des Modells ausgegangen. Die Anzahl der Warteschlangenplätze

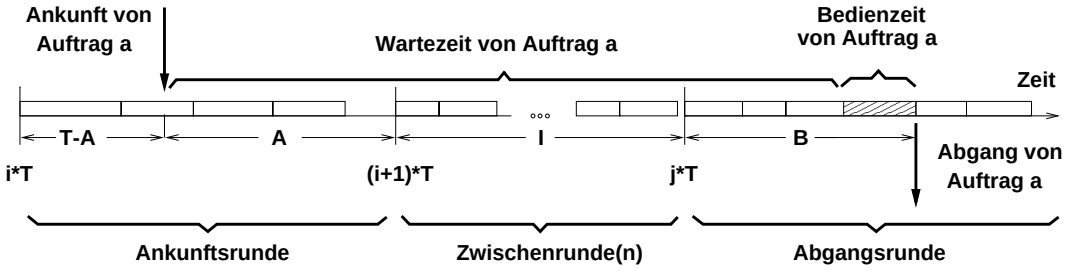



Abbildung 7.3: Komponenten der Antwortzeit eines D-Auftrages

sei mit $L_{max}$ bezeichnet. Weiterhin sei $p_b$ die *Abblockwahrscheinlichkeit*, mit der ein neu eintreffender D-Auftrag keinen freien Platz in der Warteschlange findet und somit, ohne ausgeführt zu werden, zurückgewiesen wird. Der Parameter $L_{max}$ hat im Scheduling-Algorithmus keine direkte Entsprechung, da hier die Warteschlangenlänge nur durch den zur Verfügung stehenden Speicher begrenzt ist. Um eine gute Approximation des Scheduling-Algorithmus zu erreichen, wird der Wert von $L_{max}$ in den Berechnungen deshalb so groß gewählt, daß die Abblockwahrscheinlichkeit $p_b$ vernachlässigbar klein wird. Die Länge der Warteschlange zu einem beliebigen Zeitpunkt und somit auch zum Ankunftszeitpunkt eines D-Auftrages sei mit $L$ angegeben.

Um die Notation in den Ausdrücken zu vereinfachen sei im folgenden die Dichtefunktion $f_X$ der Zufallsvariablen $X$ auch für kontinuierliche Zufallsvariablen durch $P[X = x]$ angegeben. Hiermit wird eine bessere Lesbarkeit durch die Vermeidung von länglichen Bezeichnern bei Verbundwahrscheinlichkeiten und bedingten Wahrscheinlichkeiten erreicht. Eine Zusammenfassung aller Bezeichner ist im Anhang B zu finden.

### 7.2.1 Laplace-Transformierte der Antwortzeit

Sei mit $R$ die Zufallsvariable der Antwortzeit (engl. response time) für D-Aufträge bezeichnet. Die Wahrscheinlichkeit, daß ein D-Auftrag, der nicht abgeblockt wird, eine Antwortzeit $r$ besitzt, ist dann:

$$\begin{aligned}
&P[\, R = r \mid \text{Auftrag nicht abgeblockt}\,] = P[\, A + I + B = r \mid \text{Auftrag nicht abgeblockt}\,] \\
&= \frac{1}{1-p_b}\, P[\, A + I + B = r \wedge \text{ Auftrag nicht abgeblockt}\,] \\
&= \frac{1}{1-p_b} \sum_{l=0}^{L_{max}-1} \int_0^T P\,[I + B = r - t \wedge A = t \wedge L = l]\, dt \qquad (7.1) \\
&= \frac{1}{1-p_b} \sum_{l=0}^{L_{max}-1} \int_0^T P\,[I + B = r - t \mid A = t \wedge L = l] * P\,[A = t \wedge L = l]\, dt
\end{aligned}$$

Um die Notation im folgenden zu verkürzen, seinen mit $A_l$ bzw. $I_l$ der Verbund mit der Variablen $L$ bzw. die Zufallsvariable unter der Bedingung $L = l$ bezeichnet, d.h.

$$P[\, A_l = t\,] := P[\, A = t \wedge L = l\,] \qquad P[\, I_l = t\,] := P[\, I = t \,|\, L = l\,] \qquad (7.2)$$

$P[A_l = t]$ wird im folgenden als Verbundwahrscheinlichkeit (engl. joint probability) der Variablen $A$ und $L$ bezeichnet. $I_l$ ist durch die FCFS-Auswahlstrategie zu Rundenbeginn abhängig

von der Anzahl der D-Aufträge in der Warteschlange zum Ankunftszeitpunkt aber durch den ”gated service“ unabhängig vom Ankunftszeitpunkt innerhalb der Ankunftsrunde. Die Zufallsvariable $B$ wird als unabhängig von den Variablen $A$ und $L$ postuliert und wird, wie später gezeigt, durch die Anzahl der D-Aufträge zu Rundenbeginn der Abgangsrunde bestimmt. Aus diesem Grund vereinfacht sich Gleichung 7.1 zu:

$$P[\, R = r \mid \text{Auftrag nicht abgeblockt}\,] = \frac{1}{1-p_b} \sum_{l=0}^{L_{max}-1} \int_0^T P\,[\, I_l + B = r - t\,] * P\,[\, A_l = t\,]\, dt \quad (7.3)$$

Der linke Term im Integral der Gleichung 7.3 entspricht der Faltung der beiden Zufallsvariablen $I_l$ und $B$. Das Integral selbst beschreibt die Faltung der beiden Variablen $(I_l + B)$ und $A_l$. Die Variablen $A_l$ und $I_l$ sind für ein festes $l$ unabhängig voneinander. Dasselbe gilt aufgrund der Unabhängigkeitsannahme auch für $B$. Hierdurch ist es möglich, die beiden Faltungen in Gleichung 7.3 durch eine Multiplikation der Laplace-Transformierten auszudrücken. $A_l^*$, $I_l^*$ und $B^*$ sind die Laplace-Transformierten der Zufallsvariablen $A_l$, $I_l$ und $B$. Die Laplace-Transformierte $R^*$ der Antwortzeit ergibt sich damit zu:

$$R^*(s) = \frac{1}{1-p_b} \sum_{l=0}^{L_{max}-1} A_l^*(s) * I_l^*(s) * B^*(s) \quad (7.4)$$

Aufgrund der Definition der Laplace-Transformierten hat sie an der Stelle $s = 0$ den Wert 1. Durch diese Eigenschaft läßt sich aus Gleichung 7.4 die Abblockwahrscheinlichkeit $p_b$ berechnen. Es gilt:

$$\lim_{s\to 0} R^*(s) = 1 \Longleftrightarrow p_b = 1 - \lim_{s\to 0} \sum_{l=0}^{L_{max}-1} A_l^*(s) * I_l^*(s) * D^*(s) \quad (7.5)$$

In den folgenden Abschnitten sollen die einzelnen Laplace-Transformierten abgeleitet werden. Begonnen wird mit der Analyse der Ankunftsrunde.

### 7.2.2 Analyse der Ankunftsrunde

Ziel der Analyse ist es zunächst, die Verbundwahrscheinlichkeit $P[A_l = t]$ der Restzeit in der Ankunftsrunde und der Warteschlangenlänge zu bestimmen, um hieraus anschließend die Laplace-Transformierte $A_l^*$ berechnen zu können. Das hierzu verwendete Verfahren (engl. supplementary variable technique) [Lee84, Tak91] untersucht die Verteilung der Warteschlangenlänge zu Rundenbeginn und berechnet dann die Verbundwahrscheinlichkeit zwischen $A$ und $L$ zu einem beliebigen Zeitpunkt in der Ankunftsrunde.

#### Verteilung der Warteschlangenlänge zu Rundenbeginn

Die Berechnung der Warteschlangenlänge zu Rundenbeginn erfolgt durch eine eingebettete Markov-Kette (engl. embedded discrete-time Markov chain), deren Betrachtungszeitpunkte die

Startzeitpunkte einer Runde sind. Zu jedem dieser Betrachtungszeitpunkte $t$ mit $t \in \{0, 1, \dots\}$ ist der Systemzustand $L(t)$ gegeben durch die Anzahl der D-Aufträge in der Warteschlange. Die Markov-Kette ist nicht reduzierbar (engl. irreducible), da jeder Zustand von jedem anderen Zustand erreichbar ist, und aperiodisch. Hieraus folgt aufgrund des endlichen Zustandsraumes, daß die Markov-Kette ergodisch ist und ein stationärer Zustand (engl. steady state) existiert. Im stationären Zustand gilt für die Wahrscheinlichkeit $\pi_l$ eines Systemzustandes mit dem Wert $l$:

$$\pi_l = \lim_{i \to \infty} P[\, L(i) = l \,] \quad \text{mit} \quad l = 0, \dots, L_{max} \quad . \tag{7.6}$$

Sei $M$ die Zufallsvariable, die die Anzahl der D-Aufträge angibt, die maximal pro Runde ausgeführt werden können. Sei $m_i$ die Wahrscheinlichkeit, daß $i$ D-Aufträge in einer Runde bearbeitet werden können. Der Wert von $m_i$ läßt sich wie folgt bestimmen:

$$\begin{aligned} m_i &= P[\, M = i \,] \\ &= P[\, M \geq i \,] - P[\, M \geq i+1 \,] \\ &= P[\, T_{svc}^{K,D}(N, i) \leq T \,] - P[\, T_{svc}^{K,D}(N, i+1) \leq T \,] \end{aligned} \tag{7.7}$$

In Gleichung 7.7 ist $T_{svc}^{K,D}(N, i)$ die Gesamtbedienzeit, die für $N$ K-Aufträge und $i$ D-Aufträge benötigt wird. Die dazugehörige Verteilungsfunktion $F_{svc}^{K,D}$ kann analog zu Gleichung 6.9 unter Berücksichtigung zusätzlicher Transfer- und Positionierungszeiten für D-Aufträge approximativ berechnet werden. Aus Gleichung 7.7 folgt hiermit:

$$m_i = F_{svc}^{K,D}(N, i) - F_{svc}^{K,D}(N, i+1) \quad \text{für } i = 0, \dots, M_{max} \tag{7.8}$$

Der Parameter $M_{max}$ beschreibt die maximale Anzahl der D-Aufträge, die im Modell pro Runde bearbeitet werden. Dieser Parameter dient dazu, die Durchführbarkeit späterer Berechnungen zu gewährleisten und besitzt im Scheduling-Algorithmus keine Entsprechung. Der Wert von $M_{max}$ muß deshalb so gewählt werden, daß die Wahrscheinlichkeit, daß der Algorithmus mehr als $M_{max}$ D-Aufträge in einer Runde bedienen kann, vernachlässigbar gering ist.

Sei $H$ die Zufallsvariable, die die Anzahl der D-Aufträge angibt, die in einer Runde der Dauer $T$ eintreffen und die in die Warteschlange des Systems eingefügt werden. Unter der Annahme, daß ein Poisson-Ankunftsprozeß mit der Rate $\lambda$ vorliegt, ist die Wahrscheinlichkeit $h_k$, daß während einer Runde $k$ Ankünfte eintreffen, gemäß Gleichung 5.5 gegeben durch:

$$h_k = P[\, H = k \,] = \frac{(\lambda T)^k}{k!} * e^{-\lambda T} \tag{7.9}$$

Das Warteschlangenmodell ist überlastet mit einer Abblockwahrscheinlichkeit von $p_b = 1$, wenn der Erwartungswert der Anzahl der Ankünfte pro Runde größer ist als der Erwartungswert der maximal pro Runde ausführbaren D-Aufträge. Aus diesem Grund muß für die Ankunftsrate $\lambda$ folgende Bedingung gelten:

$$E[H] = \sum_{k=0}^{\infty} k * h_k \leq \sum_{i=0}^{M_{max}} i * m_i = \lambda * T = E[M] \tag{7.10}$$

Die Leistungskapazität des Warteschlangenmodells hat hierdurch den Wert $E[M]/T$.

Die stationären Zustandswahrscheinlichkeiten $\pi_l$ lassen sich, da die Warteschlange mit $L_{max}$ Plätzen endlich ist, durch die folgenden drei Chapman-Kolmogorov-Gleichungen berechnen, die im stationären Zustand der Markov-Kette gelten:

$$l < L_{max}: \quad \pi_l = \sum_{i=0}^{M_{max}} m_i * \left( h_l * \sum_{j=0}^{i} \pi_j + \sum_{j=i+1}^{\min(i+l, L_{max})} \pi_j * h_{l-j+i} \right) \tag{7.11}$$

$$l = L_{max}: \quad \pi_l = \sum_{i=0}^{M_{max}} m_i * \left( \sum_{j=0}^{i} \pi_j * \sum_{j=L_{max}}^{\infty} h_j + \sum_{j=i+1}^{L_{max}} \pi_j * \sum_{k=L_{max}-j}^{\infty} h_{k+i} \right) \tag{7.12}$$

$$\text{Normierung}: \quad \sum_{i=0}^{L_{max}} \pi_i = 1 \tag{7.13}$$

Die erste Summe in Gleichung 7.11 iteriert über die Anzahl der in einer Runde maximal ausführbaren D-Aufträge. Die zweite Summe erfaßt alle diejenigen Fälle, in denen alle zum Rundenbeginn in der Warteschlange verfügbaren D-Aufträge bedient werden. Dies ist dann der Fall, wenn zu Rundenbeginn weniger als die maximal ausführbare Anzahl an Aufträgen in der Warteschlange vorhanden ist. Damit zum Beginn der nächsten Runde $l$ Aufträge in der Warteschlange stehen, müssen somit während der Runde $l$ Aufträge eintreffen. Die dritte Summe erfaßt die Fälle, in denen nicht alle wartenden Aufträge bearbeitet werden können. Werden von den zu Rundenbeginn vorhandenen $j$ Aufträgen insgesamt $i$ bearbeitet, so müssen $(l - j + i)$ Aufträge während der Runde eintreffen, damit zu Beginn der nächsten Runde sich $l$ Aufträge in der Warteschlange befinden. Gleichung 7.12 betrachtet den Sonderfall, bei dem Aufträge aufgrund einer vollständig belegten Warteschlange abgewiesen werden müssen. Die $L_{max} + 2$ Gleichungen aus 7.11 bis 7.13 sind linear abhängig mit dem Rang $L_{max} + 1$. Zur Berechnung der $L_{max} + 1$ Unbekannten können z.B. die Gleichungen aus 7.11 zusammen mit der Normierungsbedingung in Gleichung 7.13 verwendet werden. Die Lösung des Gleichungssystems kann durch effiziente numerische Algorithmen, z.B. Gauß-Seidel, durchgeführt werden.

**Verteilung der Warteschlangenlänge zu einem beliebigen Zeitpunkt**

Basierend auf den Wahrscheinlichkeiten $\pi_i$ des stationären Zustands zu Rundenbeginn kann nun die Verbundwahrscheinlichkeit $P[A_l = t]$ zwischen der verbleibenden Rundendauer $A$ und der Warteschlangenlänge $L$ bei Eintreffen eines Auftrages $a$ zu einem beliebigen Zeitpunkt $t \in [0, T]$ innerhalb der Runde berechnet werden. Durch $O(t)$ sei im folgenden die Anzahl der Ankünfte von D-Aufträgen in einer Zeitperiode der Dauer $t$ angegeben. Die Anzahl der Ankünfte innerhalb der Runde vor Eintreffen von $a$ ist deshalb bei konstanter Rundendauer $T$ durch $O(T - A)$ gegeben. Die Berechnung von $P[A_l = t]$ geschieht durch die Betrachtung der

folgenden beiden Fälle:

$$l < L_{max} : P\,[\,A_l = t\,] = P\,[\,A = t \wedge L = l\,]$$
$$= \sum_{i=0}^{M_{max}} m_i * \left( \sum_{j=0}^{i} \pi_j \;\; P\,[\,O(T-A) = l \wedge A = t\,] + \sum_{j=i+1}^{\min(i+l, L_{max})} \pi_j \;\; P\,[\,O(T-A) = l - j + i \wedge A = t\,] \right) \quad (7.14)$$

$$l = L_{max} : P\,[\,A_l = t\,] = P\,[\,A = t \wedge L = l\,]$$
$$= \sum_{i=0}^{M_{max}} m_i * \left( \sum_{j=0}^{M_D} \pi_j \;\; \left( 1 - \sum_{m=0}^{L_{max}-1} P\,[\,O(T-A) = m \wedge A = t\,] \right) + \sum_{j=i+1}^{L_{max}} \pi_j \;\; \left( 1 - \sum_{m=0}^{L_{max}-j+i-1} P\,[\,O(T-A) = m \wedge A = t\,] \right) \right) \quad (7.15)$$

Die Zusammensetzung obiger Terme ist vergleichbar mit der in den Gleichungen 7.11 und 7.12. Die erste Summe in Gleichung 7.14 iteriert über die maximale Anzahl der in einer Runde ausführbaren D-Aufträge. Die zweite Summe betrachtet all diejenigen Fälle, bei denen die Warteschlange zu Rundenbeginn vollständig geleert wurde. Dies ist der Fall, wenn zu Rundenbeginn weniger D-Aufträge in der Warteschlange vorhanden sind, als maximal bearbeitet werden können. Alle D-Aufträge, die dann zum Zeitpunkt $t$ in der Warteschlange stehen, müssen vom Rundenbeginn bis zum Zeitpunkt $t$ im Intervall $[0, t]$ eingetroffen sein. Die dritte Summe betrachtet die restlichen Fälle, bei denen sich die Anzahl der D-Aufträge in der Warteschlange zum Zeitpunkt $t$ aus den zu Rundenbeginn verbliebenen und den neu eingetroffenen zusammensetzt.

Die Verbundwahrscheinlichkeit $P[O(T-A) = n \wedge A = t]$ in den Gleichungen 7.14 und 7.15 wird basierend auf den Poisson-Ankünften der D-Aufträge berechnet. Es gilt:

$$P\,[\,O(T-A) = n \wedge A = t\,] = P\,[\,O(T-A) = n \mid A = t\,] * P\,[\,A = t\,]$$
$$= \frac{\lambda^n\,(T-t)^n}{n!} * e^{-\lambda\,(T-t)} * \frac{1}{T} \quad (7.16)$$

Die Laplace-Transformierte von $P[O(T-A) = n \wedge A = t]$ erhält man durch Integration über die Zeit t. Die Lösung des Integralausdrucks führt zu:

$$\int_0^T e^{-st}\, P\,[\,O(T-A) = n \wedge A = t\,]\, dt$$
$$= \int_0^T e^{-st} * \frac{\lambda^n\,(T-t)^n}{n!} * e^{-\lambda\,(T-t)} * \frac{1}{T}\, dt$$
$$\overset{\text{[Tak91]}}{=} \frac{1}{(\lambda - s)T} \left( e^{-sT} \left( \frac{\lambda}{\lambda - s} \right)^n - \sum_{m=0}^{n} a_m \left( \frac{\lambda}{\lambda - s} \right)^{n-m} \right) \quad (7.17)$$

Die Laplace-Transformierte $A_l^*$ für Gleichung 7.4 erhält man, wenn die Laplace-Transformierte der Gleichung 7.17 in die Laplace-Transformierte der Gleichungen 7.14 und 7.15 eingesetzt wird. Das Ergebnis lautet für $l < L_{max}$:

$$A_l^*(s) =$$

$$\sum_{i=0}^{M_{max}} m_i \left( \sum_{j=0}^{i} \pi_j \right) \frac{1}{(\lambda - s)T} \left( e^{-sT} \left( \frac{\lambda}{\lambda - s} \right)^l - \sum_{m=0}^{l} a_m \left( \frac{\lambda}{\lambda - s} \right)^{l-m} \right) +$$

$$\sum_{j=i+1}^{\min(i+l, L_{max})} \pi_j \frac{1}{(\lambda - s)T} \left( e^{-sT} \left( \frac{\lambda}{\lambda - s} \right)^{l-j+i} - \sum_{m=0}^{l-j+i} a_m \left( \frac{\lambda}{\lambda - s} \right)^{l-j+i-m} \right) \quad (7.18)$$

Das Ergebnis für $l = L_{max}$ wird in Gleichung 7.4 nicht verwendet, da die Summierung darin nur bis zum Index $L_{max} - 1$ durchgeführt wird.

### 7.2.3 Analyse der Zwischenrunden

Die Anzahl der Zwischenrunden, die ein Auftrag warten muß, bis die Abgangsrunde beginnt, hängt aufgrund der FCFS-Auswahl der D-Aufträge zu Rundenbeginn nur von der Anzahl der D-Aufträge zum Ankunftszeitpunkt und der Anzahl der in den Zwischenrunden bedienten D-Aufträge ab.

Es sei durch $M(l)$ die maximale Anzahl der Zwischenrunden angegeben, die ein Auftrag warten muß, wenn zu dessen Ankunftszeitpunkt bereits $l$ D-Aufträge in der Warteschlange vorhanden sind. Es gilt:

$$M(l) = \max \left\{ m \geq 0 : \left( \sum_{i=0}^{m} M_i \right) < l \right\}$$

$M_i$ ist die Zufallsvariable für die Anzahl der D-Aufträge, die in der $i$-ten Zwischenrunde von Auftrag $a$ maximal ausgeführt werden können. Die minimale Anzahl der D-Aufträge, die pro Runde bearbeitet werden, sei durch $M_{min}$ angegeben. Für $M_{min}$ gilt:

$$M_{min} = \min \{ n \geq 0 : m_n > 0 \}$$

Es wird angenommen, daß $M_{min}$ größer ist als $0$. In diesem Fall ist $M(l)$ endlich und hat den Wert:

$$M(l) = \left\lfloor \frac{l}{M_{min}} \right\rfloor \quad (7.19)$$

Als nächster Schritt soll die Wahrscheinlichkeit für die Anzahl der Zwischenrunden eines Auftrages berechnet werden. Dies geschieht unter der Voraussetzung, daß zum Ankunftszeitpunkt des Auftrages bereits $l$ Aufträge in der Warteschlange vorhanden sind. Es gilt:

$$
\begin{aligned}
&P\left[\begin{array}{c}\text{Anzahl der}\\ \text{Zwischenrunden}\end{array} = i \,\middle|\, L = l\right] \\
&\quad = P\left[\begin{array}{c}\text{Anzahl der}\\ \text{Zwischenrunden}\end{array} \geq i \,\middle|\, L = l\right] - P\left[\begin{array}{c}\text{Anzahl der}\\ \text{Zwischenrunden}\end{array} \geq i+1 \,\middle|\, L = l\right] \\
&\quad = P\left[\left(\sum_{j=1}^{i} M_j\right) \leq l\right] - P\left[\left(\sum_{j=1}^{i+1} M_j\right) \leq l\right] \qquad (7.20) \\
&\quad = F_M^{(i-1)}(l) - F_M^{(i)}(l) \qquad (7.21)
\end{aligned}
$$

$P\left[\left(\sum_{j=1}^{i} M_j\right) \leq l\right]$ in Gleichung 7.20 beschreibt die $i$-fache Faltung $F_M^{(i)}$ der Verteilungsfunktion von $M$. Sie läßt sich wie folgt berechnen:

$$
F_M^{(-1)}(l) = 1\,, \quad F_M^{(0)}(l) = \sum_{i=0}^{\min(l,M_{max})} m_i\,, \quad F_M^{(i)}(l) = \sum_{j=0}^{\min(l,M_{max})} \sum_{k=0}^{j} m_j * F_M^{(i-1)}(j-k) \qquad (7.22)
$$

Betrachtet man den Fall, bei dem ein Auftrag nach seiner Ankunft sofort in der nachfolgenden Runde bearbeitet werden kann und somit keine Zwischenrunde warten muß, so erhält man:

$$
P\left[\begin{array}{c}\text{Anzahl der}\\ \text{Zwischenrunden}\end{array} = 0 \,\middle|\, L = l\right] = P\left[\begin{array}{c}M \text{ in}\\ \text{Folgerunde}\end{array} > l\right] = 1 - \sum_{i=0}^{l} m_i
$$

Das Ergebnis dieses Sonderfalls in obiger Gleichung, läßt sich ebenfalls mit $i = 0$ in Gleichung 7.22 und das Einsetzen in Gleichung 7.21 erzielen.

Mit den bisherigen Ergebnissen läßt sich nun die Wahrscheinlichkeit $P[I_l = t]$ für den Zeitraum $t$ berechnen, in dem sich Auftrag $a$ in den Zwischenrunden befindet. Man iteriert über die mögliche Anzahl der Zwischenrunden und multipliziert die Wahrscheinlichkeit, daß $i$ Zwischenrunden vergehen mit der Wahrscheinlichkeit, daß $i$ Zwischenrunden eine Zeitdauer von $t$ benötigen. Da jede Zwischenrunde eine konstante Länge von $T$ hat, wird letzteres mit der Dirac-Funktion $\delta_0(t - i * T)$ beschrieben. Diese Funktion liefert an der Stelle $0$ den Wert $1$ und hat an den übrigen Stellen den Wert $0$. Es folgt mit den Ergebnissen aus Gleichung 7.21 und 7.19:

$$
\begin{aligned}
P[\,I_l = t\,] &= P[\,I = t \mid L = l\,] \\
&= \sum_{i=0}^{M(l)} \delta_0(t - i * T) * P\left[\begin{array}{c}\text{Anzahl der}\\ \text{Zwischenrunden}\end{array} = i \,\middle|\, L = l\right] \\
&= \sum_{i=0}^{M(l)} \delta_0(t - i * T) * \left(F_M^{(i-1)}(l) - F_M^{(i)}(l)\right) \qquad (7.23)
\end{aligned}
$$

Durch die Laplace-Transformation der Gleichung 7.23 mit der Integrationsvariablen $t$ ergibt sich die Laplace-Transformierte von $I_l$ zu:

$$I_l^*(s) = \sum_{i=0}^{M(l)} e^{-siT} * \left( F_M^{(i-1)}(l) - F_M^{(i)}(l) \right) \tag{7.24}$$

$e^{-siT}$ ist in obiger Gleichung die Laplace-Transformierte der Dirac-Funktion $\delta_0(t - i * T)$.

### 7.2.4 Analyse der Abgangsrunde

Der Anteil an der Wartezeit, die der Auftrag $a$ in der Abgangsrunde verbringt, hängt von der Anzahl der D-Aufträge ab, die sich zu Beginn dieser Runde in der Warteschlange befinden. Die Anzahl ist für den Auftrag $a$ durch die Länge $L$ der Warteschlange zum Ankunftszeitpunkt und die Anzahl der im Zeitraum $A + I$ eintreffenden D-Aufträge bestimmt. Da im stationären Zustand die Wahrscheinlichkeit $\pi_i$ für $i$ D-Aufträge zu Rundenbeginn für jede Runde gleich ist, soll statt der Abgangsrunde eine beliebige Runde und somit statt $a$ ein beliebiger D-Auftrag betrachtet werden. Es gilt:

$$P\left[ \begin{array}{c}\text{Anzahl an D-Aufträgen in Abgangs-} \\ \text{runde zu Rundenbeginn}\end{array} = i \,\middle|\, A_l = t \right] \;=\; \pi_i \tag{7.25}$$

Die Wahrscheinlichkeit, daß in einer beliebigen Runde $k$ D-Aufträge bearbeitet werden, läßt sich aus den Wahrscheinlichkeiten $\pi_i$ der Warteschlangenlänge des stationären Zustandes und den Wahrscheinlichkeiten $m_i$ für die maximal in der Runde ausführbare Anzahl an D-Aufträgen berechnen. Der Wert beträgt:

$$\frac{m_k * \sum_{j=k}^{L_{max}} \pi_j + \pi_k * \sum_{j=k+1}^{M_{max}} m_j}{\sum_{i=1}^{M_{max}} m_i * \sum_{j=i}^{L_{max}} \pi_j + \pi_i * \sum_{j=i+1}^{M_{max}} m_j} \tag{7.26}$$

Die erste Summe im Zähler in Gleichung 7.26 berücksichtigt den Fall, bei dem die Anzahl der ausführbaren D-Aufträge durch die Schranke begrenzt wird und mindestens $k$ Aufträge zu Rundenbeginn in der Warteschlange zur Verfügung stehen. Die zweite Summe erfaßt alle diejenigen Fälle, in denen genau $k$ Aufträge zur Verfügung stehen, die, da die Schranke größer als $k$ ist, alle bearbeitet werden können. Der Nenner des Quotienten dient zur Normierung der Werte.

Aus der Sichtweise eines beliebigen Auftrages ist die Wahrscheinlichkeit $q_k$, daß $k$ D-Aufträge in einer Runde ausgeführt werden, im Vergleich zu Gleichung 7.26 um den Faktor $k$ größer, da jeder Auftrag aus den $k$ D-Aufträgen ausgewählt werden kann. Hieraus folgt:

$$q_k \;=\; P\left[ \begin{array}{c}\text{Auftrag "sieht" } k \\ \text{Aufträge in Abgangsrunde}\end{array} \right] \tag{7.27}$$

$$= \frac{k * \left( m_k * \sum_{j=k}^{L_{max}} \pi_j + \pi_k * \sum_{j=k+1}^{M_{max}} m_j \right)}{\sum_{i=1}^{M_{max}} i * \left( m_i * \sum_{j=i}^{L_{max}} \pi_j + \pi_i * \sum_{j=i+1}^{M_{max}} m_j \right)} \tag{7.28}$$

Die Laplace-Transformierte $B^*$ der Zeitdauer, die ein beliebiger D-Auftrag $a$ in der Abgangsrunde verweilt, ist damit:

$$B^*(s) = \sum_{d_1=1}^{M_{max}} q_{d_1} \sum_{p=1}^{d_1+N} \frac{1}{d_1+N} * \sum_{d_2=\max(0,p-N-1)}^{min(p-1,d_1-1)} pred(N,d_1,p,d_2) * \begin{pmatrix}\text{p-fache Faltung der}\\ \text{Bedienzeit bei SCAN}\end{pmatrix} \quad (7.29)$$

In der äußersten Summe der Gleichung 7.29 wird durch die Variable $d_1$ über die Gesamtzahl der D-Aufträge iteriert, die zusammen mit dem betrachteten D-Auftrag $a$ in der gleichen Runde bedient werden. Die Wahrscheinlichkeit, in der Abgangsrunde $k$ D-Aufträge zu bedienen, ist aus der Sichtweise von $a$ durch $q_k$ aus Gleichung 7.28 gegeben. Die mittlere Summe in Gleichung 7.29 iteriert durch die Variable $p$ über die Position, die $a$ in der Abgangsrunde einnehmen kann. Unter der Voraussetzung, daß pro Runde zusätzlich $N$ K-Aufträge ausgeführt werden, ergeben sich $(d_1 + N)$ mögliche Positionen. Die Wahrscheinlichkeit, eine dieser Positionen einzunehmen, ist gleichverteilt und hat die Dichte $(\frac{1}{d_1+N})$. Die innerste Summe iteriert mit der Variablen $d_2$ über die Anzahl der D-Aufträge, die $a$ vorausgehen. Durch die Festlegung der Position ist damit auch die Anzahl der K-Aufträge bestimmt, die im SCAN nach $a$ ausgeführt werden. Insgesamt kann $a$ bis zu $(d_1 + N - 1)$ Vorgänger im SCAN haben. Die Positionen werden mit $1$ bis $(d_1 + N)$ numeriert. $pred(N, d_1, p, d_2)$ ist die Wahrscheinlichkeit, daß an der Position $p$ der Auftrag $a$ $d_2$ D-Aufträge und $(p - 1 - d_2)$ K-Aufträge als Vorgänger hat.

Der Wert von $pred(\cdot)$ kann kombinatorisch wie folgt berechnet werden:

$$pred(N,d_1,p,d_2) = \frac{\binom{d_1}{d_2+1} * \binom{N}{p-1-d_2} * (p-1)! * (d_1+N-p)! * (d_2+1)}{(d_1+N-1)! * d_1} \quad (7.30)$$

Der Zähler in Gleichung 7.30 multipliziert die Anzahl der Möglichkeiten, $(d_2 + 1)$ D-Aufträge aus einer Menge von $d_1$ D-Aufträgen auszuwählen (Anzahl der Kombinationen ohne Wiederholung), mit der Anzahl der Möglichkeiten, $(p-1-d_2)$ K-Aufträge aus insgesamt $N$ auszuwählen. Diese Aufträge werden auf den SCAN-Positionen $(1 \ldots p)$ plaziert. Es gibt $(p - 1)!$ Permutationen vor Position $p$ und $(d_1 + N - p)!$ Permutationen hinter der Position $p$. Weiterhin gibt es $(d_2 + 1)$ Möglichkeiten, einen D-Auftrag für Position $p$ auszuwählen. Multipliziert ergibt sich somit die Gesamtzahl der Möglichkeiten zur Plazierung unter den Bedingungen der Parameter $d_1$, $p$, $N$ und $d_2$. Der Nenner in Gleichung 7.30 berechnet die Gesamtzahl der Möglichkeiten, insgesamt $d_1$ D-Aufträge und $N$ K-Aufträge auf $(d_1 + N)$ Positionen zu verteilen, unter der Bedingung, daß auf Position $p$ ein D-Auftrag zu finden ist.

In Gleichung 7.29 besteht strenggenommen eine wechselseitige Abhängigkeit zwischen der Wahrscheinlichkeit $q_k$ und der Verteilung der p-fachen Faltung der Bedienzeit, da festgelegte Werte von $m_i$ die Verteilung der Bedienzeit in einer Runde bestimmen. Diese Abhängigkeit

wird im folgenden ignoriert. Hiermit läßt sich die Laplace-Transformierte der p-fachen Faltung der Bedienzeit im SCAN wie folgt berechnen:

$$\begin{pmatrix}\text{p-fache Faltung der}\\ \text{Bedienzeit bei SCAN}\end{pmatrix} = \left[{T_{pos,SZ}^{scan_1}}^*(d_1+N,s)\right] * \left[{T_{pos,SZ}^{scan_X}}^*(d_1+N,s)\right]^{p-1} * \left[{T_{trans,SZ}^{S_D}}^*(s)\right]^{d_2+1} * \left[{T_{trans,SZ}^{S_K}}^*(s)\right]^{p-1-d_2} * \left[{T_{rot}}^*(s)\right]^{p} \quad (7.31)$$

Sie setzt sich aus den Bedienzeiten der Aufträge, die vor $a$ ausgeführt werden, und der Bedienzeit von $a$ zusammen. Die einzelnen Komponenten sind:

1. Positionierungzeit des ersten Auftrags

   ${T_{pos,SZ}^{scan_1}}^*(n,s)$ ist die Laplace-Transformierte der Positionierungszeit für den ersten Auftrag bei insgesamt $n$ Aufträgen im SCAN. Die Berechnung der Dichtefunktion ist in Abschnitt 5.2.4 zu finden. Die Berechnung der Laplace-Transformierten erfolgt durch numerische Integration (siehe Gleichung 6.10).

2. Positionierungszeiten der übrigen Aufträge

   Bis Position $p$ führt die Magnetplatte insgesamt $(p-1)$ Positionierungen zwischen zwei aufeinanderfolgenden Aufträgen durch. ${T_{pos,SZ}^{scan_X}}^*(n,s)$ ist die Laplace-Transformierte einer dieser Positionierungszeiten bei insgesamt $n$ Aufträgen im SCAN. Die Berechnung erfolgt analog zu 1. Da sich die Verteilungen der Variablen $T_{pos,SZ}^{scan_X}$ und $T_{pos,SZ}^{scan_1}$ nur geringfügig unterscheiden (siehe Kapitel 5.2.1), könnte die Laplace-Transformierte ${T_{pos,SZ}^{scan_1}}^*$ durch ${T_{pos,SZ}^{scan_X}}^*$ approximiert werden, wodurch sich die Gleichung 7.31 vereinfachen ließe. Hierauf wird allerdings im folgenden verzichtet.

3. Transferzeiten vorausgehender K-Aufträge

   Durch ${T_{trans,SZ}^{S_K}}^*$ sei die Laplace-Transformierte der Transferzeit eines K-Auftrages gegeben. An der Position $p$ fallen $(p-1-d_2)$ dieser Zeiten an, bevor $a$ das System verlassen kann. Die Dichte, aus der die Laplace-Transformierte berechnet werden kann, ist in Gleichung 5.43 angegeben.

4. Transferzeiten vorausgehender D-Aufträge + Transferzeit Auftrag $a$

   In der Abgangsrunde fallen $d_2$ Transferzeiten für vorangehende D-Aufträge und die Transferzeit für den Auftrag $a$ selbst an, bevor dieser das System verlassen kann. Analog zu den K-Aufträgen kann aus der Dichte der Transferzeitverteilung in Gleichung 5.44 die Laplace-Transformierte ${T_{trans,SZ}^{S_K}}^*$ berechnet werden.

5. Rotationsverzögerung

   Bevor der Auftrag $a$, der sich an der Position $p$ in der Abgangsrunde befindet, das System verlassen kann, finden $p$ Rotationsverzögerungen statt. Die Laplace-Transformierte ${T_{rot}}^*$ einer dieser Verzögerungen berechnet sich aus der gleichverteilten Dichte in Gleichung 5.34 und ist bereits in Gleichung 6.14 angegeben worden.

Hiermit sind abschließend die drei Komponenten $A_l^*$, $I_l^*$ und $B^*$ der Laplace-Transformierten $R^*$ aus der Gleichung 7.4 bestimmt worden. Die weitere Verwendung von $R^*$ zur Vorhersage der Antwortzeit von D-Aufträgen wird im folgenden Abschnitt beschrieben.

### 7.2.5 Abschätzung der Antwortzeit

Die Laplace-Transformierte $R^*$ kann dazu benutzt werden, die Antwortzeit für D-Aufträge für gegebene System- und Lastparameter zu berechnen. Dabei ist es nicht direkt möglich, die Verteilungsfunktion der Antwortzeit aus $R^*$ zu bestimmen. Hierzu müßte die Laplace-Transformierte invertiert werden, was bei komplexen Termen, wie sie in $R^*$ vorkommen, nicht einfach möglich ist. Spezielle Ergebnisse, wie z.B. die mittlere Antwortzeit $E[R]$, das zweite Moment der Antwortzeitverteilung $E[R^2]$ sowie die Varianz der Antwortzeit $Var[R]$, können aus der Laplace-Transformierten aber einfach berechnet werden. Hierzu wird die erste und die zweite Ableitung der Laplace-Transformierten an der Stelle $0$ bestimmt. Es gilt [All90, Nel95]:

$$E[R] = -\frac{dR^*(s)}{ds}\bigg|_{s=0}, \quad E[R^2] = \frac{d^2R^*(s)}{ds^2}\bigg|_{s=0}, \quad Var[R] = E[R^2] - (E[R])^2 \tag{7.32}$$

Des weiteren läßt sich die Restwahrscheinlichkeit $P[\ R \geq r\ ]$ der Antwortzeitverteilung durch die Tschebyscheff-Schranke oder, wie in Abschnitt 6.1.2 gesehen, durch die Chernoff-Schranke abschätzen. Für letzteres gilt:

$$p_{resp}(\lambda, r) := P[\ R \geq r\ ] \leq \inf_{\theta \geq 0} \left\{ e^{-\theta r} * R^*(-\theta) \right\} =: b_{resp}(\lambda, r) \tag{7.33}$$

Das Infimum kann auch hier durch Ableitung der Funktion $h(\theta) = e^{-\theta r} * R^*(-\theta)$ mit anschließender Nullstellensuche bestimmt werden.

### 7.2.6 Varianten des analytischen Modells

Durch geringfügige Modifikationen und Vereinfachungen lassen sich drei Varianten des analytischen Modells ableiten. Die erste Variante (Variante I) (siehe hierzu auch [NMW99]) geht von einer konstanten maximal pro Runde bedienbaren Anzahl an D-Aufträgen aus. $M$ wird in diesem Fall nicht als Zufallsvariable, sondern als Konstante betrachtet. Dies entspricht einer Bedienstrategie mit anzahllimitierter statischer Begrenzung (siehe Abschnitt 3.2).

Die zweite Variante (Variante II) betrachtet eine Bedienstrategie mit getrennter Einteilungsstrategie, bei der in jeder Runde zunächst alle K-Aufträge und danach alle D-Aufträge ausgeführt werden.

Die dritte Variante (Variante III) kombiniert die beiden ersten Varianten und geht weiterhin von einer FCFS-Bedienung der D-Aufträge innerhalb der Runde aus.

#### Variante I: anzahllimitierte statische Begrenzung

Bei einer konstant vorgegebenen oberen Schranke $M_{max}$ wird vorausgesetzt, daß $M_{max}$ D-Aufträge innerhalb einer Runde vollständig ausgeführt werden können. Der Wert von $M_{max}$ muß deshalb so gewählt werden, daß die Wahrscheinlichkeit, daß die Gesamtbedienzeit bei $N$ K-Aufträgen und $M_{max}$ D-Aufträgen die Rundendauer überschreitet, sehr gering ist. Die tatsächliche Anzahl der pro Runde ausgeführten D-Aufträge hängt von der Anzahl der zu Rundenbeginn in der Warteschlange verfügbaren D-Aufträge ab. Ist die Anzahl geringer als $M_{max}$, so wird die Warteschlange vollständig geleert, ansonsten werden $M_{max}$ Aufträge entnommen.

Durch das konstante Maximum ist eine Berechnung der Verteilung von $M$ nicht notwendig. Statt dessen gilt:

$$m_i = \begin{cases} 1 & : \quad i = M_{max} \\ 0 & : \quad \text{sonst} \end{cases}$$

Hierdurch entfällt bei den Chapman-Kolmogorov-Gleichungen 7.11 und 7.12 sowie den Gleichungen 7.14 und 7.15 durch Vereinfachung die äußere Summation. Die Leistungskapazität dieses Modells hat den Wert $M_{max}/T$.

Die Anzahl der Zwischenrunden, die ein Auftrag bis zur Abgangsrunde warten muß, beträgt exakt $\lfloor \frac{l}{M_{max}} \rfloor$, wenn zum Ankunftszeitpunkt sich bereits $l$ Aufträge in der Warteschlange befinden. Die Laplace-Transformierte $I_l^*$ vereinfacht sich somit zu:

$$I_l^*(s) = \int_0^\infty e^{-sx} P[I_l = x]\, dx = e^{-sT * \left\lfloor \frac{l}{M_{max}} \right\rfloor} \tag{7.34}$$

Für diese Variante muß weiterhin die Berechnung der Wahrscheinlichkeit $q_k$, daß ein D-Auftrag $k$ D-Aufträge in einer Runde sieht, angepaßt werden. Die Berechnung, die analog zur Gleichung 7.28 erfolgt, berücksichtigt zwei Fälle:

$$k = 1 \ldots M_{max} - 1: \quad q_k = \frac{k * p_k}{\sum_{j=1}^{M_{max}-1} j * p_j + M_{max} * (1 - \sum_{i=1}^{M_{max}-1} p_i)}$$

$$k = M_{max}: \quad q_k = \frac{M_{max} * \sum_{i=M_{max}}^{L_{max}} p_i}{\sum_{j=1}^{M_{max}-1} j * p_j + M_{max} * (1 - \sum_{i=1}^{M_{max}-1} p_i)}$$

Alle anderen Gleichungen einschließlich Gleichung 7.29 bleiben unverändert.

**Variante II: getrennte Einteilung in zwei Bedienabschnitte**

Soll das Modell an eine getrennte Bearbeitung der Aufträge angepaßt werden, so ist die Gleichung 7.29 des Ausgangsmodells zu verändern.

Wird angenommen, daß in einer Runde zunächst die Ausführung von $N$ K-Aufträgen durchgeführt wird und daß daraufhin die Bedienung der D-Aufträge stattfindet, so ergibt sich folgende Gleichung für $B^*$ mit ($N > 0$):

$$B^*(s) = \sum_{d=1}^{M_{max}} q_d \sum_{p=1}^{d} \frac{1}{d} * \begin{pmatrix} \text{(p+N)-fache Faltung der} \\ \text{Bedienzeit bei SCAN} \end{pmatrix} \qquad \text{mit} \tag{7.35}$$

$$\begin{pmatrix} \text{(p+N)-fache Faltung der} \\ \text{Bedienzeit bei SCAN} \end{pmatrix} = \left[ {T_{pos,SZ}^{scan_1}}^* (d+N, s) \right] * \left[ {T_{pos,SZ}^{scan_X}}^* (d+N, s) \right]^{p+N-1} * \left[ {T_{trans,SZ}^{S_K}}^* (s) \right]^N * \left[ {T_{trans,SZ}^{S_D}}^* (s) \right]^p * \left[ {T_{rot}}^* (s) \right]^{p+N}$$

Die Laplace-Transformierte der $(N + p)$-fachen Faltung der Bedienzeit ist hierbei einfacher aufgebaut als in Gleichung 7.31, da die Anzahl der K-Aufträge, die dem betrachteten D-Auftrag vorausgehen, konstant ist. Hierdurch entfällt die Iteration über die Anzahl vorangehender K-Aufträge und die Berechnung der Wahrscheinlichkeit für diese Fälle.

Eine weitere Vereinfachung ergibt sich, wenn ein Modell betrachtet wird, das die Bedienung der D-Aufträge vor den K-Aufträgen ausführt. Hierdurch entfällt der dritte Term in der Laplace-Transformierten.

**Variante III: Variante I + Variante II + FCFS-Anordnung**

Die dritte Variante verwendet eine anzahllimitierte statische Begrenzung wie Variante I und bedient K- und D-Aufträge in den Runden getrennt voneinander. Des weiteren erfolgt die Bearbeitung der D-Aufträge innerhalb einer Runde in FCFS-Reihenfolge. Hierdurch ist zum Ankunftszeitpunkt eines Auftrages $a$ sowohl die genaue Anzahl der Zwischenrunden als auch die Anzahl $(l \mod M_{max})$ der D-Aufträge, die vor $a$ in der Abgangsrunde bearbeitet werden, bekannt. Beide Werte ergeben sich aus der Anzahl der Aufträge, die zum Ankunftszeitpunkt in der Warteschlange stehen. Die Laplace-Transformierte $I_l^*$ ist in Gleichung 7.34 der Variante I angegeben. Die Laplace-Transformierte $B_l^*$, der Zufallsvariablen $B_l$ mit $P[B_l = t] = P[B = t|L = l]$ sieht wie folgt aus:

$$B_l^*(s) = \left[ {T_{pos,SZ}^{scan_1}}^*(N,s) \right] * \left[ {T_{pos,SZ}^{scan_X}}^*(N,s) \right]^{N-1} * \left( \left[ {T_{rot}}^*(s) \right] * \left[ {T_{trans,SZ}^{S_K}}^*(s) \right] \right)^N$$
$$* \left( \left[ {T_{pos,SZ}^{fcfs}}^*(s) \right] * \left[ {T_{rot}}^*(s) \right] * \left[ {T_{trans,SZ}^{S_D}}^*(s) \right] \right)^{(l \mod M_{max})+1} \quad (7.36)$$

Die Laplace-Transformierte ${T_{pos,SZ}^{fcfs}}^*$ der Positionierungszeit eines D-Auftrages ergibt sich aus der in Abschnitt 5.2.4 hergeleiteten Dichtefunktion $f_{pos,SZ}^{fcfs}$.

## 7.3 Maximale D-Last bei festgelegter Service-Qualität

Analog zu der Vorgehensweise für K-Aufträge in Abschnitt 6.3 läßt sich basierend auf dem analytischen Modell für das Antwortzeitverhalten durch eine inverse Berechnung die maximale Last $\lambda_{max}$ bestimmen, für die eine gewünschte Antwortzeitschranke eingehalten werden kann. Hierzu werden System- und Datenparameter, Schwellwert $\delta_{resp}$ der gewünschten Restwahrscheinlichkeit sowie die Antwortzeitschranke $\epsilon_{resp}$ vorgegeben. Aufgrund der konservativen Approximationen im Modell wird die berechnete Maximallast $\lambda_{max}^{\approx}$ kleiner ausfallen als die tatsächlich mögliche, die bei Simulationen oder bei einem realen Daten-Server beobachtet werden kann. Es gilt:

$$\lambda_{max} := \max\{\ \lambda\ :\ p_{resp}(\lambda, \epsilon_{resp}) \leq \delta_{resp}\ \} \quad (7.37)$$
$$\leq \max\{\ \lambda\ :\ b_{resp}(\lambda, \epsilon_{resp}) \leq \delta_{resp}\ \}$$
$$=: \lambda_{max}^{\approx} \quad (7.38)$$

Da eine Invertierung der Gleichungen nicht direkt durchführbar ist, muß auch hier die Suche nach dem Wert von $\lambda_{max}^{\approx}$ durch eine sich mehrfach wiederholende Berechnung von $b_{resp}(\lambda, \epsilon_{resp})$

durchgeführt werden. Diese Suche nach dem Maximum kann z.B. durch eine binäre Suche und einen geeigneten Startwert verkürzt werden.

## 7.4 Evaluation des analytischen Modells

Analog zu Abschnitt 6.4 erfolgt hier eine Evaluation des Warteschlangenmodells. Es werden die mit dem Warteschlangenmodell berechneten Werte für die Antwortzeitverteilung mit experimentell ermittelten Meßergebnissen verglichen.

Die Analyse erfolgt in mehreren Schritten, wobei im ersten und dritten Schritt ein Vergleich mit Simulationsergebnissen stattfindet. Alle analytischen Ergebnisse werden mit Maple [Kof96] unter Verwendung symbolischer und numerischer Verfahren berechnet.

- Schritt 1:
  Der erste Schritt berechnet die Dichtefunktion für die Verteilung der Anzahl der maximal pro Runde ausführbaren D-Aufträge (Gleichung 7.8).

- Schritt 2:
  Im zweiten Schritt werden die stationären Zustandswahrscheinlichkeiten der Markov-Kette durch das Lösung des Gleichungssystems (Gleichung 7.11,7.13) bestimmt.

- Schritt 3:
  Der dritte Schritt stellt die Laplace-Transformierte der Antwortzeit (Gleichung 7.4) auf und berechnet Erwartungswert, zweites Moment (Gleichung 7.32) und Chernoff-Schranke (Gleichung 7.33) der Restwahrscheinlichkeit.

Zunächst erfolgt eine Beschreibung der verwendeten Parameter

### 7.4.1 Parameter der Simulation und der Analyse

#### Die Plattenparameter

In den analytischen Berechnungen und den Simulationen werden dieselben Plattenparameter verwendet, die auch schon in den Abschnitten 4.1 und 6.4 benutzt worden sind. Die Simulation beschränkt sich wie die Analyse auf eine einzige Ein-Zonen-Platte mit einer konstanten Transferrate von $8.79$MBytes/sec.

#### Die Last- und Datenparameter

Um den Aufwand der Untersuchung zu reduzieren, werden exemplarisch einige typische Last- und Datenparameter betrachtet:

- Szenario $I_D$:
  Im ersten Szenario haben die K-Aufträge eine MPEG-2-charakteristische gammaverteilte Auftragsgröße: $E[S_K] = 800000$ Bytes, $Var[S_K] = (200000 \text{ Bytes})^2$. Es wird angenommen, daß insgesamt 7 K-Aufträge pro Runde ($N = 7$) bedient werden. Die Auftragsgrößen der D-Aufträge sind normalverteilt mit den Werten $E[S_D] = 50000$ Bytes und $Var[S_D] = (25000 \text{ Bytes})^2$.

- Szenario $II_D$:
  Im zweiten Szenario haben die K-Aufträge eine MPEG-1-charakteristische Auftragsgröße: $E[S_K] = 200000$ Bytes, $Var[S_K] = (100000 \text{ Bytes})^2$, und es werden pro Runde eine konstante Anzahl von 27 Aufträgen ($N = 27$) angenommen. Die Auftragsgrößenverteilung von D-Aufträgen entspricht der von Szenario $I_D$.

- Szenario $III_D$:
  Im dritten Szenario wird im Vergleich zu Szenario $I_D$ die Anzahl der K-Aufträge pro Runde auf 2 reduziert ($N = 3$) und dafür die Größe der D-Aufträge auf $E[S_D] = 200000$ Bytes und $Var[S_D] = (100000 \text{ Bytes})^2$ erhöht.

Szenario $I_D$ charakterisiert eine Situation mit hoher K-Last und geringer D-Last. In Szenario $III_D$ sind diese Rollen vertauscht. Die Rundendauer wird in allen Experimenten auf $T = 1$sec festgelegt.

**Die Parameter des Simulationsmodells**

Es werden zwei Scheduling-Algorithmen in verschiedenen Durchläufen getrennt voneinander simuliert. Als Vergleichsgrundlage zu den analytisch bestimmten Werten dient zum einen der in Abschnitt 3.2 vorgestellte Scheduling-Algorithmus III (*gemischte, dynamische, nicht inkrementelle SCAN-Strategie*), der die Berechnung der Ausführungspläne unter Kenntnis der exakten Rotationsverzögerung durchführt und dem das Warteschlangenmodell am ehesten entspricht. Zum anderen erfolgt ein Vergleich mit dem ebenfalls in Abschnitt 3.2 vorgestellten Scheduling-Algorithmus V (*gemischte, dynamische, voll inkrementelle SCAN-Strategie*), bei dem die niedrigsten mittleren Antwortzeiten erzielt worden sind.

Die Position der Aufträge auf der Platte wird zufällig bestimmt. Es wird angenommen, daß die Daten eines Auftrages zusammenhängend ohne erneute Positionierung innerhalb des Auftrages von der Platte gelesen werden können.

**Die Parameter des Warteschlangenmodells**

Die verwendeten Verteilungsfunktionen zur Beschreibung von Positionierungszeiten, Rotationsverzögerungen sowie Transferzeiten modellieren die in der Simulation verwendete Platte. Die maximale Warteschlangenlänge wird auf die doppelte Anzahl der pro Runde ausführbaren D-Aufträge gesetzt, d.h. $L_{max} = 2 * M_{max}$. Dieser Wert reicht aus, um die Abblockwahrscheinlichkeit $p_b$ gering und gleichzeitig den Berechnungsaufwand zur Lösung der Kolmogorov-Gleichungen bei den gegebenen Lastparametern auf einem akzeptablen Wert zu halten.

### 7.4.2 Verteilung der maximal pro Runde ausführbaren D-Aufträge

Da die Berechnung der Dichtefunktion der maximal pro Runde ausführbaren D-Aufträge in Gleichung 7.8 durch einen approximativen Ansatz erfolgt, soll zunächst deren Abweichung von den tatsächlichen Werten betrachtet werden, die in einer Simulation des Warteschlangenmodells erzielt werden. In Abbildung 7.4 ist die Verteilungsfunktion ($F_M(i) = P[M \leq i] = \sum_{i=0}^{i} m_i$) der Zufallsvariablen $M$ aus Simulation und Analyse zu sehen.

Der linke Graph betrachtet Szenario $I_D$. Aus den Berechnungen wird eine minimale Anzahl ausführbarer D-Aufträge pro Runde von $8$ und eine maximale Anzahl von $39$ abgeleitet, der sich relativ genau mit den experimentellen Ergebnissen deckt. Außerhalb dieses Bereichs sind die Werte für $m_i$ vernachlässigbar klein. Da die approximative Berechnung konservativ ist, liegt die Kurve ihrer Verteilungfunktion oberhalb der experimentell ermittelten Meßwertkurve. Der Erwartungswert der pro Runde bedienbaren D-Aufträge liegt in der Approximation bei $E[M] = 24.53$. Tatsächlich erreicht die Simulation einen Mittelwert von $E[M] = 24.79$.

Im rechten Graph gilt dasselbe für Szenario $II_D$. $M_{max}$ hat den Wert $29$ und $M_{min}$ den Wert $2$. Der Erwartungswert $E[M]$ der Approximation liegt bei $15.07$. Die Simulation ermittelt $E[M] = 15.57$.

Analoges gilt für Szenario $III_D$ mit $E[M] = 23.24$, $M_{min} = 16$ sowie $M_{max} = 31$.

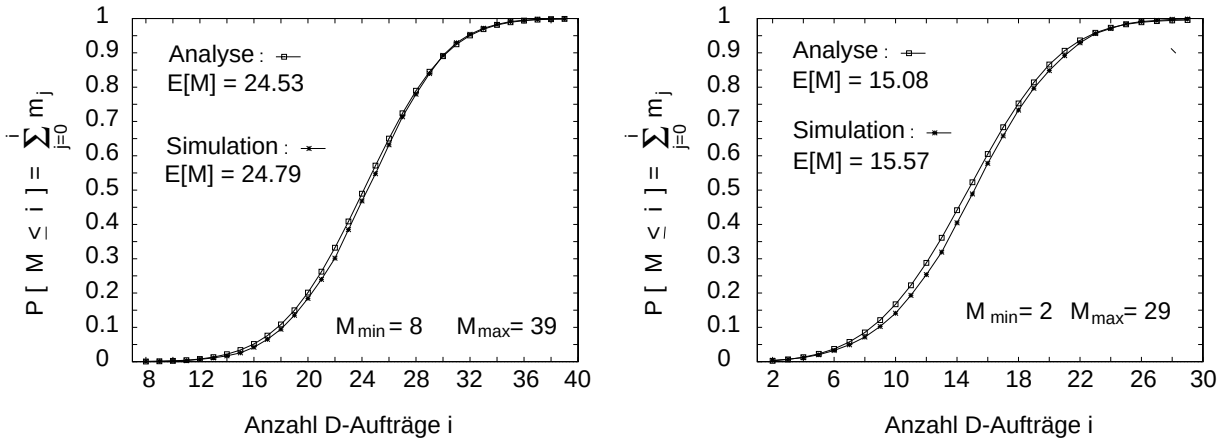



Abbildung 7.4: Verteilungsfunktion von $M$ bei analytischer Berechnung und experimenteller Simulation

### 7.4.3 Vergleich des Modells mit Scheduling-Algorithmus III

Der Vergleich zwischen der Analyse und dem Scheduling-Algorithmus III erfolgt zunächst auf Basis von Erwartungswert und zweitem Moment der Antwortzeitverteilung. In der Analyse wird hierzu die Laplace-Transformierte $R^*$ symbolisch abgeleitet und an der Stelle $0$ ausgewertet.

Tabelle 7.1 zeigt die Ergebnisse der Analyse und die Meßergebnisse der Simulation für Szenario $I_D$ bei unterschiedlichen Ankunftsraten. Alle Zeitangaben erfolgen in der Einheit Sekunden. Es ist ersichtlich, daß nur geringe Abweichungen hinsichtlich Erwartungswert und zweitem Moment zwischen Modell und Simulation existieren. Der relative Fehler des berechneten

| **Szenario $I_D$** | **Simulation** | | | **Analyse** | | | | |
|---|---|---|---|---|---|---|---|---|
| $\lambda[s^{-1}]$ | $E[R]$ | $E[R^2]$ | $p_{resp}(\lambda,2)$ | Last | $E[R]$ | $E[R^2]$ | $b_{resp}(\lambda,2)$ | $p_b$ |
| 15 | 0.96 | 1.08 | 0.001 | 87% | 0.96 | 1.21 | 0.029 | 3.60E-14 |
| 16 | 0.97 | 1.10 | 0.002 | 88% | 0.97 | 1.24 | 0.041 | 1.55E-12 |
| 17 | 0.98 | 1.13 | 0.004 | 89% | 0.99 | 1.28 | 0.059 | 5.12E-11 |
| 18 | 0.99 | 1.16 | 0.006 | 90% | 1.01 | 1.34 | 0.088 | 1.36E-9 |
| 19 | 1.02 | 1.22 | 0.009 | 92% | 1.04 | 1.41 | 0.132 | 3.00E-8 |
| 20 | 1.05 | 1.29 | 0.015 | 93% | 1.08 | 1.39 | 0.205 | 5.57E-7 |
| 21 | 1.08 | 1.39 | 0.026 | 94% | 1.14 | 1.56 | 0.324 | 8.67E-6 |
| 22 | 1.16 | 1.61 | 0.054 | 95% | 1.25 | 1.88 | 0.517 | 1.11E-4 |
| 23 | 1.30 | 2.06 | 0.113 | 97% | 1.45 | 2.61 | 0.786 | 1.09E-3 |
| 24 | 1.65 | 3.50 | 0.270 | 98% | 1.85 | 4.19 | 0.999 | 7.21E-3 |

Tabelle 7.1: Algorithmus III: Vergleich von Analyse und Simulation für Szenario $I_D$

Erwartungswertes steigt mit zunehmender Ankunftsrate an und liegt bei maximal $12\%$. Diese Abweichung ist hauptsächlich auf die konservative Approximation der Verteilung von $M$ und auf Berechnungsungenauigkeiten zurückzuführen.

Weiterhin werden die gemessene Restwahrscheinlichkeit $p_{resp}$ sowie die analytisch berechnete Restwahrscheinlichkeit $b_{resp}$ untersucht. Ausschlaggebend für die Bewertung soll im folgenden zum einen die Abweichung zwischen der Gesamtauslastung sein, die das analytische Modell bei festgelegtem Schwellwert $\delta_{resp}$ und Antwortzeitschranke $\epsilon_{resp}$ maximal erlaubt, und der Gesamtauslastung, die in der Simulation unter Einhaltung der festgelegten Service-Qualität maximal möglich ist. Die Gesamtauslastung ist der Quotient aus der Gesamtbedienzeit $T_{svc}$ aller D- und K-Aufträge einer Runde und der Rundenlänge $T$. Die Verwendung dieser Metrik berücksichtigt, daß im Warteschlangenmodell neben der D-Last auch die K-Last einfließt, die somit auch Einfluß auf die Vorhersagegenauigkeit nehmen kann.

Zum anderen soll die Abweichung zwischen der D-Last betrachtet werden, die das analytische Modell (siehe Abschnitt 7.3) maximal erlaubt, und der D-Last, die in der Simulation unter Einhaltung der festgelegten Service-Qualität maximal möglich ist. Es wird also die Abweichung zwischen $\lambda_{max}^{\approx}$ aus Gleichung 7.38 und $\lambda_{max}$ aus Gleichung 7.38 betrachtet. Hierbei werden die beiden Werte nicht explizit berechnet, sondern die Ankunftsrate $\lambda$ mit einer Schrittweite von $1$ sukzessive in der Simulation und im analytischen Modell erhöht, so daß nur Intervalle angegeben werden können, in denen die Werte von $\lambda_{max}$ und $\lambda_{max}^{\approx}$ liegen.

In Tabelle 7.1 wird eine Antwortzeitschranke $\epsilon_{resp}$ in der Größe der doppelten Rundendauer, d.h. $2$ Sekunden, sowie ein Schwellwert $\delta_{resp}$ von $5\%$ angenommen. Hieraus ergibt sich in der analytischen Berechnung eine maximale Ankunftsrate $\lambda_{max}^{\approx}$, die zwischen $16$ und $17$ liegt (Zeilen, in der dunkel hinterlegten Einträge vorhanden sind). Die mittlere Gesamtauslastung liegt bei dieser Ankunftsrate und $7$ K-Aufträgen pro Runde zwischen $88\%$ und $89\%$. Die Meßergebnisse zeigen, daß die tatsächlich mögliche Ankunftsrate $\lambda_{max}$ zwischen $21$ und $22$ liegen darf. Hierdurch ist eine höhere Auslastung zwischen $94\%$ und $95\%$ möglich. Das analytische Modell unterschätzt somit die Gesamtauslastung des Systems bei der geforderten Service-Qualität und den Parameterwerten um ca. $6\%$. Die Abweichung hinsichtlich der möglichen D-Last $\lambda_{max}$ ist größer und beträgt ca. $28\%$.

Ähnliche Ergebnisse sind auch in Tabelle 7.2 für Szenario $II_D$ bei geringerer K-Auftragsgröße

| **Szenario $II_D$** | **Simulation** | | | **Analyse** | | | | |
|---|---|---|---|---|---|---|---|---|
| $\lambda[s^{-1}]$ | $E[R]$ | $E[R^2]$ | $p_{resp}(\lambda,2)$ | Last | $E[R]$ | $E[R^2]$ | $b_{resp}(\lambda,2)$ | $p_b$ |
| 4 | 0.95 | 1.05 | 0.002 | 85% | 0.95 | 1.23 | 0.044 | 2.89E-19 |
| 5 | 0.96 | 1.07 | 0.003 | 86% | 0.96 | 1.25 | 0.057 | 1.78E-19 |
| 6 | 0.96 | 1.08 | 0.003 | 87% | 0.98 | 1.29 | 0.076 | 5.50E-16 |
| 7 | 0.98 | 1.12 | 0.004 | 88% | 0.99 | 1.34 | 0.103 | 2.23E-13 |
| 8 | 0.99 | 1.16 | 0.008 | 90% | 1.01 | 1.39 | 0.139 | 3.14E-11 |
| 9 | 1.01 | 1.19 | 0.009 | 91% | 1.04 | 1.47 | 0.193 | 2.09E-9 |
| 10 | 1.05 | 1.30 | 0.021 | 92% | 1.08 | 1.59 | 0.273 | 8.19E-8 |
| 11 | 1.09 | 1.40 | 0.032 | 94% | 1.14 | 1.77 | 0.392 | 2.11E-6 |
| 12 | 1.15 | 1.59 | 0.058 | 95% | 1.24 | 2.09 | 0.567 | 3.85E-5 |
| 13 | 1.26 | 1.95 | 0.103 | 96% | 1.42 | 2.76 | 0.795 | 5.05E-4 |
| 14 | 1.49 | 2.89 | 0.209 | 97% | 1.78 | 4.31 | 0.985 | 4.47E-3 |

Tabelle 7.2: Algorithmus III: Vergleich von Analyse und Simulation für Szenario $II_D$

| **Szenario $III_D$** | **Simulation** | | | **Analyse** | | | | |
|---|---|---|---|---|---|---|---|---|
| $\lambda[s^{-1}]$ | $E[R]$ | $E[R^2]$ | $p_{resp}(\lambda,2)$ | Last | $E[R]$ | $E[R^2]$ | $b_{resp}(\lambda,2)$ | $p_b$ |
| 18 | 0.96 | 1.09 | 0.003 | 83% | 0.97 | 1.21 | 0.028 | 1.83E-9 |
| 19 | 1.00 | 1.18 | 0.008 | 87% | 1.00 | 1.29 | 0.058 | 7.23E-7 |
| 20 | 1.04 | 1.28 | 0.014 | 90% | 1.05 | 1.42 | 0.121 | 2.27E-6 |
| 21 | 1.12 | 1.48 | 0.037 | 93% | 1.14 | 1.65 | 0.264 | 5.50E-5 |
| 22 | 1.29 | 1.99 | 0.103 | 96% | 1.32 | 2.21 | 0.562 | 9.41E-4 |

Tabelle 7.3: Algorithmus III: Vergleich von Analyse und Simulation für Szenario $III_D$

erkennbar. Um die Performance-Anforderungen einzuhalten, liegt die maximale Ankunftsrate $\lambda_{max}$ in der Simulation zwischen $11$ und $12$, wodurch der Daten-Server zwischen $94\%$ und $95\%$ ausgelastet wird. Das analytische Modell ermöglichst eine Gesamtauslastung zwischen maximal $85\%$ und $87\%$ bei einer Ankunftsrate $\lambda^{\approx}_{max}$ zwischen $4$ und $5$. In diesem Fall unterschätzt das analytische Modell die Gesamtauslastung des Systems absolut um ca. $9\%$. Die Abweichung in der Vorhersage der maximal möglichen D-Last ist besonders groß und beträgt maximal ca. $72\%$.

Tabelle 7.3 zeigt bei veränderter D-Auftragsgrößenverteilung eine größere Vorhersagegenauigkeit des analytischen Modells hinsichtlich der zu erwartenden mittleren Antwortzeit. Die Abweichung zwischen Meßwerten und analytischen Vorhersagen fallen geringer aus als bei Szenario $I_D$ und Szenario $II_D$. So beträgt der relative Fehler des berechneten Erwartungswertes maximal $2.4\%$ bei einer Ankunftsrate von $22$ Aufträgen pro Sekunde. Bei einem festgelegten Schwellwert von $\delta_{resp} = 0.05$ sagt das analytische Modell eine maximale Ankunftsrate $\lambda^{\approx}_{max}$ zwischen $18$ und $19$ mit einer Gesamtauslastung zwischen $83\%$ und $85\%$ voraus. Die tatsächlich mögliche Gesamtauslastung bei Einhaltung der Performance-Anforderungen liegt wiederum im Bereich zwischen $93\%$ und $96\%$ bei einer Ankunftsrate zwischen $21$ und $22$, so daß auch hier eine um ca. $9\%$ höhere Gesamtauslastung erreicht werden könnte. Der relative Fehler in der Vorhersage der maximal möglichen D-Last beträgt in diesem Fall ca. $19\%$.

| **Szenario $I_D$** | **Simulation** | | | | **Analyse** | | | |
|---|---|---|---|---|---|---|---|---|
| $\lambda[s^{-1}]$ | Last | $E[R]$ | $E[R^2]$ | $p_{resp}(\lambda, 2)$ | Last | $E[R]$ | $E[R^2]$ | $b_{resp}(\lambda, 2)$ |
| 15 | 91% | 0.28 | 0.13 | 0 | 87% | 0.96 | 1.21 | 0.029 |
| 16 | 92% | 0.29 | 0.14 | 0 | 88% | 0.97 | 1.24 | 0.041 |
| 17 | 93% | 0.31 | 0.15 | 0 | 89% | 0.99 | 1.28 | 0.059 |
| 18 | 95% | 0.33 | 0.17 | 0 | 90% | 1.01 | 1.34 | 0.088 |
| 19 | 96% | 0.36 | 0.20 | 0 | 92% | 1.04 | 1.41 | 0.132 |
| 20 | 97% | 0.41 | 0.26 | 0 | 93% | 1.08 | 1.39 | 0.205 |
| 21 | 98% | 0.49 | 0.39 | 0.006 | 94% | 1.14 | 1.56 | 0.324 |
| 22 | 99% | 0.69 | 0.80 | 0.037 | 95% | 1.25 | 1.88 | 0.517 |
| 23 | 100% | 1.46 | 3.97 | 0.248 | 97% | 1.45 | 2.61 | 0.786 |
| 24 | $> 100\%$ | $\infty$ | $\infty$ | $\infty$ | 98% | 1.85 | 4.19 | 0.999 |

Tabelle 7.4: Algorithmus V: Vergleich von Analyse und Simulation für Szenario $I_D$

### 7.4.4 Vergleich des Modells mit Scheduling-Algorithmus V

Um zu erkennen, inwieweit das Warteschlangenmodell auch zur Vorhersage des Verhaltens eines Algorithmus mit inkrementeller Auswahlstrategie verwendet werden kann, werden die Messungen für den Algorithmus mit gemischter, dynamischer, voll inkrementeller SCAN-Strategie wiederholt. Die analytisch ermittelten Werte bleiben unverändert. Im Gegensatz zu Algorithmus III des vorigen Abschnitts arbeitet der hier verwendete Algorithmus V mit einer Abschätzung der Rotationsverzögerung pro Auftrag. Aus diesem Grund können besonders bei hoher D-Last Ausführungspläne nicht optimal im voraus berechnet werden, was sich in einer geringeren Leistungskapazität des Algorithmus niederschlägt (siehe Abschnitt 4.3.5).

Wie zu erwarten, zeigen Erwartungswert und zweites Moment der Antwortzeit bei geringer D-Last für alle drei Szenarien ($I_D$, $II_D$, $III_D$) in den Tabellen 7.4, 7.5 und 7.6 eine hohe Abweichung. Erst in der Nähe der Leistungskapazität kommt es zu einer Annäherung der Werte, da hier die inkrementelle Auswahlstrategie D-Aufträge nicht mehr innerhalb der Runde, in der sie eintreffen, auswählen kann, sondern sie in die nächste Runde verzögern muß. Scheduling-Algorithmus V und Bedienstrategie des analytischen Modells verhalten sich in der Nähe der Leistungskapazität folglich ähnlich.

Im Vergleich zu Scheduling-Algorithmus III erfolgt der Anstieg von $p_{resp}$ von nahezu $0$ auf Werte von größer als $20\%$ sehr schnell. Es genügt bereits, $\lambda$ um ca. $2$ D-Aufträge pro Sekunde zu erhöhen. Der Schwellwert $\delta_{resp}$ hat damit nur geringen Einfluß auf die Abweichungen zwischen Simulation und Analyse.

Setzt man den Schwellwert $\delta_{resp}$ auf $5\%$ und die Antwortzeitschranke $\epsilon_{resp}$ erneut auf $2$ sec, so erlaubt der Scheduling-Algorithmus V für alle Szenarien eine nur minimal höhere D-Last $\lambda_{max}$, die circa bei einem zusätzlichen D-Auftrag pro Runde liegt. Die Auslastung des Simulationsmodells ist hierbei stets größer als $98\%$.

Als Folge der nur geringfügig größeren Werte für $\lambda_{max}$ ändern sich im Vergleich zu Scheduling-Algorithmus III die Abweichungen zwischen $\lambda_{max}$ und $\lambda^{\approx}_{max}$ nur unwesentlich, so daß auf eine Betrachtung des relativen Fehlers im folgenden verzichtet werden kann.

| **Szenario $II_D$** | **Simulation** | | | | **Analyse** | | | |
|---|---|---|---|---|---|---|---|---|
| $\lambda[s^{-1}]$ | Last | $E[R]$ | $E[R^2]$ | $p_{resp}(\lambda, 2)$ | Last | $E[R]$ | $E[R^2]$ | $b_{resp}(\lambda, 2)$ |
| 4 | 87 % | 0.28 | 0.13 | 0 | 85% | 0.95 | 1.23 | 0.044 |
| 5 | 88 % | 0.29 | 0.14 | 0 | 86% | 0.96 | 1.25 | 0.057 |
| 6 | 90 % | 0.31 | 0.15 | 0 | 87% | 0.98 | 1.29 | 0.076 |
| 7 | 91 % | 0.32 | 0.17 | 0 | 88% | 0.99 | 1.34 | 0.103 |
| 8 | 93 % | 0.34 | 0.19 | 0 | 90% | 1.01 | 1.39 | 0.139 |
| 9 | 94 % | 0.37 | 0.23 | 0.001 | 91% | 1.04 | 1.47 | 0.193 |
| 10 | 96 % | 0.43 | 0.30 | 0.003 | 92% | 1.08 | 1.59 | 0.273 |
| 11 | 97 % | 0.52 | 0.47 | 0.013 | 94% | 1.14 | 1.77 | 0.392 |
| 12 | 98 % | 0.70 | 0.90 | 0.048 | 95% | 1.24 | 2.09 | 0.567 |
| 13 | 99 % | 1.29 | 3.21 | 0.208 | 96% | 1.42 | 2.76 | 0.795 |
| 14 | $> 100\%$ | $\infty$ | $\infty$ | $\infty$ | 97% | 1.78 | 4.31 | 0.985 |

Tabelle 7.5: Algorithmus V: Vergleich von Analyse und Simulation für Szenario $II_D$

| **Szenario $III_D$** | **Simulation** | | | | **Analyse** | | | |
|---|---|---|---|---|---|---|---|---|
| $\lambda[s^{-1}]$ | Last | $E[R]$ | $E[R^2]$ | $p_{resp}(\lambda, 2)$ | Last | $E[R]$ | $E[R^2]$ | $b_{resp}(\lambda, 2)$ |
| 18 | 89% | 0.19 | 0.06 | 0 | 83% | 0.97 | 1.21 | 0.028 |
| 19 | 92% | 0.22 | 0.08 | 0 | 87% | 1.00 | 1.29 | 0.058 |
| 20 | 95% | 0.27 | 0.12 | 0 | 90% | 1.05 | 1.42 | 0.121 |
| 21 | 98% | 0.35 | 0.22 | 0.002 | 93% | 1.14 | 1.65 | 0.264 |
| 22 | 99% | 0.57 | 0.58 | 0.019 | 96% | 1.32 | 2.21 | 0.562 |
| 23 | 100% | 2.44 | 11.47 | 0.449 | 99% | 1.71 | 3.56 | 0.935 |

Tabelle 7.6: Algorithmus V: Vergleich von Analyse und Simulation für Szenario $III_D$

## 7.5 Fazit

Die in diesem Kapitel vorgestellte Analyse ist der erste Ansatz, der eine Vorhersage der Antwortzeit von Aufträgen im Kontext multimedialer Daten-Server durchführt. Im Gegensatz zu früheren Arbeiten auf vergleichbaren Gebieten, die fast ausschließlich eine FCFS-Ausführung der Aufträge annehmen, berücksichtigt das hier entwickelte analytische Modell die SCAN-Anordnung der Aufträge und benutzt ein detailliertes Plattenmodell. Durch die Verwendung von Laplace-Transformierten werden nicht nur Erwartungswerte, sondern vollständige Verteilungen erfaßt. Nur hierdurch ist es möglich, letztlich die Abschätzung der Restwahrscheinlichkeit vorzunehmen.

Die Evaluation hat gezeigt, daß eine Berechnung von Erwartungswert und Varianz sowie die Abschätzung der Restwahrscheinlichkeit der Antwortzeit praktisch möglich ist. Hierbei müssen allerdings gewisse Ungenauigkeiten in der Vorhersage hingenommen werden. Weiterhin konnte gezeigt werden, daß das analytische Modell auch bedingt die maximale D-Last berechnen kann, die der Scheduling-Algorithmus V unter Berücksichtigung der gewünschten Service-Qualität unterstützen kann.

# Kapitel 8

# Konfiguration des Daten-Servers

Basierend auf den entwickelten analytischen Modellen für Störungsrate und Antwortzeit wird in diesem Kapitel eine Konfigurationsmethode für Daten-Server mit gemischten Arbeitslasten vorgestellt. Das Ziel der Konfigurationsmethode ist es, die minimale Anzahl benötigter Magnetplatten zu bestimmen, um den Performance-Anforderungen der Benutzer bzw. Anwendungen zu genügen.

## 8.1 Der Konfigurationsalgorithmus

Die Vorhersagen der in den Kapiteln 6 und 7 entwickelten analytischen Modelle sind gültig bei Verwendung einer einzigen Magnetplatte. Damit bei diesen Modellen ebenfalls Vorhersagen über das gesamte Disk-Array gemacht werden können, sind einige Annahmen notwendig.

1. gleichmäßige, unabhängige Verteilung diskreter Aufträge
   Die Konfigurationsmethode nimmt an, daß der Poisson-Ankunftsstrom der D-Aufträge mit einer Gesamtankunftsrate von $\lambda_{tot}$ gleichmäßig über alle $P_{disks}$ Platten des Disk-Arrays verteilt wird. Hierdurch wird jede Platte von einem Poisson-Ankunftsstrom mit der Rate $\lambda = \lambda_{tot}/P_{disks}$ erreicht. Voraussetzung hierfür ist eine geeignete Datenallokation, die diese gleichmäßige Verteilung des Ankunftsstromes sicherstellt.

2. gleichmäßige, unabhängige Verteilung kontinuierlicher Aufträge
   Als weitere Annahme soll die Verteilung der K-Aufträge einer Runde ebenfalls gleichmäßig über alle Platten des Disk-Arrays erfolgen und eine Zulassungskontrolle pro Platte möglich sein. Daneben sollen die K-Aufträge auf den verschiedenen Platten unabhängig voneinander ausgeführt werden können. Dies ist z.B. dann gewährleistet, wenn grobkörniges, regelmäßiges Striping als Plazierungsverfahren (siehe Abschnitt 3.1.3) gewählt wird. Mit Hilfe der Zulassungskontrolle findet hierbei durch das Scheduling automatisch eine gleichmäßige Verteilung der K-Aufträge statt, so daß bei einer Gesamtzahl von $N_{tot}$ zu unterstützenden Datenströme maximal pro Platte $N = \lceil N_{tot}/P_{disks} \rceil$ K-Aufträge pro Runde verarbeitet werden müssen.

3. homogenes Disk-Array
   Da die Modelle an die Parameter einer einzigen Platte angepaßt werden, ist es notwen-

dig, daß das Disk-Array, auf das die Ergebnisse einer Platte hochgerechnet werden, aus identischen Platten besteht.

Die Performance-Anforderungen für D-Aufträge werden durch den Schwellwert $\delta_{resp}$ und die Antwortzeitschranke $\epsilon_{resp}$ vorgegeben. Für kontinuierliche Datenströme sind die Performance-Anforderungen durch den Schwellwert $\delta_{erate}$ und die Störungsratenschranke $\epsilon_{erate}$ bestimmt. Daten- und Plattenparameter sowie die Rundendauer $T$ sind implizit festgelegt und keine Eingabeparameter des Algorithmus.

Der Ablauf des Algorithmus, der in Abbildung 8.1 skizziert ist, sieht wie folgt aus. In einer Iteration (Zeile 6 bis 24) wird die Anzahl der Magnetplatten $P_{disks}$ sukzessive erhöht bis die Performance-Anforderungen alle erfüllt sind. Innerhalb einer Iteration wird zunächst die Höhe der Ankunftsrate $\lambda$ und die Anzahl der Datenströme $N$ pro Platte auf Basis der getroffenen Annahmen berechnet (Zeile 11, 13). Mit diesem Wert kann die Restwahrscheinlichkeit der Störungsrate bei gegebener Störungsratenschranke durch das analytische Modell aus Kapitel 6 bestimmt werden (Zeile 15). Wird der Schwellwert unterschritten, erfolgt als nächstes die Prüfung, ob die Konfiguration die D-Last verarbeiten kann (Zeile 17). Mit $N$ und den Daten- und Plattenparametern läßt sich die Verteilung der maximal pro Runde ausführbaren D-Aufträge gemäß des in Kapitel 7 vorgestellten Modells berechnen. Der Erwartungswert $E[M]$ der Anzahl der pro Runde ausführbaren D-Aufträge muß größer sein als die mittlere Anzahl $\lambda * T$ der pro Runde eintreffenden D-Aufträge (Zeile 18). Sollte diese Bedingung ebenfalls erfüllt sein, so wird zuletzt (Zeile 19) überprüft, ob die Schwelle der Restwahrscheinlichkeit für die Antwortzeit eingehalten werden kann. Erst wenn alle Performance-Anforderungen erfüllt sind, terminiert der Algorithmus mit der minimalen Anzahl $P_{disks}$ benötigter Magnetplatten.

Die Suche nach dem Minimum kann, anders als in Abbildung 8.1 gezeigt, z.B. durch eine binäre Suche effizienter und schneller durchgeführt werden.

## 8.2 Ein Konfigurationsbeispiel

Der Algorithmus soll an einem konkreten Beispiel illustriert werden, bei dem die Anzahl der Platten schrittweise um den Wert eins erhöht wird. Die Gesamtlast, die auf das zu konfigurierende System einwirken soll, ist mit einer Gesamtankunftsrate von $120$ D-Aufträgen $(\lambda_{tot} = 120)$ pro Sekunde und $50$ Datenströmen $(N_{tot} = 50)$ gegeben. Folgende Performance-Anforderungen werden gestellt. Die Wahrscheinlichkeit, daß die Antwortzeit eines D-Auftrages länger als die doppelte Rundendauer ist, soll kleiner sein als $5\%$, d.h. $b_{resp}(\lambda, 2\text{sec}) < 0.05$. Weiterhin soll die Wahrscheinlichkeit, daß die Störungsrate größer ist als $1\%$, unterhalb von $5\%$ liegen, d.h. $b_{erate}(N, 0.01) < 0.05$.

Die Auftragsgröße entspricht der von Szenario $I_D$ in Abschnitt 7.4. Die Rundendauer wird auf eine Sekunde festgelegt. Die verwendeten Plattenparameter sind die aus Abschnitt 7.4 und 6.4.

Nachdem der Algorithmus gestartet ist, hat der Wert von $P_{disks}$ nach wenigen Iterationen den Wert $7$ erreicht, bei dem jede Platte $8$ K-Aufträge $(N = \lceil \frac{50}{7} \rceil = 8)$ pro Runde zu verarbeiten hat. Entsprechend Tabelle 6.4 aus Abschnitt 6.4.3 kann für diese Konfiguration die Performance-Anforderung hinsichtlich der Störungsrate für kontinuierliche Daten erfüllt werden. Bei einem

```
1   procedure berechneAnzPlatten(λ_tot, N_tot, δ_resp, ε_resp, δ_erate, εerate)
2   begin
3    // Startwert, kann ggf. größer gewählt werden
4    P_disks = 0;
5    fertig=FALSE;
6    do
7      // Inkrement, ggf. kann P_disks in
8      // größeren Schritten erhöht werden
9      P_disks++;
10     // Ankunftsrate pro Platte
11     λ = λ_tot / P_disks;
12     // Datenströme pro Platte
13     N = ⌈ N_tot / P_disks ⌉;
14     // überprüfe Störungsrate
15     if (b_erate(N, ε_erate) < δ_erate) then
16      // überprüfe Auslastung
17      if (E[M] < λ * T) then
18       // überprüfe Antwortzeit
19       if ( b_resp(λ, ε_resp) < δ_resp) then
20         fertig = TRUE;
21       fi
22      fi
23     fi
24   while ( not(fertig) );
25   // Anzahl der benötigten Magnetplatten
26   // als Ergebnis zurückliefern
27   return P_disks;
28  end
```

Abbildung 8.1: Konfigurationsalgorithmus für Daten-Server mit gemischter Arbeitslast

geringeren Wert, z.B. mit $6$ Platten ($P_{disks} = 6$) und somit $9$ K-Aufträgen pro Runde ($N = \lceil \frac{50}{6} \rceil = 9$), ist dieses nicht gegeben.

Als nächster Schritt erfolgt die Berechnung der Verteilung für die maximale Anzahl ausführbarer D-Aufträge. Sie ergibt bei $8$ K-Aufträgen ($N = 8$) einen Erwartungswert von $E[M] = 16.99$. Die mittlere Ankunftsrate pro Platte liegt bei $17.14$ ($\lambda = \frac{120}{7}$). Dieser Wert liegt geringfügig über dem Erwartungswert $E[M]$. Die Leistungskapazität des Systems ist somit bei insgesamt $7$ Platten hinsichtlich diskreter Aufträge überschritten.

Es folgt eine weitere Iteration für $8$ Platten ($P_{disks} = 8$). Die Performance-Anforderung für kontinuierliche Datenströme ist natürlich nach wie vor erfüllt, und die mittlere Ankunftsrate pro Platte liegt mit $15$ ($\lambda = \frac{120}{8}$) unter dem Erwartungswert $E[M] = 24.51$. Des weiteren wird die Performance-Anforderung an D-Aufträge ebenfalls eingehalten. Mit $\lambda = \frac{120}{8} = 15$ und $N = \lceil \frac{50}{8} \rceil = 7$ liegt gemäß Tabelle 7.1 aus Abschnitt 7.4 die Restwahrscheinlichkeit für eine länger als $2$ Sekunden andauernde Antwortzeit bei $2.9\%$. Der Algorithmus terminiert mit dem Ergebnis $P_{disks} = 8$. Eine Zusammenfassung der Ergebnisse für dieses Konfigurationsbeispiel ist in Tabelle 8.1 zu finden.

Im Simulationsmodell werden tatsächlich ebenfalls mindestens acht Magnetplatten benötigt, damit bei den gewählten Lastparametern die Performance-Anforderungen erfüllt werden können.

| Lastparameter: $\lambda_{tot} = 120 \quad N_{tot} = 50$ |
|---|
| Service-Qualität: $b_{erate}(N, 0.01) < 0.05$ und $b_{resp}(\lambda, 2) < 0.05$ |

| **Analyse** | | | | | |
|---|---|---|---|---|---|
| $P_{disks}$ | $N$ | $\lambda$ | $E[M]$ | $b_{erate}(N, 0.01)$ | $b_{resp}(\lambda, 2)$ |
| 6 | 9 | 20 | 9.64 | 1 | 1 |
| 7 | 8 | 17.14 | 16.99 | 0 | 1 |
| 8 | 7 | 15 | 24.51 | 0 | 0.029 |

| **Simulation** | | | |
|---|---|---|---|
| $P_{disks}$ | $N$ | $b_{erate}(N, 0.01)$ | $b_{resp}(\lambda, 2)$ |
| 7 | 8 | 0.001 | 0.595 |
| 8 | 7 | 0 | 0.001 |

Tabelle 8.1: Werte zur Konfigurationsberechnung

## 8.3 Rekonfiguration

Über die Erstkonfiguration eines Systems hinaus kann der obige Algorithmus auch zur Rekonfiguration eines Daten-Servers benutzt werden, wenn sich die Lastcharakteristika verändern. Hierzu mißt der Daten-Server zu verschiedenen Zeitpunkten, z.B. mehrfach am Tag zu verschiedenen Uhrzeiten, die Lastparameter. Diese dienen dem Rekonfigurationsprogramm, eine Plattenkonfiguration zu berechnen, mit dem der Daten-Server der anspruchsvollsten Last unter den gewünschten Performance-Anforderungen gerecht werden kann.

Sollte die Analyse vorschlagen, die Anzahl der Magnetplatten um eine bestimmte Anzahl zu erhöhen, so ist eine Neuplazierung der vorhandenen Datenobjekte notwendig, um die zusätzliche Hardware auszunutzen. Bei den diskreten Daten erfolgt dies, wie in [Sal96, SWZ98] beschrieben, durch einen Online-Algorithmus, bei dem der Betrieb des Daten-Servers nicht unterbrochen werden muß. Bei kontinuierlichen Daten ist die Reorganisation bei laufendem Betrieb aufgrund der größeren Datenmenge und der periodischen Zugriffsmuster aufwendiger. Eine Möglichkeit besteht darin, das kontinuierliche Datenobjekt zunächst vollständig zu kopieren und hierbei entsprechend dem neuen Allokationsschema auf dem Disk-Array zu plazieren, bevor ein Zugriff auf das neue Objekt erlaubt wird. Zwischenzeitlich eintreffende Aufträge werden vom alten Objekt bedient. Ist der Kopiervorgang abgeschlossen, können die Aufträge auf das neue Objekt migriert und das alte Objekt gelöscht werden. Diese Methode hat den Vorteil, daß der Kopiervorgang ohne zeitliche Fristen stattfindet und deshalb unausgelastete Zeitabschnitte nutzen kann, ohne daß der normale Betrieb behindert wird. Hierzu werden die Fragmente des kontinuierlichen Objektes durch diskrete Datenzugriffe gelesen und neu geschrieben. Diesen diskreten Datenzugriffen kann eine geringere Priorität als den normalen diskreten Datenzugriffen zugeordnet werden, so daß sie nur dann bearbeitet werden, wenn keine normalen D-Aufträge in der Warteschlange stehen. Bei geringer Auslastung können so pro Runde auch mehrere Fragmente eines kontinuierlichen Datenobjektes kopiert werden. Der Nachteil dieses Ver-

fahrens besteht darin, daß das alte Objekt erst nach Abschluß des Kopiervorganges vollständig gelöscht werden kann. Alternativ hierzu ist es möglich pro Runde ein Fragment zu kopieren und die Vorlage dabei direkt zu löschen. K-Aufträge greifen nach Beginn des Kopiervorganges sofern möglich auf die noch (unvollständige) Kopie zu. Notwendig ist hierfür, daß das Lesen und Schreiben eines Fragmente in jeder Runde sichergestellt wird. Hierzu muß den K-Aufträgen des Kopiervorganges eine höhere Priorität als den normalen K-Aufträgen eingeräumt werden, da das Auslassen eines Fragments beim Kopieren zu einem unwiederbringlichen Datenverlust führt.

Während der Reorganisationsphase gelten die gemachten Annahmen der analytischen Modelle nicht. So wird z.B. das periodische Scheduling-Muster für Zugriffe auf Fragmente beeinträchtigt und durch den Kopiervorgang der Daten werden bestimmte Platten des Disk-Arrays stärker beansprucht. Daher kann während dieser Phase keine Service-Garantie für D- bzw. K-Aufträge abgegeben werden. Es ist sogar wahrscheinlich, daß sich die Performance während der Reorganisationphase verschlechtert und der Benutzer zeitlich begrenzt eine geringere Service-Qualität hinnehmen muß.

Der Beitrag der analytischen Modellierung und Performance-Vorhersage liegt darin, daß durch die genaue Berechnung die zusätzlich benötigten Magnetplatten und die dazugehörige Hardware (z.B. I/O-Karten) relativ exakt bestimmt werden kann. Ohne dieses Verfahren steht man vor dem Dilemma, entweder durch mehrfaches Hinzufügen von Hardware die Kapazität schrittweise anzupassen oder durch Hinzufügen einer größeren Anzahl von Platten die Kapazität des Daten-Servers in einem Schritt zu erweitern. Ersteres führt zu einer mehrfachen Reorganisation und den dadurch für den Benutzer verbundenen Nachteilen. Letzteres treibt, wenn die tatsächlich benötigte Kapazität weit überschritten wird, zu unangemessen hohen Kosten.

# Teil IV

# Architektur und Implementierung

# Kapitel 9

# Prototypimplementierung

Neben dem im ersten Teil vorgestellten Simulator existiert eine Implementierung eines auf realer Hardware basierenden Prototyps. Im Gegensatz zum Simulationsmodell aus Abschnitt 4.2 speichert diese prototypische Implementierung reale Daten und benutzt hierzu ein reales Disk-Array.

Im folgenden soll zunächst in Abschnitt 9.1 ein Überblick über das Gesamtsystem gegeben werden. Daraufhin erfolgt in Abschnitt 9.2 eine Kurzbeschreibung zur Implementierung des Daten-Servers. Den Abschluß in Abschnitt 9.3 bilden Performance-Messungen, die am implementierten System durchgeführt worden sind.

## 9.1 Das Gesamtsystem

Neben der Verwaltung der Daten durch den Daten-Server muß für die praktische Verwendbarkeit eines Gesamtsystems weitere Funktionalität zur Verfügung gestellt werden. Hierzu gehört die Darstellung der Daten, die Administration sowie Möglichkeiten zur Überwachung des Betriebs.

### 9.1.1 Wiedergabe und Darstellung der Daten

Die Visualisierung der auf dem Daten-Server abgelegten Daten geschieht über einen Web-Browser, der als Schnittstelle zum Endbenutzer des multimedialen Informationssystems dient. Hierdurch erreicht man eine größtmögliche Flexibilität, da diese Programme (Netscape, IExplorer, HotJava) zum einen bereits die Darstellung einer großen Anzahl unterschiedlicher Dokumenttypen (z.B. HTML, GIF, JPEG, ASCII) unterstützen, zum anderen nicht-unterstützte Formate durch die Einbindung externer Programme (z.B. Word, PowerPoint), durch *Plug-Ins* (z.B. FlashPlayer) oder durch Java-Programme berücksichtigt werden können. Letzteres ist z.B. notwendig, um Audio- oder Videodaten vom Daten-Server zu empfangen und wiederzugeben. Durch die Verwendung des Web-Browsers erfolgt die Adressierung der diskreten Datenobjekte durch Angabe einer URL. Die Kommunikation zwischen Web-Browser und Daten-Server geschieht über das HTTP-Protokoll.

Um diskrete und kontinuierliche Daten gemeinsam auf einer Seite des Web-Browsers darstellen zu können (siehe Abbildung 9.1), erfolgt die Präsentation von Audio- oder Videodaten durch ein Wiedergabe-Applet, das als Java-Programm in eine HTML-Seite integriert wird. Der Programmcode des Applets wird als diskretes Datenobjekt auf dem Server gespeichert und vom Web-Browser bei Bedarf geladen. Aufgaben des Wiedergabe-Applets sind, über eine Benutzerschnittstelle ggf. die Auswahl eines kontinuierlichen Datenobjektes zu ermöglichen, die Verbindung zum Daten-Server aufzubauen, vom Daten-Server gelieferte Fragmente ggf. zwischenzuspeichern, den kontinuierlichen Datenstrom zu dekodieren und darzustellen sowie das Anhalten und Starten der Wiedergabe durch den Benutzer zu ermöglichen.

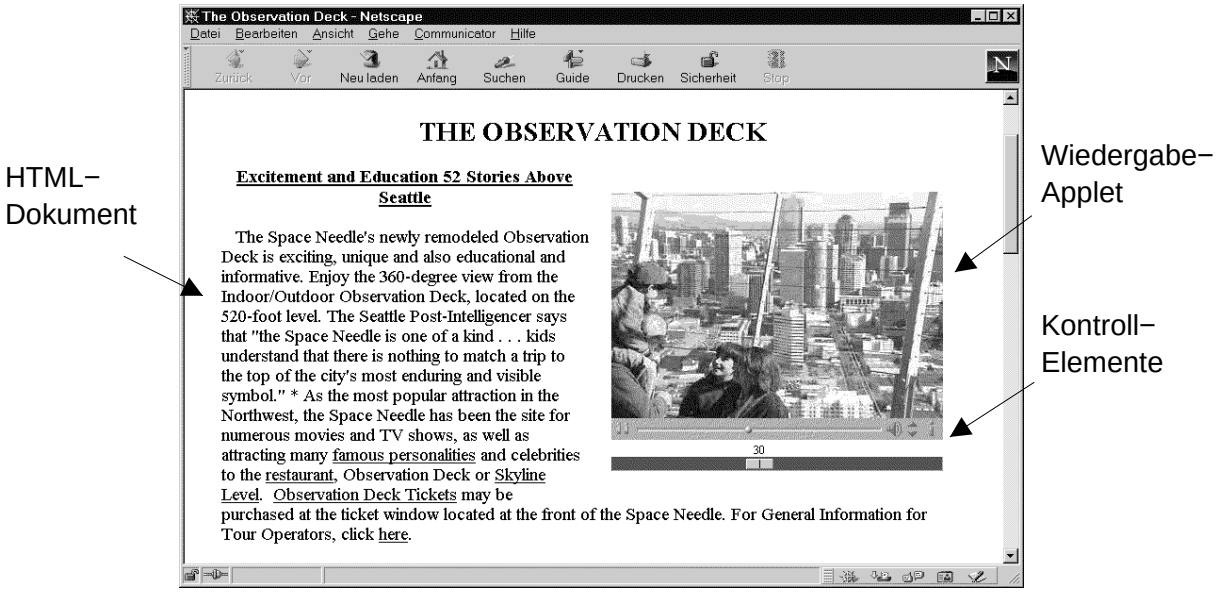



Abbildung 9.1: Integrierte Darstellung kontinuierlicher und diskreter Datenobjekte innerhalb eines Web-Browsers

Für die Dekodierung und Darstellung der Daten sowie die Interaktion mit dem Benutzer wird die von der JMF-API (Java Media Framework) [Sun98] zur Verfügung gestellte Wiedergabekomponente verwendet. Die Implementierungsaufgabe beschränkt sich damit auf die Realisierung eines Kontroll- und Puffer-Moduls (siehe Abbildung 9.2), mit dem die Anbindung und Anpassung dieser Wiedergabekomponente an den Daten-Server durchgeführt wird.

Der Daten-Server liefert über eine Datenverbindung die Fragmente an das Puffer-Modul, wo die Zwischenspeicherung erfolgt, bis die Daten von der JMF-Wiedergabekomponente gelesen werden. Das Kontroll-Modul überwacht den Füllstand des Puffers und stoppt bzw. startet gegebenenfalls über eine Kontrollverbindung den Datenstrom vom Server. Weiterhin wird das Kontroll-Modul von der Wiedergabekomponente über Benutzerinteraktionen informiert und kann so den Neustart des Datenstromes bzw. einen zeitlichen Sprung im Datenstrom durch Befehle an den Daten-Server initiieren. Der Datenaustausch zwischen Wiedergabe-Applet und Daten-Server erfolgt über ein einfaches Kommunikationsprotokoll.

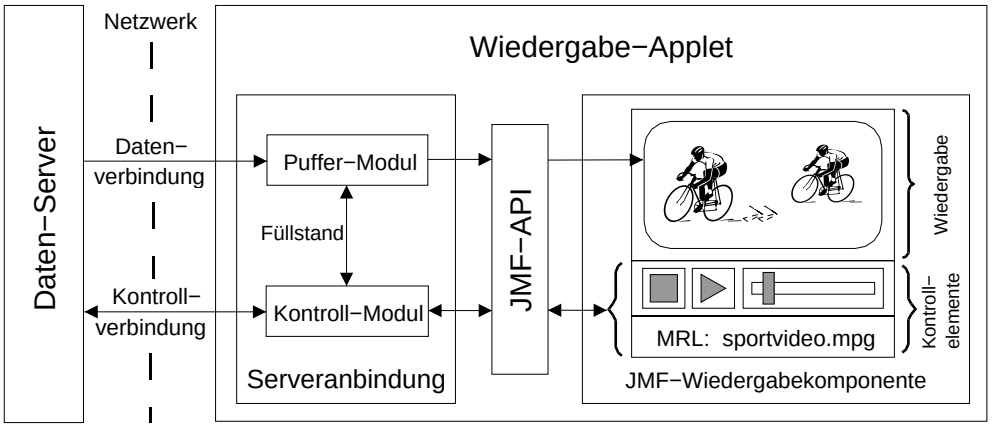
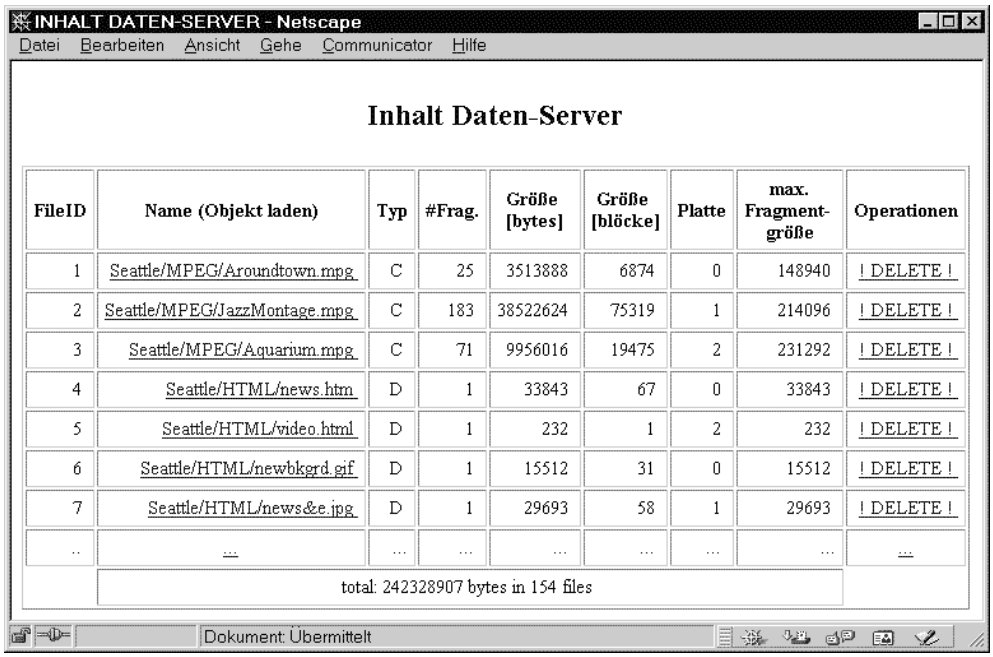



Abbildung 9.2: Komponenten des Wiedergabe-Applets und deren Anbindung an den Daten-Server

### 9.1.2 Administration des Daten-Servers

Durch externe Administrationprogramme sowie eine Web-Browser-basierte Schnittstelle (siehe Abbildung 9.3) erfolgt die Verwaltung der auf dem Daten-Server abgelegten Daten. Hierzu gehört in der prototypischen Implementierung das Anzeigen eines Inhaltsverzeichnisses mit allen auf dem Server abgelegten Datenobjekten, das Kopieren neuer Daten vom lokalen Sekundärspeicher über das Netzwerk auf den Daten-Server und das Löschen nicht mehr benötigter Datenobjekte vom Server.




| FileID | Name (Objekt laden) | Typ | #Frag. | Größe [bytes] | Größe [blöcke] | Platte | max. Fragment-größe | Operationen |
|---|---|---|---|---|---|---|---|---|
| 1 | Seattle/MPEG/Aroundtown.mpg | C | 25 | 3513888 | 6874 | 0 | 148940 | ! DELETE ! |
| 2 | Seattle/MPEG/JazzMontage.mpg | C | 183 | 38522624 | 75319 | 1 | 214096 | ! DELETE ! |
| 3 | Seattle/MPEG/Aquarium.mpg | C | 71 | 9956016 | 19475 | 2 | 231292 | ! DELETE ! |
| 4 | Seattle/HTML/news.htm | D | 1 | 33843 | 67 | 0 | 33843 | ! DELETE ! |
| 5 | Seattle/HTML/video.html | D | 1 | 232 | 1 | 2 | 232 | ! DELETE ! |
| 6 | Seattle/HTML/newbkgrd.gif | D | 1 | 15512 | 31 | 0 | 15512 | ! DELETE ! |
| 7 | Seattle/HTML/news&e.jpg | D | 1 | 29693 | 58 | 1 | 29693 | ! DELETE ! |
| .. | ... | ... | ... | ... | ... | ... | ... | ... |
| | total: 242328907 bytes in 154 files | | | | | | | |

Abbildung 9.3: Administration des Daten-Servers mit Web-Browser-basierter Schnittstelle: Anzeige des Inhaltsverzeichnisses mit der Option zum Löschen von Objekten

Ein Kopierprogramm ermöglicht es, sowohl einzelne Dateien als auch gesamte Verzeichnisstrukturen eines lokalen Dateisystems auf den Daten-Server zu übertragen. Hiermit kann die

erstmalige Initialisierung und eine fortlaufende Ergänzung der Inhalte durchgeführt werden. Das Kopierprogramm setzt voraus, daß diskrete Datenobjekte in ihrer ursprünglichen Form in lokalen Dateien vorliegen. Kontinuierliche Datenobjekte müssen vor Aufruf des Kopierprogrammes partitioniert werden. Dies geschieht durch separate Programme, die das Kodierungsformat eines kontinuierlichen Datenobjektes (z.B. MPEG, AVI, WAV) interpretieren und hierdurch die Zerlegung geeignet, z.B in Fragmente konstanter Wiedergabelänge, durchführen können. Es ist vorgesehen, daß diese Partitionierungsprogramme die Fragmente eines kontinuierlichen Datenobjektes in einem Verzeichnis des lokalen Dateisystems sammeln, indem sie dort jedes Fragment als einzelne Datei ablegen. Die Wiedergabelänge jedes Fragments, die der vom Daten-Server verwendeten Rundendauer entsprechen muß, wird den Partitionierungsprogrammen als Parameter übergeben. Zur Zeit existiert nur ein Partitionierungsprogramm für MPEG-1-kodierte kontinuierliche Objekte. Eine Ergänzung des Systems zur Unterstützung weiterer Kodierungsformate ist geplant und kann durch die Entwicklung weiterer Partitionierungsprogramme leicht realisiert werden, ohne daß hierdurch Veränderungen am Daten-Server oder an den Administrationsprogrammen notwendig werden.

Die beiden anderen erwähnten Programme zum Anzeigen des Inhaltsverzeichnisses und zum Löschen von Datenobjekten weisen eine ähnliche Funktionalität auf wie die vom lokalen Dateisystem bekannten Programme, z.B. del, rm. Nähere Erläuterungen zur Funktionweise, zum Funktionsumfang sowie zur Implementierung sind in [Wen99] zu finden.

### 9.1.3 Überwachung des Daten-Servers

Zur Beobachtung des Daten-Servers wird ein externer Überwachungsmonitor eingesetzt. Auf dem derzeitigen Implementierungsstand wird die Plattenaktivität, die Anzahl der zugelassenen, die Anzahl der angehaltenen sowie die Anzahl der aktiven Datenströme angezeigt. Der Überwachungsmonitor unterhält zum Daten-Server eine permanente Netzwerkverbindung, über die er vom Daten-Server über dessen Zustandsänderungen in Echtzeit informiert wird.

Abbildung 9.4 zeigt die Ausgabe des Überwachungsmonitors bei einem Daten-Server, dessen Disk-Array aus zwei Magnetplatten besteht. Auf der linken Seite der Abbildung sind durch zwei übereinanderliegende Rechtecke die beiden Magnetplatten mit insgesamt $8000$ Zylindern angedeutet. Das Dreieck oberhalb eines Rechtecks gibt die aktuelle Position des Plattenkopfes an. Innerhalb des Rechtecks werden wartende Aufträge durch senkrechte Linien auf der Zylinderposition eingezeichnet, an der der Datenzugriff stattfinden wird. Nachdem der Plattenkopf entsprechend der Scheduling-Strategie diese Positionen angefahren hat, werden nach der Ausführung der Aufträge die Linien gelöscht.

D-Aufträge und K-Aufträge werden farblich (blau bzw. rot) unterschieden. In der Abbildung sind D-Aufträge dunkelgrau und K-Aufträge hellgrau gekennzeichnet. Rechts zeigt die Überwachungskomponente an, wann die Datenzugriffe erfolgen. Hierzu wird ein Ausschnitt von zwei Sekunden Dauer aus der Zeitachse eingeblendet, der mit fortschreitender Zeit kontinuierlich von rechts nach links verschoben wird. Der Zeitachsenausschnitt ist wiederum in Runden mit einer Länge von einer Sekunde eingeteilt. Innerhalb einer Runde wird durch ein Rechteck der Beginn und das Ende eines Plattenzugriffs gekennzeichnet. Die Farbe bzw. Schattierung der Rechtecke gibt an, ob es sich um die Ausführung eines D- bzw. K-Auftrages handelt. Am unteren Rand zeigt die Statuszeile an, wie viele Datenströme zur Zeit beim Daten-Server insgesamt

angemeldet und zugelassen sind, wie viele zur Zeit gestoppt sind und wie viele zur Zeit mit Fragmenten beliefert werden.

Denkbar ist hier, durch Erweiterungen die aktuelle Auslastung des Daten-Servers und weitere Performance-Informationen (z.B. Antwortzeit, Startverzögerung, Störungsrate) anzuzeigen.

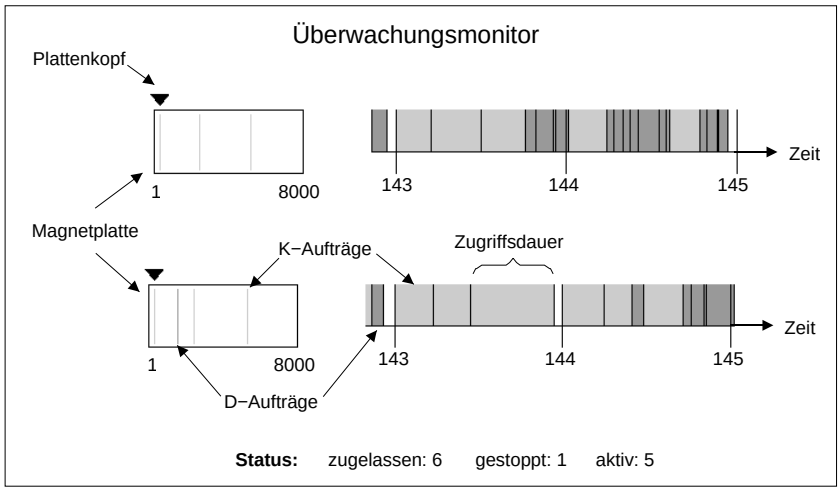



Abbildung 9.4: Überwachungsmonitor zur Darstellung der Plattenaktivität

## 9.2 Der Daten-Server

Der Daten-Server, dessen Architektur in Abbildung 9.5 skizziert ist, speichert kontinuierliche und diskrete Datenobjekte auf einem Disk-Array. Kontinuierliche Datenobjekte werden hierauf gemäß dem grobkörnigen, regelmäßigen Striping (GRS, siehe Abschnitt 2.2.2) abgelegt. Die Plazierung eines diskreten Datenobjektes erfolgt ohne Zerlegung vollständig zusammenhängend. Der Zugriff auf die Platten geschieht über eine block-basierte Schnittstelle des Betriebssystems (z.B. Unix Raw-Device). Es wird ein rundenbasiertes, periodisches Platten-Scheduling mit variabler Reihenfolge der K-Aufträge (siehe Abschnitt 3) verwendet. Die Speicherung von persistenten Meta-Daten, wie z.B. Konfigurationsdateien oder Informationen über das Server-Dateisystem, geschieht auf einer separaten Magnetplatte. Die Ablage der Meta-Daten erfolgt in Dateien des vom Betriebssystem zur Verfügung gestellten Dateisystems.

Die verschiedenen Module des Daten-Servers, auf die im folgenden näher eingegangen wird, wurden im Rahmen von zwei Diplomarbeiten implementiert. Details der Implementierung finden sich in [Sai98] und [Wen99].

### 9.2.1 Die Dateiverwaltung

Das Dateisystem des Daten-Servers übernimmt die Buchführung über die auf dem Daten-Server abgelegten Daten. Hierzu zählt z.B. die Verwaltung des Freispeichers auf den Magnetplatten und der Aufbau von Strukturen, die es erlauben, die Adresse eines Datenobjektes, d.h. die Plattennummer sowie Zylinder und Blocknummer, zu ermitteln. Zudem werden Namen und Größe

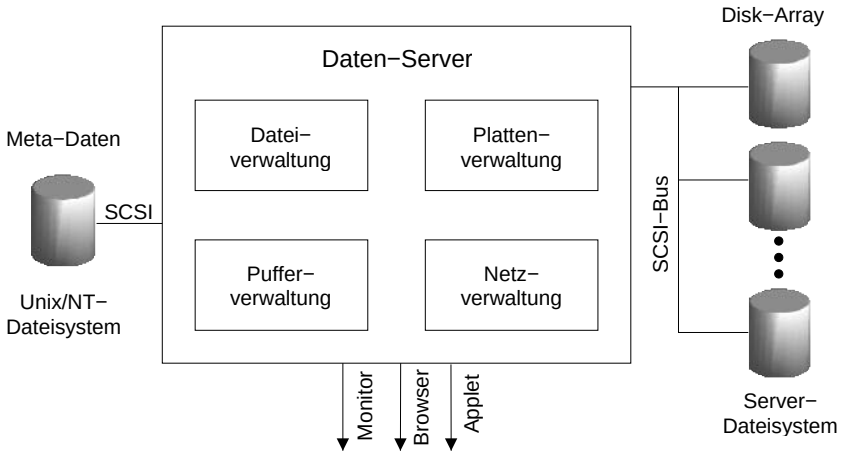



Abbildung 9.5: Architektur des Daten-Servers

der Datenobjekte gespeichert. Diese Meta-Daten werden persistent in mehreren Dateien auf einer separaten Platte abgelegt, von wo sie während der Initialisierung des Servers einmalig gelesen werden. Während des Betriebs werden alle Meta-Daten im Hauptspeicher gehalten, so daß ein hinreichend schneller Zugriff auf die von den Scheduling-Algorithmen benötigten Informationen gewährleistet werden kann.

### 9.2.2 Die Pufferverwaltung

Die Pufferverwaltung alloziert Speicher im Hauptspeicher, in dem Fragmente und diskrete Datenobjekte nach dem Lesen von der Magnetplatte zwischengespeichert werden. Das Verschicken der Daten über die Netzwerkverbindung an den Web-Browser bzw. das Wiedergabe-Applet wird hierdurch vom Lesevorgang entkoppelt, so daß der Zugriff auf die Magnetplatten und der Zugriff auf das Netzwerk mit Hilfe des Betriebssystems parallel stattfinden kann.

In weiteren möglichen Implementierungsstufen ließen sich innerhalb dieser Komponente geeignete Caching-Algorithmen [RZ95, DDM+95, MNO+96, GLM94] realisieren.

### 9.2.3 Die Netzverwaltung

Die Netzverwaltung kontrolliert die Verbindungen zum Überwachungsmonitor, zum Wiedergabe-Applet und zum Web-Browser. Um die Anfragen des Web-Browsers zur Lieferung diskreter Daten entgegennehmen zu können, implementiert die Netzverwaltung eine Teilmenge des HTTP-Protokolls. Die Implementierung weiterer Protokolle, z.B. des FTP-Protokolls, für den Zugriff auf diskrete Datenobjekte ist denkbar.

Das Protokoll zwischen Wiedergabe-Applet und Daten-Server umfaßt Befehle zum Auf- und Abbau der Verbindung sowie Kommandos zur Steuerung des kontinuierlichen Datenstroms. Zwischen Wiedergabe-Applet und Daten-Server werden pro Datenstrom zwei Verbindungen geöffnet. Die erste Verbindung (Datenverbindung) ist unidirektional und dient zur Übertragung des kontinuierlichen Datenstromes. Über die zweite Verbindung (Kontrollverbindung) werden in beide Richtungen Kommandos und Informationen zwischen Applet und Server ausgetauscht.

Die Netzverwaltung leitet die Kommandos an die Plattenverwaltung weiter bzw. empfängt von hier die Rückgabedaten.

Die Netzverwaltung verwendet in der vorliegenden Implementierung die Möglichkeit, durch das Betriebssystem mehrere Threads (parallele Prozesse innerhalb desselben Adressraumes) zu erzeugen, indem es für jede Datenverbindung zum Wiedergabe-Applet und für jede HTTP-Verbindung einen eigenen Thread benutzt. Jeder Thread initiiert und führt das Verschicken der Daten auf der ihm zugeordneten Verbindung selbsttätig quasi parallel zu den anderen Threads durch. Das Scheduling des Netzwerkzugriffs für mehrere parallele Verbindungen obliegt hierbei dem Betriebssystem.

Wird ähnlich dem Plattenzugriff eine vollständige Kontrolle für den Zugriff auf die Netzwerkressource gewünscht, um z.B. eine Netzüberlastung besser kontrollieren zu können, so kann in dieser Komponente ein Netzwerk-Scheduler implementiert werden, der den Zugriff auf das Netzwerk, z.B. für verschiedene Scheduling-Strategien, steuert.

### 9.2.4 Die Plattenverwaltung

Die Plattenverwaltung steuert den Zugriff auf das gesamte Disk-Array und führt die Schreib- und Leseaufträge aus. Der Aufbau der Plattenverwaltung ist in Abbildung 9.6 skizziert.

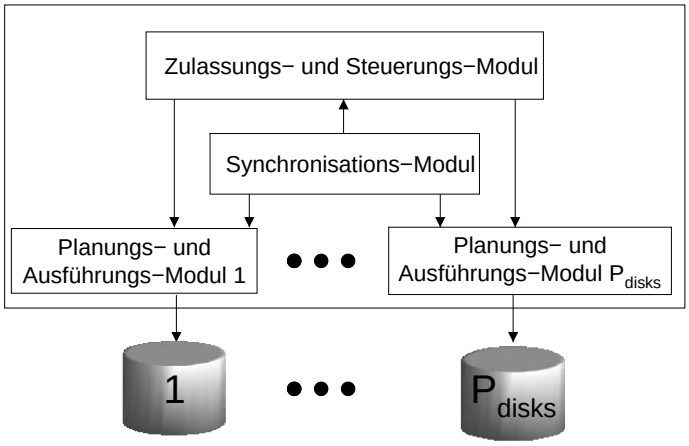



Abbildung 9.6: Module der Plattenverwaltung

#### Das Zulassungs- und Steuerungs-Modul

Das Zulassungs- und Steuerungs-Modul überwacht die Anzahl und den Zustand der vom Daten-Server gelieferten Datenströme. Die maximale Anzahl paralleler Datenströme wird mit Hilfe des in Kapitel 6 vorgestellten analytischen Modells berechnet und diesem Modul als Parameter in einer Konfigurationsdatei übergeben. Sollte die maximale Anzahl erreicht sein, so werden Anfragen zum Starten neuer Datenströme mit einer entsprechenden Fehlermeldung abgelehnt, die über die Netzverwaltung an das Wiedergabe-Applet zurückgeschickt werden. Das Zulassungs- und Steuerungs-Modul ermöglicht es, kontinuierliche Datenströme ab einem beliebigen Fragment zu starten und anzuhalten. Außerdem können Ausschnitte eines kontinuierlichen Datenobjektes angefordert werden. Eine besondere Behandlung erfahren D-Aufträge,

deren Bedienung aufgrund ihrer Größe zeitlich nicht innerhalb einer Runde ausführbar ist. Ab einer festgelegten Größe erfolgt die Zerlegung von D-Aufträgen in mehrere Teilaufträge, die unabhängig voneinander bearbeitet und hierdurch von dem im Ausführungs-Modul arbeitenden Scheduling-Algorithmus flexibler in den Ausführungsplan integriert werden können.

#### Das Planungs- und Ausführungs-Modul

Das Planungs- und Ausführungs-Modul koordiniert die Ausführung der Schreib- und Leseaufträge auf dem Disk-Array. Zur gemeinsamen Ausführung von D- und K-Aufträgen ist eine gemischte, dynamische, beschränkt inkrementelle SCAN-Scheduling-Strategie in Form von Algorithmus IV (siehe Abschnitt 3.3) implementiert. Ausschlaggebend für die Wahl dieses Algorithmus ist dessen geringer Berechnungsaufwand bei der Aufstellung der Ausführungspläne, so daß ein Betrieb auch auf Systemen mit geringer Prozessorleistung möglich ist. Damit die Zugriffe auf die Platten des Disk-Array parallel stattfinden können, existiert für jede Platte ein eigenes Planungs- und Ausführungs-Modul, in dem ein Thread eigenständig die Plattenzugriffe ausführt. Durch einen weiteren Thread werden parallel zum Plattenzugriff K-Aufträge für die Folgerunde erzeugt und in eine Warteschlange eingefügt.

Die vom Scheduling-Algorithmus zur Erstellung der Ausführungspläne angenommenen Systemparameter, wie z.B. Plattenparameter und Rundendauer sind frei konfigurierbar. Im Betrieb verwendet der Daten-Server die Rundendauer von einer Sekunde. Die Plattenparameter sind an die verwendete Hardware angepaßt.

#### Das Synchronisations-Modul

Das Synchronisations-Modul koordiniert den Datenaustausch zwischen den einzelnen Planungs- und Ausführungs-Modulen sowie dem Zulassungs- und Steuerungs-Modul. Hierzu sendet es einen periodischen Zeittakt am Anfang jeder Runde und wartet danach auf eine Rückmeldung der einzelnen Module.

## 9.3 Performance-Messungen

Um die Funktions- und Leistungsfähigkeit der Implementierung zu demonstrieren, sind verschiedene Performance-Messungen durchgeführt worden. Die Messungen beruhen auf zufällig erzeugten Daten, die vom Daten-Server in einem speziellen Testmodus erzeugt werden können. Diese Testdaten gehören zu $27$ kontinuierlichen Datenobjekten mit einer Wiedergabedauer von $500$ Sekunden und ca. $5000$ diskreten Datenobjekten. Die gammaverteilte Größe der Fragmente beträgt im Mittel $200$ KBytes mit einer Standardabweichung von $50$ KBytes. Die normalverteilte Größe der diskreten Datenobjekte hat im Mittel einen Wert von $40$ KBytes mit einer Standardabweichung von $20$ KBytes.

Gemessen wird die Leistungsfähigkeit des Servers hinsichtlich des Plattenzugriffs. Netzwerkverzögerungen werden nicht berücksichtigt. Die Aufträge werden innerhalb des Daten-Servers durch ein Test-Modul erzeugt und unter Umgehung des Netzwerk-Moduls direkt an das Zulassungs- und Kontroll-Modul weitergeleitet. Die gemessene Antwortzeit der D-Aufträge ist die

Zeitspanne zwischen dem Erzeugen eines Auftrages und der Bereitstellung der gewünschten Daten durch das Puffer-Modul im Hauptspeicher des Daten-Servers. Das Versenden der Daten über das Netzwerk findet nicht statt.

Die Messungen werden auf einer SUN Mehrprozessor-Architektur (SparcCenter $2000$, $10$ Prozessoren à $40$ MHz) durchgeführt. Die Speicherkapazität der verwendeten Magnetplatten beträgt insgesamt $2.1$ GBytes. Die Transferrate hat einen maximalen Wert von ca. $2$ MBytes/sec.

In der ersten Messung wird das Skalierungsverhalten der Implementierung untersucht. Hierzu werden pro Platte und Runde jeweils $3$ K-Aufträge ausgeführt. Abbildung 9.7 zeigt die mittlere Antwortzeit von D-Aufträgen bei unterschiedlichen Ankunftsraten. Diese Messung wird für zwei Disk-Arrays mit einer und mit drei Magnetplatten durchgeführt. Das Ergebnis zeigt, daß bei drei Magnetplatten und somit dreifacher Last (sowohl D- als auch K-Last) die Antwortzeit nur geringfügig über der Antwortzeit liegt, die mit einer Platte erreicht wird. Dieses Ergebnis ist auf die Parallelisierung innerhalb des Daten-Servers in Kombination mit der ausreichenden Gesamtprozessorleistung zurückzuführen. Die Berechnungen des Scheduling-Algorithmus können von mehreren Prozessoren des Systems parallel ausgeführt werden. Deshalb wird die Transferrate des Disk-Arrays zum Engpaß und bestimmt so die mittlere Antwortzeit.

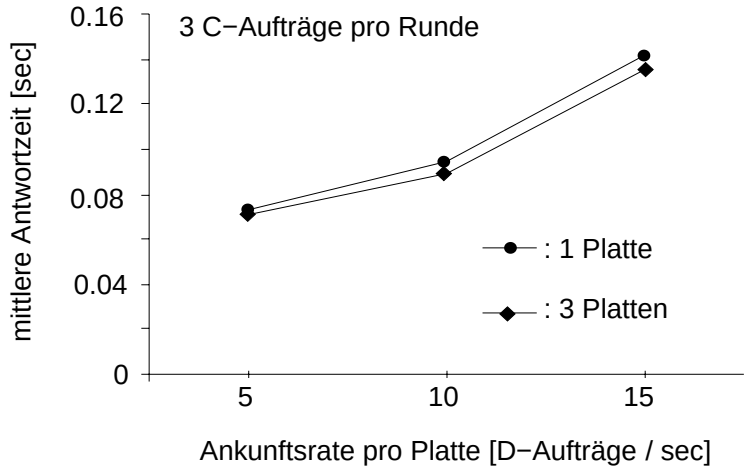



Abbildung 9.7: Meßergebnisse zur Überprüfung des Skalierungsverhaltens

In einer zweiten Messung wird die K- und D-Last variiert. Es ergeben sich ähnliche Ergebnisse (siehe Abbildung 9.8) wie bei den Messungen, die mit dem Simulationsmodell zur Evaluation der Scheduling-Algorithmen in Abschnitt 4.3 durchgeführt worden sind.

Insgesamt wurde mit der prototypischen Implementierung gezeigt, daß eine praktische Umsetzung der in Abschnitt 3.3 entwickelten Scheduling-Algorithmen auf eine reale Architektur möglich ist. Des weiteren zeigen die Meßergebnisse, daß die Implementierung des Daten-Servers relativ zu den verwendeten Hardwareressourcen ein sehr gutes skalierbares Performance-Verhalten besitzt.

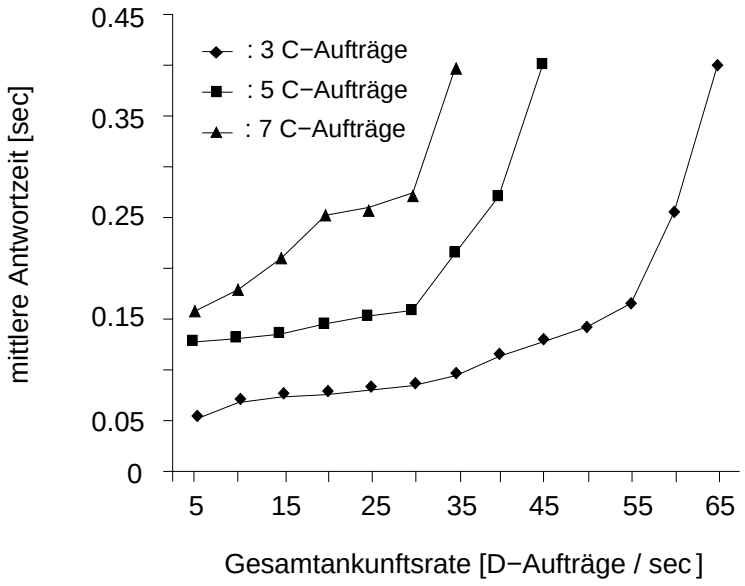



Abbildung 9.8: Meßergebnisse für verschiedene K- und D-Lasten

# Teil V

# Ausblick und Zusammenfassung

# Kapitel 10

# Ausblick

In diesem Kapitel soll zum Abschluß ein Ausblick auf mögliche Erweiterungen gegeben werden, die in dieser Arbeit nicht berücksichtigt worden sind.

## 10.1 Erweiterungen für synchronisierte Präsentationen

Die in dieser Arbeit betrachteten Scheduling-Algorithmen und die Zulassungkontrolle berücksichtigen keine Abhängigkeiten, die ggf. zwischen der Ausführung mehrerer Aufträge bestehen. Es wird angenommen, daß die Auswahl der Datenobjekte durch den Benutzer zufällig geschieht.

Dies ist nicht der Fall, wenn im Falle einer synchronisierten Präsentation multimedialer Inhalte, eine Menge verschiedener kontinuierlicher und diskreter Datenobjekte in festgelegter zeitlicher Reihenfolge ohne Einwirkung des Benutzers vom Daten-Server angefordert wird. Hierbei ergeben sich neue Probleme für die Zulassungkontrolle und das Scheduling der Plattenzugriffe.

Der Daten-Server muß bei zeitlich synchronisierten Präsentationen, wie sie z.B. mit der Beschreibungssprache SMIL (Synchronized Multimedia Integration Language) [Con98] festgelegt werden können, den Zeitpunkt festlegen, wann sie frühestmöglich gestartet werden können. Mit der Bereitstellung des ersten Objektes durch den Daten-Server sind nämlich die zeitlichen Fristen für die nachfolgenden Datenobjekte festgelegt. Dies entspricht den zeitlichen Abhängigkeiten, die unter den Fragmenten eines kontinuierlichen Datenobjektes bestehen, nur daß in diesem Fall ebenfalls diskrete Datenobjekte betroffen sind und daß keine Periodizität in den Fristen zu erwarten ist. Die Zulassungskontrolle hat somit zu entscheiden, wann der Ablauf der Präsentation gestartet wird und wann die einzelnen diskreten Datenobjekte im Server-Puffer bereitgestellt werden müssen. Dabei ist die Größe des Client-Puffers, der die diskreten Datenobjekte bis zur Präsentation zwischenspeichern muß, zu berücksichtigen. So sollte z.B. vermieden werden, daß ein diskretes Datenobjekt zu früh an den Client geschickt wird und dabei knappe Puffer-Ressourcen belegt. Hierbei könnte es nämlich bei unzureichender Planung passieren, daß ein Datenobjekt, das früher präsentiert, aber später vom Daten-Server bereitgestellt wird, aufgrund der begrenzten Kapazität des Client-Puffers nicht mehr zwischengespeichert werden kann. Für das Scheduling der D-Aufträge bedeutet dies, daß die Zulassungkontrolle für jeden D-Auftrag ein Intervall definieren sollte, in dem dessen Ausführung und damit die Bereitstellung der Daten erfolgen muß. Dieses Ausführungsintervall muß vom Scheduling-Algorithmus

bei der Anordnung der Aufträge berücksichtigt werden. Die Problemstellung, die sich hier bei gleichzeitiger Optimierung des Anordnungsreihenfolge (SCAN-Algorithmus) ergibt, läßt sich auf ein TSP mit zusätzlichen Zeitbedingungen (traveling salesman problem with time windows) [DDSS95] abbilden. Es müßte untersucht werden, inwieweit und unter welchen Bedingungen eine praktische Umsetzung aufgrund der großen Berechnungskomplexität möglich ist.

Da der Präsentationsablauf in vielen Fällen bereits im voraus feststeht, bietet sich eine weitere einfachere Alternative an, wie mit synchronisierten Präsentationen verfahren werden kann. Hierbei werden alle Datenobjekte der Präsentation in ein einziges kontinuierliches Datenobjekt kodiert. Die Kodierung muß dabei die Größe des Client-Puffers und die Präsentationsreihenfolge und -zeitpunkte berücksichtigen. Ist die Verteilung der Fragmentgrößen dieser zusammengesetzten kontinuierlichen Datenobjekte bekannt, so lassen sich die in dieser Arbeit vorgestellten Verfahren zur Zulassungkontrolle und zur Performance-Verhersage anwenden.

## 10.2 Gleichzeitiger Einsatz stochastischer und deterministischer Service-Garantien

In dieser Arbeit ist die gleichzeitige Verwendung von stochastischen und deterministischen Service-Garantien nicht betrachtet worden. Es ist aber denkbar, daß für bestimmte Anwendungen bzw. einen bestimmten Benutzerkreis deterministische Service-Garantien abgegeben werden und gleichzeitig anderen Anwendungen bzw. Anwendern eine stochastische Performance garantiert wird. Dies setzt natürlich voraus, daß die Scheduling-Algorithmen und die analytischen Modelle entsprechend angepaßt werden.

Als ersten Ansatz könnte man beim Scheduling eine Subrundenstrategie mit zwei Subrunden pro Runde verfolgen. In der ersten Subrunde erfolgt die ausschließliche Bedienung von kontinuierlichen Datenströmen mit deterministischer Service-Garantie, damit die Ausführung der zugehörigen K-Aufträge auf jeden Fall sichergestellt werden kann. Die Berechnung, wie viele K-Aufträge in der ersten Subrunde zugelassen werden können, erfolgt auf Basis eines deterministischen, analytischen Modells. Hierbei muß festgelegt werden, wie groß der Anteil der ersten Subrunde an der Rundendauer maximal sein darf. In der zweiten Subrunde könnte dann die Ausführung der übrigen K-Aufträgen und D-Aufträge gemäß dem vorgestellten Scheduling-Algorithmus V erfolgen. Die Zulassungkontrolle beruht hierbei auf dem in dieser Arbeit vorgestellten stochastischen Modell, das bei den Berechnungen sowohl die K-Aufträge der ersten Subrunde als auch die K-Aufträge der zweiten Subrunde in die Berechnungen einbezieht.

Ähnliches ließe sich auch auf der Basis einer gemischten Ausführung der Aufträge realisieren. Hierbei erstellt der Scheduling-Algorithmus unter Berücksichtigung der auftretenden Verzögerungen den Ausführungplan so, daß die Ausführung der K-Aufträge mit deterministischer Garantie auf jeden Fall sichergestellt wird. Als nächstes werden dann die K-Aufträge mit stochastischer Garantie berücksichtigt. Die geringste Priorität haben hierbei die D-Aufträge. Das in dieser Arbeit entwickelte analytische Modell läßt sich mit nur geringfügigen Veränderungen auch in diesem Fall für die Zulassungkontrolle verwenden.

Darüber hinaus besteht zusätzlich die Möglichkeit, durch eine Klasseneinteilung verschiedene Stufen der stochastischen Service-Qualität anzubieten. Hierdurch könnte man Anwendungen

und Benutzer berücksichtigen, die einerseits eine relativ hohe oder andererseits nur eine relativ niedrige Störungsrate bzw. Antwortzeit tolerieren.

## 10.3 Weitere analytische Betrachtungen

### 10.3.1 Ein stochastisches Modell für die Startverzögerung

Neben der Störungsrate und der Antwortzeit existieren weitere Metriken, die die Service-Qualität bestimmen. Eine dieser Metriken für kontinuierliche Datenobjekte ist die Startverzögerung, d.h. die Zeit, die vergeht, bis nach der Zulassung eines kontinuierlichen Datenstromes das erste Fragment im Puffer des Daten-Servers zur Verfügung gestellt wird. In Abschnitt 3.1.3 dieser Arbeit ist nur die maximal mögliche Startverzögerung betrachtet worden, die beim grobkörnigem Striping über mehrere Platten im "worst case" bei maximaler Auslastung auftreten kann. Die tatsächliche Startverzögerung ist aber generell kleiner, wenn der Server unterhalb der maximalen Auslastung $N_{max} * P_{disks}$ betrieben wird. In diesem Fall erhöht sich nämlich die Wahrscheinlichkeit, daß die Platte, von der das erste Fragment gelesen werden muß, den zugehörigen Auftrag in der Folgerunde ausführen kann, da sie weniger als $N_{max}$ K-Aufträge bearbeitet.

Erste Ansätze, die Startverzögerung durch ein stochastisches Modell zu charakterisieren, sind in [GK95a, GKSZ96] zu finden. Das vollständige Modell müßte in der Lage sein, ähnlich wie bei der Antwortzeit, die Restwahrscheinlichkeit abzuschätzen, daß die Startverzögerung oberhalb eines tolerierbaren Wertes liegt, also z.B. $P[\text{Startverzögerung} > 2T] \leq 1\%$, wenn insgesamt weniger als $N_{max} * P_{disks}$ Datenströme von der Zulassungkontrolle akzeptiert werden.

### 10.3.2 Ein stochastisches Modell für die Wartezeit

Auf Grundlage des in dieser Arbeit vorgestellten Modells für die Störungsrate läßt sich relativ einfach ein grobes Modell zur Berechnung der Wartezeit ableiten. Hierzu wird ein Warteschlangenmodell ($M/M/(N_{max} * P_{disks})$) verwendet, wie es in der Literatur [Jai91] zu finden ist. Der Daten-Server wird hierbei als abstrakter Server mit $N_{max} * P_{disks}$ Bedienstationen aufgefaßt, der somit parallel $N_{max} * P_{disks}$ Datenströme verarbeiten kann. Ein Datenstrom entspricht im Warteschlangenmodell einem Auftrag, dessen Bedienzeit durch die Präsentationsdauer des zugehörigen kontinuierlichen Datenobjektes festgelegt ist. Das Warteschlangenmodell nimmt hierbei vereinfachend eine Exponentialverteilung für die Bedienzeit an. Inwieweit dieses der Realität entspricht, müßte näher untersucht werden und ggf. ein Modell für eine allgemein verteilte Bedienzeit entwickelt werden. Die Modellierung des Benutzers, der den Auftrag zum Start eines kontinuierlichen Datenstromes gibt, wird in diesem offenen System durch einen Poisson-Prozeß modelliert.

### 10.3.3 Ein stochastisches Modell für den Speicherbedarf

Eine weitere Maßnahme zur effizienten Auslastung der Systemressourcen ist eine effiziente Benutzung des zur Verfügung stehenden Server-Puffers. Der in dieser Arbeit in Abschnitt 3.1.3

angegebene Bedarf stellt nur eine obere Schranke des tatsächlichen Bedarfs dar. Wird angenommen, daß der Transfer der Daten eines Fragments über das Netzwerk an den Client gleichmäßig über die gesamte Runde verteilt erfolgt und nicht plötzlich am Ende einer Runde, so kann in vielen Fällen der frei werdende Speicher eines Fragments bereits vor Ablauf einer Runde wieder zur Zwischenspeicherung von Daten genutzt werden. Der Pufferbedarf hängt dabei von der Transferrate der Platte, von der Transferrate des Netzwerks und der Fragmentgröße ab. Betrachtet man alle drei Größen als Zufallsvariablen, so kann analog zu dem Modell für die Störungsrate ein stochastisches Modell für den benötigten Pufferbedarf entwickelt werden. Auch hier liefert die vorliegende Arbeit deshalb den prinzipiellen Ansatz zur Lösung.

# Kapitel 11

# Zusammenfassung

Durch den verstärkten Einsatz von multimedialen Daten in neuartigen Anwendungen, wie z.B. Tele-Shopping oder News-On-Demand, und durch die Entwicklung des Internets zum Massenmedium, ist eine kostengünstige Speicherung und Bereitstellung der multimedialen Inhalte notwendig. Hierdurch muß besonders bei Konzeption und Einsatz multimedialer Daten-Server, die die Speicherung und Bereitstellung der Daten übernehmen, auf ein gutes Preis-Leistungs-Verhältnis geachtet werden. Aufgrund der hohen Performance-Anforderungen von Seiten der Anwender und Anwendungen stellt vor allem das Speichersystem einen großen Kostenfaktor dar. In dieser Arbeit werden Verfahren vorgestellt, die eine hohe Auslastung des Speichersystems zulassen und somit einen Beitrag zu einer kostengünstigen Realisierung eines Daten-Servers leisten.

Ausgangspunkt der Arbeit bildet die *gemischte* Speicherung kontinuierlicher Daten (Audio, Video) und diskreter Daten (Text, Bild, Graphik) auf einem Disk-Array des Daten-Servers. Durch diese gemischte Datenhaltung kann eine gleichmäßige Auslastung besser gewährleistet werden als bei strikter Ressourcentrennung, da sich zeitliche Schwankungen in den Zugriffen auf beide Datentypen z.T. automatisch ausgleichen können. Bedingt durch diese Form der gemischten Datenhaltung muß jede Magnetplatte des Disk-Arrays auch eine gemischte Last, d.h. Zugriffe auf kontinuierliche *und* diskrete Daten, verarbeiten können. Hierfür untersucht die Arbeit verschiedene Scheduling-Algorithmen, die den anwendungsorientierten Performance-Anforderungen beider Datentypen hinsichtlich der Service-Qualität Rechnung tragen. Für kontinuierliche Daten wird als Metrik der Service-Qualität die Störungsrate verwendet, d.h. die Häufigkeit, mit der die Lieferung des kontinuierlichen Datenstroms temporär unterbrochen werden muß. Für diskrete Daten wird als Metrik der Service-Qualität die Antwortzeit festgelegt, während der auf die Bereitstellung der diskreten Daten gewartet werden muß. Aufbauend auf bereits bekannten Verfahren der Datenplazierung und des Schedulings kontinuierlicher Daten gibt es verschiedene Möglichkeiten, das Scheduling gemischter Arbeitslasten durchzuführen. Diese Möglichkeiten werden anhand einer Taxonomie in Form verschiedener Teilstrategien beschrieben. Die Zusammenstellung verschiedener Teilstrategien erlaubt es dann, verschiedene erfolgversprechende Scheduling-Algorithmen abzuleiten. Durch eine experimentelle Evaluation dieser Algorithmen mit Hilfe von Simulationsmodellen wird das Verfahren ermittelt, das die geringste mittlere Antwortzeit für Zugriffe auf diskrete Daten aufweist und dabei keine negativen Auswirkungen auf die Störungsrate kontinuierlicher Datenzugriffe zeigt.

Daß neben dem Simulationsmodell auch eine Realisierung und praktische Implementierung der vorgestellten Scheduling-Algorithmen unter Verwendung realer Hardware möglich ist, zeigt

die Arbeit, indem sie einen Überblick über das Prototypsystem gibt, in dem ein Großteil der entwickelten Ideen und Konzepte praktisch umgesetzt worden ist.

Die Auslastung des Daten-Servers im Betrieb wird zum großen Teil auch dadurch bestimmt, wie viele kontinuierliche Datenströme von der Zulassungkontrolle unter Einhaltung der geforderten Service-Qualität gleichzeitig zugelassen werden können. Diese Anzahl läßt sich, wie in der Arbeit gezeigt wird, beträchtlich steigern, wenn statt deterministischer Service-Garantien den einzelnen Datenströmen eine stochastische Garantie gewährt wird. Für die Zulassungskontrolle, die die Einhaltung der Service-Garantien überwacht, ist es deshalb notwendig, die stochastische Performance des Servers durch ein analytisches Modell vorhersagen zu können.

Für dieses analytische Modell wird zunächst eine detaillierte stochastische Modellierung des Plattenzugriffs durchgeführt, bei der neben Ein-Zonen-Platten auch Mehr-Zonen-Platten mit variabler Transferrate betrachtet werden. Für letzteres wird die Dichtefunktion der Transferratenverteilung hergeleitet. Weiterhin wird die Dichtefunktion für die Positionierungszeit sowohl für FCFS-Algorithmen als auch für SCAN-Algorithmen analysiert. Unter Berücksichtigung dieser Ergebnisse und zusätzlicher variabler Auftragsgrößen läßt sich eine obere Schranke der Wahrscheinlichkeit angeben, mit der die Gesamtdauer zur Ausführung einer Menge von Plattenzugriffen eine vorgegebene Zeitschranke überschreitet. Dieses Resultat kann schließlich dazu benutzt werden, eine obere Schranke für die Häufigkeit zu berechnen, mit der eine festgelegte Störungsrate überschritten wird. Es werden unterschiedliche Berechnungsverfahren zur Abschätzung der Wahrscheinlichkeiten vorgestellt, die auf einer numerischen Faltung der Dichtefunktionen bzw. auf einer Abschätzung der Laplace-Transformierten durch die Chernoff-Schranke beruhen. Die unterschiedlichen Verfahren werden in einer Evaluation hinsichtlich ihres Berechnungsaufwandes und der dabei erzielten Vorhersagegenauigkeit untersucht. Als Vergleichsmaßstab für die Vorhersagegenauigkeit werden die Meßergebnisse aus der Simulation der im ersten Teil vorgestellten Scheduling-Algorithmen herangezogen.

Die Berechnung der stochastischen Service-Qualität für diskrete Daten geschieht auf Basis eines Warteschlangenmodells, das eine obere Schranke für die Wahrscheinlichkeit berechnet, daß eine festgelegte Antwortzeit überschritten wird. Auch hier wird eine stochastische Verteilung der Auftragsgrößen vorausgesetzt, und die Häufigkeit der Zugriffe auf diskrete Daten durch einen stochastischen Poisson-Ankunftsprozeß berücksichtigt. Die abschließende Evaluation untersucht die Vorhersageergebnisse des Modells und vergleicht sie mit Simulationsresultaten, die für verschiedene Scheduling-Algorithmen ermittelt worden sind.

Neben dem Einsatz für die Zulassungkontrolle bildet das stochastische Verhaltensmodell die Grundlage zur Konfiguration des Speichersystems. Es wird eine Konfigurationsmethode vorgestellt, die unter gegebenen Systemparametern, einer gegebenen Last und festgelegten Service-Garantien für beide Datentypen die minimale Anzahl benötigter Magnetplatten berechnet.

# Kapitel 12

# Summary

Due to the increasing usage of multimedia data in new applications like news-on-demand or tele-teaching and the growth of the internet, the need arises to store and retrieve huge amounts of multimedia data in a very efficient way. Obtaining good cost/performance ratios poses a great challenge in the design and implementation of a multimedia data server, whose storage system is one of the biggest expense factors in building a multimedia information system. This work contributes to this goal by providing methods to use storage resources, i.e. disk-arrays, efficiently with very high server utilization.

A starting point for a good cost/performance ratio is the usage of a shared resource pool. With respect to the storage system this means that continuous data (audio, video) and discrete data (text, picture, graphics) is laid out on the disks in a mixed way. This makes the system rather insensitive against temporal load imbalances and allows a more even utilization. The latter cannot be guaranteed with dedicated resource allocations for each of both data types. As a result of this mixed allocation of multimedia data on the server's disk-array, each disk has to cope with a mixed workload consisting of accesses to both continuous *and* discrete data.

This thesis examines different disk scheduling algorithms that are handling this kind of workload considering quality-of-service requirements of both data types. For continuous data the quality-of-service is expressed in terms of the glitch rate, that is the frequency of temporarily suspending the delivery of the continuous data stream. The metric for discrete data is given by the response time of data requests. This thesis examines known techniques for data placement and scheduling of continuous data and identifies policies how these techniques can be extended for mixed workload scheduling. A framework describes all policies in terms of few parameters and allows deriving the most promising scheduling algorithm for the given performance requirements. Using a simulation model it is experimentally shown that a newly proposed scheduling algorithm outperforms all other algorithms. It provides the lowest mean response time of accesses to discrete data, and has no negative influence on the glitch rate of continuous data streams.

A prototype system demonstrates the practical feasibility of implementing and using all developed algorithms.

The number of data streams that are allowed by the admission control mainly determines the utilization of the data server during its operation mode. This number of data streams can be significantly increased when quality-of-service is guaranteed stochastically rather than deterministically. This thesis develops a stochastic performance model that predicts the quality-of-

service that can be sustained for both data types assuming a given number of disks and a given workload.

In this thesis disk accesses are modeled stochastically for both single-zone and multi-zone disks. The latter have a variable transfer rate, so the density function for the transfer rate distribution of these disks must be considered. Furthermore the time for positioning the disk head is calculated stochastically for FCFS and for SCAN scheduling algorithms. Both results can be used to derive the time that is needed to perform a given number of disk accesses retrieving a certain amount of data. It is shown how the tail probability for the service time of these disk accesses exceeding a given limit can be computed by a numerical convolution of the density functions or by using Chernoff's theorem and Laplace transforms of the density functions. Based on this value the tail probability of the glitch rate can be bounded. Several methods for calculating this bound are presented, and calculation overhead and accuracy of the methods are studied. The accuracy of the predictions is compared to the results of using a simulation model.

The quality-of-service for discrete data is predicted using a queueing model which assumes a Poisson arrival process of discrete data accesses. This model derives the Laplace transform of the response time distribution and bounds its tail by Chernoff's theorem. The final evaluation compares the prediction of the analytic model with simulation results.

The performance predictions are used also for configuring a server by computing a configuration with a minimum of required resources while meeting specific quality-of-service requirements for a given workload.

# Anhang

# Anhang A

# Glossar

In diesem Abschnitt sind einige zentrale Begriffe aufgeführt, die in der Arbeit wiederholt verwendet werden.

**Abblockwahrscheinlichkeit** Die Wahrscheinlichkeit, daß ein D-Auftrag im analytischen Modell aufgrund einer vollständig gefüllten Warteschlangen abgewiesen werden muß.

**Abgangsrunde** Die Runde, in die der Abgangszeitpunkt eines D-Auftrages fällt.

**Abgangszeitpunkt** Zeitpunkt, an dem die Ausführung eines D-Auftrages beendet wird und die gelesenen Daten im Server-Puffer zur Verfügung stehen.

**Ankunftsrunde** Die Runde, in die der Ankunftszeitpunkt eines D-Auftrages fällt.

**Ankunftszeitpunkt** Zeitpunkt, an dem ein D-Auftrag in die FCFS-Warteschlange des Scheduling-Algorithmus eingefügt wird.

**Antwortzeit** Zeitdauer zwischen Ankunfts- und Abgangszeitpunkt.

**Audio/Video-Streaming** Methode, die Audio/Videodaten in einem kontinuierlichen Datenstrom vom Daten-Server zum Client versendet.

**Auftrag** siehe K-Auftrag bzw. D-Auftrag

**Ausführungsplan** Reihenfolge, in der der Scheduling-Algorithmus die Ausführung von D- und K-Aufträgen durchführt.

**Auslastung** Anteil der Zeit, in der der Daten-Server mit der Ausführung von Aufträgen beschäftigt ist. Beim periodischen rundenbasierten Scheduling ist es auf eine Runde bezogen der Quotient aus Gesamtbedienzeit und Rundendauer, d.h. $T_{svc}/T$.

**Bedienrate** Die Bedienrate der Platte ist der Quotient aus Datenmenge, die beim Plattenzugriff gelesen oder geschrieben wird, und aus der hierfür benötigten Zeit.

**Bedienzeit** Zeitdauer für einen Plattenzugriff, d.h. Zeitdauer zur Ausführung eines Auftrages. Die Bedienzeit setzt sich zusammen aus der Positionierungszeit, der Rotationsverzögerung und der Transferzeit.

**D-Auftrag** Auftrag zum Lesen oder Schreiben eines diskreten Datenobjektes.

**D-Last** Anzahl der D-Aufträge, die pro Zeiteinheit eintreffen und ausgeführt werden sollen. Die D-Last wird in Simulation und Analyse durch einen stochastischen Poisson-Ankunftsprozeß beschrieben.

**Data-Striping** Verteilung eines Datenobjektes auf mehrere Platten eines Disk-Arrays.

**Datenstrom** kontinuierliche Folge von Fragmenten eines kontinuierlichen Datenobjektes.

**Datenobjekt** Logische Einheit, in der der Daten-Server Audio/Videodaten (kontinuierliche Datenobjekte) sowie Texte, Bilder, Graphiken (diskrete Daten) speichert.

**Datenrate** Geschwindigkeit, mit der die Daten eines kontinuierlichen Datenobjektes wiedergegeben werden, d.h. die Geschwindigkeit mit der der Client auf die Daten eines kontinuierlichen Datenobjektes zugreift (siehe auch KDR-/VDR-Datenobjekte).

**Disk-Striping** siehe Data-Striping.

**Durchsatz** Anzahl der pro Zeiteinheit ausgeführten Aufträge. Der maximal mögliche Durchsatz ist die Leistungskapazität des Daten-Servers.

**Ein-Zonen-Platten** Magnetplatten mit einer konstanten Anzahl Sektoren pro Spur. Die Transferrate von Ein-Zonen-Platten ist konstant.

**Fahrstuhlalgorithmus** siehe SCAN-Algorithmus.

**FCFS-Algorithmus** Aufträge werden in der Ankunftsreihenfolge bearbeitet.

**Fragment** Logisch zusammenhängender Bereich eines kontinuierlichen Datenobjektes. Kleinste Einheit, auf die der Daten-Server innerhalb des kontinuierlichen Datenobjektes zugreifen kann.

**Gesamtbedienzeit** Summe der Bedienzeiten aller Aufträge, die innerhalb einer Runde ausgeführt werden.

**Gesamtpositionierungszeit** Summe der Positionierungszeiten aller Aufträge, die innerhalb einer Runde ausgeführt werden.

**Gesamtrotationsverzögerung** Summe der Rotationsverzögerungen aller Aufträge, die innerhalb einer Runde ausgeführt werden.

**Gesamttransferzeit** Summe der Transferzeiten aller Aufträge, die innerhalb einer Runde ausgeführt werden.

**K-Auftrag** Auftrag zum Lesen oder Schreiben eines Fragments.

**K-Last** Anzahl der K-Aufträge, die pro Zeiteinheit eintreffen und ausgeführt werden sollen. Für das periodische rundenbasierte Scheduling ist die K-Last die Anzahl der K-Aufträge, die pro Runde ausgeführt werden sollen, gleichzusetzen mit der Anzahl der Datenströme, die der Server an die Client liefert.

**KDL-Partitionierung** Zerlegung eines kontinuierlichen Datenobjektes in Fragmente mit konstanter Datenlänge, d.h., alle Fragmente beinhalten die gleiche Datenmenge in Bytes. Siehe auch KWL-Partitionierung.

**KDR-Datenobjekt** kontinuierliches Datenobjekt mit konstanter Datenrate. Siehe auch VDR-Datenobjekt.

**KWL-Partitionierung** Zerlegung eines kontinuierlichen Datenobjektes in Fragmente mit konstanter Wiedergabelänge, d.h., die Wiedergabedauer der in den Fragmenten gespeicherten Informationen hat einen konstanten Wert. Siehe auch KDL-Partitionierung.

**Last** siehe D-Last und K-Last.

**Leistungskapazität** Maximaler Durchsatz an Aufträgen, der pro Zeiteinheit ausgeführt werden kann.

**Mehr-Zonen-Platten** Magnetplatten mit einer variablen Anzahl Sektoren pro Spur. Spuren mit gleicher Sektoranzahl werden zu benachbarten Zonen zusammengeschlossen. Die Transferrate ist variabel.

**Partitionierung** Zerlegung eines Datenobjektes in mehrere Teilobjekte, z.B. Segmente oder Fragmente. Siehe auch KWL-/KDR-Partitionierung.

**Performance-Anforderungen** Anforderungen an die Leistungsfähigkeit des Daten-Servers. Eine systemorientierte Performance-Anforderungen ist die Maximierung des Durchsatzes. Anwendungsorientierte Performance-Anforderungen betreffen die Antwortzeit von D-Aufträgen sowie die Störungsrate bei kontinuierlichen Datenströmen (siehe auch Service-Qualität).

**Performance-Garantie** siehe Service-Garantie

**Positionierungsdistanz** Anzahl der Zylinder, die der Plattenarm zwischen zwei aufeinanderfolgenden Plattenzugriffen zu überqueren hat.

**Positionierungszeit** Zeitdauer, die der Plattenarm zum Überqueren einer bestimmten Positionierungsdistanz benötigt.

**Restwahrscheinlichkeit** Wahrscheinlichkeit, daß Werte der Zufallsvariablen $X$ oberhalb einer festgelegten Schranke $\epsilon_X$ liegen, d.h. $P[X > \epsilon_X]$.

**Rotationsverzögerung** Zeitdauer nach der Positionierung des Plattenarmes bis der gewünschte Block sich unter dem Schreib-Lese-Kopf befindet.

**Runde** Zeitabschnitt konstanter Länge, in der die Ausführung von D- und K-Aufträgen stattfindet.

**SCAN-Algorithmus** Aufträge werden entsprechend ihrer Zylinderposition sortiert und in dieser Anordnung bearbeitet. Der Plattenarm bewegt sich hierbei nur in einer Richtung d.h. entweder von außen nach innen oder umgekehrt.

**Scheduling** Koordinierung der Plattenzugriffe. Aufstellen von Ausführungsplänen, die die Reihenfolge der Bearbeitung von D- und K-Aufträgen bestimmen.

**Segmente** Logisch zusammenhängender Bereich eines diskreten Datenobjektes. Siehe auch Partitionierung

**Service-Garantie** Zusicherung des Servers an den Client, daß eine vereinbarte anwendungsorientierte Performance eingehalten wird. Deterministische Service-Garantien garantieren zu $100\%$, daß eine festgelegte Antwortzeit bzw. eine Störungsrate von $0\%$ eingehalten wird. Stochastische Service-Garantien beschränken hingegen die Restwahrscheinlichkeit, daß die Antwortzeit bzw. die Störungsrate über einer festgelegten Schranke liegt.

**Service-Qualität** Metrik, die bestimmt, in welchem Maße die Performance-Anforderungen eingehalten werden. Gleichzusetzen mit Service-Garantie.

**Störung** Wird ein Fragment nicht fristgerecht im Server-Puffer zur Verfügung gestellt, entsteht bei der Wiedergabe der Daten eine Störung.

**Störungsrate** Häufigkeit, mit der eine Störung während der Präsentation eines kontinuierlichen Datenobjektes auftritt.

**Startverzögerung** Zeitraum zwischen der Zulassung eines Datenstromes und der Bereitstellung des ersten Fragments im Server-Puffer.

**Striping** Siehe Data-Striping.

**Transferrate** Geschwindigkeit, mit der Daten von der Platte gelesen werden können. Die Transferrate ist abhängig von der Rotationsgeschwindigkeit der Platte und der Anzahl der Blöcke auf einer Spur.

**Transferzeit** Zeitdauer für das ausschließliche Lesen bzw. Schreiben der Daten beim Plattenzugriff.

**VDR-Datenobjekt** kontinuierliches Datenobjekt mit variabler Datenrate. Siehe auch KDR-Datenobjekt.

**Wartezeit** Die Wartezeit eines D-Auftrages ist der Zeitraum zwischen dessen Ankunftszeitpunkt und dem Zeitpunkt, an dem seine Ausführung beginnt. Die Wartezeit eines kontinuierlichen Datenstromes beschreibt die Zeitdauer, mit der die Zulassungkontrolle die Zulassung dieses Datenstromes verzögert.

**Zulassungskontrolle** Die Zulassungskontrolle bestimmt die Anzahl der kontinuierlichen Datenströme, die parallel durch den Daten-Server bedient werden. Beim periodischen rundenbasierten Scheduling legt sie die Anzahl der K-Aufträge fest, die pro Runde und Platte ausgeführt werden.

**Zwischenrunde** Eine Runde, die zwischen der Ankunfts- und der Abgangsrunde eines D-Auftrages liegt.

# Anhang B

# Symbolverzeichnis

Dieser Abschnitt enthält eine sortierte Liste aller verwendeten Bezeichner und ermöglicht so einen Quereinstieg in die einzelnen Kapitel.

| | |
|---|---|
| $b_{erate}^{BNI,NC}(N,e)$ | obere Schranke für die Wahrscheinlichkeit $p_{erate}(N,e)$ bei Verwendung von $b_{error}^{BNI,NC}$ |
| $b_{erate}^{BNI,LTI}(N,e)$ | obere Schranke für die Wahrscheinlichkeit $p_{erate}(N,e)$ bei Verwendung von $b_{error}^{BNI,LTI}$ |
| $b_{erate}^{BNI,LTII}(N,e)$ | obere Schranke für die Wahrscheinlichkeit $p_{erate}(N,e)$ bei Verwendung von $b_{error}^{BNI,LTII}$ |
| $b_{erate}^{BNII,NC}(N,e)$ | obere Schranke für die Wahrscheinlichkeit $p_{erate}(N,e)$ bei Verwendung von $b_{error}^{BNII,NC}$ |
| $b_{erate}^{BNII,LTI}(N,e)$ | obere Schranke für die Wahrscheinlichkeit $p_{erate}(N,e)$ bei Verwendung von $b_{error}^{BNII,LTI}$ |
| $b_{erate}^{BNII,LTII}(N,e)$ | obere Schranke für die Wahrscheinlichkeit $p_{erate}(N,e)$ bei Verwendung von $b_{error}^{BNII,LTII}$ |
| $b_{error}^{BNI,NC}(N,C,g)$ | obere Schranke für die Wahrscheinlichkeit $p_{error}(N,C,g)$ bei Verwendung von $b_{glitch}^{NC}$ und numerischer Berechnung der Binomialverteilung |
| $b_{error}^{BNI,LTI}(N,C,g)$ | obere Schranke für die Wahrscheinlichkeit $p_{error}(N,C,g)$ bei Verwendung von $b_{glitch}^{LTI}$ und numerischer Berechnung der Binomialverteilung |
| $b_{error}^{BNI,LTII}(N,C,g)$ | obere Schranke für die Wahrscheinlichkeit $p_{error}(N,C,g)$ bei Verwendung von $b_{glitch}^{LTII}$ und numerischer Berechnung der Binomialverteilung |
| $b_{error}^{BNII,NC}(N,C,g)$ | obere Schranke für die Wahrscheinlichkeit $p_{error}(N,C,g)$ bei Verwendung von $b_{glitch}^{NC}$ und Abschätzung der Binomialverteilung durch die Chernoff-Schranke |
| $b_{error}^{BNII,LTI}(N,C,g)$ | obere Schranke für die Wahrscheinlichkeit $p_{error}(N,C,g)$ bei Verwendung von $b_{glitch}^{LTI}$ und Abschätzung der Binomialverteilung durch die Chernoff-Schranke |

| | |
|---|---|
| $b_{error}^{BNII,LTII}(N,C,g)$ | obere Schranke für die Wahrscheinlichkeit $p_{error}(N,C,g)$ bei Verwendung von $b_{glitch}^{LTII}$ und Abschätzung der Binomialverteilung durch die Chernoff-Schranke |
| $b_{glitch}^{NC}(N)$ | obere Schranke für die Wahrscheinlichkeit $p_{glitch}(N)$ bei Verwendung von $b_{late}^{NC}$ |
| $b_{glitch}^{LTI}(N)$ | obere Schranke für die Wahrscheinlichkeit $p_{glitch}(N)$ bei Verwendung von $b_{late}^{LTI}$ |
| $b_{glitch}^{LTII}(N)$ | obere Schranke für die Wahrscheinlichkeit $p_{glitch}(N)$ bei Verwendung von $b_{late}^{LTII}$ |
| $b_{late}^{NC}(N)$ | obere Schranke für die Restwahrscheinlichkeit $p_{late}(N)$ bei Anwendung numerischer Faltungen |
| $b_{late}^{LTI}(N)$ | obere Schranke für die Restwahrscheinlichkeit $p_{late}(N)$ bei Verwendung der Laplace-Transformierten $T_{svc}^{LTI*}$ |
| $b_{late}^{LTII}(N)$ | obere Schranke für die Restwahrscheinlichkeit $p_{late}(N)$ bei Verwendung der Laplace-Transformierten $T_{svc}^{LTII*}$ |
| $f_D$ | allgemeine Dichtefunktion für die Verteilung der Positionierungsdistanz |
| $f_D^{fcfs}$ | allgemeine Dichtefunktion für die Verteilung der Positionierungsdistanz bei FCFS-Anordnung der Aufträge |
| $f_{D,SZ}^{fcfs}$ | Dichtefunktion für die Verteilung der Positionierungsdistanz bei FCFS-Anordnung der Aufträge und Verwendung einer Ein-Zonen-Platte |
| $f_{D,MZ}^{fcfs}$ | Dichtefunktion für die Verteilung der Positionierungsdistanz bei FCFS-Anordnung der Aufträge und Verwendung einer Mehr-Zonen-Platte |
| $f_{D,SZ}^{scan_1}(n,d)$ | Dichtefunktion für die Verteilung der Positionierungsdistanz bei SCAN-Anordnung mit $n$ Aufträgen zwischen Zylinder 1 und Position des ersten Auftrages, betrachtet wird eine Ein-Zonen-Platte |
| $f_{D,SZ}^{scan_X}(n,d)$ | Dichtefunktion für die Verteilung der Positionierungsdistanz bei SCAN-Anordnung mit $n$ Aufträgen zwischen zwei beliebigen Aufträgen, betrachtet wird eine Ein-Zonen-Platte |
| $f_{S_K}$ | Dichtefunktion für die Verteilung der Größe von K-Aufträgen |
| $f_{S_D}$ | Dichtefunktion für Verteilung der Größe von D-Aufträgen |
| $f_{V,MZ}$ | Dichtefunktion für die Verteilung der Transferrate einer Mehr-Zonen-Platte |
| $f_Z$ | allgemeine Dichtefunktion für die Verteilung der Zylinderauswahl |
| $f_{Z,SZ}$ | Dichtefunktion für die Verteilung der Zylinderauswahl bei einer Ein-Zonen-Platte |

| | |
|---|---|
| $f_{Z,MZ}$ | Dichtefunktion für die Verteilung der Zylinderauswahl bei einer Mehr-Zonen-Platte |
| $f_{pos}$ | Dichtefunktion der Verteilung für die Positionierungszeit eines Auftrages |
| $f^{\approx}_{\sum pos}$ | approximierte Dichtefunktion der Verteilung für die Gesamtpositionierungszeit innerhalb einer Runde |
| $f_{rot}$ | Dichtefunktion der Verteilung für die Rotationsverzögerung eines Auftrages |
| $f^{\approx}_{\sum rot}$ | approximierte Dichtefunktion der Verteilung für die Gesamtrotationsverzögerungen innerhalb einer Runde |
| $f_{trans}$ | allgemeine Dichtefunktion der Verteilung für die Transferzeit |
| $f_{trans,SZ}$ | allgemeine Dichtefunktion der Verteilung für die Transferzeit einer Ein-Zonen-Platte |
| $f_{trans,MZ}$ | allgemeine Dichtefunktion der Verteilung für die Transferzeit einer Mehr-Zonen-Platte |
| $f^{S_K}_{trans,SZ}$ | Dichtefunktion der Verteilung für die Transferzeit einer Ein-Zonen-Platte bei gammaverteilter Größe der Aufträge |
| $f^{S_D}_{trans,SZ}$ | Dichtefunktion der Verteilung für die Transferzeit einer Ein-Zonen-Platte bei normalverteilter Größe der Aufträge |
| $f^{S_K}_{trans,MZ}$ | Dichtefunktion der Verteilung für die Transferzeit einer Mehr-Zonen-Platte bei gammaverteilter Größe der Aufträge |
| $f^{S_K\approx}_{trans,MZ}$ | Dichtefunktion einer Gammaverteilung, die $f^{S_K}_{trans,MZ}$ approximiert |
| $f^{S_D}_{trans,MZ}$ | Dichtefunktion der Verteilung für die Transferzeit einer Mehr-Zonen-Platte bei normalverteilter Größe der Aufträge |
| $h(\theta)$ | Infimum dieser Funktion ist die kleinste Abschätzung der Restwahrscheinlichkeit |
| $h_k$ | Wahrscheinlichkeit, daß in einer Runde $k$ D-Aufträge eintreffen |
| $m_i$ | Wahrscheinlichkeit, daß in einer Runde $i$ D-Aufträge ausgeführt werden können |
| $p_b$ | Abblockwahrscheinlichkeit, d.h. die Wahrscheinlichkeit, daß ein D-Auftrag im analytischen Modell aufgrund einer vollständig gefüllten Warteschlangen abgewiesen wird |
| $p_{erate}(N, e)$ | Restwahrscheinlichkeit, daß bei einem beliebigen Datenstrom bei $N$ K-Aufträgen die Störungsrate größer oder gleich $e$ ist |

| | |
|---|---|
| $p_{error}(N, C, g)$ | Restwahrscheinlichkeit, daß bei einem beliebigen Datenstrom bei $N$ K-Aufträgen mehr oder genau $g$ Störungen in einem Zeitraum von $C$ Runden auftreten |
| $p_{glitch}(N)$ | Wahrscheinlichkeit, daß bei einem beliebigen Datenstrom bei $N$ K-Aufträgen pro Runde eine Störung auftritt |
| $p_{late}(N)$ | Restwahrscheinlichkeit, mit der die Gesamtbedienzeit für $N$ K-Aufträgen die Rundendauer überschreitet |
| $q_k$ | Wahrscheinlichkeit, daß ein D-Auftrag in seiner Abgangsrunde mit $k$ D-Aufträgen ausgeführt wird |
| $t_{pos}(d)$ | Funktion zur Berechnung der Positionierungszeit bei einer Positionierungsdistanz von $d$ |
| $t_{pos}^{-1}(t)$ | inverse Funktion zu $t_{pos}$ |
| $t_{pos}^{max}(N)$ | Funktion zur Berechnung der oberen Schranke der Gesamtpositionierungszeit bei insgesamt $N$ Aufträgen im SCAN |
| $t_{trans}(s)$ | Transferzeit einer Ein-Zonen-Platte bei der Auftragsgröße $s$ |
| $t_{trans}^{i}(s)$ | Transferzeit in der $i$-ten Zone einer Mehr-Zonen-Platte bei der Auftragsgröße $s$ |
| $A$ | Zufallsvariable der Zeit, die ein D-Auftrag in der Ankunftsrunde verbringt |
| $A_l$ | Zufallsvariable der Verbundverteilung aus der Zeit, die ein D-Auftrag in der Ankunftsrunde verbringt, und der Anzahl der D-Aufträge, die zu seinem Ankunftszeitpunkt in der Warteschlange vorhanden sind |
| $A_l^*$ | Laplace-Transformierte der Zufallsvariablen $A_l$ |
| $B$ | Zufallsvariable der Zeit, die ein beliebiger D-Auftrag in der Abgangsrunde verbringt |
| $B_l$ | Zufallsvariable der Zeit, die ein D-Auftrag in der Abgangsrunde verbringt unter der Voraussetzung, daß zu seinem Ankunftszeitpunkt $l$ D-Aufträge in der Warteschlange vorhanden sind |
| $B^*$ | Laplace-Transformierte der Zufallsvariablen $B$ |
| $B_l^*$ | Laplace-Transformierte der Zufallsvariablen $B_l$ |
| $C$ | Wiedergabedauer eines kontinuierlichen Datenobjektes in Runden |
| $D$ | Zufallsvariable der Positionierungsdistanz |
| $F_D$ | Verteilungsfunktion der Positionierungsdistanz $D$ |

| | |
|---|---|
| $F_M^{(i)}$ | $i$-fach gefaltete Verteilungsfunktion der Zufallsvariablen $M$ |
| $F_{pos}$ | Verteilungsfunktion der Positionierungszeit $T_{pos}$ |
| $F_{svc}^{\approx}$ | Verteilungsfunktion der approximierten Gesamtbedienzeitverteilung |
| $F_{svc}^{K,D}(N,i)$ | Verteilungsfunktion der Gesamtbedienzeitverteilung bei $N$ K-Aufträgen und $i$ D-Aufträgen |
| $G$ | Gruppengröße beim Scheduling mit mehreren Zugriffsgruppen |
| $H$ | Zufallsvariable für die Anzahl der D-Aufträge, die pro Runde eintreffen |
| $I$ | Zufallsvariable der Zeit, die ein D-Auftrag in Zwischenrunden verbringt |
| $I_l$ | Zufallsvariable der Zeit, die ein D-Auftrag in Zwischenrunden verbringt unter der Voraussetzung, daß zu seinem Ankunftszeitpunkt $l$ D-Aufträge in der Warteschlange vorhanden sind |
| $I_l^*$ | Laplace-Transformierte der Zufallsvariablen $I_l$ |
| $L$ | Zufallsvariable der Warteschlangenlänge des analytischen Modells zu einem beliebigen Zeitpunkt |
| $L(t)$ | Warteschlangenlänge an einem Betrachtungszeitpunkt $t$ der Markov-Kette zu Rundenbeginn |
| $L_{max}$ | maximale Länge der Warteschlange im analytischen Modell |
| $M$ | Zufallsvariable für die Anzahl der pro Runde ausführbaren D-Aufträge |
| $M_{max}$ | maximale Anzahl an D-Aufträgen, die im analytischen Modell pro Runde ausgeführt werden können |
| $M_{min}$ | minimale Anzahl an D-Aufträgen, die im analytischen Modell pro Runde garantiert ausgeführt werden können |
| $M(l)$ | Anzahl der Zwischenrunden, die ein D-Auftrag maximal bis zur Abgangsrunde warten muß, wenn zu seinem Ankunftszeitpunkt $l$ D-Aufträge in der Warteschlange vorhanden sind |
| $N$ | Anzahl der K-Aufträge pro Runde und Platte |
| $N_{tot}$ | Gesamtzahl der K-Aufträge, die pro Runde vom Disk-Array ausgeführt werden |
| $N_{tot}^{wc}$ | maximale Anzahl an K-Aufträgen, die pro Runde vom Disk-Array bei deterministischer Service-Garantie bearbeitet werden können |
| $N_{max}$ | maximal erlaubte Anzahl an K-Aufträgen pro Runde und Platte, bei der die gewünschte Service-Qualität eingehalten werden kann |

| | |
|---|---|
| $N_{max}^{\approx}$ | untere Schranke von $N_{max}$ bei der Berechnung durch das stochastische Modell |
| $N_{max}^{wc}$ | maximale Anzahl an K-Aufträgen pro Runde und Platte bei deterministischer Service-Garantie |
| $O$ | Zufallsvariable für die Anzahl der D-Aufträge, die innerhalb eines festgelegten Zeitintervalls eintreffen |
| $P_{cap}$ | Speicherkapazität einer Spur einer Ein-Zonen-Platte |
| $P_{cap}^{i}$ | Speicherkapazität einer Spur der Zone $i$ einer Mehr-Zonen-Platte |
| $P_{cap}^{min}$ | Speicherkapazität der innersten Spur einer Mehr-Zonen-Platte |
| $P_{cap}^{max}$ | Speicherkapazität der äußersten Spur einer Mehr-Zonen-Platte |
| $P_{cap}^{tot}$ | Gesamtspeicherkapazität einer Magnetplatte |
| $P_{cpz}^{i}$ | Anzahl der Spuren in Zone $i$ einer Mehr-Zonen-Platte |
| $P_{cyls}$ | Anzahl der Zylinder |
| $P_{disks}$ | Anzahl der Platten im Disk-Array |
| $P_{heads}$ | Anzahl der Plattenköpfe |
| $P_{plat}$ | Anzahl der Scheiben der Magnetplatte |
| $P_{pos}^{i}$ | Parameter der Funktion $t_{pos}$, $i = 1, \ldots, 5$ |
| $P_{pos}^{min}$ | Positionierungszeit für einen Zylinderwechsel |
| $P_{pos}^{avg}$ | durchschnittliche Positionierungzeit |
| $P_{pos}^{max}$ | maximale Positionierungszeit, d.h. Positionierungszeit bei einer Distanz von $P_{cyls} - 1$ Zylindern |
| $P_{rate}^{min}$ | minimale Transferrate einer Mehr-Zonen-Platte |
| $P_{rate}^{max}$ | maximale Transferrate einer Mehr-Zonen-Platte |
| $P_{rate}^{i}$ | Transferrate in der Zone $i$ einer Mehr-Zonen-Platte |
| $P_{rate}$ | konstante Transferrate einer Ein-Zonen-Platte |
| $P_{rot}$ | maximale Rotationsverzögerung |

| | |
|---|---|
| $P_{zones}$ | Anzahl der Zonen einer Mehr-Zonen-Platte |
| $R$ | Zufallsvariable der Antwortzeitverteilung bei D-Aufträgen |
| $R^*$ | Laplace-Transformierte der Zufallsvariablen $R$ |
| $S$ | Zufallsvariable einer allgemeinen Auftragsgrößenverteilung |
| $S_D$ | Zufallsvariable für die Größe von D-Aufträgen |
| $S_K$ | Zufallsvariable für die Größe von K-Aufträgen |
| $S_K^{max}$ | maximale Fragmentgröße der kontinuierlichen Datenobjekte |
| $S_K^{95\%}$ | $95\%$-Quantil von $S_K$ |
| $S_K^{99\%}$ | $99\%$-Quantil von $S_K$ |
| $T$ | Rundendauer beim periodischen rundenbasierten Scheduling |
| $T_{svc}$ | Zufallsvariable der Gesamtbedienzeit |
| $T_{svc}^{LTI*}$ | Laplace-Transformierte der Verteilung für die Gesamtbedienzeit innerhalb einer Runde, Gesamtpositionierungszeit wird durch konstantes Maximum abgeschätzt |
| $T_{svc}^{LTII*}$ | Laplace-Transformierte der Verteilung für die Gesamtbedienzeit innerhalb einer Runde, Laplace-Transformierte der Gesamtpositionierungszeit wird durch numerische Integration berechnet |
| $T_{pos}$ | Zufallsvariable der Positionierungszeit eines Auftrages |
| $T_{\sum pos}^{LTI*}$ | Laplace-Transformierte der Verteilung für die Gesamtpositionierungszeit innerhalb einer Runde, Abschätzung durch konstantes Maximum |
| $T_{\sum pos}^{LTII*}$ | Laplace-Transformierte der Verteilung für die Gesamtpositionierungszeit innerhalb einer Runde, Berechnung durch numerische Integration |
| $T_{\sum pos}$ | Zufallsvariable der Gesamtpositionierungszeit innerhalb einer Runde |
| $T_{pos,i}$ | Zufallsvariable der Positionierungszeit des Auftrages $i$ |
| $T_{rot}$ | Zufallsvariable der Rotationsverzögerung eines Auftrages |
| $T_{rot}^*$ | Laplace-Transformierte der Verteilung für die Rotationsverzögerung eines Auftrages |
| $T_{\sum rot}$ | Zufallsvariable der Gesamtrotationsverzögerung innerhalb einer Runde |

| | |
|---|---|
| $T^*_{\sum rot}$ | Laplace-Transformierte der Verteilung für die Gesamtrotationsverzögerung innerhalb einer Runde |
| $T_{rot,i}$ | Zufallsvariable der Rotationsverzögerung des Auftrages $i$ |
| $T_{trans}$ | Zufallsvariable der Transferzeit eines Auftrages |
| $T^*_{trans}$ | Laplace-Transformierte der Verteilung für die Transferzeit eines Auftrages |
| $T_{trans,i}$ | Zufallsvariable der Transferzeit des Auftrages $i$ |
| $T_{\sum trans}$ | Zufallsvariable der Gesamttransferzeiten innerhalb einer Runde |
| $T^*_{\sum trans}$ | Laplace-Transformierte der Verteilung für die Gesamttransferzeit innerhalb einer Runde |
| $V$ | Zufallsvariable der Transferrate bei einer Mehr-Zonen-Platte |
| $Z$ | Zufallsvariable der Zylinderauswahl |
| $\alpha_S$ | Parameter der gammaverteilten Größenverteilung für K-Aufträge |
| $\alpha_{trans}$ | Parameter der gammaverteilten Transferzeit eines K-Auftrages für Ein-Zonen-Platten |
| $\alpha^{\approx}_{trans}$ | Parameter der gammaverteilten Transferzeit eines K-Auftrages für Mehr-Zonen-Platten |
| $\alpha_{\sum trans}$ | Parameter der gammaverteilten Gesamttransferzeit für Ein-Zonen-Platten |
| $\alpha^{\approx}_{\sum trans}$ | Parameter der gammaverteilten Gesamttransferzeit für Mehr-Zonen-Platten |
| $\beta_S$ | Parameter der gammaverteilten Größenverteilung für K-Aufträge |
| $\beta_{trans}$ | Parameter der gammaverteilten Transferzeit eines K-Auftrages für Ein-Zonen-Platten |
| $\beta^{\approx}_{trans}$ | Parameter der gammaverteilten Transferzeit eines K-Auftrages für Mehr-Zonen-Platten |
| $\beta_{\sum trans}$ | Parameter der gammaverteilten Gesamttransferzeit für Ein-Zonen-Platten |
| $\beta^{\approx}_{\sum trans}$ | Parameter der gammaverteilten Gesamttransferzeit für Mehr-Zonen-Platten |
| $\delta_{erate}$ | Schwellwert für die Restwahrscheinlichkeit $p_{erate}$ bei der Konfigurationsberechnung |

| | |
|---|---|
| $\delta_{resp}$ | Schwellwert für die Restwahrscheinlichkeit $p_{resp}$ bei der Konfigurationsberechnung |
| $\delta_0(t)$ | Dirac-Funktion |
| $\epsilon_{erate}$ | Schranke für die Störungsrate, deren Restwahrscheinlichkeit bei der Berechnung von $N_{max}$ berechnet wird |
| $\epsilon_{resp}$ | Schranke für die Antwortzeit, deren Restwahrscheinlichkeit bei der Berechnung von $\lambda_{max}$ berechnet wird |
| $\lambda$ | Ankunftsrate der D-Aufträge pro Platte |
| $\lambda_{tot}$ | Gesamtankunftsrate der D-Aufträge für das gesamte Disk-Array |
| $\lambda_{max}$ | maximal erlaubte Ankunftsrate der D-Aufträge, bei der die gewünschte Service-Qualität eingehalten werden kann |
| $\lambda_{max}^{\approx}$ | untere Schranke von $\lambda_{max}$ bei der Berechnung durch das stochastische Modell |
| $\mu_S$ | Parameter der normalverteilten Größenverteilung für D-Aufträge |
| $\mu_{rot}^{\approx}$ | Parameter für approximierte normalverteilte Rotationsverzögerung |
| $\mu_{\Sigma rot}^{\approx}$ | Parameter für approximierte normalverteilte Gesamtrotationsverzögerung |
| $\pi_l$ | Wahrscheinlichkeit des stationären Zustandes $l$ der Markov-Kette |
| $\sigma_S$ | Parameter der normalverteilten Größenverteilung für D-Aufträge |
| $\sigma_{rot}^{\approx}$ | Parameter für approximierte normalverteilte Rotationsverzögerung |
| $\sigma_{\Sigma rot}^{\approx}$ | Parameter für approximierte normalverteilte Gesamtrotationsverzögerung |
| $\theta_{opt}$ | Stelle, an der die Funktion $h(\theta)$ ihr Minimum annimmt |

# Literaturverzeichnis


[All90] Arnold O. Allen. *Probability, Statistics and Queueing Theory with Computer Science Applications*. Academic Press, 2nd edition, 1990.

[BDEM+94] S. Berson, A. Dashti, M. Escobar-Molano, S. Ghandeharizadeh, L. Golubchik, R. Muntz and C. Shahabi. Design of a Scalable Multimedia Storage Manager. Technical Report CSD TR-940042, University of California, Los Angeles, 1994.

[Ben97] Herve Benoit. *Digital Television: MPEG-1, MPEG-2 and Principles of the DVB System*. Wiley, 1997.

[BGMJ94] Steven Berson, Shahram Ghandeharizadeh, Richard R. Muntz and Xiangyu Ju. Staggered Striping in Multimedia Information Systems. In *Proceedings of the International Conference on Management of Data (SIGMOD'94), Minneapolis, Minnesota*, pages 79–90, May 1994.

[BMW96] Steven Berson, Richard R. Muntz and Wai-Man R. Wong. Randomized Data Allocation for Real-time Disk I/O. In *Proceedings of COMPCON'96, Santa Clara, California*, pages 286–290, February 1996.

[BSM98] Ilja N. Bronstein, Konstantin A. Semendjajew and Gerhard Musiol. *Taschenbuch der Mathematik*. Harri Deutsch, Frankfurt, 1998.

[CBR94] Ariel Cohen, Walter A. Burkhard and P. Venkat Rangan. Storage Architectures for Digital Movie Retrieval. Technical Report CS94-390, CSE Dept., University of California, San Diego, 1994.

[CBR95] Ariel Cohen, Walter A. Burkhard and P. Venkat Rangan. Pipelined Disk Arrays for Digital Movie Retrieval. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS'95), Washington D.C.*, pages 312–317, May 1995.

[CH82] Edward G. Coffman Jr. and Micha Hofri. On the Expected Performance of Scanning Disks. *Siam Journal on Computing*, 11(1):60–70, 1982.

[CH90] Edward G. Coffman Jr. and Micha Hofri. Queueing Models of Secondary Storage Devices. In Hideaki Takagi, editor, *Stochastic Analysis of Computer and Communication Systems*. North-Holland, 1990.

[CKY93] Mon-Song Chen, Dilip D. Kandlur and Philip S. Yu. Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams. In *Proceedings of the ACM International Conference on Multimedia (ACM Multimedia '93), Anaheim, California*, pages 235–242, August 1993.

[CL95] Peter M. Chen and Edward K. Lee. Striping in a RAID Level 5 Disk Array. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'95), Ottawa, Ontario, Canada*, pages 136–145, May 1995.

[CL96] Huang-Jen Chen and Thomas D.C. Little. Storage Allocation Policies for Time-Dependent Multimedia Data. *IEEE Transactions on Knowledge and Data Engineering*, 8(5):855–864, October 1996.

[CLG+94] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz and David A. Patterson. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.

[Con98] W3C: World Wide Web Consortium. *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification , W3C Recommendation*. Available from www.w3c.org, 1998.

[CP90] Peter M. Chen and David A. Patterson. Maximizing Performance in a Striped Disk Array. In *Proceedings of the 17th International Symposium on Computer Architecture (SIGARCH'90), Seattle, WA*, pages 322–331, May 1990.

[CPRV98] Igor D.D. Curcio, Antonio Puliafito, Salvatore Riccobene and Lorenzo Vita. Design and Evaluation of a Multimedia Storage Server for Mixed Traffic. *Multimedia Systems*, 6(6):367–381, 1998.

[CT93] Shenze Chen and Don Towsley. The Design and Evaluation of RAID 5 and Parity Striping Disk Array Architectures. *Journal of Parallel and Distributed Computing*, 17(1):58–74, January 1993.

[CZ94a] Ed Chang and Avideh Zakhor. Admissions Control and Data Placement for VBR Video Servers. In *Proceedings of the 1st IEEE International Conference on Image Processing (ICIP'94), Austin, Texas*, pages 278–282, November 1994.

[CZ94b] Ed Chang and Avideh Zakhor. Variable Bit Rate MPEG Video Storage on Parallel Disk Arrays. In *Proceedings of the 1st International Workshop on Community Networking Integrated Multimedia Services to the Home, San Francisco, California*, pages 127–137, July 1994.

[CZ96] Ed Chang and Avideh Zakhor. Cost Analyses for VBR Video Servers. In *Proceedings of IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology, Volume 2667: Multimedia Computing and Networking (MMCN'96), San Jose, California*, pages 381–397, January 1996.

[DBB96] Johannes Dengler, Christoph Bernhardt and Ernst Biersack. Deterministic Admission Control Strategies in Video Servers with Variable Bit Rate Streams. In

*Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Services (IDMS'96), Berlin, Germany*, March 1996.

[DDM+95] Asit Dan, Daniel Dias, Rajat Mukherjee, Dinkar Sitaram and Renu Tewari. Buffering and Caching in Large Scale Video Servers. In *Proceedings of COMPCON'95*, March 1995.

[DDSS95] Jacques Desroisiers, Yvan Dumas, Marius M. Solomon and François Soumis. Time Constrained Routing and Scheduling. In G.L. Nemhauser and A.H.G. Rinnooy Kan, editors, *Handbooks in Operations Research and Management Science: Volume 8: Network Routing*. Elsevier, 1995.

[Dos90] Bharat Doshi. Single Server Queues with Vacations. In Hideaki Takagi, editor, *Stochastic Analysis of Computer and Communication Systems*. North-Holland, 1990.

[Eis72] Martin Eisenberg. Queues with Periodic Service and Changeover Time. *Operations Research*, 20:440–451, 1972.

[Fel71] William Feller. *An Introduction to Probability Theory and Its Applications*, volume Volume 2. John Wiley & Sons, 1971.

[FW88] Steve W. Fuhrmann and Y.T. Wang. Analysis of Cyclic Service Systems with Limited Service: Bound and Approximations. *Performance Evaluation*, 9:35–54, 1988.

[GC92] Jim Gemmell and Stavros Christodoulakis. Principles of Delay-Sensitive Multimedia Data Storage and Retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, January 1992.

[GH94] D. James Gemmell and Jiawei Han. Multimedia Network File Servers: Multichannel Delay-Sensitive Data Retrieval. *ACM Multimedia Systems*, 1(6):240–252, 1994.

[GHBC94] D. James Gemmell, Jiawei Han, Richard J. Beaton and Stavros Christodoulakis. Delay-Sensitive Multimedia on Disks. *IEEE Multimedia*, pages 57–67, 1994.

[GHW90] Jim Gray, Bob Horst and Mark Walker. Parity Striping of Disk Arrays: Low-Cost Reliable Storage with Acceptable Throughput. In *Proceedings of the 16th International Conference on Very Large Data Bases (VLDB'90), Brisbane, Queensland, Australia*, pages 148–161, August 1990.

[GK95a] Shahram Ghandeharizadeh and Seon Ho Kim. An Analysis of Striping in Scalable Multi-Disk Video Servers. Technical Report USC-CS-TR95-623, University of Southern California, 1995.

[GK95b] Shahram Ghandeharizadeh and Seon Ho Kim. Striping in Multi-Disk Video Servers. In *Proceedings of High-Density Data Recording and Retrieval Technologies, Philadelphia, USA*, October 1995.

[GKS96] Shahram Ghandeharizadeh, Seon Ho Kim and Cyrus Shahabi. On Disk Scheduling and Data Placement for Video Servers. *ACM Multimedia Systems*, 1996.

[GKSZ96] Shahram Ghandeharizadeh, Seon Ho Kim, Weifeng Shi and Roger Zimmermann. On Minimizing Startup Latency in Scalable Continuous Media Servers. Technical Report USC-CS-TR96-627, University of Southern California, 1996.

[GLdG98] Leana Golubchik, John C.S. Lui, Edmundo de Silva e Souza and H. Richard Gail. Evaluation of Tradeoffs in Resource Management Techniques for Multimedia Storage Servers. Technical Report TR-3904, University of Maryland, Department of Computer Science, 1998.

[GLM94] Leana Golubchik, John C. S. Lui and Richard R. Muntz. Reducing I/O Demand in Video-On-Demand Storage Servers. Technical Report CSD TR-940037, University of California, Los Angeles, 1994.

[GVK+95] D. James Gemmel, Harrick M. Vin, Dilip D. Kandlur, P. Venkat Rangan and Lawrence A. Rowe. Multimedia Storage Servers: A Tutorial. *IEEE Computer*, pages 40–49, May 1995.

[HKR97] Silvia Hollfelder, Achim Kraiß and Thomas C. Rakow. A client-controlled Adaption Framework for Multimedia Database Systems. In *Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Telecommunications Services (IDMS'97), Darmstadt, Germany*, pages 397–409, September 1997.

[HPN96] Barry G. Haskell, Atul Puri and Arun N. Netravali. *Digital Video: An Introduction to MPEG-2 (Digial Multimedia Standard Series)*. Chapman & Hall, 1996.

[HR89] Torben Hagerup and Christiane Rüb. A Guided Tour of Chernoff Bounds. *Information Processing Letters*, 33:305–308, 1989.

[ITU90] ITU. *Recommendation 601: Encoding parameters of digital television for studios, Recommendations of the CCIR, Volume XI - Part 1*. International Telecommunication Union, Geneva, 1990.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

[Jan98] Anja Jantke. *Stochastische Performance-Analyse einer Bedienstrategie in einem Multimedia-Informationssystem*. Diplomarbeit, Universität des Saarlandes, Fachbereich Informatik, Mai 1998.

[KEGLA98] Franklin Kuo, Wolfang Effelsberg and J.J. Garcia-Luna-Aceves. *Multimedia Communications: Protocols and Applications*. Prentice Hall, 1998.

[KGK96] Jörg Keller, Thomas Grün and Jörg Keller. Video-on-Demand on the SB-PRAM. In *Proceedings of the 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'96), Zushi, Japan*, pages 105–111, April 1996.

[KH95] Marwan Krunz and Herman Hughes. A Traffic Model for MPEG-Coded VBR Streams. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'95), Ottawa, Canada*, May 1995.

[Kim86] Michelle Y. Kim. Synchronized Disk Interleaving. *IEEE Transactions on Computers*, 35(11):978–988, November 1986.

[Kof96] Michael Kofler. *Maple V Release IV: Einführung und Leitfaden für den Praktiker*. Addison-Wesley, 1996.

[Kor97] Jan Korst. Random Duplicated Assignment: An Alternative to Stiping in Video Servers. In *Proceedings of the ACM International Multimedia Conference (ACM Multimedia 97, Seattle, Washington, USA*, pages 219–226, November 1997.

[Kre86] Wolfgang Kreutzer. *Systems Simulation - Programming Styles & Languages*. Addison-Wesley, 1986.

[LaM90] Richard O. LaMaire. M/G/1 Vacation Model with Varying E-Limited Service Discipline. Technical Report RC 16554, IBM Research Division T.J. Watson Research Center, Yorktown Heights, NY, December 1990.

[LaM91] Richard O. LaMaire. M/G/1 Vacation Model with Varying G-Limited Service Discipline. Technical Report RC 17275, IBM Research Division T.J. Watson Research Center, Yorktown Heights, NY, August 1991.

[LaM92] Richard O. LaMaire. M/G/1/N vacation model with varying E-limited service discipline. *Queueing Systems*, 11:357–375, 1992.

[Lan92] Horst Langendörfer. *Leistungsanalyse von Rechensystemen, Messen, Modellieren, Simulation*. Hanser Verlag, 1992.

[LE91] Kin K. Leung and Martin Eisenberg. A Single-Server Queue with Vacations and Non-Gated Time-Limited Service. *Performance Evaluation*, 12:115–125, 1991.

[Lee84] Tony T. Lee. M/G/1/N Queue with Vacation Time and Exhaustive Service Discipline. *Operations Research*, 32(4):774–784, 1984.

[Lee89] Tony T. Lee. M/G/1/N Queue with Vacation Time and Limited Service Discipline. *Performance Evaluation*, 9:181–190, 1989.

[LKB87] Miron Livny, Setrag Khoshafian and Haran Boral. Multi-Disk Management Algorithms. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'87)*, pages 69–77, 1987.

[LL94] Kin K. Leung and David M. Lucantoni. Two Vacation Models for Token-Ring Networks where Service is controlled by Timers. *Performance Evaluation*, 20:165–184, 1994.

[MNO+96] Cliff Martin, P. S. Narayan, Banu Özden, Rajeev Rastogi and Avi Silberschatz. The Fellini Multimedia Storage Server. In Soon M. Chung, editor, *Multimedia Information Storage and Management*. Kluwer, 1996.

[MSB98] Richard R. Muntz, Jose Renato Santos and Steven Berson. A Parallel Disk Storage System for Realtime Multimedia Applications. *International Journal on Intelligent Systems, Special Issue on Multimedia Computing Systems*, 1998.

[MSS96] David R. Musser, Atul Saini and Alexander Stepanov. *STL Tutorial & Reference Guide: C++ Programming With the Standard Template Library*. Addison-Wesley, 1st edition, 1996.

[Nel95] Randolph Nelson. *Probability, Stochastic Processes, and Queueing Theory : The Mathematics of Computer Performance Modeling*. Springer, 1995.

[NMP+97] Guido Nerjes, Peter Muth, Michael Paterakis, Yannis Romboyannakis, Peter Triantafillou and Gerhard Weikum. On Mixed-Workload Multimedia Storage Servers with Guaranteed Performance and Service Quality. In *Proceedings of the 3rd International Workshop on Multimedia Information Systems (MIS'97), Como, Italy*, pages 63–72, October 1997.

[NMP+98] Guido Nerjes, Peter Muth, Michael Paterakis, Yannis Romboyannakis, Peter Triantafillou and Gerhard Weikum. Scheduling Strategies for Mixed Workloads in Multimedia Information Servers. In *Proceedings of the IEEE International Workshop on Research Issues in Data Engineering (RIDE'98), Orlando, Florida*, February 1998.

[NMP+99] Guido Nerjes, Peter Muth, Michael Paterakis, Yannis Romboyannakis, Peter Triantafillou and Gerhard Weikum. Incremental Scheduling of Mixed Workloads in Multimedia Information Servers. *Special Issue of the Journal of Multimedia Tools and Applications*, to appear 1999.

[NMW97a] Guido Nerjes, Peter Muth and Gerhard Weikum. Stochastic Performance Guarantees for Mixed Workloads in a Multimedia Information System. In *Proceedings of the IEEE International Workshop on Research Issues in Data Engineering (RIDE'97), Birmingham, UK*, pages 131–140, April 1997.

[NMW97b] Guido Nerjes, Peter Muth and Gerhard Weikum. Stochastic Service Guarantees for Continuous Data on Multi-Zone Disks. In *Proceedings of the 16th Symposium on Principles of Database Systems (PODS'97), Tucson, Arizona*, pages 154–160, May 1997.

[NMW99] Guido Nerjes, Peter Muth and Gerhard Weikum. A Performance Model of Mixed-Workload Multimedia Information Servers. In *10. ITG/GI-Fachtagung: Messen, Modellieren und Bewerten von Rechen- und Kommunikationssystemen (MMB'99), Trier*, September 1999.

[NY94] Raymond T. Ng and Jinhai Yang. Maximizing Buffer and Disk Utilization for News On-Demand. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94), Santiago de Chile, Chile*, pages 451–462, September 1994.

[One75] Walter C. Oney. Queueing Analysis of the Scan Policy for Moving-Head Disks. *Journal of the ACM*, 22(3):397–412, 1975.

[ORS95] Banu Özden, Rajeev Rastogi and Avi Silberschatz. Disk Striping in Video Server Environments. *Bulletin of the Technical Committee on Data Engineering*, 18(4):4–16, December 1995.

[ORS96] Banu Özden, Rajeev Rastogi and Avi Silberschatz. Disk Striping in Video Server Environments. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS'96), Hiroshima, Japan*, June 1996.

[OS97] Michael Orzessek and Peter Sommer. *ATM & MPEG-2: Integrating Digital Video into Broadband Networks (Hewlett-Packard Professional Books)*. Prentice Hall, 1997.

[Ott87] Teunis J. Ott. The Single-Server Queue with Independent GI/G and M/G Input Streams. *Advances in Applied Probability*, 19:266–286, 1987.

[Oya95] Yen-Jen Oyang. A Tight Upper Bound of the Lumped Disk Seek Time for the Scan Disk Scheduling Policy. *Information Processing Letters*, 54:355–358, 1995.

[Pan95] Davis Pan. A Tutorial on MPEG/Audio Compression. *IEEE Multimedia*, pages 60–74, Summer 1995.

[PGK88] David A. Patterson, Garth A. Gibson and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the International Conference on Management of Data (SIGMOD'88), Chicago, Illinois*, pages 109–116, June 1988.

[Qua99a] Quantum. *Drives and Computer System Performance*. Quantum Corporation, Whitepaper available at www.quantum.com, 1999.

[Qua99b] Quantum. *New Technologies at a Glance*. Quantum Corporation, Whitepaper available at www.quantum.com, 1999.

[RNP+98] Yannis Romboyannakis, Guido Nerjes, Michael Paterakis, Peter Triantafillou and Gerhard Weikum. Disk Scheduling for Mixed-Media Workloads in a Multimedia Server. In *Proceedings of the ACM International Conference on Multimedia (ACM Multimedia'98), Bristol, UK*, September 1998.

[Rom98] Ioannis K. Romboyannakis. *On Disk Scheduling and Performance Modeling of Mixed Workload Multimedia Information Servers*. Master thesis, Department of Electronics and Computer Engineering, Chania, Greece, 1998.

[Ros95] Oliver Rose. Statistical Properties of MPEG Video Traffic and Their Impact on Traffic Modeling in ATM Systems. Technical Report Technical Report 101, Institute of Computer Science, University of Würzburg, Germany, 1995.

[RV93] P. Venkat Rangan and Harrick M. Vin. Efficient Storage Techniques for Digital Continuous Multimedia. *IEEE Transactions on Knowledge and Data Engineering: Special Issue on Multimedia Information Systems*, 5(4), August 1993.

[RVT96] Sriram Rao, Harrick M. Vin and Ashis Tarafdar. Comparative Evaluation of Server-push and Client-pull Architectures for Multimedia Servers. In *Proceedings of the 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'96), Zushi, Japan*, April 1996.

[RW94a] A. L. Narasimha Reddy and J. Wyllie. I/O Issues in a Multimedia System. *IEEE Computer*, 27(3):69–74, March 1994.

[RW94b] Chris Ruemmler and John Wilkes. An Introduction to Disk Modeling. *IEEE Computer*, 27(3):17–28, March 1994.

[RZ95] Doron Rotem and J. Leon Zhao. Buffer Management for Video Database Systems. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95), Taipei, Taiwan*, pages 439–448, March 1995.

[Sai98] Bare Said. *Konzeption und Implementierung eines Mixed-Workload Multimedia Informationssystems: die Disk-Scheduling-Komponente*. Diplomarbeit, Universität des Saarlandes, Fachbereich Informatik, Juli 1998.

[Sal96] Betty Salzberg. Editor. *IEEE Data Engineering Bulletin: Special Issue on Online Reorganisation*, 19(2), June 1996.

[Sch74] R. Schassberger. A Broad Analysis of Single Server Priority Queues with Two Independent Input Streams, One of them Poisson. *Advances in Applied Probability*, 6:666–688, 1974.

[Sch96] Richard Schaphorst. *Videoconferencing and Videotelephony : Technology and Standards (Artech House Telecommunications Library)*. Artech House, 1996.

[Sch98] Herb Schwetman. Model-Based Systems Analysis Using CSIM. In *Proceedings of the 1998 Winter Simulation Conference (WSC'98), Washington, D.C*, Dezember 1998.

[SG94a] Cyrus Shahabi and Shahram Ghandeharizadeh. Continuous Display of Presentations Sharing Clips. Technical Report USC-CS-TR94-587, University of Southern California, 1994.

[SG94b] Abraham Silberschatz and Peter Galvin. *Operating System Concepts*. Addison-Wesley, 4th edition, 1994.

[SGM86] Kenneth Salem and Hector Garcia-Molina. Disk Striping. In *Proceedings of the 2nd International Conference on Data Engineering (ICDE'86), Los Angeles, California, USA*, pages 336–342, February 1986.

[Str97] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, 3rd edition, 1997.

[Sun98] Sun. *JMF Programmer's Guide: Java Media Players*. Sun Microsystems Incorporated, Guide available at www.javasoft.com, 1998.

[SWZ98] Peter Scheuermann, Gerhard Weikum and Peter Zabback. Data Partitioning and Load Balancing in Parallel Disk Systems. *VLDB Journal*, 7(1):48–66, February 1998.

[SZKT97] Sambit Sahu, Zhi-Li Zhang, Jim Kurose and Don Towsley. On the Efficient Retrieval of VBR Video in a Multimedia Server. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS'97), Ottawa, Ontario, Canada*, pages 46–53, June 1997.

[Tak91] Hideaki Takagi. *Queueing Analysis : A Foundation of Performance Analysis, Volume 1 : Vacation and Priority Systems*. North-Holland, Amsterdam, 1991.

[Tay97] Jim Taylor. *DVD Demystified: The Guidebook for DVD-Video and DVD-ROM*. McGraw-Hill, 1997.

[TCG98] Peter Triantafillou, Stavros Christodoulakis and Costas Georgiadis. Optimal Data Placement on Disks: A Comprehensive Solution for Different Technologies. *IEEE Transactions on Knowledge and Data Engineering*, to appear 1998.

[TKC+97] Heiko Thimm, Wolfgang Klas, Crispin Cowan, Jonathan Walpole and Calton Pu. Optimization of Adaptive Data-Flows for Competing Multimedia Presentational Database Sessions. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS'97), Ottawa, Canada*. IEEE, June 1997.

[TKKD96] Renu Tewari, Richard P. King, Dilip Kandlur and Daniel Dias. Placement of Multimedia Blocks on Zoned Disks. In *Proceedings of IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology, Volume 2667: Multimedia Computing and Networking (MMCN'96), Volume 2667, San Jose, California*, January 1996.

[TP72] Toby J. Teorey and Tad B. Pinkerton. A Comparative Analysis of Disk Scheduling Policies. *Communications of the ACM*, 15(3):177–184, March 1972.

[TPBG93] Fouad A. Tobagi, Joseph Pang, Randall Baird and Mark Gang. Streaming RAID - A Disk Array Management System for Video Files. In *Proceedings of the ACM International Conference on Multimedia (ACM Multimedia'93), Anaheim, California*, August 1993.

[VGG95] Harrick M. Vin, Alok Goyal and Pawan Goyal. Algorithms for Designing Large-Scale Multimedia Servers. *Computer Communications*, March 1995.

[VGGG94] Harrick M. Vin, Pawan Goyal, Alok Goyal and Anshuman Goyal. A Statistical Admission Control Algorithm for Multimedia Servers. In *Proceedings of the the ACM International Conference on Multimedia (ACM Multimedia '94), San Francisco, CA*, October 1994.

[VR93] Harrick M. Vin and P. Venkat Rangan. Designing a Multi-User HDTV Storage Server. *IEEE Journal on Selected Areas in Communications*, 11(1):153–164, January 1993.

[VRG95] Harrick M. Vin, Sriram Rao and Pawan Goyal. Optimizing the Placement of Multimedia Objects on Disk Arrays. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'95), Washington, D.C*, pages 158–165, May 1995.

[Wen99] Andreas Wendling. *Konzeption und Implementierung eines Mixed-Workload Multimedia Informationssystems: die Dateiverwaltungs- und Netzwerkkomponente*. Diplomarbeit, Universität des Saarlandes, Fachbereich Informatik, Januar 1999.

[WGP94] Bruce L. Worthington, Gregory R. Ganger and Yale N. Patt. Scheduling Algorithms for Modern Disk Drives. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'94), Nashville, Tennessee, USA*, pages 241–251, May 1994.

[WGPW95] Bruce L. Worthington, Gregory R. Ganger, Yale N. Patt and John Wilkes. On-Line Extraction of SCSI Disk Drive Parameters. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'95), Ottawa, Ontario, Canada*, May 1995.

[Wol96] Stephen Wolfram. *Das Mathematica Buch*. Addison-Wesley, 1996.

[YCK93] Philip S. Yu, Mon-Song Chen and Dilip D. Kandlur. Grouped Sweeping Scheduling for DASD-based Multimedia Storage Management. *ACM Multimedia Systems*, 1(3):99–109, 1993.

[Zab94] Peter Zabback. *Parallelism in Database Systems - Design, Implementation and Evaluation of a Storage System for Parallel Disks (in German)*. Doctoral thesis, Department of Computer Science ETH Zurich, 1994.