Messung und Modellierung von SAP R/3- und Storage-Systemen für die Kapazitätsplanung

Dissertation
zur Erlangung des Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

Vorgelegt beim
Fachbereich Mathematik
der Universität Duisburg-Essen
von

Kay Wilhelm, geboren in Bottrop

Datum der mündlichen Prüfung: 28.07.2003
Gutachter:

Prof. Dr. Bruno Müller-Clostermann
(Universität Duisburg-Essen, Standort Essen)

Prof. Dr. Klaus Echtle
(Universität Duisburg-Essen, Standort Essen)

Kurzfassung

Die betriebswirtschaftliche Standardsoftware SAP R/3 wird von vielen Firmen zur Verarbeitung ihrer Geschäftsprozesse eingesetzt. Hohe Antwortzeiten bei der Transaktionsverarbeitung in einem R/3-System und eine daraus folgende mangelnde Dienstgüte oder mangelnde Systemverfügbarkeit können weitreichende Folgen für die Firmen nach sich ziehen, so dass z. B. Produkte nicht rechtzeitig fertiggestellt werden können. Verursacht werden derartige Leistungsengpässe u. a. durch einen Anstieg der Benutzeranzahl, einen R/3-Release-Wechsel oder Lastzuwächse, die z. B. durch die Installation weiterer Business Components verursacht werden. Das Ziel muss somit sein, derartige Szenarien frühzeitig zu erkennen, um bereits im Voraus Leistungsengpässen durch z. B. die rechtzeitge Durchführung notwendiger System-Upgrades entgegenzuwirken.

In dieser Arbeit wird ein Vorgehensmodell zur Kapazitätsplanung von SAP R/3-Systemen vorgestellt, das zum einen das Monitoring, die Messung und die Analyse von R/3-Performance-Daten umfasst und zum anderen die Performance-Prognose für mögliche Zukunftsszenarien auf Basis der zuvor gewonnenen Messdaten beinhaltet. Im Rahmen der Datenerfassung wird ein Konzept zur Workload-Charakterisierung einschließlich Dienstgütedefinition beschrieben. Die Performance-Prognose wird mit Hilfe von effizient lösbaren mathematisch-analytischen Modellen (Warteschlangennetzwerken) durchgeführt. Sowohl für die Analyse als auch für die Modellierung werden in dieser Arbeit Konzepte, Methoden und Werkzeuge detailliert beschrieben und anhand von verschiedenen Fallstudien demonstriert.

Nahezu jede Transaktion in einem SAP-System muss auf die zentrale Datenbank zugreifen und somit in Abhängigkeit des Datenbank-Cache auf das zugehörige IO-System. Die Antwortzeit pro Transaktion steht daher neben der Prozessorleistung der Server auch in starker Abhängigkeit zur IO-Performance. Einen weiteren Schwerpunkt dieser Arbeit bildet somit die Performance-Analyse und -Prognose von IO-Systemen. Es werden Konzepte zur Lastbeschreibung der IO-Aktivität in Abhängigkeit zu der Last in einem SAP R/3-System sowie ein analytisches Festplattenmodell erarbeitet. Das Modell unterstützt die Festplatteneigenschaft Tagged Command Queueing sowie die Caching-Strategien für Lese- und Schreibzugriffe und berücksichtigt Einflussgrößen des Betriebssystems auf die IO-Verarbeitung. Es wird im Gegensatz zu bisherigen Arbeiten anhand von realen Messdaten validiert, d. h. es werden verschiedene Lastszenarien betrachtet, die mittels eines IO-Benchmarks generiert und mit Hilfe eines IO-Monitoring-Werkzeugs analysiert werden.

Abstract

Many companies make use of the standard business software SAP R/3 to carry out their business processes. High response times while processing transactions in an R/3 system and, as a result, a lack of service quality or a poor system availability may have serious consequences for the companies, so that e. g. products cannot be finished in time. Such performance bottlenecks are partly due to a growing number of users, an R/3 release upgrade or an increased workload caused by the installation of additional business components. Therefore the aim must be to make out such scenarios as early as possible in order to avoid performance bottlenecks e. g. by carrying out necessary system upgrades in time.

This PhD thesis is meant to present a feasible approach to the capacity planning of SAP R/3 systems, which deals with monitoring, measuring, and analysing R/3 performance data on the one hand, and predicting possible future scenarios on the basis of the data acquired earlier on the other. In the context of data measurement a model of characterizing workload including a concept to define service quality is presented. Quickly solvable mathematical and analytical models (queueing networks) have been employed to predict performance. Concepts, methods, and tools for analysing performance data and modelling SAP R/3 systems have been described in detail and explained on the basis of several case studies.

Almost every transaction in an SAP system accesses the central database and, depending on the cache of the database, the corresponding IO-system. Accordingly, the response time per transaction is determined by the CPU performance of the servers as well as by their IO performance. Thus, this PhD thesis also focuses on the analysis and prediction of the performance of IO systems. Concepts to describe the workload of IO activity in interdependence with the load in an SAP R/3 system have been worked out along with an analytical disk model. The latter supports the disk feature Tagged Command Queueing and caching strategies for read- and write-requests. Moreover, it takes into account the parameters of the operating system having an influence on IO processing. In contrast to earlier research this model has been evaluated on the basis of real data, i. e. various load scenarios have been examined, which were produced with the help of an IO benchmark and analysed by using an IO monitoring tool.

Danksagung

Bei allen, die auf das Zustandekommen dieser Arbeit durch ihre Unterstützung positiven Einfluss genommen haben, möchte ich mich herzlich bedanken.

Allen voran gilt der Dank natürlich dem Doktorvater dieser Arbeit, Prof. Dr. Bruno Müller-Clostermann, der durch fachliche Anregungen bis in die Endphase hinein die Gestaltung dieser Arbeit wesentlich beeinflusst hat. Seine Kenntnisse, sein profundes Fachwissen, sowie seine sehr guten industriellen Kontakte, welche die Kooperation mit Fujitsu Siemens Computers und somit meine Industrieprojektstelle ermöglichten, bildeten eine entscheidende Voraussetzung für den erfolgreichen Abschluss der vorliegenden Dissertation.

Ganz herzlich möchte ich auch Herrn Prof. Dr. Klaus Echtle für die Übernahme des Zweitgutachtens danken, sowie Herrn Prof. Dr. Michael Goedicke für die reibungslose und unkomplizierte Durchführung des Promotionsverfahrens.

Ferner möchte ich mich bei Herrn Joachim Witte für die fachlich interessante Kooperation und die finanzielle Unterstützung von Fujitsu Siemens Computers (CC Walldorf) bedanken. Hervorheben möchte ich hierbei Herrn Michael Paul und Herrn Jürgen Pfister, die fachlich stets zu Diskussionen bereit waren und wertvolle Anregungen und Unterstützung für meine Arbeit lieferten.

Schließlich möchte ich allen Kolleginnen, Kollegen und Studenten, insbesondere Corinna Flüs, André Best, Torsten Heverhagen, Markus Jochim, Anne Fischer, Jens Neumann und Thorsten Fischer für die vielen Anregungen und Ideen und nicht zuletzt für das angenehme Arbeitsklima in der Arbeitsgruppe Systemmodellierung meinen Dank aussprechen.

Ein sehr herzlicher Dank gilt meinen Eltern und meiner Schwester, die mir dies alles ermöglicht und mich fortlaufend unterstützt haben. Danken möchte ich insbesondere auch meiner Freundin Kerstin Dröse, die mich während der verschiedenen Arbeitsphasen inspiriert, motiviert und immer wieder aufgebaut hat.

INHALTSVERZEICHNIS

TABELLEN

ABBILDUNGEN

KAPITEL

1

EINLEITUNG UND MOTIVATION

In diesem Kapitel wird eine Einführung und Motivation zu den beiden Themenschwerpunkten dieser Arbeit, SAP R/3-Kapazitätsplanung und Modellierung von Storage-Systemen (Speichersubsysteme), beschrieben. Weiterhin werden die Zielsetzungen sowie die Gliederung der Arbeit dargestellt.

1.1. SAP R/3-Kapazitätsplanung

Fünf ehemalige IBM-Mitarbeiter (Dietmar Hopp, Hans-Werner Hector, Hasso Plattner, Klaus Tschira, Claus Wellenreuther) gründeten 1976 in Mannheim mit Geschäftssitz in Weinheim die SAP GmbH, die seit 1997 weltweit unangefochtener Marktführer bei betriebswirtschaftlicher Standardsoftware (Enterprise Resource Planning; ERP) ist [68]. Die Abkürzung SAP steht für die Bezeichnung „Systeme, Anwendungen, Produkte in der Datenverarbeitung“. Bereits nach acht Jahren hatte das Softwarehaus die Hälfte der 100 größten deutschen Industrieunternehmen als Kunden; heute sind es mehr als 90% dieser Unternehmen. Im Vertriebsbereich baut die SAP auf das Partnerkonzept auf, so dass sie mit einer Vielzahl namhafter Unternehmen strategische Partnerschaften besitzt. Die Partnerunternehmen stammen aus unterschiedlichen Branchen, wie z. B. führende Hardware-Hersteller, wie IBM oder Fujitsu Siemens Computers, sowie Unternehmensberatungen, wie Origin oder Accenture, so dass Kompetenzen und Ressourcen der SAP um qualifizierte Dienstleistungen ergänzt werden.

Das Hauptprodukt der SAP ist die betriebwirtschaftliche Standardsoftware R/3, die derzeit mit Einzug der Internet-Technologie in die Unternehmen von mySAP.com abgelöst bzw. integriert wird. Die Bezeichnung R/3 bedeutet zum einen Realtime („R“), d. h. es findet in einem R/3-System eine sofortige Verbuchung und Aktualisierung der Daten statt. Die „3“ steht hingegen für die dreischichtige Architektur nach dem Client-Server-Prinzip mit den Ebenen Präsentation, Applikation und Datenbank.

Die Standardsoftware SAP R/3 wird von den Firmen zur Abwicklung und Verarbeitung ihrer Geschäftsprozesse eingesetzt. Für nahezu jedes betriebswirtschaftliche Aufgabenfeld einer Firma bietet R/3 unterschiedliche Module (Business Components) an, wie beispielsweise Rechnungswesen, Logisitik, Personalwirtschaft. Entscheidet sich ein Unternehmen für die Standardsoftware von SAP, so werden von dieser alle betriebswirtschaftlichen Abläufe des

Unternehmens abgedeckt und das System ermöglicht somit eine vollständige betriebswirtschaftliche Steuerung der Firma.

Nach Installation von R/3 ist das Unternehmen von einer performanten Arbeitsgeschwindigkeit des SAP-Systems abhängig. Performant bedeutet, dass innerhalb eines bestimmten Zeitintervalls ein vorgegebener Durchsatz unter definierten Antwortzeitkriterien erreicht wird. Überlastete Systeme oder gar Ausfälle führen zu hohen Antwortzeiten und sinkenden Durchsätzen. Mögliche Folgen sind Frustration der Benutzer, Mehrarbeit, Produktionsverzögerungen und im Extremfall finanzielle Verluste oder Bankrott der Firma. Ein Beispiel hierfür ist der US-Pharmagroßhändler FoxMeyer Drug Co. (Umsatz 5 Mrd. $ pro Jahr, [113]). Nach Einführung von SAP R/3 wurden 10.000 Aufträge pro Tag von diesem System verarbeitet, wohingegen der tägliche Durchsatz des bisherigen Unisys Mainframe-Systems bei 420.000 Aufträgen lag. Die Konsequenz war nach kurzer Zeit der Konkurs der Firma (1997).

Der Fall FoxMeyer zeigt in drastischer Form die aus nicht ausreichender Planung des IT-Systems resultierenden Konsequenzen für das betroffene Unternehmen. Unter der Planung eines IT-Systems versteht man hierbei, dass ein System derart konfiguriert wird, dass es den geforderten Lastansprüchen und Dienstgüteanforderungen standhält. Erforderlich ist eine Planung immer dann, wenn Änderungen in einem bestehenden System vorgenommen bzw. ein neues System aufgebaut werden soll. Als Ergebnis der Planung werden aufgrund der definierten Anforderungen und Aufgabenstellungen Vorschläge oder Alternativen für IT-Systeme geliefert. Der Prozess zur Planung eines IT-Systems wird als Kapazitätsplanung definiert ([52], [9]).

In dieser Arbeit wird ein Vorgehensmodell zur Kapazitätsplanung von SAP R/3-Systemen vorgestellt, das zum einen das Monitoring, die Messung und die Analyse von R/3-Performance-Daten umfasst und zum anderen die Performance-Prognose für mögliche Zukunftsszenarien auf Basis der zuvor gewonnenen Messdaten beinhaltet. Im Rahmen der Datenmessung wird ein Konzept zur Workload-Charakterisierung einschließlich Dienstgütekonzept beschrieben. Das dargestellte Vorgehensmodell soll als Leitfaden für die Durchführung einer Kapazitätsplanung in einem Unternehmen dienen, wobei diese nicht als punktuelles Ereignis sondern als eigenständiger Prozess im Unternehmen etabliert werden sollte.

Es folgt als weitere Motivation die Beschreibung verschiedener Szenarien mit typischen Aufgabenstellungen der R/3-Kapazitätsplanung sowie verwandten Problemstellungen:

- Unternehmen stehen beim Einsatz von SAP R/3 häufig vor Aufgaben, wie z. B. Anstieg der Benutzerzahlen, R/3-Release-Wechsel oder Lastzuwächse durch Installation weiterer R/3-Module. Die Schwierigkeit bei der Lösung dieser Aufgaben liegt insbesondere in der Einschätzung des daraus resultierenden Systemverhaltens zur Einhaltung des gewünschten Service-Levels.
- Produktive R/3-Systeme benötigen ein zuverlässiges Performance-Monitoring- und Reporting-System, das zum einen vorhandene Systemengpässe erkennt und Ursachenforschung ermöglicht sowie die augenblickliche Systembelastung beschreibt.
- Die Erwartungen und Performance-Ansprüche an die IT-Systeme steigen mit zunehmender Globalisierung der Märkte. Die Präsenz von Firmen und die Komnunikation mit ihren Kunden und Partnern findet immer stärker unter Verwendung der Internettechnologie statt. Waren, die von einem Unternehmen hergestellt werden, können direkt über einen Internetshop verkauft werden. Der Kunde kann dann zwischen den verschiedenen Internetshops nach bestimmten Kriterien auswählen, wie z. B. Preis, Produktvielfalt und Präsentation. Entscheidend sind hierbei Faktoren wie Verfügbarkeit und Performance, denn bei zu langen Antwortzeiten bzw. Nicht-Verfügbarkeit der Web-Site hat der Kunde die Möglichkeit mit „einem Maus-Klick“ einen Shop der Konkurrenz auszuwählen. Der Unterschied für den Betreiber des Internetshops zu einem realen Geschäft ist insbesondere hierbei, dass im Internet 7 x 24 h pro Woche eingekauft werden und dass zu jeder Zeit eine „beliebige“ Anzahl an Kunden auf den Shop zugreifen kann. Die einem Internetshop

zugrundeliegenden Systeme müssen somit hohen Verfügbarkeits- und Performance-Anforderungen standhalten. Mit Einführung von mySAP.com und der damit verbundenen Internettechnologie sind somit diese Anforderungen zu bewältigen und ein Performance-Management zur ausreichenden Dimensionierung der IT-Systeme für die Einhaltung der definierten Sevice Level und zur Gewährleistung der Verfügbarkeit notwendig.

- Das Thema Systemdimensionierung und Zusicherung der versprochenen Dienstgüten ist für die Hardware-Lieferanten eine der wichtigsten Aufgaben. Der Hardware-Lieferant steht in der Pflicht, dem Kunden ein für seine Ansprüche entsprechend dimensioniertes System zu konfigurieren, so dass die Dienstgüteanforderungen des Kunden erfüllt werden. Die Schwierigkeit bzw. das Ziel sollte sein, Kundenwünsche in Form von Durchsätzen und Antwortzeiten als Garantieleistungen quantifizieren zu können. Zur Lösung dieser Aufgabe ist eine adäquate Lastcharakterisierung und eine lastabhängige Dienstgütedefinition von entscheidender Bedeutung.
- Eng verbunden mit dem Thema der Garantieleistungen bei Hardware-Lieferanten ist das Application Service Providing (ASP). Firmen mieten die Dienstleistung SAP R/3 über einen ASP-Dienstleister, d. h. sowohl Hardware als auch die Sicherstellung eines performanten Betriebs von SAP R/3 werden von dem ASP-Dienstleister angeboten und stehen somit in seiner Verantwortung. Die ASP-Dienstleister besitzen daher einen hohen Bedarf an Performance-Management-Unterstützung sowie die Aufgabe der Verrechnung ihrer IT-Dienste gegenüber dem Kunden. Die Frage ist, auf welcher Basis die IT-Leistung verrechnet werden soll. Auch hier spielt das Performance-Monitoring und die Workload-Charakterisierung eine entscheidende Rolle.

Sowohl für die Analyse als auch für die Modellierung werden in dieser Arbeit Konzepte, Methoden und Werkzeuge detailliert beschrieben. Ein Großteil der in den folgenden Kapiteln beschriebenen Methodik zur SAP R/3-Kapazitätsplanung entwickelte sich im Rahmen verschiedener Firmenprojekte, so dass bereits eine erfolgreiche Erprobung dieser Vorgehensweise stattgefunden hat.

1.2. Modellierung von Storage-Systemen

In immer geringeren Abständen werden in der Computerindustrie die Zyklen der Hardware-Entwicklungen verkürzt und bestehende Hardware durch leistungsstärkere Nachfolgemodelle ersetzt, so dass die Geschwindigkeit der Rechner stetig ansteigt. Die Computergeschwindigkeit wird häufig mit der Taktfrequenz der Prozessoren gleich gesetzt, wobei derzeitig Taktfrequenzen von 2,4 GHz im Bereich der Personal Computer (PC) dem Standard entsprechen. Neben der Taktfrequenz der Prozessoren besitzt die Geschwindigkeit der Sekundärspeicher ebenfalls einen entscheidenden Einfluss auf die Performance eines Rechners. Je höher die Taktfrequenz eines Prozessors, desto mehr Instruktionen können pro Zeiteinheit verarbeitet werden und desto mehr Speicherzugriffe werden pro Zeiteinheit generiert. Hohe Antwortzeiten aufgrund von langsamen IO-Systemen führen somit zu Wartezeiten auf Seiten der Prozessoren, so dass die Geschwindigkeit der Prozessoren sinkt.

Vergleicht man die Hardware-Entwicklung der IO-Systeme mit der von den Prozessoren, so ist mit Hilfe der in Tabelle 1.1 beschriebenen Leistungssteigerung für CPU-Taktfrequenzen gegenüber den Steigerungsfaktoren für Transferraten, Plattendrehzahlen und Zugriffszeiten von Festplatten ein signifikanter Unterschied zu erkennen. Es ergeben sich im Zeitraum von 1980 bis 2001 Steigerungen für die CPU-Taktfrequenzen um den Faktor 500, wohingegen die Performance der Festplatten nur um einen maximalen Faktor von 10 anwächst. Ein wesentlicher Grund für die relativ geringen Steigerungsraten der Festplatten-Performance ist, dass Festplatten auf

mechanischen Prozessen beruhen, wie z. B. Bewegungen des Schreib-/Lesekopfes und Plattenumdrehungen.

	1980	2001	Leistungsverbesserung
CPU-Taktung [MHz]	4,66	2400	515x
IO-Drehzahl [RPM]	3600	10000	2,8x
IO-Tranferrate [MB/s]	2,4	23,7	9,9x
IO-Seek Time avg. [ms]	24	5	4,8x

Tabelle 1.1 Historischer Geschwindigkeitsvergleich von Festplatten und CPUs[1]

1. Angaben wurden [2], [8], [139] entnommen

Die Auswirkungen der dargestellten unterschiedlichen Entwicklungen für CPU- und IO-Geschwindigkeiten können anhand von Amdahl's Law verdeutlicht werden, welches die Abhängigkeit der System-Performance von der langsamsten Systemkomponente, d. h. in diesem Fall dem IO-System, demonstriert. Angenommen wird hierbei, dass in einem System die CPU-Zeit einen 90 %igen Anteil an der Antwortzeit besitzt, d. h. die restlichen 10 % der Zeit entsprechen, aus Sicht der CPU, durch das IO-System verursachten Wartezeiten. Die CPU-Leistung wird nun bei gleichbleibender Geschwindigkeit des IO-Systems um den Faktor 10 erhöht. Laut Amdahl's Law ergibt sich dann für die CPUs ein Speedup-Faktor von 5, d. h. es resultiert nur die Hälfte der Performance-Verbesserung und es ergibt sich somit ein 50 %iger Verlust für die CPU-Performance. Die Berechnung des Speedup-Faktors beruht auf folgender Formel:

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

wobei $\text{Fraction}_{\text{enhanced}}$ den zu beschleunigenden Zeitanteil angibt, in diesem Fall den Anteil der CPU-Zeit an der Antwortzeit, und $\text{Speedup}_{\text{enhanced}}$ den Beschleunigungsfaktor definiert, in diesem Fall den Beschleunigungsfaktor für die CPU-Geschwindigkeit.

Nach Amdahl's Law ergibt somit ein zunehmender Leistungsunterschied zwischen Prozessoren und IO-Systemen eine steigende Abhängigkeit der System-Performance von der IO-Leistung. Weiterhin steigt der Bedarf an Speicherkapazität, teilweise begünstigt durch den stetig sinkenden Preis pro Megabyte. Als Beispiel dienen hierzu Business Warehouse-Systeme, die Daten von bis zu mehreren TeraBytes verwalten. Es wurden somit in den letzten 20 Jahren Konzepte und Architekturen für IO-Systeme entwickelt, die zum einen große Datenmengen zur Verfügung stellen können und zum anderen gegenüber herkömmlichen Festplatten bedeutende Performance-Gewinne hinsichtlich Datenübertragung und Zugriffszeit bieten. Entstanden sind hierdurch auf Festplatten basierende IO-Architekturen, wie z. B. RAID (Redundant Array of Inexpensive Disks), NAS (Network Attached Storage) und SAN (Storage Area Network).

Insbesondere durch die Einführung von mySAP.com und der damit verbundenen Produkte, wie z. B. das Business Information Warehouse, wachsen auch die Speicheranforderungen im Bereich von SAP. Die Unternehmensdaten, d. h. beispielsweise Kundentabellen, Verkaufszahlen, Verkaufsaufträge, stellen das Herzstück einer jeden Firma dar. Bei Verlust dieser Daten ist die Existenz der Firma nicht mehr gesichert, so dass Datensicherungen in Form von z. B. Spiegelungen der Daten (RAID-Level 1) durchgeführt werden müssen, die wiederum zu einem erhöhten Speicherbedarf führen. Die Daten eines SAP-Systems werden in einer zentralen Datenbank verwaltet. Bei nahezu jedem Zugriff eines Anwenders auf das R/3-System resultieren ein

oder mehrere Datenbankzugriffe, die dann in Abhängigkeit des Datenbank-Cache IO-Aufträge generieren. Die Antwortzeit pro Transaktion in einem R/3-System steht somit neben der Prozessorleistung der Server in starker Abhängigkeit zur Datenbankperformance und der damit verbundenen IO-Performance.

Aufgrund der beschriebenen Probleme und Entwicklungen im Bereich der Storage-Systeme, gewinnt die Performance-Analyse und -Prognose von IO-Systemen stetig an Bedeutung, so dass die Modellierung und Analyse von Storage-Systemen, und zwar Festplatten, einen weiteren Schwerpunkt dieser Arbeit bilden. Eine Betrachtung weiterer IO-Architekturen (RAID, NAS, SAN) würde den Rahmen dieser Arbeit überschreiten. Es werden Konzepte zur Lastbeschreibung der IO-Aktivität in Abhängigkeit der Last in einem SAP R/3-System sowie ein analytisches Festplattenmodell erarbeitet. Als Anforderung an das Modell soll dieses die Eigenschaften aktueller Festplatten abbilden können, d. h. insbesondere Caching-Strategien für Lese- und Schreibzugriffe sowie Tagged Command Queueing. Ziel ist es hierbei, mit Hilfe der IO-Lastcharakterisierung für SAP R/3 und des zu erarbeitenden Modells, eine IO-Prognose für die SAP R/3-Kapazitätsplanung zu entwickeln.

Bisher erschienene Arbeiten (wie die zuletzt publizierte Dissertation von Elisabeth Shriver [99]), die eine analytische Modellierung von Festplatten behandeln, validieren ihre Modelle auf Basis von Simulationsergebnissen. Die Simulationen werden hierbei mit Hilfe von Traces realer IO-Messungen durchgeführt, wobei jedoch die notwendigen realen Messungen mittels Monitoring-Werkzeugen unter Verwendung von Zusatzinstrumentierungen des Systems überwacht werden. Die Schwierigkeit der Verknüpfung von realen Messungen und analytischen Modellen unterstreicht Shriver in ihrer Arbeit, indem sie eine Validation ihres Modells auf Basis von realen Messungen für verschiedene Festplatten für nicht realisierbar hält, da hierzu jeweils Messinstrumentarien benötigt werden, die nicht bei jeder Festplatte anwendbar sind. Ein realer Einsatz in einem Kapazitätsplanungsprojekt ist für derartige Modelle somit nicht denkbar.

Eine weitere wesentliche Forderung an das in dieser Arbeit zu erstellende Festplattenmodell ist somit der Bezug zu realen Messdaten sowie zum realen System. Das Modell soll anhand von realen Messdaten validiert werden und weiterhin neben den Festplattenparametern Einflussgrößen des Betriebssystems auf die IO-Verarbeitung mit berücksichtigen können. Als Betriebssystem wird Solaris von Sun Microsystems verwendet, das auch in aktuellen R/3-Konfigurationen häufig Verwendung findet. Die Validation des Modells wird mit Hilfe verschiedener Lastszenarien vorgenommen, die mittels eines IO-Benchmarks generiert und mittels eines frei verfügbaren Monitoring-Werkzeugs, d. h. ohne Zusatzinstrumentierung, gemessen werden. Weiterhin werden die Hardware-Beschreibungen der Festplatte allein auf Basis von frei erhältlichen Produktblättern vollzogen, so dass der praktische Einsatz der in dieser Arbeit vorzustellenden Festplattenmodellierung gewährleistet ist.

1.3. Gegenstand und Ziel der Arbeit

Die Arbeit besitzt die beiden Themenschwerpunkte SAP R/3-Kapazitätsplanung und Modellierung von Storage-Systemen. Die für diese beiden Themen definierten Ziele der Arbeit sind wie folgt:

- Die Vorstellung eines Vorgehensmodells zur SAP R/3-Kapazitätsplanung, wobei insbesondere die Entwicklung von Konzepten, Methoden und Werkzeugen zur Lastcharakterisierung, Lastanalyse und Modellierung im Vordergrund stehen.
- Die Modellierung von Festplatten auf Basis realer Messungen unter Verwendung von Standard-Monitoring-Werkzeugen.

- Zur Einbindung des Festplattenmodells in die SAP R/3-Kapazitätsplanung wird ein Konzept zur IO-Lastcharakterisierung für den Anwendungsfall SAP R/3 erarbeitet, d. h. es werden Korrelationen zwischen R/3-Aktivitäten und auf Betriebssystemebene gemessenen IO-Aktivitäten untersucht.

1.4. Gliederung der Arbeit

In Kapitel 2 werden neben einer Einführung in die Technologie von SAP R/3-Systemen das Vorgehensmodell sowie Konzepte, Methoden und Werkzeuge für die SAP R/3-Kapazitätsplanung beschrieben. Insbesondere wird hierbei auf das Modellierungswerkzeug WLPSizer eingegangen, das zur Modellierung der R/3-Systeme verwendet wird. Das Kapitel wird mit zum Teil aus realen Projekten stammenden Fallstudien und Beispiele beschlossen.

In Kapitel 3 wird neben einer Beschreibung derzeitiger Konzepte und Verfahren zur IO-Prognose und zum IO-Sizing, ein Konzept zur IO-Lastcharakterisierung in Korrelation zur R/3-Last vorgestellt, welches dann für das in den Kapiteln 4 und 5 dargestellte Festplattenmodell verwendet wird.

Kapitel 4 beinhaltet neben einer Einführung in die Festplattentechnologie die für das Festplattenmodell verwendeten IO-Lastszenarien und zugehörigen Messergebnisse sowie die verwendeten IO-Monitoring- und IO-Benchmarking-Werkzeuge.

In Kapitel 5 wird das Festplattenmodell sowie die Modellvalidation anhand der in Kapitel 4 beschriebenen IO-Lastszenarien dargestellt.

Kapitel 6 beinhaltet eine Zusammenfassung der in dieser Arbeit erzielten Ergebnisse sowie einen Ausblick auf weitere auf dieser Arbeit aufbauende Themen.

KAPITEL

2

DIE SAP R/3-KAPAZITÄTSPLANUNG

In diesem Kapitel wird die SAP R/3-Kapazitätsplanung beschrieben. Es folgen zunächst einführende Beschreibungen und Grundbegriffe für SAP R/3 und mySAP.com. Nachfolgend werden bereits existierende und in der Industrie verwendete Verfahren zur Leistungsbewertung und -prognose im Bereich SAP R/3 dargestellt. Anschließend wird das im Rahmen dieser Arbeit vorzustellende Vorgehensmodell zur R/3-Kapazitätsplanung dargelegt. Die beiden Hauptsäulen der Kapazitätsplanung, Daten-Analyse und Modellierung, werden in den darauf folgenden Kapiteln detailliert beschrieben. Das Kapitel wird mit verschiedenen Beispielen und Fallstudien, die sich teilweise aus der Projektarbeit ergeben haben, abgeschlossen.

2.1. Einführung in SAP R/3 und mySAP.com

In den folgenden Abschnitten werden die Architekturen und Funktionsweisen von SAP R/3 und mySAP.com aus technischer Sicht präsentiert. Dieses Kapitel dient dazu, ein Grundverständnis zur Funktionsweise eines SAP-Systems zu schaffen sowie eine Basis an SAP-Grundbegriffen zu vermitteln, die nachfolgend in allen weiteren Kapiteln verwendet werden. Eine detaillierte technische Beschreibung der Funktionsweise eines SAP R/3-Systems befindet sich in [11] und [12].

2.1.1. SAP R/3 - Grundlagen

SAP ist eine integrierte, branchenneutrale Standardsoftware, die alle betriebswirtschaftlichen Anwendungsbereiche eines Unternehmens abdeckt, integriert und verbindet. Die Funktionen eines R/3-Systems sind auf verschiedene Module (Business Components) aufgeteilt, die jeweils den entsprechenden betriebswirtschaftlichen Bereichen eines Unternehmens, wie Rechnungswesen, Logistik, Personalwesen etc., zugeordnet sind. Man spricht hierbei auch von Hauptmodulen, da diese wiederum aus Submodulen bestehen. Beispielsweise setzt sich das Hauptmodul Logistik aus den Submodulen Allgemeine Logisitik, Materialwirtschaft, Instandhaltung, Vertrieb, Produktionsplanung und -steuerung zusammen [55]. Aus Modulsicht kann der schematische Aufbau von SAP R/3 wie in Abbildung 2.1 dargestellt werden.

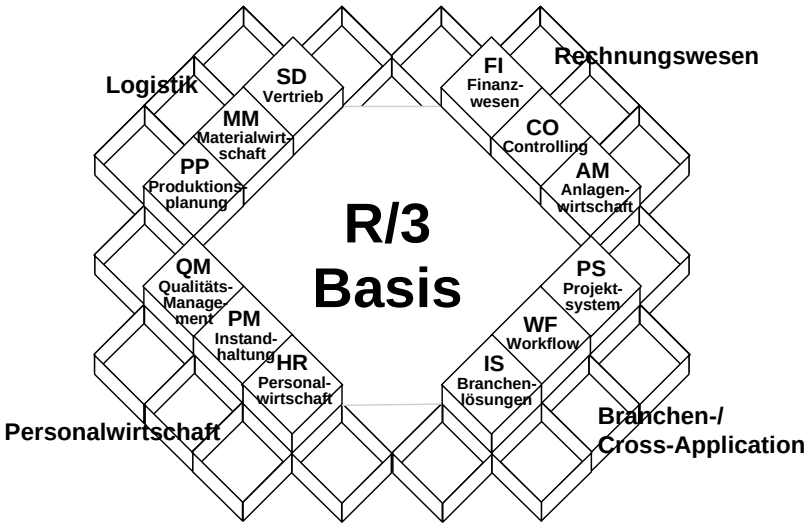


Abbildung 2.1 Das SAP R/3-Integrationsmodell

Neben den bereits genannten Bereichen Logistik, Rechnungswesen und Personalwirtschaft existiert der Bereich Branchen-/Cross-Application, welcher u. a. modifizierte und erweiterte R/3-Standard-Komponenten umfasst, die eine Vielzahl an industriespezifischen Lösungen (Industry Solutions, IS) repräsentieren. Weiterhin werden in diesem Bereich Lösungen für das Worklflow und Project-Management angeboten. Eine umfassende Beschreibung der verschiedenen R/3-Module befindet sich in [55].

Alle vier Bereiche in Abbildung 2.1 bilden eine Wabe, welche die R/3-Basis-Komponente umschließt. Unter R/3-Basis versteht man das jedem R/3-System zugrundeliegende Basissystem, das u. a. die folgenden Komponenten enthält:

- *Computer Center Management System (CCMS)* zur Verwaltung von SAP- und Datenbanksystemen und technischen Systemadministration
- *Berechtigungskonzept* zur Implementierung des Datenschutzes
- *Workflow-Management* zur Organisation effizienter Arbeitsabläufe
- *Basis-Reporting* zur Erstellung von Auswertungen und Berichten
- *ABAP/4 Development Workbench* zur Entwicklung von SAP-Anwendungen

Im Hinblick auf die R/3-Kapazitätsplanung, insbesondere bzgl. des Monitoring, der Daten-Messung und -Analyse, bietet das CCMS der R/3-Basis die folgenden Funktionen und Möglicheiten:

- *Systemüberwachung*, d. h. relevante Leistungsdaten der R/3-Installationen werden gesammelt (Workload Management) und alphanumerisch oder graphisch dargestellt (Performance Management). Bei Bedarf werden auch Alarmmeldungen generiert (Monitoring und Alert Management).
- *Systemsteuerung* zur Regelung der administrativen Abläufe innerhalb von R/3-Installationen. Eine wichtige Komponente der Systemsteuerung ist die Lastverteilung (Load Balancing). Es existieren hierzu drei Dienste: Lastbalancierung für Dialog-Arbeitsgruppen (vordefinierte Anwendergruppen verteilt über mehrere Appl.-Server in Abhängigkeit der Auslastung der Server; Logon Balancing), Hintergrundverarbeitung (Verteilung über mehrere Hintergund-Server oder Hintergrund-Workprozesse) und Ausgabe-Aufträge (gleichmäßige Verteilung auf Spool-Server oder sonstige Ausgabegeräte).

- *Management-Schnittstellen* zur Integration externer System-Management-Dienste (Application Programming Interfaces). Hierbei ist insbesondere die Datenkollektor-Schnittstelle für den Export leistungsrelevanter Daten von besonderem Interesse, da im R/3 -System z. B. die R/3-Statistik-Sätze in vorgegebenen Zeitabständen überschrieben werden.

Die Systemüberwachung des CCMS basiert auf den Statistical Records des R/3-Systems. Alle verarbeiteten Dialogschritte werden in den Statistical Records protokolliert und festgehalten. Als Dialogschritt wird das Vorbereiten und anschließende Senden der Eingabemaske zum Bildschirm des Benutzers sowie die Entgegennahme des vom Benutzer eingegebenen Bildschirminhalts und die Analyse und Verarbeitung der im empfangenen Bild enthaltenen Eingabedaten definiert (siehe auch SAP R/3-Grundbegriffe). Jedem Dialogschritt sind in den Statistical Records über 90 Attribute zugeordnet, wie z. B. System, Instanz, Antwortzeit, CPU-Zeit-Verbrauch und Datenbank-Verweilzeit. Eine detaillierte Beschreibung befindet sich in den nachfolgenden Kapiteln.

Wie bereits beschrieben ist das CCMS standardmäßig in der R/3-Basis-Komponente enthalten. Die System-Überwachung begrenzt sich jedoch auf das SAP-System und die darin verwendeten Datenbanksysteme. Für eine anwendungs- und systemübergreifende Sicht auf die IT-Infrastruktur ist das CCMS somit nicht ausreichend und es werden weitere Softwaresysteme von z. B. Siemens, BMC oder Mercury Interacive zur ganzheitlichen Überwachung des Systems benötigt (siehe Kapitel 2.4.2.).

2.1.1.1. Architektur von SAP R/3

Die Softwarearchitektur von SAP R/3 entspricht dem Client/Server-Prinzip. Es werden sowohl zentralisierte Konfigurationen als auch verteilte Systeme mit dezidierten Servern unterstützt. Jede R/3-Konfiguration besteht aus den folgenden drei Komponenten:

- *Präsentationsserver (Frontends):* Sie dienen der Datenein- und -ausgabe und werden in der Regel als PCs realisiert. Die Präsentationskomponente bildet die Schnittstelle zwischen Anwender (User) und dem R/3-System.
- *Applikationsserver:* Alle R/3-Module eines R/3-Systems können über eine Vielzahl von Aplikationsservern installiert werden. Die Verarbeitung der Geschäftsprozesse bzw. Transaktionen geschieht somit über die Applikationsserver. Die System-Architektur von SAP R/3 ist derart ausgelegt, dass eine beliebige Anzahl an Applikationsservern eingesetzt werden kann, so dass bei steigendem Lastaufkommen die Rechenleistung beliebig skalierbar ist.
- *Datenbankserver:* Er dient der Speicherung und Bereitstellung aller Daten eines R/3-Systems. Sowohl die R/3-Module als auch die eigentlichen Kundendaten werden in dieser Datenbank gespeichert. Bis auf wenige Ausnahmen existiert in einem R/3-System nur eine zentrale Datenbank. Alle Applikationsserver greifen somit auf dieselbe Datenbank zu. Mit zunehmender Anzahl an Applikationsservern stellt die Datenbank bei jedem SAP R/3-System eine potentielle Schwachstelle dar. Seit längerem gibt es hingegen zertifizierte Datenbanksysteme für SAP (Oracle Parallel Server, IBM DB2 und Microsoft SQL Server Enterprise), die für die Applikationsschicht nicht merklich eine horizontale Verteilung auf mehrere Rechner unterstützen (Database-Cluster). Ein Erfahrungsbericht zu diesem Thema wird von Zeller und Kemper in [144] gegeben.

Abbildung 2.2 demonstriert die möglichen Client/Server-Konfigurationen eines SAP R/3-Systems.

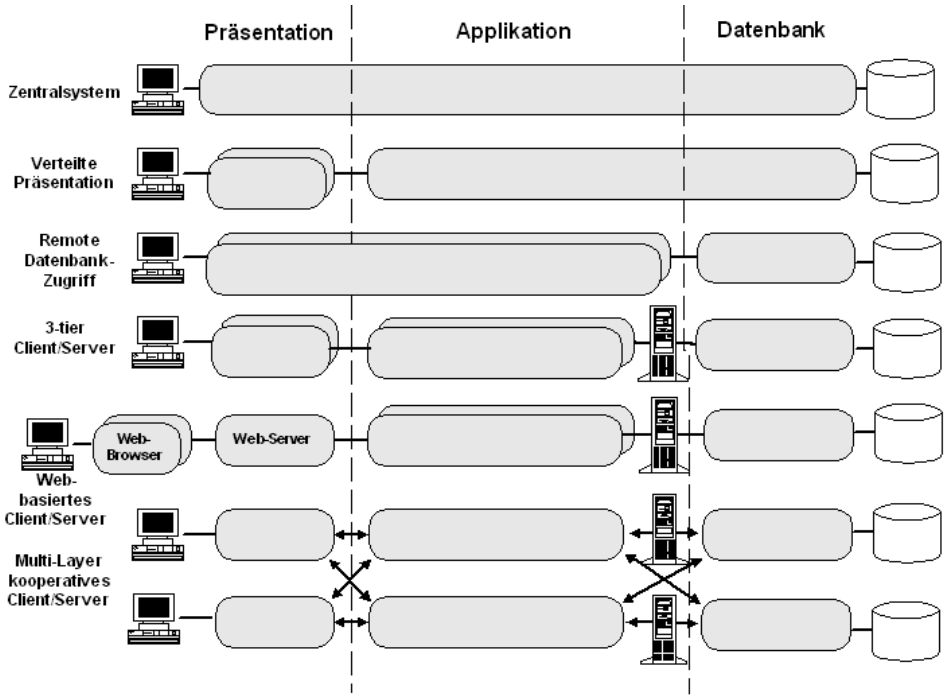


Abbildung 2.2 Client/Server-Kofigurationen für ein SAP R/3-System (Quelle [11])

Das Zentralsystem für kleinere Systeme und die 3-stufige Client/Server-Konfiguration für größere Systeme sind die in der Praxis gebräuchlichsten System-Varianten. Der Unterschied dieser beiden Konfigurationen liegt in der Verteilung der drei Komponenten Präsentation, Applikation und Datenbank auf die Hardware. Beim Zentralsystem werden alle drei Ebenen auf einem Hardware-System betrieben (klassische Großrechnerverarbeitung) und bei 3-stufig wird hingegen pro Ebene ein separates Hardware-System verwendet, wobei die Applikation auf mehrere Server verteilt werden kann.

Die von SAP unterstützten Betriebssystem-Plattformen reichen von IBM AS/400 über MS Windows NT/2000/XP bis hin zu Unix (Reliant Unix, Solaris, HP-UX, Linux etc.). Häufig wird die Präsentations-Ebene unter MS Windows NT/2000/XP betrieben und die Applikations- und Datenbank-Ebene unter Unix.

2.1.1.2. SAP R/3-Grundbegriffe

Es folgen nun einige wenige SAP-Grundbegriffe, die zum Verständnis der Funktionalität und Arbeitsweise eines SAP-Systems beitragen sollen (siehe [85], [12], [92]):

- *Dialogschritt*: Ein Dialogschritt wird durch eine Eingabemaske (Dynpro; Dynamic Program = Maske und Steuerlogik) repräsentiert. Aus R/3-Sicht besteht ein Dialogschritt aus der Verarbeitung der vom Benutzer eingegebenen Daten (PAI=Process After Input) und dem Vorbereiten und Senden der nachfolgenden Bildschirmmaske (PBO=Process Before Output). In Abbildung 2.3 wird die Definition eines Dialogschritts veranschaulicht.

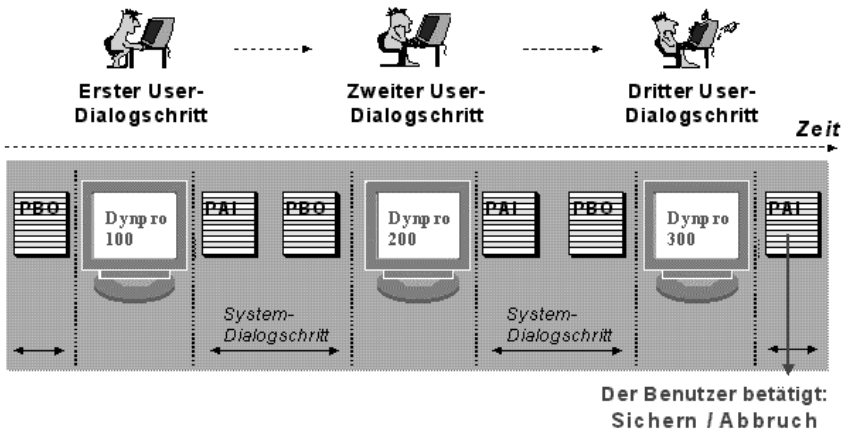


Abbildung 2.3 Definition eines Dialogschritts (Quelle: [85])

- *SAP-Transaktion*: Eine SAP-Transaktion ist eine Folge von betriebswirtchaftlich konsistenten, funktional und logisch zusammenhängenden Dialogschritten. Aus Systemsicht umfasst eine SAP-Transaktion die Gesamtheit aller Dialogschritte einschließlich dazugehöriger Änderungen im Datenbanksystem. Sie ist nicht unterbrechbar und kann wieder in den ursprünglichen Systemzustand zurückversetzt werden.
- *SAP-LUW*: Die Bezeichnung LUW bedeutet „Logical Unit of Work“ und entspricht dem bisher erläuterten SAP-Transaktionsbegriff. Der Unterschied zu Datenbank-LUWs ist bei SAP der prozessübergreifende Transaktionsablauf. Das bedeutet, dass eine SAP-LUW z. B. mehrere DB-Prozesse beinhalten kann. Die Konsistenz-Anforderungen sowie die Forderung nach einem logisch korrekten Zustand vor und nach der Ausführung einer LUW sind bei Datenbank- und SAP-LUWs gleich. In Abbildung 2.4 wird die Verarbeitung einer SAP-LUW demonstriert. Der Beginn einer SAP-Transaktion ist gleichzeitig der Anfang einer SAP-LUW. Beendet wird eine SAP-LUW mittels „Commit Work“ im ABAP-Coding bzw. dem Ende der zugehörigen asynchronen Verbuchung (zweiter Abschnitt der SAP-LUW). Jeder Dialogschritt einer SAP-LUW wird von einem Workprozess ausgeführt. Die Verarbeitungsstrategie für die im zweiten. Teil der SAP-LUW dargestellten Verbuchungen ist vorwiegend asynchron, d.h. die Verbuchungsaufträge werden gesammelt und nach Ende der Dialogphase in getrennten Verbuchungs-Workprozessen ausgeführt.
- *SAP-Sperren (Enqueues)*: In einer SAP-LUW können sich mehrere DB-LUWs befinden, da u. a. die betriebswirtschaftlichen Datenobjekte über mehrere Datenbank-Tabellen verteilt sein können. Somit sind die Sperrmechanismen der Datenbank nicht ausreichend. Aus diesem Grund besitzt SAP R/3 einen eigenen Sperrmechanismus, die sog. Enqueues. Koordiniert werden die Sperren mit dem Enqueue-Prozess, der pro R/3-System einmal existiert. Wie man anhand von Abbildung 2.4 erkennen kann, werden im ersten Abschnitt einer SAP-LUW R/3-Sperren gesetzt und nach der Fortschreibung in die Datenbank im zweiten Teil wieder frei gegeben.

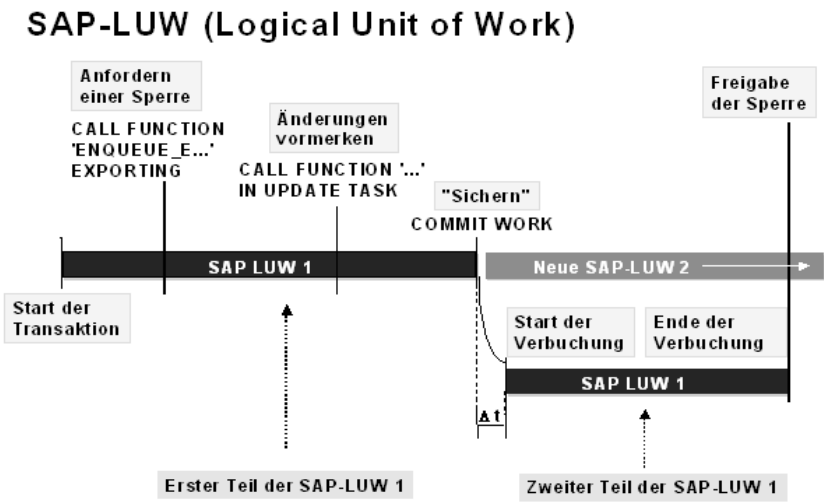


Abbildung 2.4 Definition einer SAP-LUW (Quelle: [85])

- *Workprozesse*: Vergleichbar mit den Prozessen eines Betriebssystems, werden im R/3-System Workprozesse zur Ausführung der Transaktionen bzw. Dialogschritte verwendet. Zu den Eigenschaften von Workprozessen zählen u. a. die Parallelität und Kooperation untereinander. Die Anzahl der Workprozesse ist von den zur Verfügung stehenden Ressourcen sowie den Konfigurations-Parametern abhängig. In einem R/3-System existieren verschiedene Typen von Workprozessen für:

  - *Dialogverarbeitung (Dialog):* Die Dialog-Workprozesse führen die Transaktionen der Benutzer online aus, wobei ein Dialogprozess die Anfragen mehrerer Benutzer bearbeiten kann. Für jede R/3-Instanz existieren immer mindestens zwei Dialog-Workprozesse.

  - *Verbuchung (Update):* Die zu den Dialogtransaktionen gehörenden Dialogprogramme erzeugen einen Protokollsatz für die Datenbankfortschreibung, der nach Abschluss des Dialogabschnitts verarbeitet wird. Die somit erfolgende Aktualisierung der Datenbank bezeichnet man als Verbuchung, die synchron oder asynchron erfolgen kann. Synchron bedeutet hierbei, dass das Verbuchungsprogramm direkt vom Dialog-Workprozess ausgeführt wird und der Dialogbenutzer auf den Abschluss der Verbuchung warten muss, bevor eine neue Eingabe erfolgen kann. Bei der asynchronen Verbuchung arbeiten der Dialogteil und die Fortschreibung der Datenbank in unterschiedlichen entkoppelten Workprozessen. Die Verarbeitung eines Protokollsatzes kann in mehreren Abschnitten (Verbuchungskomponenten) erfolgen. Es wird hierbei zwischen primären (V1) und sekundären (V2) Verbuchungskomponenten unterschieden. Die Bearbeitung der primären Komponenten muss abgeschlossen sein, bevor mit den sekundären Komponenten eines Protokollsatzes begonnen werden kann.

  - *Dialogfreie Hintergundverarbeitung (Batch):* Im Hintergrund werden Aufgaben abgearbeitet, die keine Online-Aktion (Dialog-Verarbeitung) erfordern.

  - *Spooling:* Unterstützung des Drucker-Spooling zur Druck-Auftrags-Verarbeitung.

  Eine gesonderte Funktion besitzen die folgenden Prozesse, die pro R/3-System nur einmal existieren:

  - *Gateway:* Koordination der Kommunikation zwischen den Anwendungen innerhalb des Systems oder mit externen Systemen (z. B. zwischen R/3- und R/2-System).

- *Sperrverwaltung (Enqueue):* Der Enqueue-Prozess verwaltet eine übergeordenete Sperrtabelle, um die Konsistenz auf betriebswirtschaftlicher Ebene über verschiedene DB-Transaktionen hinweg zu gewährleisten.
- *Message-Prozess:* Koordination der Kommunikation zwischen den verschiedenen Anwendungsservern, d. h. beispielsweise bei Logon-Gruppen werden Benutzer durch den Message-Prozess auf dem Anwendungsserver mit der geringsten Auslastung angemeldet.

Die Anzahl der verschiedenen Workprozesse kann im Instanzen-Profil festgelegt werden, so dass z. B. in Abhängigkeit der Tageszeit die Workprozess-Konfiguration geändert werden kann (Dialog-/Batch-Zeitfenster). In Abbildung 2.5 wird der Workprozess im Gesamtzusammenhang eines R/3-Systems dargestellt. Wie man anhand der Abbildung erkennen kann, besteht der Workprozess intern aus je einem Dynpro- und ABAP-Prozessor zur Steuerung der Dynpro-Ablauflogik bzw. Abarbeitung von ABAP-Anweisungen, sowie einer Datenbankschnittstelle. Koordiniert werden diese drei Komponenten mittels eines Task-Handlers, der auch für den Roll-In und Roll-Out zuständig ist, d. h. der User-Context (Daten, die den Benutzer charakterisieren) wird am Anfang eines Dialogschritts zur Verfügung gestellt (Roll-In) und am Ende gespeichert (Roll-Out).

- *SAP-Dispatcher*: Er dient als Steuerungsinstanz zur Verwaltung der Ressourcen für die R/3-Anwendungen und der dazu notwendigen Abstimmung mit dem zugrundeliegenden Betriebssystem. Die Hauptaufgaben des Disptachers umfassen dabei die Verteilung der Transaktionslast auf die Workprozesse, die Anbindung der Präsentations-Ebene und die Organisation von Kommunkationsvorgängen. Bei der Verteilung der Transaktionslast auf freie Workprozesse, werden die Arbeitsaufträge vom Dispatcher zunächst in einer Queue (Dispatcher-Queue) gespeichert und dann abgearbeitet. Pro R/3-Instanz existiert immer nur ein zentraler SAP-Dispatcher. Bei mehreren R/3-Instanzen dient ein sog. Message-Server zur Kommunikation zwischen den verteilten Dispatchern. Der Message-Server existiert nur einmal in einem R/3-System.

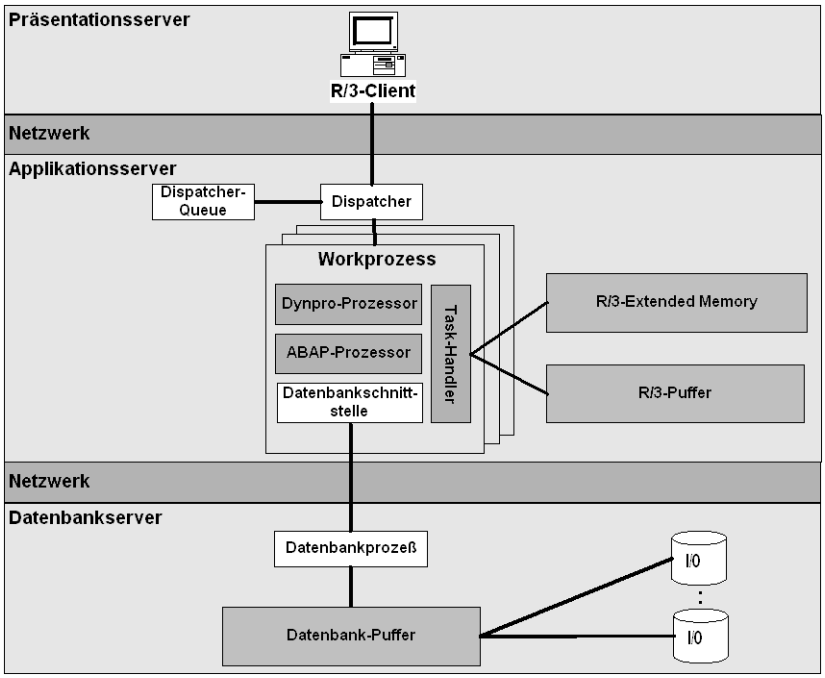


Abbildung 2.5 Grundkomponenten eines SAP R/3-Systems

- *R/3-Instanz*: Hierbei handelt es sich um eine administrative Einheit, welche diejenigen Komponenten eines R/3-Systems zusammenfasst, die einen oder mehrere Dienste anbieten. Wie bereits für Workprozesse beschrieben, werden die Komponenten einer R/3-Instanz über ein Instanzenprofil parametrisiert. Eine R/3-Instanz besitzt zudem eigene Prozesse und Pufferbereiche sowie einen zentralen SAP-Dispatcher.
- *R/3-System*: Die Installation von SAP R/3 mit allen Instanzen nennt man R/3-System. Es besteht immer aus einer DB-Instanz und einer oder mehrer R/3-Instanzen.
- *Mandant*: Dieser ist eine organisatorisch selbstständige Einheit im R/3-System (Unternehmung in einem Konzern) mit eigenem Datenumfeld und demzufolge eigenen Stamm- und Bewegungsdaten, zugeordneten Benutzerstämmen, Kontenplänen und spezifischen Customizing-Parametern. Benutzer unterschiedlicher Mandanten koexistieren mit ihren Daten unabhängig und isoliert. Der Mandant bildet die höchste Hierarchieebene im R/3-System (Mandantenebene = Ebene des Konzerns).

2.1.2. mySAP.com - Grundlagen

Mit mySAP.com verlagert die SAP mit der Jahrtausendwende den Schwerpunkt ihrer Geschäftstätigkeit ins Internet und bietet umfassende Softwarelösungen für die integrierte überbetriebliche Zusammenarbeit im Internet an. mySAP.com umfasst das vollständige Softwareportfolio der SAP einschließlich der Lösungen für Personalwirtschaft, Rechnungswesen, Customer Relationship Management, Supply Chain Management, Business Intelligence und Product Lifecycle Management.

mySAP.com ist die Weiterentwicklung von SAP R/3 und tritt dessen Nachfolge an. Sowohl in der Architektur als auch im Paradigma hat sich mySAP.com stark von SAP R/3 abgegrenzt. Das E-Business steht nun im Mittelpunkt von mySAP.com. Nach [93] gab es im SAP R/3 Umfeld hoch spezialisierte Benutzer, die von festen Arbeitsplätzen über installierte SAP GUIs auf das ERP-System zugriffen. Durch Internet und Intranet wird diese Art von Endbenutzer hingegen „überflüssig“. Mitarbeiter eines Unternehmens besitzen direkt über ihren Browser Zugriff auf das ERP-System. Auch Kunden können direkt via Internet Produkte eines Unternehmens bestellen. Die Lösungen zu diesen Anforderungen soll mySAP.com als E-Business-Plattform liefern. In diesem Kapitel wird mySAP.com und seine Architektur in Kurzform beschrieben. Für eine detaillierte Beschreibung siehe [93] und [27].

Einer der wesentlichen Unterschiede von mySAP.com zu SAP R/3 ist das für die Abbildung der Geschäftprozesse verwendete SAP System, das sich nicht mehr wie bei R/3 auf ein System mit der R/3-Basis als Grundlage beschränkt (siehe Abbildung 2.1), sondern sich über verschiedene Softwarekomponenten erstreckt.

Die mySAP.com-Software kann zum einen in diejenigen Komponenten unterteilt werden, die auf dem SAP Web Application Server (früher SAP-Basis) basieren (d. h. nach [93] die Development Workbench und die Basiswerkzeuge, wie z. B. das CCMS) sowie eine eigene Datenbank besitzen. Zusätzlich zu dem SAP Web Application Server werden anwendungsspezifische Software-Fragmente ergänzt und es ergeben sich die folgenden Systeme:

- SAP R/3 Enterprise
- SAP Business Warehouse (BW)
- SAP Strategic Enterprise Management (SEM)
- SAP Advanced Planner and Optimizer (APO)
- SAP Customer Relationship Management (CRM)
- SAP Enterprise Buyer Professional (EBP)

- SAP Workplace

Zusätzlich zu den beschriebenen Softwarekomponenten existieren andererseits in einem mySAP.com-System eigenständige Anwendungen ohne SAP Web Application Server:

- SAP Internet Transaction Server (ITS)
- SAP liveCache
- SAP Business Connector (BC)

Ein mySAP.com-System ist somit ein Zusammenspiel der verschiedenen installierten und autark agierenden Software-Komponenten. Die Kommunikation verläuft via Remote Function Calls (RFC) und die Daten selbst werden als IDocs versendet (siehe [76]). Das klassiche SAP R/3-System ist eine „Unterkomponente“ eines mySAP.com-Systems und kann in diesem mehrfach vertreten sein. Die Komplexität eines solchen Systems ist somit ungleich höher als die eines SAP R/3-Systems. Ein Beispiel für eine mySAP.com-Lösung wird in Abbildung 2.6 dargestellt.

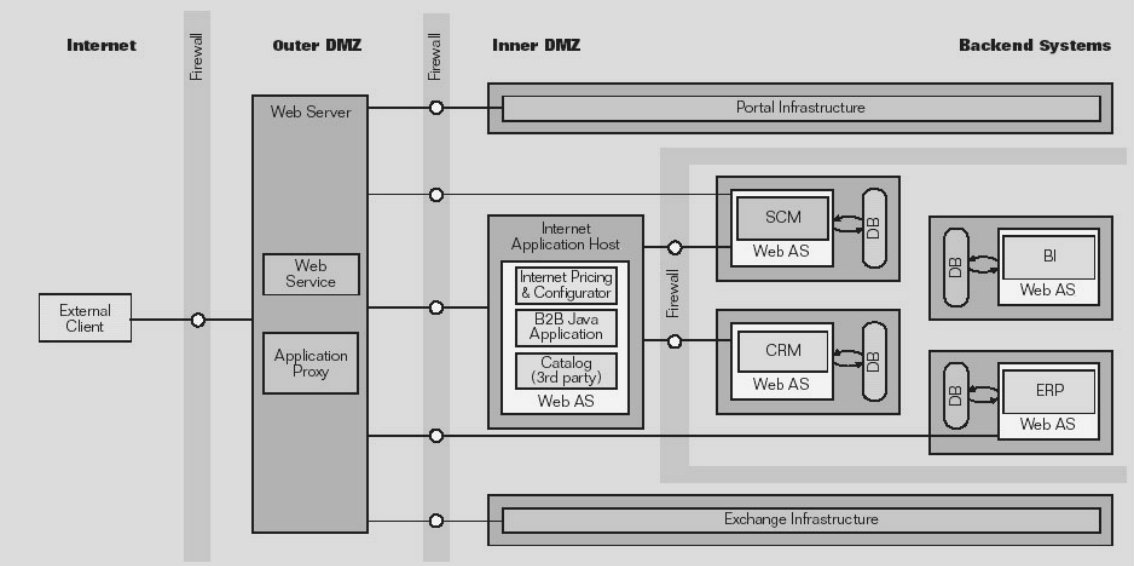


Abbildung 2.6 Beispiel für eine mySAP.com-Konfiguration

Die Anmeldung an das System kann über drei verschiedene Arten erfolgen:

- SAP GUI for Windows, SAP GUI for Java, SAP GUI for HTML
- Web-Browser
- Mobile Clients (Laptop oder Handheld)

In einem mySAP.com-System existieren unter Umständen eine Vielzahl autarker Systeme. Eine Schwierigkeit bei der Administration eines solchen Systems ist die Benutzerverwaltung. Für diese Aufgabe existiert der Workplace-Server. Ein Benutzer muss sich nur einmal im Workplace anmelden und hat dann via einer eigenen Portal-Seite Zugriff auf die von ihm benötigten Systeme.

Die Kommunikation der Benutzer mit den Applikationsservern kann im Gegensatz zum klassichen SAP R/3 über die folgenden drei Arten verlaufen:

- SAP Internet Transaction Server (ITS) und einen Web-Server

- Business Server Pages (BSP), d.h. mit ABAP oder JavaScript werden HTML-Seiten dynamisch generiert
- JSP-Server (Java Server Pages) oder Java Application Server

2.2. Methoden und Vorgehensweisen zur Leistungsbewertung und -prognose für SAP R/3

Die in der Praxis gebräuchliste Methode zur Leistungsprognose für SAP R/3 ist das Sizing. Für verschiedene Szenarien wie z. B. Release-Wechsel oder Hardware-Upgrades existieren entsprechende Sizing-Prozesse, die größtenteils auf Benchmark-Ergebnissen beruhen. Neben dem Sizing existieren weitere Vorgehensweisen für die Leistungsprognose, wie z. B. Lasttests oder die Modellierung. Sowohl SAP-Benchmarking, als auch Sizing und Lasttests werden in den folgenden Abschnitten detailliert beschrieben.

Abschnitt 2.2.4. beschreibt die in verschiedenen Arbeiten bereits publizierten Vorgehensmodelle und Methoden zur Messung und Modellierung von SAP R/3-Systemen. Werkzeuge wie z. B. von BMC oder Mercury Interactive, welche die Datenanalyse und Modellierung unterstützen, werden in Abschnitt 2.4 beschrieben.

2.2.1. SAP-Benchmarking

Benchmarks können in Standard- und Applikationsbenchmarks unterteilt werden. Als Standardbenchmarks versteht man z. B. TPC-C oder SPECint, die auf der Definition einer synthetischen Last basieren. Sie werden hauptsächlich zum Hardware-Vergleich herangezogen. Die Applikationsbenchmarks hingegen basieren auf realen Anwendungen und somit auf real existierenden Lasten. Die folgenden Ausführungen zum SAP-Benchmarking basieren auf [130] und [60].

Die SAP-Benchmarks decken beide Gruppen von Benchmarks ab. Zum einen existieren aufbauend auf verschiedenen Modulen von SAP R/3 bereits definierte Benchmark-Szenarien und zum anderen hat sich der SAP-Benchmark zu einem Standard entwickelt, der von den Hardware-Partnern zum Leistungsvergleich ihrer Hardware-Systeme herangezogen wird (siehe SAPS-Definition). Für einen Leistungsvergleich sind 100% reproduzierbare Messungen notwendig, die im SAP Standard Application Benchmark mittels vorgegebener betriebswirtschaftlich sinnvoller Abläufe und einem definierten Customizing auf festgelegten Stammdatenbeständen ermöglicht werden.

Ein wesentlicher Vorteil der SAP-Benchmarks ist ihre Integration in die Applikation SAP R/3. In [25] wird die Implementierung eines TPC-D-Benchmarks unter SAP R/3 beschrieben. Die dabei erzielten Ergebnisse werden mit den Standard-TPC-D-Ergebnissen verglichen, die durch das Benchmarking der Datenbank ohne SAP R/3 erzielt wurden. Die gemessenen Laufzeiten der verschiedenen Benchmark-Transaktionen unter SAP R/3 lagen im Vergleich zum Standard-TPC-D-Benchmark um ein Vielfaches höher. Die Aussagekraft eines TPC-D-Benchmarks ohne Implementierung in eine reale Anwendung ist somit für die Performance-Beurteilung eines Gesamtsystems fraglich und motiviert die Verwendung von SAP Standard Benchmarks für die Leistungsbewertung von R/3-Systemen.

Die Durchführung der SAP-Benchmarks geschieht in der Regel durch die SAP-Hardware-Partner. Zum einen können hierdurch verschiedene Hardware-Systeme mit den den Benchmarks zugrundeliegenden Business Components getestet und analysiert und die aus den Analysen gewonnenen Erfahrungswerte als Eingabe-Parameter für das Sizing genutzt werden. Zum anderen führen die Hardware-Partner im Wettstreit untereinander Benchmarks zum Leistungsver-

gleich ihrer jeweiligen Hardware-Systeme durch. Die hierbei verwendeten Benchmarks müssen jedoch zur Überprüfung der korrekten Durchführung von der SAP zertifiziert werden. Die erzielten Ergebnisse sind dann öffentlich zugänglich (siehe [88], [89], [50]).

Die SAP Standard Application Benchmarks definieren für eine Vielzahl der SAP Business Components (siehe [86], [87]) Benchmark-Messungen. Tabelle 2.1 enthält alle Komponenten, zu denen SAP-Benchmarks existieren (Stand Dezember 2000, [54]).

• **mySAP Supply Chain Management**	• **mySAP Human Resources**
Materials Management (MM)	Cross Application Time Sheets (CATS)
Sales & Distribution (SD)	Payroll
Production Planning (PP)	• **mySAP Financials**
Warehouse Management (WM)	Financial Accounting (FI)
Assemble-To-Order (ATO)	• **Industry Solutions**
Advanced Planning and Optimization (APO)	mySAP Retail
• **mySAP Product Lifecycle Management**	mySAP Banking (BCA)
Project System (PS)	mySAP Utilities
• **mySAP Business Intelligence**	• **mySAP Customer Relationship Mgmt**
Business Information Warehouse	Internet Sales (ISA)
• **E-Commerce**	Mobile Sales (MSA)
Online-Store	Customer Interaction Center (CIC)

Tabelle 2.1 SAP-Komponenten, die zugehörige SAP-Benchmarks besitzen

Der bekannteste Benchmark ist der auf dem SD-Modul basierende, da dieser zur Ermittlung der SAPS-Leistungen verwendet wird. Nachfolgend wird als Beispiel die Funktionsweise des SD-Benchmarks beschrieben.

2.2.1.1. Ablauf eines SD-Benchmarks

Der SD-Benchmark ist ein dialogorientierter Benchmark, d. h. es werden Dialogbenutzer simuliert, die in einer Schleife mehrmals hintereinander Dialogschritte ausführen. Aus betriebswirtschaftlicher Sicht werden die folgenden Transaktionen durchgeführt:

- Anlegen eines Auftrags mit 5 Positionen (Line Items)
- Erzeugen einer Lieferung für die angelegte Position
- Anzeigen der Kunden-Bestellung
- Änderung der Lieferung und Senden der Waren-Ausgabe
- Anzeigen von 40 Bestellungen für einen Kunden
- Erstellung einer Rechnung

Es wird somit vom Auftragseingang bis zur Rechnungsschreibung eine vollständige Bearbeitung simuliert. Aus technischer Sicht werden 15 Dialogschritte verteilt auf sechs R/3-Transaktionen verarbeitet, wobei vier Transaktionen eine Verbuchung beinhalten. Zwischen zwei Dialogschritten existiert jeweils eine Denkpause von 10 Sekunden.

Die mittlere Antwortzeit muss nach Benchmark-Definition unter zwei Sekunden liegen. Als Antwortzeit wird diejenige Zeit bezeichnet, die zur vollständigen Verarbeitung eines Dialogschritts vom System benötigt wird. Unter dieser Vorgabe wird die maximale Anzahl paralleler Benutzer mit dem bereits beschriebenen Anwenderverhalten ermittelt (siehe Abbildung 2.7). Der Zeitraum mit der Maximalanzahl an parallelen Benutzern bezeichnet man als Hochlastphase, die nach Benchmark-Definition einen Zeittraum von mindestens 15 Minuten umfassen muss. Die resultierenden Benchmark-Ergebnisse (Durchsatz und Antwortzeit) beziehen sich auf die Hochlastphase.

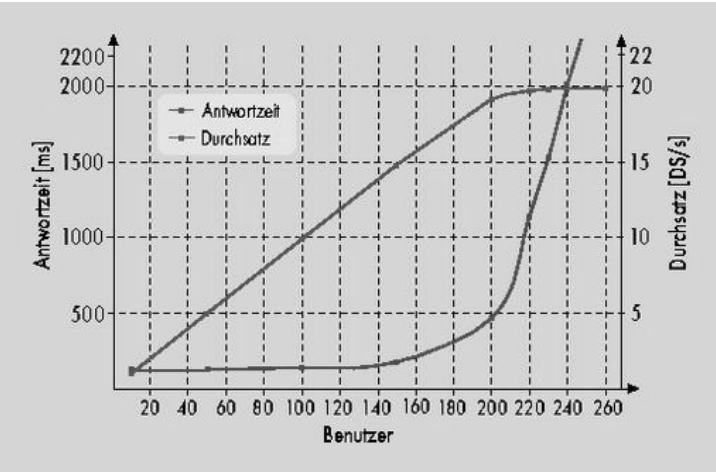


Abbildung 2.7 Antwortzeit vs. Durchsatz beim SD-Benchmark

Der Benchmark liefert die folgenden Ergebnisse:

- *Anzahl der Benutzer:* Sie entspricht der Anzahl parallel gestarteter GUIs und steht somit bereits beim Start des Benchmarks fest.
- *Antwortzeit:* Mittelwert der Antwortzeiten aller Dialogschritte mit Tasktyp Dialog
- *Durchsatz:* Er entspricht der Anzahl Dialogschritte pro Sekunde. Aufgrund der Benchmarkdefinition kann man von der Dialogschrittrate auf die Transaktionen und verarbeiteten Auftragspositionen pro Stunde schliessen. Speziell im SD-Benchmark kann anhand der Angabe Dialogschritte/Sekunde auch das Leistungsmaß SAPS berechnet werden (siehe 3.1.3).
- *Auslastung:* CPU-Auslastung der Server
- *Plattenplatz:* Insgesamt benötigter Plattenplatz

Zusätzlich zu den genannten Ergebnissen werden die Konfigurationsbeschreibungen und Versionen der SAP-, Datenbank- und Betriebssystem-Software genannt.

Als Beispiel für einen SD-Benchmark soll der Weltrekord von Fujitsu Siemens Computers mit 100.000 SAPS bei einer 3-stufigen Konfiguration vom März 2001 dienen [137]. Die 100.000 SAPS wurden durch 23.000 simulierte SD-User bei einer mittleren Antwortzeit von 1,73 Sekunden erreicht. Als Datenbankserver war eine PRIMEPOWER 2000 mit 64 CPUs (450 MHz und 8 MB Second Level Cache) installiert und über 8 FibreChannel-Controller mit einer EMC^2 Symmetrix 8730-36 als IO-System verbunden. Die Anzahl der Applikationsserver belief sich auf 160 mit jeweils 4 CPUs. Der Benchmark lief unter dem R/3-Release 4.6B.

Im Zeitalter von mySAP.com und der wachsenden Komplexität der SAP-Systeme verliert der SD-Benchmark als der „Standard-Benchmark“ zunehmend an Bedeutung. Der Grund hierfür liegt darin, dass der SD-Benchmark nur einen sehr geringen Ausschnitt der SAP-Funktionalitä-

ten abdeckt und auch nur sehr einfache, auf ein Modul beschränkte Transaktionen beinhaltet. Ein derartiges Lastszenario ist für heutige Systeme mit vielen verschiedenen miteinander agierenden Modulen nicht mehr realitätsgetreu.

Im Gegenzug gewinnt nach [72] der ATO (Assembly To Order) Benchmark immer mehr an Bedeutung und wird dahingehend auch von der SAP unterstützt. Der ATO-Benchmark simuliert die gesamte Lieferkette für z. B. den Autoverkauf unter Beteiligung vieler verschiedener Business Components von mySAP.com (inklusive einer Verfügbarkeitsprüfung). Nach SAP zeichnet sich der ATO-Benchmark durch seine implementierten Eigenschaften (hohe Verkaufsraten, kurze Produktionszeiten und individuelle Bausätze) für jede Bestellung aus. Beteiligte SAP-Komponenten sind Financials, Logistics, Basis, Human Resources sowie Cross Applications. Das Ergebnis eines ATO-Benchmarks beinhaltet Durchsatzzahlen in Form von „Assembly Orders“ pro Stunde. Anhand der verarbeiteten „Assembly Orders“ werden die verschiedenen Hardware-Systeme miteinander verglichen. Stellt man die Benchmarks ATO und SD gegenüber, so entspricht ein ATO-Benchmark-User nach [72] 2,3 SD-Benchmark-Usern.

2.2.1.2. Vergleich von Benchmark-Ergebnissen

Die Durchführung von SAP-Benchmark-Messungen für verschiedene Business Components ermöglicht den Vergleich der verschiedenen SAP-Anwendungen auf Basis ihres Ressourcenverbrauchs. Ein Beispiel hierfür wird in Abbildung 2.8 dargestellt, in welcher der CPU-Verbrauch unterteilt nach Dialog-, Verbuchungs- und Datenbankservice gegenübergestellt wird. Zum besseren Vergleich wurde der Verbrauch von FI auf Eins normiert. Die Abbildung veranschaulicht, welche Unterschiede im Ressourcenverbrauch zwischen z. B. einem FI- und einem PP-Benutzer existieren. Es handelt sich in diesem Beispiel um den Faktor 5, der sich in allen drei Bereichen des CPU-Verbrauchs (Database, Update, Dialog) widerspiegelt. Ergebnisse dieser Art werden beim Sizing für die Rechnerdimensionierung verwendet. Anhand der unterschiedlichen Ressourcenverbräuche der einzelnen Business Components können sog. Lastfaktoren definiert und im Sizing berücksichtigt werden (siehe Kapitel 2.2.2.).

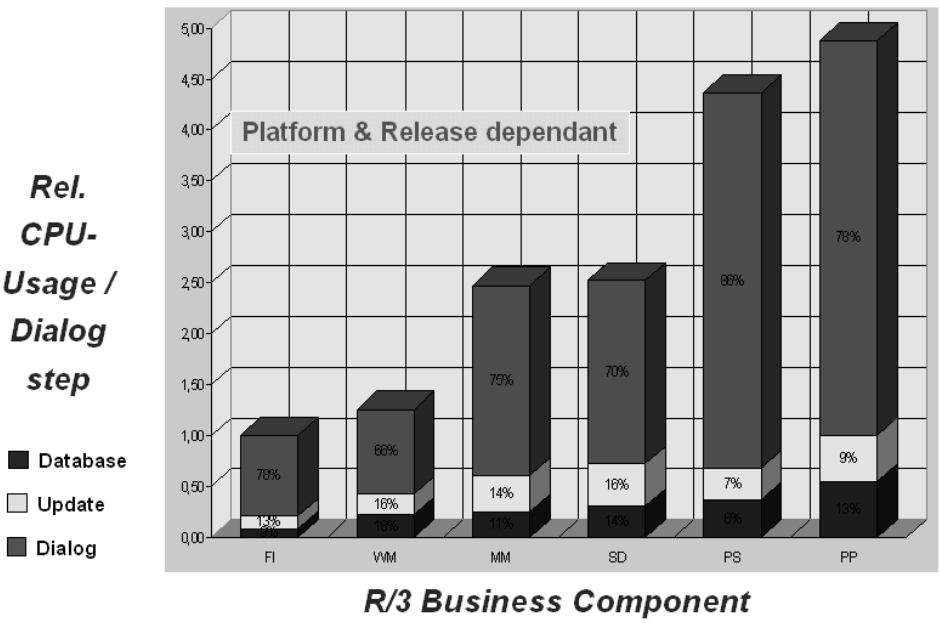


Abbildung 2.8 CPU-Verbrauch pro Business Component

2.2.1.3. SAPS-Definition

SAPS ist ein Leistungsmaß, das zur Beschreibung der Leistungsfähigkeit von Servern verwendet wird. Die Definition von SAPS basiert auf den Durchsatzergebnissen des SD-Benchmarks und bezeichnet somit einen Durchsatzwert analog zu TPM (Transactions Per Minute) beim TPC-C-Benchmark oder SPECint bei SPEC. Nach [72] kann bis zum SAP-Release 4.5 das Leistungsmaß SAPS in die Einheit tpm-C umgerechnet werden. Die Bezeichnung SAPS bedeutet „SAP Applciation Benchmark Performance Standard“ und wird folgendermaßen definiert:

> *„100 SAPS sind definiert als 2000 vollständig verarbeitete Auftragspositionen pro Stunde im Standard SD Application Benchmark. Dieser Durchsatz wird erreicht bei 6.000 Dialogschritten (Bildschirmwechsel) und 2.000 Verbuchungen pro Stunde im SD-Benchmark oder bei der Verarbeitung von 2.400 SAP-Transaktionen.“*

Diese Definition basiert auf dem betriebswirtschaftlichen Prozess und ist somit unabhängig von einer speziellen Konfiguration bzw. der verwendeten Software-Version.

2.2.2. Sizing zur Rechnerdimensionierung für SAP-Systeme

Die Dimensionierung der Hardwaresysteme zur Einführung oder Erweiterung von mySAP.com bzw. SAP R/3 bezeichnet man als Sizing. Anhand von Eingabeparametern wie z. B. zukünftige Anzahl der User, zu installierende Business Components und Business Objects werden die notwendigen CPU-Leistungen sowie Hauptspeicher- und Festplattengrößen ermittelt. Nach Mißbach und Hoffmann [72] besteht das Sizing aus zwei Abschnitten: „der Abschätzung der maximal zu erwartenden Systemlast und der Ermittlung der minimal dafür notwendigen Hardwarekonfiguration“. Angeboten wird das Sizing zum einen von den Hardware-Partnern zur Verkaufsunterstützung (Presales) und zum anderen von der SAP mittels entsprechendem Werkzeug-Angebot (Quicksizer, GoingLive Check, EarlyWatch Service).

Das Sizing basiert auf der Grundlage von Erfahrungen mit anderen Kundenprojekten und auf den Resultaten von Benchmark-Messungen. Nach [72] gibt es drei Verfahren zur Durchführung eines Sizing:

- *Benutzerbasiertes Sizing:* Eingabe der Anzahl und des Aktivitätsgrads der Benutzer für die verschiedenen SAP-Anwendungen. Der Aktivitätsgrad besagt, wieviele Dialogschritte pro Stunde ein Benutzer an das SAP System sendet. Auf Grundlage dessen werden CPU- und Speicherverbrauch pro Anwendung berechnet. Die unterschiedlichen Business Components von SAP werden mittels sog. Lastfaktoren, die anhand von Benchmark-Vergleichsmessungen erzielt wurden, berücksichtigt (siehe Kapitel 2.2.1.2.). Aufgrund der geringen Eingabeparameter stützt sich das benutzerbasierte Sizing auf eine Vielzahl von Erfahrungswerten. Das zu betrachtende SAP-System sollte somit aufgrund von Eigenentwicklung und Customizing nur in geringem Ausmaß vom SAP-Standard abweichen.
- *Transaktionsbasiertes Sizing:* Als Eingabeparameter werden hierbei die Anzahl der zu verarbeitenden Geschäftsprozesse (Transaktionen) übergeben. Das somit zu spezifizierende Mengengerüst enthält für jede Business Component die Anzahl und Größe in Form von Line Items der wesentlichen Transaktionen und beschreibt so z. B. die Anzahl von Kundenaufträgen, Lieferungen und Produktionsaufträgen. Im Gegensatz zum benutzerbasierten Sizing können hierbei auch Batch-Jobs und Datenübernahmen via ALE und RFC sowie tageszeitabhängige Lastverteilungen („M-Kurve“) berücksichtigt werden. Es ist zu beachten, dass Business Components mit aktivierten Customizing-Funktionalitäten, die mittels Standard-Lastfaktoren nicht abgedeckt werden, die Systemlast um z. B. 20 % oder auch mehr als 100 % erhöhen können (Hot Spots). Nach Möglichkeit sollten die Lastfaktoren derartiger Hot Spots dem Customizing entsprechend angepasst werden.

- *Lastbasiertes Sizing:* Existiert bereits ein SAP-System, so wird anhand von Messungen und des somit ermittelten Auslastungsgrades des Systems die Hardware-Dimensionierung für zusätzliche User oder einen Release-Wechsel berechnet. Es werden Angaben über das existierende System (Anzahl und Typ der Prozessoren, Größe des Hauptspeichers, IO-System etc.), die Anzahl der aktiven Benutzer und die Höchstwerte der Prozessor- und Hauptspeicherauslastung benötigt. Nach [72] ist dieses Verfahren zur Betrachtung der Installation weiterer mySAP.com-Komponenten und R/3-Module nicht geeignet und auch für einen Release-Wechsel nur bedingt einsetzbar.

Zusätzlich zu den drei genannten Sizing-Verfahren wird in [54] und [53] ein „Customer Performance Test“ beschrieben. Die Grundidee ist hierbei, dass Einzeltransaktionen in einem Testsystem vermessen und für einen Multi-User-Betrieb hochgerechnet werden.

Das benutzerbasierte und transaktionsbasierte Sizing entsprechen den Standard-Methoden eines Sizing. Nach Angaben von SAP und [93] wird das benutzerbasierte Sizing aufgrund seiner starken Einschränkungen nur in seltenen Fällen allein verwendet. SAP gibt in [10] an, dass nur 30% aller Kunden und Hardware-Partner sich allein auf das benutzerbasierte Sizing verlassen und empfielt, dass diese Form des Sizings nur für kleine und mittelgroße Installationen verwendet werden sollte. Für alle weiteren wird entweder eine Mixtur aus transaktionsbasiertem und benutzerbasiertem oder ein rein transaktionsbasiertes Sizing empfohlen.

Nach [72] bilden im Sizing die Eigenentwicklungen (Z-ABAPs bzw. Z-Transaktionen) des Kunden, d. h. die vom SAP-Standard abweichenden Programme, das Hauptproblem, da diese in ihrem Ressourcenbedarf nicht abschätzbar sind. Der dabei auftretende Ressourcenbedarf ist in hohem Maße von der Qualität der Programmierung abhängig (siehe [138]). Es können somit nur Kundenangaben oder auf Testsystem-Messungen (Customer Performance Test) basierende Annahmen verwendet werden.

Weitere Informationen zum Thema Sizing befinden sich im SAPNet sowie in [93] und [72]. In [72] werden Beispiele für Sizing-Verfahren beschrieben sowie Vorgehensweisen für SAP-Komponenten, die nicht der Transaktionsverarbeitung unterliegen, wie z. B. Business Information Warehouse, Advanced Planner and Optimizer oder der Workplace Server.

2.2.2.1. Quicksizer, GoingLive Check und EarlyWatch Service

Zur Durchführung eines Sizing verwendet jeder Hardware-Partner seine eigenen Werkzeuge, wie z. B. Fujitsu Siemens Computers sein eigenentwickeltes R/3 SizingTool [42]. Diese Werkzeuge werden zum größten Teil nur von Mitarbeitern der Firmen bedient und sind für die Kunden meist nicht zugänglich. Die SAP bietet hingegen seit 1997 für jeden Kunden den Quicksizer an (http://service.sap.com/sizing). Dieser unterstützt sowohl das benutzerbasierte als auch das transaktionsbasierte Sizing. Der Zugriff auf den Quicksizer geschieht online über den SAP Service Marketplace, so dass jeder Kunde in Eigeninitiative sein Sizing vornehmen kann. Der Quicksizer berechnet auf Basis von publizierten SAP-Benchmark-Ergebnissen und Erfahrungswerten über den CPU- und Speicherverbrauch die notwendige CPU-Kapazität, die Größe des Hauptspeichers und das benötigte Festplattenvolumen. Als Grundlage für die CPU-Verbrauchs-Berechnung geht die SAP bei einem benutzerbasierten Sizing von einer Server-Auslastung von 33% und bei einem transaktionsbasierten Sizing von 65% aus. Die dabei verwendeten Algorithmen werden in [10] beschrieben. Die Ergebnisse des Quicksizer werden in Form von SAPS angegeben, so dass anhand dieser Resultate Angebote bei den Hardware-Partnern eingeholt werden können. Nach Angaben der SAP basiert die beschriebene Sizing-Methodik auf über 18.000 SAP-Projekten und Performance-Messungen.

Im Rahmen eines Software-Wartungsvertrags (TeamSAP Support) bietet die SAP ihren Kunden einen SAP GoingLive Check an. Steht ein Unternehmen kurz vor der Produktivphase ihres neuen SAP-Systems, wird ein GoingLive Check durchgeführt. Die Hauptaufgabe besteht darin

zu überprüfen, ob die im Sizing prognostizierten Systeme für den Produktivbetrieb ausreichend dimensioniert wurden. Der GoingLive-Service wird nach [93] auch für andere Szenarien angeboten:

- *SAP GoingLive Functional Upgrade Check* für den Wechsel des SAP-Release
- *SAP GoingLive Migration Check* für den Wechsel der Datenbankplattform oder Hardwareplattform (Betriebssystem- oder Herstellerwechsel)

Ebenfalls im Wartungspreis inbegriffen ist der von der SAP angebotene EarlyWatch Service, der im laufenden Betrieb zwei Mal pro Jahr durchgeführt wird. Im Wesentlichen werden, wie auch beim GoingLive Check, Hinweise zur optimalen Parametrisierung des Systems, zur Performance-Verbesserung und zu möglicherweise zukünftig auftretenden Engpässen gegeben. Es werden hierzu die Systemressourcenverbräuche bzgl. CPU und Hauptspeicher gemessen sowie die Systemparameter analysiert. Im EarlyWatch Report werden für die R/3-Workload, die Datenbanklast und die Gesamtantwortzeit der Transaktionen die Verursacher des prozentual größten Anteils der Last dargestellt. Anhand der Transaktionen können die Programme bzw. Business Components identifiziert werden, die einer genaueren Betrachtung bedürfen. Für die einzelnen Module werden die prozentualen Anteile an der CPU- und DB-Last berechnet und graphisch präsentiert. Wichtig ist hierbei, dass auch die eigenprogrammierten Anwendungen (Z-Transaktionen bzw. Z-ABAPs) als eine Gruppe „Customer“ berücksichtigt werden.

Der EarlyWatch Service ist somit als „kontinuierliche“ Kapazitätsüberwachung anzusehen und der GoingLive Check wird nur bei geplanten hohen Laststeigerungen von über 30% (nach Angabe der SAP) eingesetzt.

2.2.3. Lasttests

Lasttests stellen zur Prognose von Leitungskennzahlen die sicherste aber auch kostspieligste Methode dar. Bei einer konkreten Aufgabenstellung eines Kunden, wie z. B. Release-Wechsel oder Lastzuwachs durch eine erhöhte Benutzeranzahl bzw. Installation einer neuen Business Component, wird das reale Produktivsystem inklusive Software nachgebaut und es werden die zu untersuchenden Aufgabenstellungen unter Laborbedingungen real erprobt. Die Anwender werden hierbei mittels Lastgeneratoren simuliert.

Eine derartige Vorgehensweise ergibt die bestmöglichen Ergebnisse, ist jedoch hinsichtlich der Kosten durch die Anschaffung doppelter Hardware und Software-Lizenzen sowie dem zeitlichen Umfang die finaziell aufwendigste Variante der Leistungsprognose.

2.2.4. R/3-Kapazitätsplanung - Methoden und Vorgehensweisen

In der Literatur sind verschiedene Arbeiten, die dem Gebiet der R/3-Kapazitätsplanung zugeordnet werden können, zu finden. Thematisch behandeln diese Arbeiten die folgenden drei Sachgebiete mit jeweils unterschiedlicher Schwerpunktsetzung:

- Vorgehensmodelle
- Daten-Messung, -Charakterisierung und -Analyse
- Performance-Prognose

Im Jahre 1996 wurde von G. Thompson ein allgemeines Vorgehensmodell für ein „IT Enterprise Resource Management“ (IT ERM) definiert (siehe [118] und [117]), das er 1997 zusammen mit J. Munoz und J. DeBruhl auf den Fall SAP R/3 anwendete [119]. Das von Thompson deklarierte Ziel von IT ERM sollte die Unterstützung eines Unternehmens in der Planung und Aquirierung

von IT-Ressourcen sein. In [117] werden hierfür die in einem Unternehmen zu etablierenden „Enterprise Resource Management Processes“ und die damit zu erreichenden Ziele beschrieben, wobei Thompson die ERM-Prozesse in die folgenden vier Gruppen unterteilt:

- *Infrastructure Improvement Activities:* Workload-Charakterisierung nach Attributen wie z. B. User oder Applikation und Aufbau eines ERM Data Repository. Weiterhin Entwicklung von Methoden zur Performance-Prognose, Daten-Korrelation zwischen Business Objects und Resourcen-Verbräuchen sowie Techniken für die IT-Kosten-Reduktion und Aufbau einer ERM-Wissensdatenbank.
- *Accounting Period Activities:* Messung und Darstellung von Dienstgüten (Service Level), Resourcen-Verbäuchen und -Auslastungen sowie Schwellwert- und Fehler-Überwachung (Early Problem Detection). Aufbereitung der Daten für die Prognose und Durchführung von Trendanalysen. Analyse der Messdaten hinsichtlich Kosten-Reduktion und Performance-Steigerung und Ableitung bzw. Durchführung von Tuning-Maßnahmen [56].
- *Forecast Planning Period Activities:* Erstellung von Performance-Prognosen auf Basis der gewonnenen Daten im ERM Data Repository zur Last- und Kapazitäts-Prognose. Überprüfung der Prognose-Ergebnisse mit Service Level Agreements des Kunden.
- *Resource Investment Planning:* Nutzung der aufgebauten Wissensdatenbank zur Bildung von professionellen ERM-Organisationen oder Abteilungen. Aquisitions-Strategien können dann aus den gewonnenen Ergebnissen entwickelt und dem Management als Entscheidungshilfe zur Verfügung gestellt werden. Unterstützung von Projekten hinsichtlich Alternativen und Vorschlägen in Kapazitätsplanungsfragen. Erstellung von Kosten-, Applikations- und Kapazitäts-Prognosen „auf Anfrage“.

Zur Umsetzung seines Konzepts hebt Thompson als Basis von IT ERM eine kundenorientierte, hoch automatisierte und integrierte Last-Messungs- und -Prognose-Infrastruktur hervor. In [118] werden die Last-Messung und -Charakterisierung sowie Ansätze zur Performance-Prognose mittels Trendanalysen detailliert beschrieben, wobei hier der Schwerpunkt auf der Lastcharakterisierung liegt.

In [119] werden die im IT ERM-Framework beschriebenen Ansätze zur Workload-Charakterisierung auf den Fall SAP R/3 angewandt. Es wird eine Klassifizierung der Last nach Anwender, Applikation oder Tasktyp vorgeschlagen, die Ressourcenverbräuche (CPU-Zeit im Workprozess, DB-Request-Time, Requested KB) und Antwortzeiten ebenfalls beinhaltet. Es werden mögliche Datenquellen wie z. B. das CCMS, der R/3-Accounting-Exit und die R/3-Statistik-Sätze für das Data Repository beschrieben. Einen Schwerpunkt der Arbeit bildet die Untersuchung der Güte der gewonnenen Messdaten. Insgesamt werden die vorhandenen Performance-Daten, insbesondere aufgrund ihres hohen Detaillierungsgrads, und die durch das CCMS bereitgestellten Reportmöglichkeiten als solide und gute Basis für ein Kapazitätsmanagement bewertet. Im Detail betrachtet werden hingegen aufgetretene Probleme bei der Workload-Analyse genannt, wobei als die zwei wichtigsten die folgenden genannt werden:

- *Datenbank-Last:* In den Statistiken von SAP R/3 werden nur die Verweilzeiten der DB-Prozesse in der Datenbank beschrieben. Es existieren keine Angaben über CPU-Verbrauch, Wartezeiten und IO-Aktivität. Als Lösungs-Ansatz für dieses Problem wird die Instrumentierung der Anwendungen mit Hilfe von ARM (Application Response Measurement) vorgeschlagen.
- *Modul-Interaktion:* Modul-Interaktionen sind in den R/3-Statistiken nicht erkennbar. Es kann nicht nachvollzogen werden, ob und in welcher Form R/3-Module z. B. auf die Basis-Komponente (BC) zurückgreifen bzw. miteinander kommunizieren.

Zusätzlich zu den in [119] beschriebenen Datenquellen im SAP R/3-System werden nach Angaben von Simon Ansfield [1] für die R/3-Kapazitätsplanung weitere Performance-Messdaten wie z. B. solche aus Sicht der Datenbank und des Betriebssystems für eine ganzheitliche Sicht auf

das System benötigt. Die Schwierigkeit liegt dann in der Korrelation dieser verschiedenen Messdaten. Weiterhin betont Ansfield, dass für das R/3-System Monitoring-Programme erforderlich sind, die automatisch die Performance-Daten aus dem R/3-System extrahieren und in eine Datenbank schreiben, da die R/3-Statistiken zyklisch auf Stunden-, Tages- oder Wochen-Basis vom R/3-System verdichtet werden und somit für nachträgliche detaillierte Performace-Untersuchungen nicht mehr zur Verfügung stehen.

Bei den oben genannten Arbeiten stehen die Bereiche Vorgehensmodelle und Daten-Messung, -Charakterisierung und -Analyse im Vordergrund. In den Arbeiten von G. Giacone und J. Munoz ([36], [37]) sowie von Y. Somin [110] wird hingegen die Performance-Prognose mittels analytischer Modellierung von SAP R/3-Systemen beschrieben.

Die Erstellung eines entsprechenden Modells basiert nach [36] auf den Messdaten des R/3-Systems (CCMS), des Betriebssystems und der Datenbank. Die Workload-Charakterisierung wird auf Applikations-Ebene, d. h. Klassifizierung nach R/3-Modulen, vorgenommen. Giacone und Munoz geben die folgenden vier Attribute zur Beschreibung der R/3-Modul-Verbräuche an:

- *CPU:* CPU-Zeit-Verbrauch
- *I/O:* User- und System-IOs
- *Netzwerk:* TCP/UDP-Statistiken über Paket-Verarbeitung
- *Wartezeiten:* Wartezeiten im Fall von Transaktionsverarbeitung bei Multi-Programming

Anhand der CCMS-Reports können dann für die Applikationen aller Instanzen folgende Parameter ermittelt werden:

- *Durchsatz:* Anzahl der Dialogschritte pro Stunde
- *Antwortzeit:* Antwortzeit pro Dialogschritt
- *CPU:* CPU-Zeit-Verbrauch pro Dialogschritt im Workprozess
- *DB-Req:* Anzahl der Datenbank-Anfragen pro Dialogschritt

Mittels Korrelation der CCMS- und der Betriebssystem-Daten (auf Prozess-Ebene) werden die Workload-Beschreibungen pro Applikation erstellt. Zur Workload-Charakterisierung im Modell werden Transaktionen definiert, die zum einen die SAP-Last und zum anderen die Datenbank-Last repräsentieren. Weiterhin wird im Modell jeweils unterteilt nach Last verursacht durch Workprozesse bzw. Datenbank-Prozesse und der übrigen Last im R/3-System bzw. in der Datenbank. Eine letzte Gruppe repräsentiert die restliche Last, die zusätzlich auf den Servern, sowohl auf SAP- als auch auf DB-Seite, verarbeitet wurde.

Die mit einem solchen Modell erzielten Ergebnisse hinsichtlich der Antwortzeit liegen bei 10% - 20% Abweichung zwischen Modell und Messung. Ein mit vergleichbarer Workload-Charakterisierung parametrisiertes Modell wird in [110] am Beispiel des SD-Benchmarks beschrieben.

2.3. Vorgehensmodell für die R/3-Kapazitätsplanung

In diesem Abschnitt wird das in dieser Arbeit vorzustellende Vorgehensmodell für die R/3-Kapazitätsplanung definiert ([126], [127]). Das Konzept und die Struktur dieses Modells basieren auf dem im Dokument [122] dargestellten MAPKIT-Vorgehensmodell und dem in Abschnitt 2.2.4. vorgestellten IT Enterprise Resource Management-Framework (IT ERM, [118], [117]), wobei die folgenden Phasen zur Durchführung einer R/3-Kapazitätsplanung definiert werden:

1. *Vorbereitung der Kapazitätsplanung und Festlegung der Rahmenbedingungen:* Dies beinhaltet Präsentationen und Kundengespräche, die u. a. zur Definition des Projektziels

und der Darstellung der entscheidenden Geschäftsprozesse dienen. Weiterhin erfolgt die Installation der Messwerkzeuge (myAMC.LNI, SAR, Windows NT Systemmonitor etc.) und von dem zu betrachtenden R/3-System wird eine Beschreibung angelegt, die alle Angaben bzgl. Prozessoren, Hauptspeicher, Netze, IO-System etc. beinhaltet.

2. *Erfassung und Darstellung der IST-Situation:* Es werden Messungen durchgeführt und die Ergebnisse in geeigneter Form dargestellt. Performance-Probleme in der existierenden Konfiguration sollten in dieser Phase erkannt werden, so dass diese für die nachfolgenden Schritte der R/3-Kapazitätsplanung beseitigt oder berücksichtigt werden können.
3. *Durchführung einer Basismodellierung:* Anhand der vorliegenden Daten wird ein repräsentatives Hochlast-Zeitfenster ausgewählt und für dieses Zeitintervall ein Modell erstellt, welches das reale System adäquat widerspiegelt. Das entstandene Modell wird als Basismodell bezeichnet, wobei evtl. auch verschiedene Zeitfenster (z. B. Tag und Nacht) und somit unter Umständen verschiedene Basismodelle entstehen können.
4. *Performance-Prognosen und Dimensionierung:* Mit Hilfe des Basismodells können Prognosen für Software- oder Hardware-Neuerungen erstellt werden. Im Software-Bereich können beispielsweise Release-Wechsel und bzgl. der Hardware sogenannte vertikale bzw. horizontale Upgrades betrachtet werden. Horizontales Upgrade kann hierbei der Inbetriebnahme eines oder mehrerer zusätzlicher Applikationsserver entsprechen und unter vertikalem Upgrade versteht man die Aufrüstung der Server mit z. B. Hauptspeicher oder weiteren Prozessoren.

Die nachfolgenden Abschnitte beschreiben die Messung und das Monitoring von SAP R/3 (Phase 2) sowie die Performance-Prognose mittels analytischer Modellierung (Phasen 3 und 4). Die dominierenden Werkzeuge zur Umsetzung dieser beiden Bereiche sind der R/3 Live Monitor (=myAMC.LNI), welcher der Daten-Messung und dem Monitoring dient, sowie der WLPSizer (Workload Profile Sizer), der zur analytischen Modellierung von R/3-Systemen genutzt wird. Das Zusammenspiel dieser beiden Werkzeuge wird in Abbildung 2.9 dargestellt.

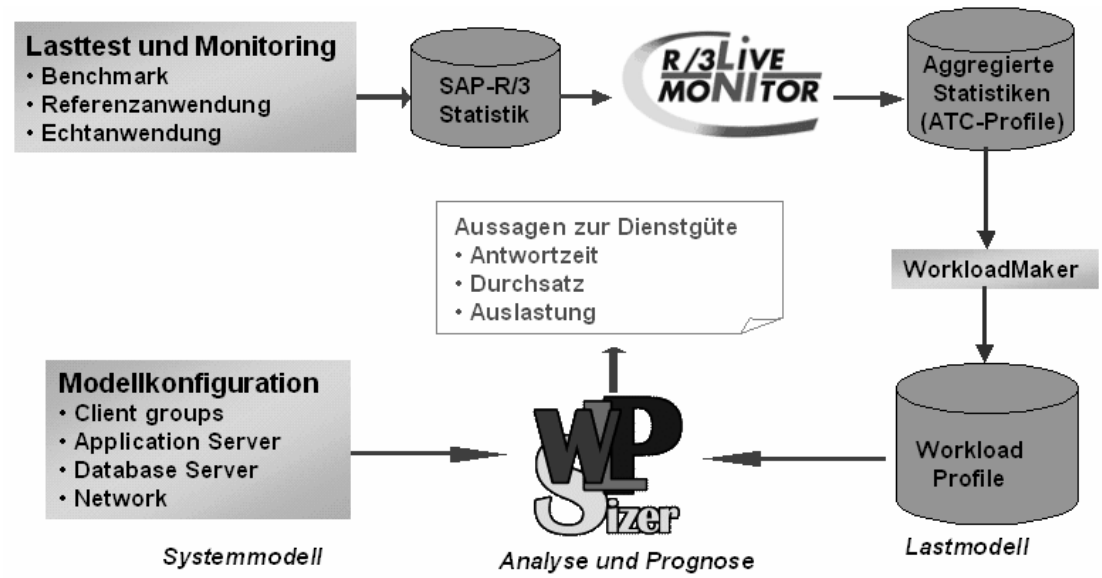


Abbildung 2.9 Vorgehensmodell unter Verwendung der Werkzeuge WLPSizer und myAMC.LNI

Der myAMC.LNI greift auf die R/3-Statistik-Sätze zu, die z. B. die Last eines Benchmarks, einer Referenzanwendung oder einer Echtanwendung beschreiben, und anhand der gesammelten Daten werden mittels Aggregation Lastprofile (TCQ-Profile) erstellt und in einer Datenbank

gespeichert. Die Komponente WLPMaker des WLPSizers greift auf diese Profile zu und erstellt zur Lastbeschreibung für die Modellierung verwendbare Workloadprofile. Weitere Modellkomponenten werden mit Hilfe von Bibliotheken in das Modell eingefügt. Die Modellergebnisse liefern dann Leistungsmaße in Form von Durchsatz, Auslastung und Antwortzeit.

Ein wesentlicher Vorteil des beschriebenen Vorgehensmodells besteht darin, dass die Interaktion zwischen Messung und Modellierung hoch automatisiert abläuft, d. h. die Messdaten werden automatisch aus dem R/3-System extrahiert und als Profile verdichtet und der WLPsizer kann mit Hilfe des WLPMakers automatisch auf Basis der LNI-Datenbank die Profile zur Lastcharakterisierung im Modell erstellen. Eine derart hoch automatisierte Infrastruktur zur R/3-Kapazitätsplanung wird bereits von Thompson in der Beschreibung von IT ERM als Basis zur Einführung und Etablierung von IT ERM oder in diesem Fall R/3-Kapazitätsplanung hervorgehoben.

Die in Abschnitt 2.2.4. beschriebenen Ansätze von Giacone und Munoz zur Modellierung von R/3-Systemen basieren auf einer zum Teil manuellen Beschreibung der R/3-Last auf Basis von CCMS-Reports mit einem Detaillierungsniveau auf Applikations-Ebene. In den folgenden Abschnitten wird eine weitere Lastaggregation mit zugehöriger Dienstgütedefinition beschrieben, die sich insbesondere durch ihren höheren Detaillierungsgrad von den oben genannten abgrenzt. Die Modellierung eines R/3-Systems wird dann nachfolgend in Abschnitt 2.5. beschrieben.

2.4. Monitoring, Messung und Datenanalyse von SAP R/3-Systemen

Die Messung von Performance-Daten und eine anschließende Analyse der Messdaten bildet die Grundlage einer jeden SAP R/3-Kapazitätsplanung (Erfassung und Darstellung der IST-Situation, 2. Phase). In einem repräsentativen Zeitfenster werden Messdaten für das Gesamtsystem erhoben, die dann in der Analyse den aktuellen Systemzustand aus Performance-Sicht (z. B. CPU-Auslastungen, Antwortzeiten, Durchsätze) widerspiegeln.

Die Messdaten-Erhebung besitzt die Zielsetzung einen Überblick über das Gesamtsystem zu liefern und muss somit auf verschiedenen Ebenen vollzogen werden. In zahlreichen Projekten haben sich, wie bereits von Ansfield [119] beschrieben, die folgenden Ebenen als sinnvoll und notwendig herausgestellt:

- SAP R/3-System
- Betriebssystem
- Datenbank

In den folgenden Abschnitten wird ein Messkonzept für SAP R/3-Daten, die aus den R/3-Statistiken gewonnen werden, beschrieben. Anschließend werden für jede der genannten Ebenen spezifische Monitoring- und Messwerkzeuge vorgestellt, wobei insbesondere die zu untersuchenden Leistungsparameter dargestellt werden, die für eine IST-Analyse im Rahmen einer R/3-Kapazitätsplanung von Bedeutung sind. Eine detailliertere Beschreibung befindet sich in [125].

2.4.1. Messkonzept für R/3-Workload

SAP R/3 ist eine transaktionsorientierte Anwendung, in der alle durchgeführten Transaktionen vom R/3-System protokolliert werden. Sie werden mit Hilfe der vom Dispatcher verwalteten Workprozesse verarbeitet, die in vier Tasktypen eingeteilt werden können:

- Dialog (Dialogverarbeitung)
- Update (Verbuchungen)
- Batch (Hintergundverarbeitung)
- Other (Spool, Enqueue etc.)

Die Bewertung der ermittelten Performance-Daten erfolgt im Workload-Monitor des CCMS anhand von Mittelwerten für z. B. die Antwortzeit über alle gemessenen Dialogschritte oder mittels eines Tasktyp-Profils gruppiert nach Tasktypen oder Applikationen. Es wird hierbei nicht unterschieden, ob z. B. der gemessene Dialogschritt einen Zeilenwechsel oder eine komplexe Suchanfrage repräsentiert. Es ist jedoch davon auszugehen, dass die Zahl der kleineren Dialogschritte, beispielsweise Menü-Klicks, gegenüber den komplexeren in Abhängigkeit des Benutzerverhaltens überwiegen. Somit glätten die kleineren Dialogschritte die Ressourcen-Verbräuche und Antwortzeiten der komplexeren, so dass die Aussagekraft des globalen Mittelwerts als alleiniges Bewertungskriterium für die Dienstgüte eines Systems in Frage zu stellen ist.

Es wird somit eine von der Komplexität der Dialogschritte abhängige Antwortzeitbewertung benötigt. Wie bereits in [126] dargestellt, wird die Komplexität eines Dialogschritts hierbei anhand der verursachten Datenbankaktivität definiert, die im folgenden als Database Service Unit (DBSU) bezeichnet wird. Es folgt eine kurze Beschreibung der Database Service Units und eine darauf aufbauende Definition der Komplexitäts- und Dienstgüteklassen auf Basis der Dialogschrittebene. Das grundlegende Konzept zur Workload-Charakterisierung wurde von Michael Paul [126] erarbeitet und im Rahmen dieser Arbeit erweitet und mittels realer Messdaten erprobt.

2.4.1.1. Database Service Units

Database Service Units (DBSU) stellen ein Maß zur Beschreibung der Datenbankaktivität eines Dialogschritts dar. Nahezu jeder Dialogschritt im R/3-System greift auf Daten der Datenbank zu. Diese Zugriffe werden mittels einer Vielzahl von Attributen in den R/3-Statistiken beschrieben, die allgemein in lesende und schreibende Operationen unterteilt werden können. Für die lesenden Zugriffe existieren Direct und Sequential Reads, wobei die Bezeichnung Direct Read stellvertretend für Zugriffe auf einzelne Datensätze und Sequential Read für solche, die auf eine aufeinanderfolgende Kette von Datensätzen zugreifen, steht. Für Direct und Sequential Read existieren jeweils die folgenden Attribute:

- Anzahl der vom R/3-Puffer (des Applikationsservers) erfüllten Anfragen (Buffer)
- Anzahl der Anfragen an die Datenbankschnittstelle
- Anzahl der gelesenen Zeilen (Rec)
- Verweilzeit in der Datenbank
- Anzahl der Datenbankzugriffe (Phycnt)

Die schreibenden Zugriffe werden in Delete, Insert und Update aufgeteilt und jeweils durch die folgenden Attribute beschrieben:

- Anzahl der Anfragen an die Datenbankschnittstelle (Buffer)
- Anzahl der modifizierten Zeilen (Rec)
- Verweilzeit in der Datenbank
- Anzahl der Datenbankzugriffe (Phycnt)

Mit Ausnahme der Verweilzeiten werden die beschriebenen Attribute zur Berechnung des Datenbankaktivitätsmaßes DBSU verwendet (Tabelle 2.2). Für die Anzahl der Datenbankzu-

griffe der Lesezugriffe wird in Tabelle 2.2 nicht zwischen Direct und Sequential Reads unterschieden, so dass diese mit Hilfe des Parameters Read beschrieben werden.

Anhand der Attribute in Tabelle 2.2 und der CommitTime, welche die benötigte Zeit für ein Datenbank-Commit beschreibt, wird pro Dialogschritt eine gewichtete Summe berechnet, die als Database Service Unit (DBSU) bezeichnet wird. Die hierzu verwendeten Gewichte werden in Tabelle 2.2 dargestellt.

DB-Operation	Buffer	Rec	Phycnt
Insert	10	5	5
Update	10	5	5
Delete	10	5	5
Readdirect	0,4	0,4	
Readsequential	0,4	0,4	
Read			20
CommitTime			50

Tabelle 2.2 Gewichte für DBSU-Definition

Mit Ausnahme der CommitTime basiert die DBSU-Definition somit allein auf quantitativen Angaben bzgl. der Anzahl der Datenbankoperationen und Datenmengen der im R/3-System protokollierten Datenbankzugriffe. Aufgrund dessen ist die Definition unabhängig von der Hardware-Plattform, von der verwendeten Applikation und dem R/3-Release sowie der Datenbank-Software und dem Betriebssystem.

2.4.1.2. Komplexitätsklassen

Die Definition der Komplexitätsklassen beruht auf den im vohergehenden Abschnitt beschriebenen DBSUs. Sie werden als geeichte Summe über die Datenbankaktivität der Dialogschritte für die Tasktypen Dialog, Update, Batch und Other berechnet. In Abhängigkeit der berechneten Anzahl der DBSU werden die folgenden fünf verschiedenen Komplexitätsklassen definiert:

- sehr einfach ≤ 100 DBSU
- einfach ≤ 1.000 DBSU
- mittel ≤ 10.000 DBSU
- komplex ≤ 100.000 DBSU
- sehr komplex > 100.000 DBSU

Die Komplexitätsklassen-Grenzen wurden anhand von Benchmark-Messungen definiert und in R/3-Kapazitätsplanungs-Projekten verifiziert. Ein im R/3-System gemessener Dialogschritt wird somit einem Tasktyp und einer Komplexitätsklasse zugeordnet, wie z. B. ein Menüklick dem Tasktyp Dialog und der Komplexitätsklasse 1 entspricht, da dieser in der Regel keine oder nur einer sehr geringe Datenbankaktivität verursacht.

2.4.1.3. Dienstgüteklassen

Die Dienstgüteklassen stellen eine Bewertung der Antwortzeit im Verhältnis zur Komplexität der Dialogschritts dar. Es werden die Güteattribute „gut“, „mäßig“ und „schlecht“ vergeben, so

dass ein zutreffendes Bild der Dienstgüte erzielt wird. Dienstgüteklassen sind realisiert als Schwellwerte pro Komplexitätsklasse mit der folgenden Definition:

- sehr einfach 250 ms (konstant)
- einfach 1.000 ms (konstant)
- mittel 1 ms pro DBSU
- komplex 1 ms pro DBSU
- sehr komplex 1 ms pro DBSU

Ab Komplexitätsklasse 3 werden die Schwellwerte für die Dienstgüte in Abhängigkeit der DBSU-Anzahl definiert mit der Annahme, dass ein DBSU einer Millisekunde Verarbeitungszeit entspricht. Diese Defintion ist sinnvoll, da die Definitionsbereiche der einzelnen Klassen ab Komplexitätsklasse 3 sehr groß sind und somit ein fester Schwellwert eine zu ungenaue Dienstgüte-Definition darstellen würde.

Die Dienstgüteklassen repräsentieren eine komplexitätsabhängige Antwortzeit-Bewertung, indem die gemessene Antwortzeit (Messwert) mit dem zuvor definierten Schwellwert wie folgt verglichen wird:

- gut Messwert ≤ Schwellwert
- mäßig Messwert ≤ 4 * Schwellwert
- schlecht Messwert > 4 * Schwellwert

Es folgen zwei Beispiele zur Verdeutlichung der Dienstgütedefinition:

1. Ein Dialogschritt der Komplexitätsklasse 1 wird der Dienstgüteklasse gut (mäßig) zugeordnet, wenn seine Antwortzeit (RespTime) unter 200 (800) ms liegt. Übersteigt die Antwortzeit 800 ms, besitzt der Dialogschritt die Dienstgüte schlecht.
2. Ein Dialogschritt der Komplexitätsklasse 3 mit 1500 DBSU besitzt als Schwellwert 1500 ms. Die Antwortzeit für Dienstgüte gut (mäßig) muss demnach unter 1,5 (6) Sekunden liegen. Über 6 Sekunden wird der Dialogschritt der Dienstgüte schlecht zugeordnet.

Die Klassifizierung der Dialogschritte wird in Abbildung 2.10 demonstriert.

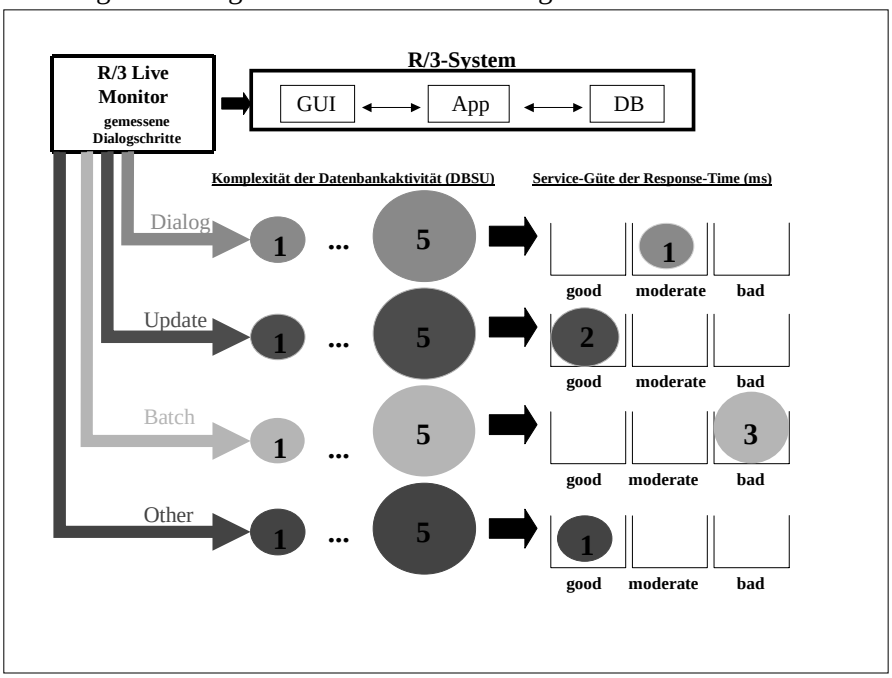


Abbildung 2.10 Klassifizierung der Dialogschritte

2.4.2. Performance-Monitoring- und Analyse-Werkzeuge für SAP R/3

Es wird in diesem Kapitel ein Überblick über ausgewählte zum Zeitpunkt der vorliegenden Arbeit auf dem Markt befindliche SAP-Performance-Monitoring- und Analyse-Werkzeuge gegeben. Es handelt sich hierbei um Werkzeuge von Mercury Interactive, Compuware, BMC und Siemens. Das Produkt myAMC.LNI von Siemens wird detailliert beschrieben, da dieses die beschriebene Messmethodik unterstützt und somit die Datengrundlage für die Modellierung liefert.

2.4.2.1. BMC - Patrol for R/3 Suite

Die Firma BMC bietet in ihrem Produkt-Portfolio eine Komponente „PATROL for R/3 Suite“, die eine Vielzahl an Werkzeugen zum Monitoring und für die Analyse von Applikationen, Datenbanken, Middleware-Produkten, Internet-Applikationen und Betriebssystemen bietet. Nach [5] kann die PATROL R/3 Suite in drei Bereiche unterteilt werden:

- *Daten sammeln:* Es werden Daten über Ressourcenverbräuche, Verfügbarkeit und Systemauslastungen gesammelt.
- *Daten analysieren und Performance-Prognose:* Die gesammelten Daten werden z. B. zur Ursachenforschung aufgetretener Bottlenecks analysiert und Performance-Prognosen auf Basis von Trendanalysen durchgeführt.
- *Aktionen ausführen:* Auf Basis der gewonnenen Informationen der Analysefunktion können automatisch Aktionen z. B. für die Verteilung von Systemressourcen auf Basis von Business Prioritäten, das Management der Datenbank oder auch die Ausführung von 3rd-Party Programmen ausgeführt werden.

Die R/3 Suite von BMC bietet für die beschriebenen Funktionen eine Vielzahl an unterschiedlichen Werkzeugen, wie z. B. PATROL for R/3, PATROL for R/3 - DB-Maintain, - Manager etc. Eine detailliertere Beschreibung befindet sich in [5] und [6].

2.4.2.2. Mercury Interactive und Compuware

Mercury Interactive hat sich im Bereich SAP R/3 insbesondere auf das „Enterprise Testing“, d. h. die Durchführung von Funktions- und Lasttests sowie deren Verwaltung spezialisiert. Neben dem Testen von SAP R/3 bietet Mercury Interactive mit der Komponente Topaz ein Application Performance Management an. Ähnliches gilt auch für die Firma Compuware, die sich vornehmlich auf das Testen von SAP R/3-Systemen spezialisiert hat, aber auch Performance Management betreibt. Das Testen von SAP R/3-Systemen soll in dieser Arbeit nicht näher beschrieben werden. Informationen zu diesem Thema befinden sich z. B. in [77].

2.4.2.3. R/3 Live Network Integrator (myAMC.LNI)

Das von Siemens entwickelte Werkzeug „R/3 Live Network Integrator“ (seit 2000 mit neuem Namen: myAMC.LNI) ist ein Enterprise IT Management Tool für SAP R/3-Systeme. Der LNI ist ein Bestandteil der Performance-Suite myAMC (my Application Management Center), die eine Vielzahl von Performance-Monitoren und Analysewerkzeugen für z. B. Datenbanken, Netzwerke oder IO-Systeme unterstützt [104]. Mit diesem Werkzeug können verschiedene SAP R/3-Systeme von einer zentralen Stelle aus überwacht werden (Single Point of R/3 Control). Der myAMC.LNI besteht aus einer Basis- und einer Expert-Komponente. Das „R/3 Live Monitor Basis-Management“ dient hauptsächlich dem Online-Monitoring (siehe Abbildung 2.11) und besitzt die folgenden Funktionen:

- Darstellung der Beziehungen, Verteilungen und Aktivitäten von R/3-Datenbanken und ihrer zugehörigen R/3-Applikations-Instanzen
- Komplexitäts- und dienstgütebezogene Erfassung von R/3-Lasten basierend auf den Dialogschritten
- Anzeige der aufgetretenen Einträge in das R/3-Systemlog
- Überwachung von vordefinierten Schwellwerten für z. B. die Datenbank, das IO-System
- Präsentation von Zeitreihen für Auslastungs- und Performancewerte

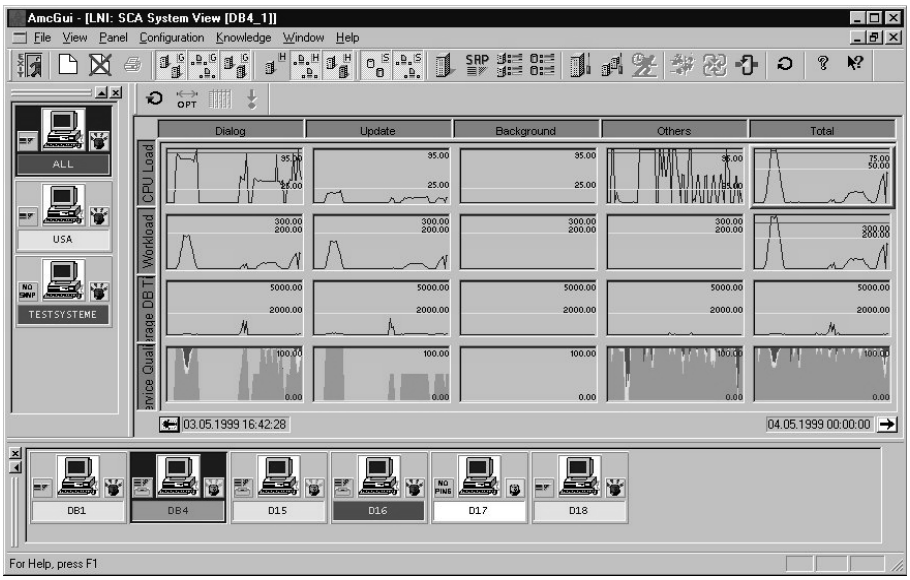


Abbildung 2.11 Basis-Management des myAMC.LNI

Eine Erweiterung des Basis-Managements stellt das „R/3 Live Monitor Expert-Management“ dar, das zusätzlich zu den Basis-Eigenschaften die Komponente LNI-SCA (Service Quality, Complexity, Accounting) beinhaltet. Die Erweiterung des R/3 Live Monitor liegt in der Analyse der in einem SAP R/3-System erfassten Messdaten, die das System standardmäßig für jede Transaktion in Form von Dialogschritten protokolliert. Die Funktionen des Expert-Managements sind:

- Auslesen der R/3-Statistiksätze und Speicherung der Messdaten in einer Datenbank
- Dialogschritte werden beschrieben durch z. B. Antwortzeit, CPU-Verbrauch, DB-Zeit, Wartezeiten
- Zusammenfassung aller Dialogschritte für vorgegebene Zeitintervalle gruppiert nach Tasktyp, Komplexität und Dienstgüte. Die Messdaten dienen dann als Grundlage für die Workload-Generierung in der Modellierung, die automatisch auf Basis der LNI-Daten (TCQ-Profile) durchgeführt werden kann.

Das LNI-Expert-Management bietet neue Funktionalitäten für SAP R/3 Service Quality, Capacity und Accounting-Management und besitzt hierfür die folgenden Ausgabemöglichkeiten:

- *V2-Records:* Sie umfassen die vollständigen R/3-Statistiksätze eines R/3-Systems.
- *V2-Significant:* Es können Rule-Sets für die Spezifikation von Filtern definiert werden, so dass z. B. performance-relevante Dialogschritte oder solche bestimmter Business Components bzw. Transaktionen gemessen werden können. Die Selektionskriterien basieren auf den Attributen der V2-Records. Der Vorteil für die Definition von Filtern ist

die Datenreduktion. In einem produktiven R/3-System werden pro Stunde z. B. 100.000 Dialogschritte verarbeitet. Diese werden in den V2-Records jeweils mit ca. 90 Attributen beschrieben, d. h. der Speicherbedarf der Datenbank steigt kontinuierlich an. Die Beschränkung auf wesentliche Dialogschritte ist zumindest für eine Langzeit-Messung somit die bessere Alternative.

- *User Activity:* Die User Activity liefert eine nach Mandant, User-Account und Tasktyp gruppierte Sicht auf die gemessenen Dialogschritte. Es werden jeweils die Attribute für die Datenbankaktivität, CPU-Verbräuche und transferierte Datenmengen zwischen Datenbank und Applikations-Server dargestellt.
- *Service Quality:* Die Service Quality liefert eine nach Tasktyp gruppierte Sicht der prozentualen Anteile an guten, mäßigen und schlechten Dialogschritten (siehe Kapitel 5.1.3 für Dienstgütedefinition).
- *TCQ-Profil:* Die TCQ-Profile besitzen das in Kapitel 5.1 beschriebene Format der Last-Charakterisierung. Die Dialogschritte werden ergänzend zu den vier Tasktypen in fünf Komplexitätsklassen und drei Dienstgüteklassen unterteilt, wodurch auch die Namensgebung TCQ für „Tasktype, Complexity, Service Quality“ zu erklären ist. Die Zeilen der TCQ-Profile enthalten keine einzelnen, sondern eine Gruppe der in einem bestimmten Zeitraum auftretenden Dialogschritte. Die Attribute der Profile entsprechen den aufsummierten Verbräuchen aller Dialogschritte einer Gruppe (siehe Tabelle 2.3).

Attribut	Einheit	Beschreibung
Datum	---	Datum für den zu beschreibenden Datensatz
Zeit	---	Zeit für den zu beschreibenden Datensatz
T	---	Tasktyp (Dialog, Update, Batch, Other)
C	---	Komplexitätsklasse (1 ... 5)
SQ	---	Dienstgüte (gut, mäßig, schlecht)
DBSU	Anzahl	Maß für die Datenbankaktivität (dargestellt in Database Service Units)
CPUTI	ms	CPU-Zeit-Verbrauch am Application-Server im Workprozess
DBKB	KB	Datentransfer zwischen Applikations- und Datenbank-Server
OTHBytes	Byte	Summe der transferierten Bytes für ABAP/4 Quelltext, ABAP/4-Programm laden und DYNPRO/Screen Quelltext, DYNPRO/Screen laden
RespTime	ms	Antwortzeit der Dialogschritte
DBTime	ms	Verweilzeit der Dialogschritte in der DB (gemessen an der DB-Schnittstelle)
LockTime	ms	Zeitverbrauch für das Warten auf Sperrfreigaben (keine DB-Locks)
QueueTime	ms	Verweilzeit in der Queue vom Dispatcher
ClBy	Byte	Datentransfer zwischen Applikationsserver und Client
RecCnt	Anzahl	Anzahl der Dialogschritte in der jeweiligen Gruppe

Tabelle 2.3 Attribute der TCQ-Profile

Die TCQ-Profile können die R/3-Last auf beliebig detaillierter Ebene beschreiben, wobei die oberste Ebene die R/3-Instanz bildet. Der Detaillierungsgrad kann somit auf Modul-, Sub-Modul- oder auch Transaktions-Ebene erhöht werden, d. h. es können beispielsweise alle Module eines R/3-Systems mittels TCQ-Profilen dargestellt werden, wie es z. B. von Giacone und Munoz in [36] und [37] beschrieben wird.

2.4.3. Betriebssystem-Monitoring

Die Auswahl der Monitoring-Werkzeuge ist abhängig von dem verwendeten Betriebssystem. MS Windows NT/2000 bietet standardmäßig den Systemmonitor ([24],[32]), UNIX besitzt zahlreiche Monitoring-Werkzeuge, wie z. B. SAR, IOSTAT, NETSTAT (für Reliant UNIX siehe z. B. [103]), und für SUN Solaris existiert beispielsweise das Werkzeug SE Performance Toolkit von Richard Petit und Adrian Cockcroft ([97], [14]).

Allgemein existieren auf Betriebssystemebene die Messdaten für:

- *Prozessoren*: Auslastung der Prozessoren unterteilt in User (Zeit im Benutzerprozess), Sys (Overhead-Zeitverbräuche durch das System), WaitIO (Wartezeiten, falls alle IO-Prozesse belegt sind) und Idle (nicht aktive Zeit). Abbildung 2.12 zeigt ein Beispiel für ein CPU-Auslastungsprofil.

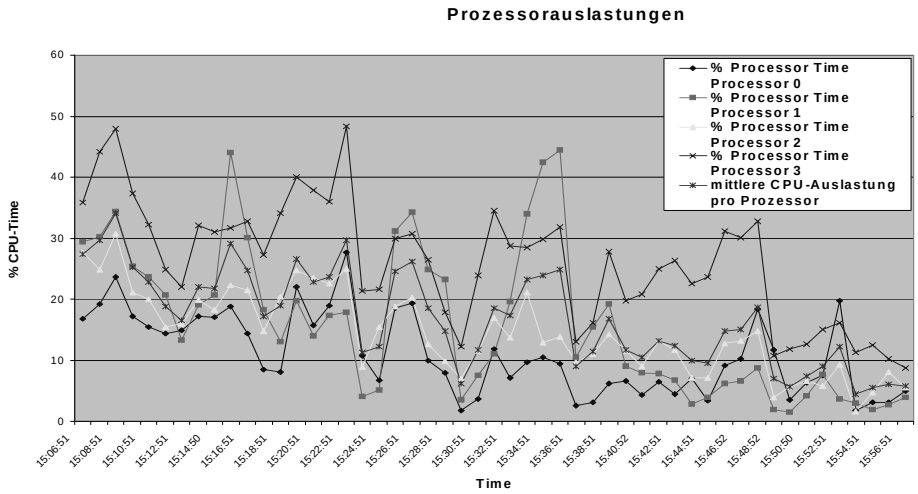


Abbildung 2.12 Beispiel für Prozessorauslastungen unter Windows NT

- *Hauptspeicher*: Belegung des Speichers (Used Memory und Free Memory) und Paging-Aktivität (Page In und Page Out). Werden Daten angefordert, die nicht im Hauptspeicher liegen, müssen diese vom IO-System geladen und in den Hauptspeicher transferiert werden (Page In). Ist der Hauptspeicher bereits belegt, so müssen zunächst Daten aus dem Hauptspeicher verdrängt, d. h. evtl. im IO-System gespeichert werden (Page Out). Diese Vorgänge bezeichnet man als Paging. Für die Paging-Aktivität sind je nach Betriebssystem verschiedene Schwellwerte und Eigenschaften zu beachten, z. B. sollten bei UNIX die Page Outs gleich Null sein, wohingegen bei Windows die Page Ins eine entscheidende Rolle spielen und somit nahezu Null sein sollten. Abbildung 2.13 zeigt ein Beispiel für die Paging-Aktivität eines Systems.

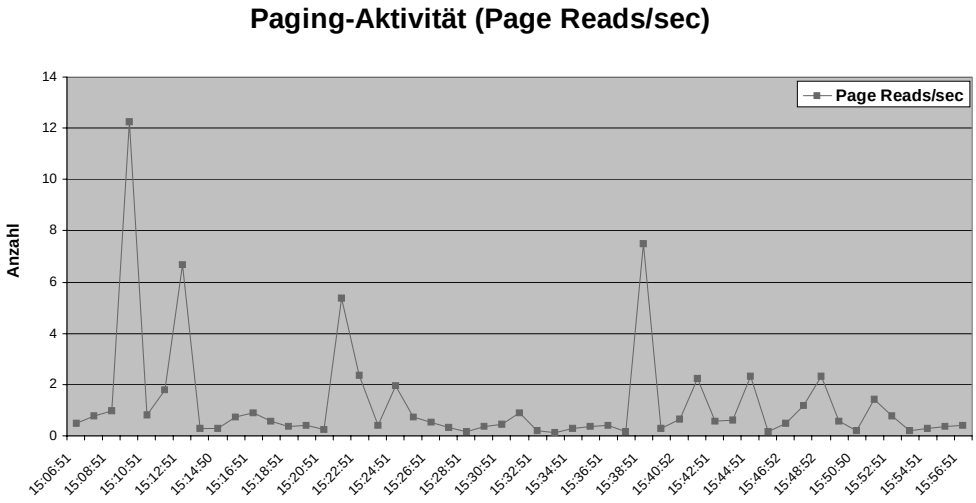


Abbildung 2.13 Beispiel für Paging-Aktivitäten unter Windows NT

- *I/O-System*: Datentransfer (KB/s), IO-Operationen (IOs/s), Warteschlangenlänge (Queue-Length), Antwortzeit und Bedienzeit pro IO (ServiceTime). Die Bedienzeit pro IO ist, falls als Messgröße vorhanden, für die verschiedenen Betriebssysteme in Abhängigkeit des verwendeten Monitoring-Werkzeugs entsprechend zu interpretieren. UNIX-Programme, wie z. B. IOSTAT, liefern einen Parameter ServiceTime pro IO, der als Bedienzeit bezeichnet wird, jedoch tatsächlich der Antwortzeit pro IO entspricht. Unter Solaris existiert ein Monitoring-Werkzeug SE Performance Toolkit, das im Gegensatz zum IOSTAT eine Trennung zwischen Bedienzeit und Antwortzeit vornimmt (siehe Kapitel 4). Bei Windows NT wird ebenfalls eine Bedienzeit pro IO angegeben, die jedoch nach M. Friedman ebenfalls der Antwortzeit entspicht. Durch Einführung eines weiteren Parameters ist dieser Fehler bei Windows 2000 behoben. Die Angabe der Warteschlangenlänge ist je nach Betriebssystem unterschiedlich detailliert oder gar nicht vorhanden. Sun Solaris bietet hierbei den höchsten Detaillierungsgrad, indem sowohl für den Device Driver des Betriebssystems (Wait Queue) als auch für das IO-System selbst inklusive Bus-System (Active Queue) eine Warteschlange gemessen wird. Weitere Informationen werden in den Kapiteln 4 und 5 gegeben.
- *Netzwerk*: Messung der Auslastung des Netzes und des Durchsatzes, d. h. Datenvolumen pro Zeiteinheit.

2.4.4. Datenbank-Monitoring

Die Datenbanküberwachung kann mit Hilfe der Werkzeuge des Datenbank-Herstellers oder von Fremdfirmen, wie z. B. BMC, Siemens oder Mercury Interactive durchgeführt werden. SAP selbst bietet ebenfalls ein eigenständiges Dienstprogramm SAPDBA zur Unterstützung der Administration der Datenbank an, welches Informationen zu allen relevanten Parametern der Datenbank anbietet, wie z. B. Tablespaces, Fragmentierung, Datendateien, Extents, Verzeichnisse, Freiplatz, Redo-Log-Dateien, SGA-Parameter und init.ora-Parameter (Oracle-Parameter). Zahlreiche Datenbank-Administrationsfunktionen des SAPDBA wurden inzwischen direkt in das CCMS des R/3-Systems intergiert (Database System Check, Cost Based Optimizer).

Siemens bietet ein Database Management Center myAMC.DBMC [106] an, das durch Zugriff auf die MIB des Datenbank-Herstellers beispielsweise die folgenden Parameter liefert:

- Cache Hit Ratio
- Anzahl der Physical Writes
- Anzahl der Physical Reads
- Anzahl der Datenbank-Zugriffe
- Anzahl der Datenbank-Transaktionen
- Mittlere Auslastung der Speichersysteme (Plattenplatz)
- Anzahl der Datenbank-Sperren

Es werden von myAMC.DBMC verschiedene Datenbanksysteme unterstützt, wie z. B. Informix, Oracle und Microsoft.

Die Datenbanken selbst liefern ebenfalls Werkzeuge, die zur Performance-Betrachtung genutzt werden können. Da Oracle die im SAP-Umfeld am häufigsten installierte Datenbank ist, wird diese im folgenden näher beschrieben. Oracle enthält die „Oracle dynamic performance views“ (V$ views), die als Tabellen mit einer Vielzahl an statistischen Performance-Daten in der Datenbank verwaltet werden, wie z. B.:

- CPU-Verbrauch aller Oracle-Sessions
- Lesezugriffe unterteilt in physikalisch gelesene Blöcke, gelesene Blöcke in „consistent mode“ und gelesene Blöcke in „current mode“
- Zugriffe auf die Datenbank (gemessen in User Calls aus Sicht der User)
- Wartezeiten (V$SESSION-WAIT, V$SYSTEM_EVENT): Wartesituationen innerhalb einer Session bzw. im System
- Enqueues (V$LOCK): Sperren im Datenbanksystem

Oracle bietet zwei Skripte UTLBSTAT und UTLESTAT zur Erstellung eines Performance-Reports über ein vordefiniertes Zeitintervall, der auf den V$-Views basiert. UTLBSTAT generiert einen Snapshot der V$-Views, UTLESTAT erzeugt nach Ablauf eines festegelegten Zeitintervalls einen zweiten Snapshot. Im Report werden dann die Differenzen der beiden Snapshots als Ergebnisse wiedergegeben. Eine detaillierte Beschreibung bzgl. der V$-Views befindet sich in [78], [13], [40], [22], [39].

Anhand der Ergebnisse von UTLB/ESTAT können Parameter berechnet werden, wie z. B. die DB-Hit-Ratio, welche die prozentuale Häufigkeit angibt, dass ein Datenbank-Prozess einen Datenblock aus dem Buffer-Cache laden kann:

$$\text{DB-Hit-Ratio} = \frac{\text{physical reads} \bullet 100}{\text{consistent gets} + \text{db block gets}}$$

Der Parameter „consistent gets“ beschreibt die Anzahl der sich im Puffer befindlichen Blöcke, auf die ohne Verwendung einer Update-Routine zugegriffen wurde. Hingegen beschreiben die „db block gets“ die Anzahl Blöcke im Puffer, auf die mit einer Update-Routine zugegriffen wurde. Eine detaillierte Beschreibung befindet sich in [40].

2.4.5. Beispiele für die Messdatenanalyse

Die folgenden Beispiele beschreiben Problemstellungen, Aufgabenfelder und erzielbare Ergebnisse unter Verwendung der vorstehend beschriebenen Verfahren und Werkzeuge bei der Datenanalyse [124].

1) Ressourcenmehrbedarf durch Release-Wechsel

Der Release-Wechsel einer Software beinhaltet die Erweiterung des Funktionsumfangs und eine dadurch bedingte Änderung des Ressourcenbedarfs. In den meisten Fällen steigen im R/3-Umfeld bei einem Release-Wechsel die Hardware-Anforderungen in außergewöhnlichem Maße, so dass diese Umstellung häufig Hardware-Erweiterungen erfordert. Der zu erwartende Ressourcenmehrbedarf kann mit Hilfe von SAP-Standard-Benchmarks quantifiziert werden.

Bei einem Vergleich der Benchmark-Ergebnisse der unterschiedlichen R/3-Releases erhält man für die CPU-Mehrbelastungen eines Release-Wechsels von Version X auf Y zum Beispiel die Tabelle 2.4. Hier werden modulspezifische Faktoren für die CPU-Mehrbelastung aufgeteilt nach Dialog-, Update- und Datenbank-Service dargestellt, mit denen die prozentualen Aufschläge berechnet werden. Es wird zum Beispiel für FI unter Release Y 47% mehr CPU-Leistung im Dialog-Betrieb beansprucht als unter dem R/3-Release X. Man erkennt, dass sich die Performance-Veränderungen in den unterschiedlichen Modulen sehr stark unterscheiden. Für die einzelnen Module können nach Angaben der SAP selbst auf Transaktionsebene Leistungsunterschiede angegeben werden.

R/3-Modul	DIA	UPD	DB
Finanzwesen (FI)	1,47	1,00	1,14
Materialwirtschaft (MM)	1,13	1,03	1,03
Produktionsplanung (PP)	1,50	1,00	1,67
Vertrieb (SD)	1,48	1,01	1,20
Workflow-Management (WM)	1,46	0,91	1,26

Tabelle 2.4 CPU-Mehrbelastungen auf Grund eines Release-Wechsels von X auf Y

2) Messung der vom Kunden implementierten ABAP/4-Anwendungen

Die Messung und Identifizierung von unperformanten Eigenentwicklungen, d. h. Z-Transaktionen, geschieht wie in Abschnitt 3.2.1. beschrieben durch die SAP-Reports GoingLive Check und EarlyWatch. Mit Hilfe der Filter-Funktionalität des R/3 Live Monitors (Significant Dialogsteps) können die in den Reports dargestellten Lastverursacher für zukünftige Messungen getrennt von der Gesamtlast betrachtet werden. Es können für die gefilterten Dialogschritte bzw. Anwendungen Workloadprofile zur Analyse der gemessenen Last erstellt werden. Es können dann exemplarisch die folgenden Fragestellungen behandelt werden:

- *Wie hoch ist die Datenbank-Verweilzeit an der Gesamtantwortzeit im Dialogbetrieb?* Es wird in [92] ein Schwellwert von 40% angegeben, der nicht überschritten werden sollte. Ein zu hoher Datenbankanteil deutet auf Datenbank-Performance-Probleme hin und ein zu hoher CPU-Zeit-Bedarf auf Seiten der Applikationsserver könnte ein Indiz für Laufzeitprobleme der ABAP/4-Anwendungen sein.
- *Wie hoch ist die Datenbank-Verweilzeit und CPU-Zeit am Application-Server im Vergleich zu den DBSU?* Wie bereits in Abschnitt 2.4.1. beschrieben, geben die DBSU die

Datenbankaktivität und zugleich auch die Komplexität der Dialogschritte an, d. h. beispielsweise ist bei einer relativ geringen Anzahl an DBSU und somit relativ geringer Komplexität eines Dialogschritts eine hohe Datenbank-Verweilzeit bzw. ein hoher CPU-Zeit-Verbrauch am Application-Server auffällig.

3) Datenbank-Tuning

In einem SAP R/3-System werden in der Praxis zentrale Datenbanken verwendet, die nicht über mehrere Server verteilt werden können. Die Datenbank wird somit häufig zum Performance-Engpass des gesamten Systems. Eine wichtige Rolle spielt somit das Monitoring und Tuning der Datenbank.

Die Analyse einer Datenbank kann mit Hilfe der Datenbank-Werkzeuge und SAP-Reports erfolgen. Der SAP-Report GoingLive Check liefert z. B. folgendes:

- Überprüfung der Parametereinstellungen der Datenbank (z. B. Puffer-Einstellungen)
- Überprüfung der Indexe
- Auflistung „teurer“ SQL-Anweisungen
  - Es werden die SQL-Anweisungen und deren Anteil an der Gesamt-DB-Last beschrieben.
  - Mit Hilfe der EXPLAIN-Funktion in Oracle werden Zugriffspfade der SQL-Anweisungen dargestellt und welche Indexe für die Anfrage verwendet werden.
  - Die Datenbank-Last einer SQL-Anweisung wird anhand der gemessenen Datenbank-Puffer-Zugriffe in Relation zur Gesamt-Anzahl definiert.
  - Es wird die Anwendung ausgegeben, die die SQL-Anweisung enthält.
  - Der Report bietet Verbesserungsvorschläge zur Optimierung der SQL-Anweisungen, zum Beispiel Definition eines weiteren Index, Erweiterung der WHERE-Klausel mit weiteren Attributen oder auch eine verbesserte Verwendung des Datenbank-Optimierers für die Wahl der Zugriffspfade.

Neben den Analysewerkzeugen für Datenbanken, wie z. B. UTLB/ESTAT von Oracle, kann auch mit Hilfe des R/3 Live Monitors anhand der Workloadprofile und der V2-Statistiken die Datenbank analysiert werden. Die Workloadprofile bieten die Möglichkeit des Vergleichs von DBSU und Datenbankverweilzeit, so dass bei einer hohen Verweilzeit Wartezeiten z. B. durch Locks zu vermuten sind.

Mittels der V2-Statistiken können aus R/3-Sicht die Datenbankaktivitäten (Read, Insert, Delete und Update) im Detail analysiert werden. Ursachen für ein Datenbankproblem liefern diese Statistiken jedoch nicht, da alle Aktivitäten nur bis zur R/3-Datenbankschnittstelle protokolliert werden. Die Performance-Daten der Datenbank werden hingegen z. B. vom Skript UTLB/ESTAT gemessen. Eine Verknüpfung dieser beiden Sichtweisen würde somit zu einer verbesserten Ursachenforschung von Datenbankproblemen führen. Eine derartige Verbindung existiert über die DB-Calls von SAP R/3 und die User-Calls von Oracle.

Datenbankanfragen in SAP R/3 werden unterschieden in DB-Requests und DB-Calls. Unter DB-Requests versteht man die in den ABAP-Programmen generierten SQL-Anfragen, die an die Datenbankschnittstelle gesendet werden, in der dann analysiert wird, ob die Anfrage durch den R/3-Puffer erfüllt werden kann oder ob die Anfrage an die Datenbank weitergeleitet werden muss. Bei einer Weitergabe der Anfrage an die Datenbank wird dies als DB-Call protokolliert. Die Zugriffe auf die Datenbank werden in den DB-Statistiken als User-Calls angegeben. Es wurden für zwei SAP-Standard-Benchmarks (Zentralserversysteme) und zwei Messungen in einem

Produktiv-System (3-stufig) die Zusammenhänge zwischen DB-Calls und User-Calls untersucht. Die Ergebnisse sind in Tabelle 2.5 dargestellt.

System	DB-Calls	User-Calls	Fehler
SD-Benchmark	496.711	423.039	14,83%
FI-Benchmark	331.446	316.934	4,38%
Prod.Syst., Messung 1	9.152.368	8.812.386	3,71%
Prod.Syst., Messung 2	28.159.501	25.931.617	7,91%

Tabelle 2.5 Zusammenhang zwischen DB-Calls und User-Calls

Wie aus der Untersuchung zu erkennen ist, liegen die Differenzen zwischen DB- und User-Calls im Mittel bei 10%, d. h. das R/3-System liefert 10% mehr Zugriffe auf die Datenbank als durch die User-Calls widergespiegelt wird. Dies ist dadurch zu erklären, dass jeder DB-Call der R/3-DB-Schnittstelle an die OCI-Schnittstelle von Oracle gesendet wird, diese jedoch nicht jeden DB-Call in die Datenbank weiterleitet (wie z. B. PREPARE) bzw. mehrere DB-Calls zu einem Datenbank-Aufruf zusammenfasst.

Die beschriebenen Ergebnisse zeigen, dass Zusammenhänge zwischen der R/3- und DB-Sicht auf Ebene der Messdaten existieren. Es bleibt zu untersuchen, in wie weit z. B. Datenbank-Locks sowie Warte- und Sperrzeiten durch die beschriebene Verbindung auf die R/3-Daten übertragen werden können.

2.5. Modellierung und Prognose für SAP R/3-Systeme mit dem WLPSizer

Die bislang dargestellten Methoden und Werkzeuge beschreiben die Erfassung und Darstellung der IST-Situation (2. Phase der R/3-Kapazitätsplanung) für ein R/3-System. Auf Basis der hierbei gewonnenen Daten und Erkenntnisse können Modelle zur Performance-Prognose erstellt werden. Das Ziel der Modellierung ist es, das Systemverhalten in Form von Durchsätzen, Antwortzeiten und Auslastungen für zukünftige Szenarien, wie z. B. R/3-Release-Wechsel, Lastzuwächse oder Installation weiterer Business Components zu prognostizieren. Die Modelle können Systemengpässe aufdecken und notwendige Hardware-Upgrades für deren Beseitigung vorschlagen.

Wie auch bei Giacone und Munoz ([36], [37]) wird in dieser Arbeit eine analytische Modellierung verwendet, d. h. das R/3-System wird mit Mitteln der klassischen Warteschlangentheorie ([52], [57], [58]) beschrieben und mit Hilfe des effizienten Bard-Schweitzer-Algorithmus [95] gelöst. Zur Modelllösung wird das Werkzeug TOTO [121] verwendet, das eine Implementierung des Bard-Schweitzer-Algorithmus enthält. Der Lösungsalgorithmus benötigt selbst für größere Modelle, wie z. B. über 10 Application-Server und 20 Workloadprofile, nur wenige Sekunden.

Für die Modell-Erstellung wird in diesem Kapitel das Modellierungswerkzeug WLPSizer, das im Rahmen einer Diplomarbeit [111] entworfen und anschließend durch die in zahlreichen Projekten gewonnenen Erfahrungen weiterentwickelt wurde. In den nun folgenden Abschnitten steht die Funktionsweise und Anwendung des WLPSizers im Vordergrund. Eine detaillierte Beschreibung der im WLPSizer implementierten Abbildung eines R/3-Systems in ein Warteschlangenmodell befindet sich in [111]. Im Anschluss an die Werkzeugbeschreibung werden

Vorgehensweisen und Beispiele für die Modellierung von z. B. R/3-Release-Wechseln oder Last-steigerungen beschrieben.

2.5.1. Modellierungswerkzeug WLPSizer - Architektur

Der WLPSizer [129] ist ein Werkzeug zur Modellierung von SAP R/3-Systemen. Die Abkürzung WLPSizer steht für "Workload Profile Sizer". Ein zentraler Bestandteil des WLPSizers ist die Beschreibung der R/3-Last in Form von Workoad-Profilen, die automatisch anhand von Messdaten des myAMC.LNI erstellt werden können. Der Begriff Sizer steht stellvertretend für die Möglichkeit Prognosemodelle zu erstellen, die Informationen bzgl. der in einem R/3-System zu erwartenden Performance und der notwendigen Hardware-Ausstattung liefern.

In diesem Abschnitt wird ein Überblick über die Architektur des WLPSizers gegeben. Es werden hierzu die in Abbildung 2.14 dargestellten WLPSizer-Komponenten (Bibliotheken und WLPMaker), sowie die Schnittstellen zu weiteren Werkzeugen (TOTO, VITO und MS Excel) beschrieben.

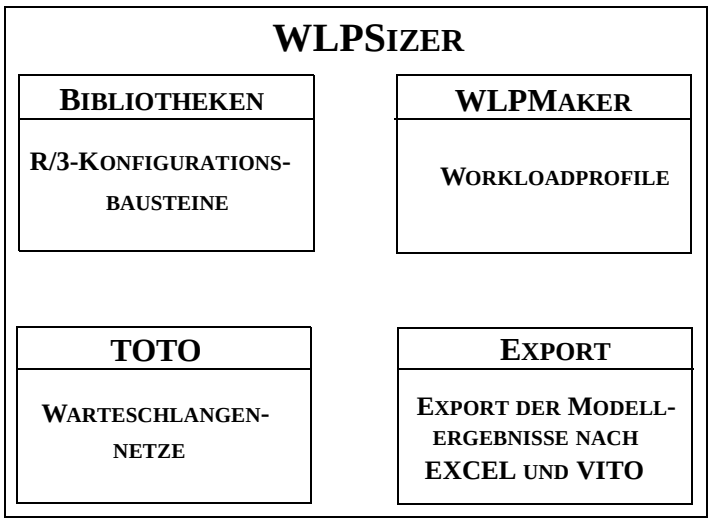


Abbildung 2.14 WLPSizer-Architektur

Zur Modellierung eines R/3-Systems mit Hilfe des WLPSizers benötigt man die Eingabe der R/3-Konfiguration (Beschreibung der Server, Netze, IO-Systeme), die mit Hilfe von Bibliotheken eingegeben wird, und eine Beschreibung der auf dem System befindlichen Last (Workloadprofile). Wurde das R/3-System mit dem Werkzeug myAMC.LNI [105] überwacht, können mit Hilfe des WLPMakers die Workloadprofile automatisch aus den Messdaten generiert werden. Die Bibliotheken und der WLPMaker sind, wie in der Abbildung 2.14 dargestellt, Bestandteile des WLPSizers.

Der WLPSizer kann nun anhand der eingegebenen Daten ein Warteschlangenmodell erstellen, das dem Warteschlangenlöser TOTO [74] übergeben wird. Die zurückgelieferten Modell-Ergebnisse können im WLPSizer dargestellt und nach MS Excel bzw. als ASCII-Datei exportiert werden. Weiterhin kann das im WLPSizer erstellte Modell zur Weiterbearbeitung in das Modellierungswerkzeug VITO [73] importiert werden. Im Folgenden wird auf das Bibliotheken-Konzept und die WLPSizer-Komponente WLPMaker sowie auf die Export-Möglichkeiten der im WLPSizer erstellten Konfigurationen eingegangen.

Bibliotheken

Eine für SAP R/3 typische 3-stufige Konfiguration wird in Abbildung 2.15 dargestellt.

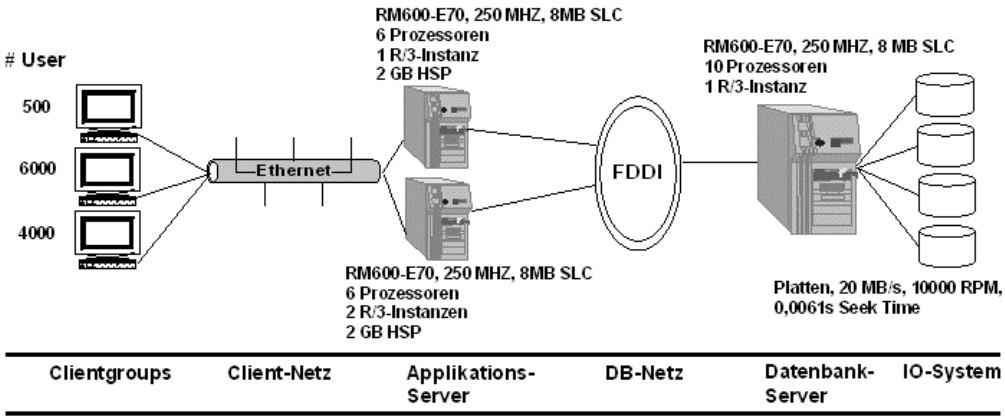


Abbildung 2.15 Beispiel für eine SAP R/3-Konfiguration

Die Konfiguration eines R/3-Systems besteht aus den vier Komponenten: Clients, Netze, Application-Server und Database-Server. Der WLPSizer unterstützt die Modellierung von 3-stufigen und Zentralserver-Systemen (Applikation und Datenbank befinden sich auf einem Server). Die vier Komponenten eines R/3-Systems können in beliebiger Anzahl, mit Ausnahme der Datenbank, in den WLPSizer eingegeben und zu einem System verbunden werden. In der Praxis existiert in einem 3-stufigen R/3-System eine zentrale Datenbank ([12], [92]), die sich auf einem einzigen Server befindet, so dass jedes WLPSizer-Modell nur einen Database-Server besitzt. Die Eingabe der einzelnen Systemkomponenten wird durch Bibliotheken, die für die CPUs der Server, die IO-Systeme und die Netze existieren, unterstützt.

- *CPU-Bibliothek*

  Die CPU-Bibliothek wird zur Beschreibung der CPUs der Applikations- und Datenbankserver in SAPS verwendet. Es können im WLPSizer verschiedene CPU-Bibliotheken angelegt werden, so dass die CPUs nach den verschiedenen R/3-Releases klassifiziert werden können. Dies ist notwendig, da jedes R/3-Release unterschiedliche Ressourcen-Anforderungen besitzt und somit auch unterschiedliche SAPS-Werte, da diese auf Basis der verarbeiteten Dialogschritte definiert sind (siehe Abschnitt 2.2.1.3.).

- *IO-Bibliothek*

  Für den Datenbankserver muss im WLPSizer neben den CPUs auch das IO-System spezifiziert werden. Für die Applikationsserver werden keine IO-Syseme berücksichtigt, da diese hauptsächlich nur für das Betriebssystem genutzt werden und somit für die Performance-Betrachtung eines SAP R/3-Systems nicht relevant sind.

- *Netzwerk-Bibliothek*

  Zusätzlich zu den CPUs und IO-Systemen existiert im WLPSizer eine Bibliothek zur Definition und Paramterisierung der Netze. Bei der dreischichtigen Architektur von SAP R/3, werden Netze zwischen der Präsentations- und Applikationsserverebene sowie zwischen Applikationsserver- und Datenbankserverebene benötigt.

WLPMaker

Der WLPMaker (Workload Profile Maker) ist ein Bestandteil des WLPSizers und wird zur Erstellung von Workloadprofilen verwendet. Diese werden auf Basis der vom myAMC.LNI gemessenen R/3-Last nach Eingabe der gewünschten R/3-Instanz und dem entsprechenden Zeitfenster automatisch generiert. Sie können anschließend direkt in das Modell als Lastbeschreibung eingegeben werden.

- *Konzept zur Beschreibung der R/3-Last mittels Workloadprofilen*

  Das Werkzeug myAMC.LNI liefert neben vielen weiteren Ausgabe-Daten eine Darstellung der gemessenen Dialogschritte in Form von TCQ-Profilen. Es wird hierzu auf die Statistical Records (V2-Records) des R/3-Systems zugegriffen, die alle verarbeiteten Dialogschritte mit ihren Ressourcenverbräuchen enthalten. In den TCQ-Profilen werden die Dialogschritte nach Tasktyp, Komplexitätsklasse und Dienstgüteklasse klassifiziert (siehe Abschnitt 2.4.1). Das TCQ-Profil beschreibt typischerweise die R/3-Last einer R/3-Instanz.

  In der nun folgenden Beschreibung der im WLPSizer verwendeten Workloadprofile wird im Gegensatz zu den TCQ-Profilen nur eine Klassifizierung nach Komplexität und Tasktyp verwendet, da die Dienstgüte anhand der Modellergebnisse ermittelt wird. Es existieren für den WLPSizer zwei Arten von Profilen: Simple- und Extended-Workloadprofile. Wie in Abbildung 2.16 dargestellt, ist ein Simple-Workloadprofil durch die Attribute CPUTI, DBSU, DBKB und ClBy definiert. Das Extended-Profil besitzt hingegen zusätzlich die Spalten RespTime, DBTime, LockTime und QueueTime. Beide Profile können vom WLPSizer eingelesen werden, wobei jedoch nur die Spalten der Simple-Workloadprofile zur Modellierung verwendet werden. Die Extended-Workloadprofile werden zur Kalibrierung des Modells herangezogen. Im weiteren Verlauf dieser Arbeit wird unter der Bezeichnung Workloadprofil das Extended-Workloadprofil verstanden.

T	C	ratio	CPUTI	DBSU	DBKB	ClBy	RespTime	DBTime	LockTime	QueueTime
D	1	0,5228	19,50	72,84	0,57	325	87,77	20,36	0,14	0
D	2	0,1193	69,46	425,48	11,98	415	421,08	276,03	1,33	0
D	3	0,0322	932,84	1888,59	104,81	603	3959,03	2650,82	6,26	0
D	4	0,0045	6406,56	19454,60	2355,77	715	37959,57	28602,39	12,33	0
D	5	0,0007	73417,97	390184,00	123115,67	1015	379533,32	319135,92	0,11	2
U	1	0,0529	6,54	131,98	0,78	112	20,50	7,31	0,00	0
U	2	0,2076	28,51	595,96	31,14	336	88,35	40,38	0,28	10
U	3	0,0274	99,01	1568,43	191,28	645	676,18	543,27	0,33	6
U	4	0,0018	555,47	6360,15	456,03	812	3455,48	2912,10	0,24	0
U	5	0,0000	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
B	1	0,0044	12,76	170,98	0,13	225	29,29	10,86	0,31	3
B	2	0,0015	43,20	414,47	20,71	336	607,08	214,62	0,86	2
B	3	0,0011	183,59	1411,25	52,95	548	2286,45	678,01	6,66	1
B	4	0,0000	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
B	5	0,0000	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
O	1	0,0143	12,21	59,86	0,00	96	366,25	1,58	0,00	313
O	2	0,0066	29,98	478,31	24,10	223	1606,70	29,55	0,00	8
O	3	0,0028	59,48	1467,97	30,83	469	2072,20	67,20	0,00	864
O	4	0,0001	93,75	5468,00	21,89	712	239,56	178,57	0,00	0
O	5	0,0000	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
*	*	**1,0000**	**139,59**	**707,05**	**114,79**	**344**	**734,87**	**516,10**	**0,56**	**9**

Abbildung 2.16 Workloadprofil

Die Abkürzungen im Workloadprofil werden in Tabelle 2.6 beschrieben.

Parameter	Einheit	Beschreibung
T	---	Tasktyp des Dialogschritts (Dialog, Update, Batch, Other)
C	---	Komplexitätsklasse (1 ... 5)
ratio	---	Anteil an der gesamten Last
CPUTI	ms	mittlere normierte CPU-Zeit eines Dialogschritts am Application-Server im Workprozess
DBSU	Anzahl	mittlere Anzahl an DBSU
DBKB	KB	mittlere Anzahl an Daten in KB, die zwischen Database- und Application-Server versendet wurden
OTHKB	KB	Summe der transferierten Bytes für ABAP/4-Quelltext und Programm laden sowie Dynpro/Screen-Quelltext und Dynpro/Screen laden
ClBy	Byte	mittlere Anzahl an Bytes, die zwischen Client und Application-Server transferiert wurden
RespTime	ms	mittlere Antwortzeit des Dialogschritts
DBTime	ms	Verweilzeit des Auftrags in der DB (gemessen an der DB-Schnittstelle des Application-Servers, somit inkl. Netzzeit und Wartezeiten in der DB)
LockTime	ms	mittlere Sperrzeit im R/3-System (keine Datenbank-Sperren)
QueueTime	ms	Wartezeit am Dispatcher
*	---	gewichteter Mittelwert der jeweiligen Spalte über alle Tasktypen und Komplexitätsklassen mit Ausnahme der Spalte "ratio", die summiert wird

Tabelle 2.6 Parameter eines Workloadprofils

Es können mit Hilfe des WLPMakers über beliebige Zeitintervalle Workloadprofile dargestellt werden. Für die Modellierung werden hingegen die Profile immer als Lastrepräsentation einer Stunde interpretiert, wobei die Durchsätze entsprechend normiert werden. Die Einträge des Workloadprofils entsprechen den mittleren Verbräuchen eines Dialogschritts im betrachteten Zeitintervall. Wie bereits beschrieben, werden die Dialogschritte nach Tasktyp und Komplexitätsklasse gruppiert. Die Spalte ratio gibt den Anteil der jeweiligen Gruppe von Dialogschritten gemessen am Gesamtdurchsatz des Workloadprofils an. Für die CPU-Zeit werden die gemessenen Zeitverbräuche bzgl. eines Referenzrechners normiert, d. h. es wird berechnet, wie viel CPU-Zeit ein Rechner mit einer vorgegebenen Referenzleistung für die im Workloadprofil angegebene Last benötigt. Die Referenzleistung ist auf 250 SAPS festgelegt. Die Normierung der Workloadprofile liefert die Möglichkeit, sie in verschiedenen Konfigurationen wieder zu verwenden und miteinander vergleichen zu können.

Export von Modellen und Ergebnissen

Zur Weiterbearbeitung der Modelle können diese mit Einschränkungen in das Modellierungswerkzeug VITO übertragen werden. Weiterhin können in Form von Reports Modelle und Ergebnisse nach MS Excel exportiert werden.

2.5.2. WLPSizer im Überblick

In diesem Abschnitt werden die Grundeigenschaften des WLPSizers beschrieben, welche die Eingabe einer Konfiguration, das Lösen des Modells und die Darstellung der Ergebnisse ermöglichen.

- *Konfigurationen*

  Die mit Hilfe des WLPSizers beschriebenen R/3-Systeme werden als Konfigurationen bezeichnet. Eine solche besteht aus den folgenden Komponenten:

  - Datenbankserver und IO-System
  - Applikationsserver
  - Netze
  - Benutzergruppen
  - Lastbeschreibungen (Workloadprofile)
  - Konfigurationsparameter

  Die Erstellung einer neuen Konfiguration im WLPSizer erfordert die Eingabe eines DB-Servers, eines Netzwerks und einer Clientgroup sowie eines Namens der Konfiguration. Die dabei entstandene Konfiguration hat die in Abbildung 2.17 dargestellte Form und kann als Grundkonfiguration bezeichnet werden.

Abbildung 2.17 Grundkonfiguration im WLPSizer

Wie in der Abbildung dargestellt, wird der Server als Database- und Application-Server interpretiert, also als Zentralserver. Werden der Konfiguration Application-Server hinzugefügt, entspricht diese einer 3-stufigen R/3-Konfiguration. Die erstellte Konfiguration wird in der Config-View dargestellt (siehe Abbildung 2.18).

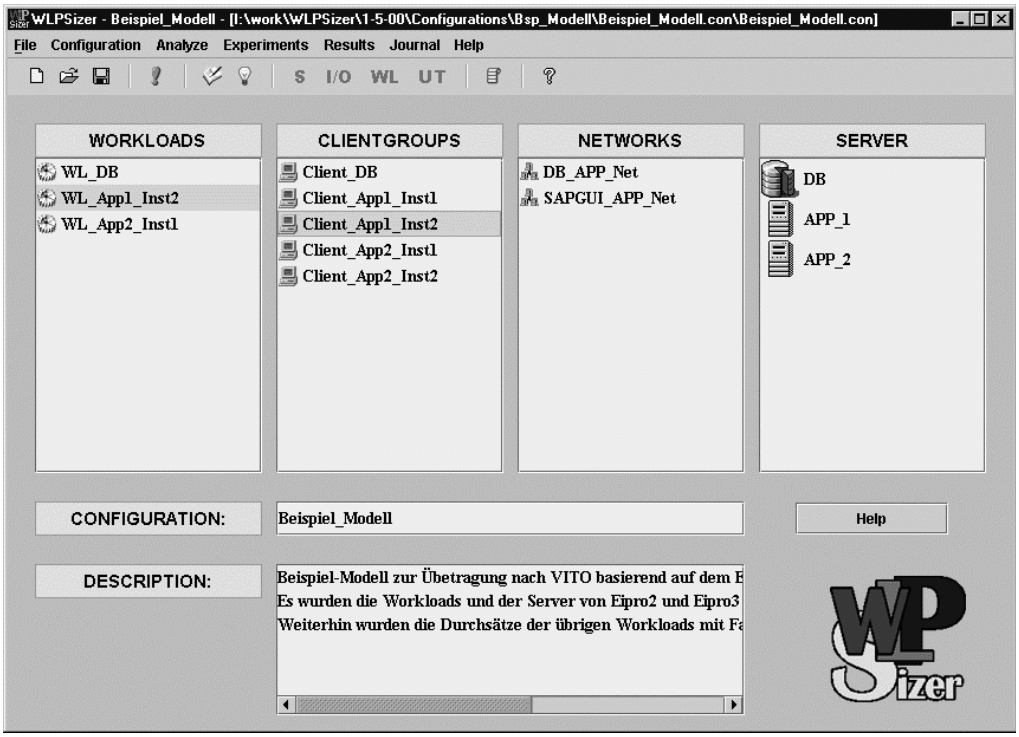


Abbildung 2.18 Config-View vom WLPSizer zur Darstellung der R/3-Konfiguration

In den nun folgenden Abschnitten werden in kurzer Form die einzelnen Komponenten einer Konfiguration und deren Parameter beschrieben. Eine detailliertere Darstellung befindet sich in [129] und [128].

- *Lastbeschreibungen - Workloads*

  Eine Workload repräsentiert die von einer R/3-Instanz oder einer bestimmten Applikation erzeugte Last. Sie wird durch die folgenden Parameter charakterisiert:

  - Bezeichnung: Name der Workload
  - Profil: Normiertes Workloadprofil
  - Durchsatz: Anzahl der Dialogschritte pro Stunde

  Eine Workload besitzt eine fest vorgegebene Quelle und ein Ziel, wobei als Quelle eine Clientgroup (Lastverursacher) und als Ziel ein Server der Konfiguration ausgewählt werden können. Das Ziel ist hierbei der Server, der die Instanz bzw. Applikation betreibt. Der WLPSizer bestimmt dann automatisch, welche Stationen der Konfiguration von der Workload besucht werden, um das Ziel zu erreichen.

  Der Workload im Modell liegt ein aus den Messdaten erstelltes Workloadprofil zugrunde. Wie bereits oben beschrieben, werden die Durchsätze der Workload jeweils auf eine Stunde normiert. Der WLPSizer unterstützt insgesamt drei verschiedene Workload-Typen:

  - *Standard:* Die Workload besitzt eine Quelle (Clientgroup) und ein Ziel (Server). Der Pfad zwischen Quelle und Ziel wird automatisch vom WLPSizer festgelegt.

- *Background:* Eine Background-Workload dient der Belastung einzelner Modellkomponenten zur Abbildung von Hintergrundlasten, wie z. B. weiteren Nicht-R/3-Anwendungen, die vom Monitoring-Werkzeug myAMC.LNI nicht erfasst wurden. Die Workload besitzt ein Ziel, das eine beliebige Komponente des Modells sein kann, wie z. B. Netz, IO-System oder CPU(s). Der Workload wird automatisch eine Quelle zugewiesen, die im WLPSizer als nicht sichtbare Modellkomponente existiert.
- *Custom:* Die Workload besitzt eine Quelle (Clientgroup). Der Pfad für die Workload bzw. die zu besuchenden Komponenten des Modells ist frei definierbar. Für alle Komponenten können Besuchszahlen angegeben werden, die festlegen, wie oft sie von der Workload besucht werden sollen.

• *Clientgroups*

Eine Clientgroup repräsentiert eine Gruppe von Benutzern (bzw. Terminals) in einem gemeinsamen Netzsegment, die eine identische Last erzeugen. Eine Clientgroup wird parametrisiert durch:

- Active User: Anzahl aktiver Benutzer
- Netz: Netz, das die Clientgroups mit den Servern verbindet

Jeder Clientgroup wird ein Netz zugeordnet, dass die Verbindung zu den Applikationsservern herstellt. Im Modell besitzt jeder Benutzer eine Denkzeit, also die Zeit, die zwischen dem Abschluss und dem Neubeginn eines Dialogschritts vergeht. Es besteht die Auswahl zwischen der automatischen Berechnung und Eingabe der Denkzeiten, wobei die Eingabe pro Komplexitätsklasse erfolgt. Die automatische Berechnung der Denkzeiten wird mit der Zielsetzung durchgeführt, den in den Workloads vorgegebenen Durchsatz zu erreichen.

• *Applikationsserver*

Die Applikationsserver werden im WLPSizer allein durch ihre CPUs charakterisiert. Die eventuell lokal vorhandenen Platten werden nicht betrachtet, da sie hauptsächlich für das Betriebssystem genutzt werden. Somit sind für einen Applikationsserver allein die Parameter der CPUs entscheidend:

- Performance-Index: Leistungsmaß für die CPUs in SAPS (siehe 2.2.1.3.)
- CPU-Anzahl: Anzahl der CPUs
- Skalierungsfaktoren: Faktoren, die das Ansteigen der Leistung beim Multiprozessor-betrieb beschreiben.

Die Auswahl der CPU geschieht über die CPU-Bibliothek. Wählt man die gesuchte CPU aus, so werden die Parameter Performance-Index und Skalierungsfaktor automatisch der Bibliothek entnommen. Wurde ein CPU-Typ mit mindestens drei Prozessoren ausgewählt (Anzahl N), so werden für die Fälle 2,...,N-1 die Skalierungsfaktoren mit Hilfe des WLPSizers wahlweise nach Amdahl oder Gustafson [41] interpoliert.

• *Datenbankserver und IO-System*

Der Datenbankserver ist eine Erweiterung des Applikationsservers, so dass auch hier Name, CPU und Skalierungsfaktor angegeben werden muss. Die Erweiterung des Datenbankservers umfasst die folgenden Parameter:

- *DB-Hit-Ratio [%]:* Anteil an DB-Zugriffen, die aus dem Cache der DB bedient werden können und somit keinen Zugriff auf den Sekundärspeicher benötigen.

- *IO-Systemparameter:* SeekTime, Rotationsgeschwindigkeit, Blocksize, Transferrate, Cache-Hit-Ratio und Controller-Time (Parameter für eine Festplatten-Modellierung, siehe Kapitel 4 bzw. [69], [70]) sowie bei mehreren Platten die Access-Distribution, d. h. Verteilung auf die Platten (Gleichverteilung oder Sägezahnverteilung).

Die Spezifikation der CPUs und des IO-Systems werden jeweils von den Bibliotheken unterstützt, so dass die Parameter bereits durch Auswahl eines Bibliothek-Eintrags automatisch spezifiziert werden. Werte für die DB-Hit-Ratio, Cache-Hit-Ratio des IO-Systems und die Blocksize müssen den Messdaten entnommen werden.

- *Netze*

Netze werden im WLPsizer zur Verbindung der Clientgroups mit den Applikationsservern und der Applikationsserver mit dem Datenbankserver benötigt. Es können auch Netze direkt miteinander verbunden werden. Die Parameter eines Netzes beschreiben die Länge des Mediums, die Signalgeschwindigkeit, die Bandbreite, die Paketlänge, das Protokoll, den Overhead pro Paket und die maximale Paketlänge. Anhand dieser Parameter wird ein analytisches Submodell in Anlehnung an Garry Higginbottom [45] erstellt. Die Parameter eines Netzes werden durch die Netz-Bibliothek bereitgestellt.

- *Settings einer Konfiguration*

Die Settings einer Konfiguration sind Parameter, die zum Teil anhand der Messdaten berechnet werden müssen bzw. auf Erfahrungswerten basieren. Die in Tabelle 2.7 dargestellten Parameter können für jede Konfiguration spezifiziert werden.

Parameter	Einheit	Beschreibung
ComApp	Sek.	Basis-CPU-Bedarf für die Kommunikation am Appl.-Server
ComKB	Sek.	CPU-Bedarf für die Kommunikation am Appl.-Server zw. DB-Server pro KB
CPUDBSU	Sek.	CPU-Bedarf pro DBSU am DB-Server
IODBSU	Anzahl	Anzahl der IO-Zugriffe pro DBSU
RP	SAPS	Referenzleistung zur Normierung der Workloads (RP = 250 SAPS)
BT 1...5	Sek.	Basis-CPU-Bedarf für Dialogschritte der Komplexitätsklassen 1...5, wie z. B. CPU-Zeit am Dispatcher

Tabelle 2.7 Settings-Parameter

Die Referenzleistung kann nicht modifiziert werden, da für alle Konfigurationen von einer festen Referenzleistung von 250 SAPS ausgegangen wird. Die Konstanten CPUDBSU und IODBSU können anhand der Betriebssystem-Messdaten berechnet werden (siehe hierzu [129] oder [128]). Eine detaillierte Beschreibung zur Berechnung der CPUDBSU-Konstante befindet sich in [129]. Eine nähere Betrachtung und Analyse der IODBSU-Konstante befindet sich in Kapitel 3.

2.5.3. Modell-Evaluation

Die Modell-Evaluation vom WLPSizer beinhaltet die Erzeugung und Berechnung des Modells. Die definierte Konfiguration wird hierbei in ein Warteschlangennetz überführt. Bei Inkonsistenzen in der Konfiguration wird eine Fehlermeldung angezeigt. Inkonsistenzen betreffen hierbei die Topologie der Konfiguration, wie zum Beispiel, dass ein Applikationsserver nicht mit dem Datenbankserver verbunden ist.

Mit der Evaluate-Funktion kann das Warteschlangennetz mittels TOTO [74] gelöst werden. Der WLPSizer stellt die Ergebnisse dann mit Hilfe verschiedener Sichten dar (siehe folgenden Abschnitt 2.5.4.).

2.5.4. Modell-Ergebnisse

Die mit der Modell-Evaluation ermittelten Ergebnisse werden im WLPSizer in verschiedenen Sichten dargestellt und mittels drei verschiedener Dienstgüten (gut, mäßig, schlecht) bewertet (siehe 2.5.1.3.). Es folgt eine Beschreibung der verschiedenen Ergebnissichten des WLPSizers. Abschließend wird die Report-Funktion des WLPSizers dargestellt, mit der die Ergebnisse und Konfigurationsbeschreibungen zusammengefasst und exportiert werden können.

2.5.4.1. Modell-Ergebnissichten

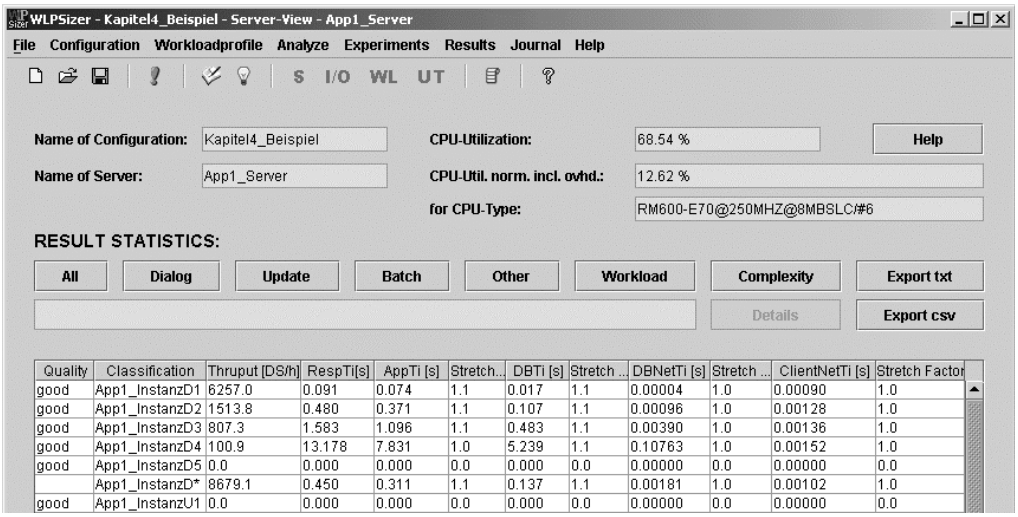
Die Ergebnisse der Modell-Evaluation werden im WLPSizer in vier verschiedenen Sichten dargestellt: Server-, Workload-, Utilization- und IO-View. Die Ergebnisse werden in gleicher Form wie die Workloadprofile, d. h. gruppiert nach Tasktyp und Komplexitätsklasse, beschrieben. Es werden für jede Klasse (Kombination aus Tasktyp und Komplexitätsklasse) Mittelwerte zur Beschreibung des Ressourcen-Verbrauchs und der Antwortzeit pro Dialogschritt sowie die Durchsätze angegeben.

Server View

Die Server-View stellt die Modell-Ergebnisse aus Sicht eines bestimmten Servers (Database- oder Application-Server) dar. Es werden die folgenden Daten angezeigt:

- Name der Konfiguration
- Name des Servers
- CPU-Auslastung des Servers laut Modellrechnung

Zusätzlich sind Analyseergebnisse über die auf diesem Server verarbeiteten Dialogschritte in einer Tabelle zusammengestellt (siehe Abbildung 2.19).



Quality	Classification	Thruput [DS/h]	RespTi[s]	AppTi [s]	Stretch...	DBTi [s]	Stretch ...	DBNetTi [s]	Stretch ...	ClientNetTi [s]	Stretch Factor
good	App1_InstanzD1	6257.0	0.091	0.074	1.1	0.017	1.1	0.00004	1.0	0.00090	1.0
good	App1_InstanzD2	1513.8	0.480	0.371	1.1	0.107	1.1	0.00096	1.0	0.00128	1.0
good	App1_InstanzD3	807.3	1.583	1.096	1.1	0.483	1.1	0.00390	1.0	0.00136	1.0
good	App1_InstanzD4	100.9	13.178	7.831	1.0	5.239	1.1	0.10763	1.0	0.00152	1.0
good	App1_InstanzD5	0.0	0.000	0.000	0.0	0.000	0.0	0.00000	0.0	0.00000	0.0
	App1_InstanzD*	8679.1	0.450	0.311	1.1	0.137	1.1	0.00181	1.0	0.00102	1.0
good	App1_InstanzU1	0.0	0.000	0.000	0.0	0.000	0.0	0.00000	0.0	0.00000	0.0

Abbildung 2.19 Server-View als Ergebnis-Sicht vom WLPSizer

Die Parameter der Server-View werden in Tabelle 2.8 beschrieben.

Parameter	Einheit	Beschreibung
Quality	---	Dienstgüte (good, moderate, bad)
Classification	---	Name der Workload, gefolgt von Tasktyp und Komplexitätsklasse der Dialogschritts
Thruput	DS/h	Durchsatz
RespTi	Sek.	Antwortzeit des Dialogschritts (ClientNetTi+AppTi+DBNetTi+DBTi)
AppTi	Sek.	Zeit am Appl.-Server inkl. Wartezeiten (=0 bei DB-Server)
Stretch-Factor (Appl.-Server)	---	Dehnfaktor: Verhältnis von Verweilzeit zur Bedienzeitanforderung am Appl.-Server
DBTi	Sek.	Zeit am DB-Server inkl. der IO-Zeiten und Wartezeiten
Stretch-Factor (DB-Server)	---	Dehnfaktor: Verhältnis von Verweilzeit zur Bedienzeitanforderung am DB-Server
DBNetTi	Sek.	Zeit auf dem Netz bzw. den Netzen zwischen Appl.- und DB-Server (inkl. Wartezeiten)
Stretch Factor (DB-Net)	---	Dehnfaktor: Verhältnis von Verweilzeit zur Bedienzeitanforderung am DB-Netz
ClientNetTi	Sek.	Zeit auf dem Netz bzw. den Netzen zwischen Appl.-Server und Clientgroups
Stretch Factor (Client-Net)	---	Dehnfaktor: Verhältnis von Verweilzeit zur Bedienzeitanforderung am Client-Netz

Tabelle 2.8 Parameter der Server-View

In der tabellarischen Ergebnisdarstellung wird das Symbol "*" als Platzhalter für den Tasktyp bzw. die Komplexitätsklasse verwendet. Die Werte dieser Zeile sind eine Zusammenfassung aller Dialogschritte mit fester Komplexitätsklasse und beliebigem Tasktyp oder umgekehrt. Dabei wird der Durchsatz als Summe aller Durchsätze der einzelnen Dialogschritte berechnet. Alle weiteren Werte sind gewichtete Mittelwerte, die auf Grundlage des Durchsatzes bestimmt werden.

Die weiteren Ergebnissichten vom WLPSizer besitzen einen vergleichbaren Aufbau und werden im Rahmen dieser Arbeit daher nur in Kurzform beschrieben, können jedoch in [129] detaillierter nachgelesen werden.

Workload-View

Die Workload-View liefert die Modell-Ergebnisse aus Sicht einer ausgewählten Workload. Es werden die folgenden Daten dargestellt:

- Name der Konfiguration
- Name der Workload
- Name der Clientgroup (Quelle der Workload)
- Anzahl der Terminals dieser Clientgroup
- Name des Servers (Ziel der Workload)
- CPU-Auslastung des Servers

In einer Tabelle werden die berechneten Zeiten für die Dialogschritte dieser Workload zusammengestellt (vergleichbar mit der Server-View). Ein wesentlicher Unterschied zur Server-View ist die zusätzliche Trennung der DB-Verweilzeit in DB-CPU und IO sowie die Angabe der Denkzeiten für die Clients.

Utilization View

In der Utilization-View werden die durch Dialogschritte verursachten Auslastungen der Systemkomponenten dargestellt. Diese Darstellung wird jeweils für eine vorher ausgewählte Workload erstellt. Es werden mit Ausnahme der Ergebnistabelle die gleichen Daten wie für die Workload-View angezeigt. Die Ergebnistabelle beinhaltet zum einen klassifiziert nach Tasktyp und Komplexitätsklasse die Dienstgüte, den Durchsatz und die Antwortzeit. Zum anderen werden für das ClientNet, den Application-Server (Ziel der Workload), das DB-Net, die DB-CPU und das IO-System die durch die Workload verursachten Auslastungen dargestellt.

IO-View

Die IO-View liefert die Auslastung des IO-Systems und die folgenden Angaben:

- Name der Konfiguration
- Name des Database-Servers
- CPU-Auslastung am Database-Server

Die IO-View liefert für das IO-System die Auslastung jeder einzelnen Festplatte in Prozent.

Reports

Im WLPSizer können Berichte über die erstellten Konfigurationen generiert werden. Es können bereits vorkonfigurierte Berichte oder einzelne Abschnitte der vorhandenen Ergebnisse und Konfigurationsbeschreibungen aufgerufen werden.

2.5.5. Modell-Validation und -Kalibrierung

Die in den Ergebnissichten dargestellten Modellergebnisse hinsichtlich Auslastungen, Antwortzeiten und Durchsätzen können im Vergleich zu den real gemessenen Leistungsmaßen zur Validation und Kalibrierung des Modells herangezogen werden. Der WLPSizer unterstützt die von TOTO angebotene Funktion der Durchsatzsteuerung, d. h. die in dem Modell verwendeten Denkzeiten werden von TOTO derart berechnet, dass die in den Workloadprofilen vorgegebenen Durchsätze exakt erreicht werden. Realisiert wird die Durchsatzsteuerung durch die Synchronisation von Lastketten, so dass z. B. „schnellere Lasten“ auf langsamere unter Verwendung von künstlichen Verzögerungen warten müssen. Schätter und Totzauer beschreiben diese Eigenschaft in Anwendung auf Warteschlangensysteme in [91] und [120]. Bei Aktivierung dieser Funktion sind die berechneten Durchsätze und Auslastungen mit den real gemessenen nahezu identisch.

In Abhängigkeit der in das Modell eingegebenen Workloadprofile müssen zur Validation und Kalibrierung der Antwortzeit pro Profil maximal 20 Antwortzeiten (nicht alle Zeilen eines Profils müssen belegt sein) der jeweiligen Dialogschrittgruppen (4 Tasktypen mit jeweils 5 Komplexitätsklassen) betrachtet werden. Bereits bei fünf Profilen können es somit bereits bis zu 100 Antwortzeitwerten werden. Eine Kalibrierung aller Antwortzeiten ist somit nahezu unmöglich und bedarf einer Priorisierung weniger Werte. Typischerweise sind bei der Modellierung eines R/3-Systems insbesondere die Dialoglasten von entscheidender Bedeutung. Die Hauptlast der Dialogverarbeitung wird den Komplexitätsklassen 2 und 3 zugeordnet, so dass für die Kalibrierung versucht wird, die Antwortzeiten dieser beiden Klassen entsprechend den real gemessenen anzupassen. Bei der Kalibrierung der Antwortzeit kann nach Lazowska [63] in einem Multi-Class-Modell eine Abweichung von 10% bis 30% toleriert werden. Verwendet man im WLPSizer keine Durchsatzsteuerung, so sind auch Durchsätze und Auslastungen für die Validation und

Kalibrierung des Modells näher zu betrachten. Auch hierfür gibt Lazowska in [63] Toleranzgrenzen an.

2.6. Methoden und Fallstudien

Die im WLPSizer generierten Modelle können zum einen ein bereits existierendes System in seiner Topologie und Performance (Antwortzeiten, Auslastungen und Durchsätze) nachbilden (Basismodellierung) oder aufbauend auf ein solches Modell Prognosen für zukünftige Szenarien liefern (Phasen 3 und 4 der R/3-Kapazitätsplanung). Mit Hilfe des WLPSizers können Prognosen in den folgenden Aufgabengebieten erstellt werden:

- R/3-Release-Wechsel
- Hardware-Upgrade
- Zukünftige Laststeigerungen

Die Methoden zur Behandlung dieser Aufgaben und die Anwendungsmöglichkeiten des WLPSizer in diesen Gebieten werden in den folgenden Abschnitten beschrieben. Abschließend werden als Fallstudien drei ausgewählte Projekte dargestellt.

2.6.1. R/3-Release-Wechsel

Ein R/3-Release-Wechsel bedeutet häufig einen Ressourcenmehrbedarf an CPU-Zeit, DB-Zeit und IO. Zur Veranschaulichung werden in der folgenden stark vereinfachten Abbildung 2.20 die SAPS-Leistungen eines Server für verschiedene R/3-Release-Wechsel beschrieben.

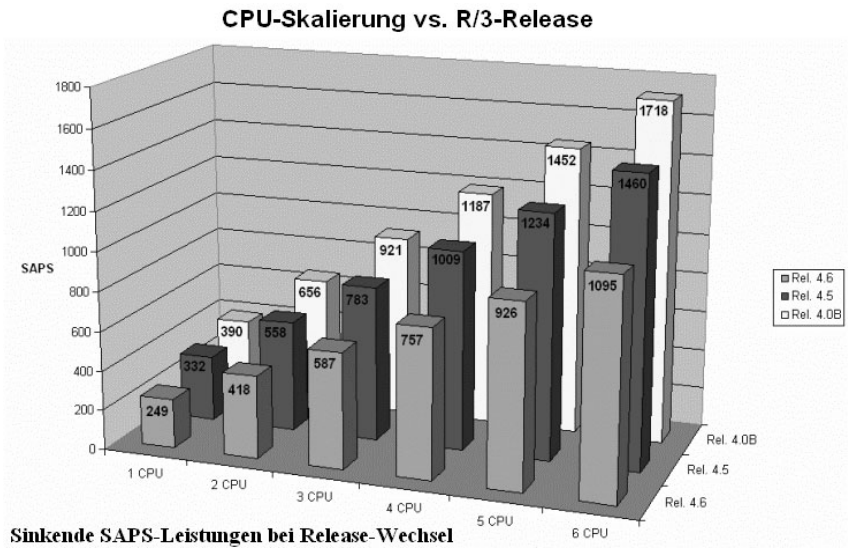


Abbildung 2.20 SAPS-Leistung vs. R/3-Release

Entscheidend ist somit die Frage, ob das System nach vollzogenem Release-Wechsel die Dienstgüteanforderungen bzgl. Durchsatz und Antwortzeit weiterhin einhält. Mittels WLPSizer kann der bevorstehende Release-Wechsel auf zwei verschiedene Arten modelliert werden:

- *Anpassung der SAPS-Werte*

  Die Server werden im Modell mit Leistungskennzahlen in SAPS beschrieben, die aufgrund von Benchmarkmessungen ermittelt wurden. Da SAPS einem Durchsatzmaß entspricht und jedes R/3-Release einen unterschiedlichen Ressourcenbedarf aufweist, muss jedem Server für jedes R/3-Release eine entsprechende SAPS-Kennzahl zugeordnet werden (siehe Tabelle 2.9).

Server-Name	#CPU	1	2	3	4	5	6
R/3-Release	**Faktor**						
X	35 %	390	656	921	1187	1452	1718
Y	15 %	332	558	783	1009	1234	1460
Z	25 %	249	418	587	757	926	1095

Tabelle 2.9 SAPS-Tabelle für verschiedene R/3-Releases

Im Modell können somit die jeweiligen Server mit Hilfe der Bibliotheken ausgetauscht werden (siehe Abbildung 2.21). Es existieren hierfür bereits verschiedene CPU-Bibliotheken für die R/3-Releases.

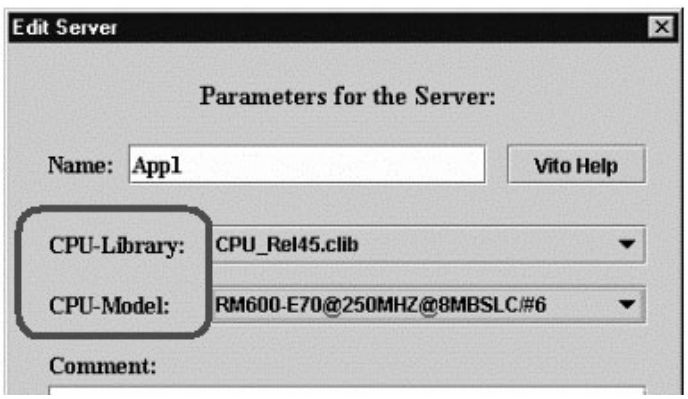


Abbildung 2.21 Verringerung der SAPS-Leistung im Modell für den Release-Wechsel

- *Erhöhung der CPU-Zeit-Mehrbedarfe für die Dialogschritte*

  Auf Basis von Benchmarkmessungen können für die Business Components eines R/3-Systems Faktoren berechnet werden, die den CPU-Zeit-Mehrbedarf für Dialog, Update und Datenbank im Falle eines Release-Wechsels angeben (siehe Tabelle 2.10). Es kann somit im Vergleich zur vorherigen Methode eine detaillierte Parametrisierung des Modells erfolgen.

R/3-Modul	DIA	UPD	DB
Finanzwesen (FI)	1,47	1,00	1,14
Materialwirtschaft (MM)	1,13	1,03	1,03
Produktionsplanung (PP)	1,50	1,00	1,67
Vertrieb (SD)	1,48	1,01	1,20
Workflow-Management (WM)	1,46	0,91	1,26

Tabelle 2.10 Faktoren für CPU-Zeiten der App- und DB-Server für einen Release-Wechsel

Der jeweilige CPU-Zeit-Mehrbedarf wird im Modell durch Modifikation der Workloads berücksichtigt, indem die Spalte CPUTI der Workloads mit dem entsprechenden Faktor multipliziert wird (siehe Abbildung 2.22). Der CPU-Zeit-Mehrbedarf für die Datenbank wird durch Änderung der CPUDBSU-Konstante realisiert.

T	C	ratio	CPUTI [ms]	DBSU	DBKB [KB]	RespTime [ms]	DBTime [ms]	LockTime [ms]	QueueTime [ms]
D	1	0,541	14,654	70,645	0,476	124,640	108,750	0,145	0,201
D	2	0,094	64,584	435,918	15,739	487,664	418,977	1,681	0,165
D	3	0,040	294,171	1.784,767	86,423	2.718,844	2.389,842	4,483	0,175
D	4	0,009	3.860,400	18.673,049	2.380,628	20.357,157	15.349,553	21,356	0,262
D	5	0,001	26.929,687	535.277,125	25.285,747	235.416,309	213.958,521	43,875	0,000
U	1	0,048	5,645	131,894	0,713	13,688	8,482	0,003	0,229
U	2	0,178	25,276	596,005	31,251	63,886	41,722	0,221	0,252
U	3	0,028	85,801	1.526,503	176,137	586,209	507,172	0,265	0,224
U	4	0,001	329,327	5.748,615	467,253	1.432,547	1.148,376	0,305	0,000
U	5	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
B	1	0,000	0,000	171,000	0,132	27,543	15,689	0,214	0,000
B	2	0,044	45,154	336,740	29,762	270,996	207,950	0,048	0,082
B	3	0,007	128,227	1.974,489	45,324	736,797	413,915	20,016	1,359
B	4	0,001	911,184	11.760,895	242,769	3.275,340	2.372,602	52,653	0,000
B	5	0,000	64.531,248	255.937,000	259.131,063	224.584,736	204.016,992	3,062	0,000
O	1	0,009	18,186	68,803	0,030	2.039,263	223,247	0,000	1.792,123
O	2	0,004	44,271	406,646	24,806	2.414,767	31,334	0,000	588,521
O	3	0,001	164,063	1.825,583	340,089	17.878,779	176,282	0,000	7.627,583
O	4	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
O	5	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
*	*	1,000	77,500	795,529	64,972	575,250	453,603	0,811	26,246

Abbildung 2.22 Erhöhung des CPU-Zeit-Mehrbedarfs für die Dialogschritte

2.6.2. Hardware-Upgrade

Die Durchführung eines Release-Wechsels, die Installation weiterer Business Components oder der Anstieg der Benutzeranzahl stellt häufig die Ursache für einen Hardware-Upgrade dar. Ein Hardware-Upgrade kann in zwei Fälle unterschieden werden:

- *Vertikaler Upgrade*

  Die Modellkomponenten werden durch leistungsstärkere Bausteine ersetzt. So könnten zum Beispiel in einem Server weitere CPUs eingesetzt oder schnellere Festplatten im IO-System verwendet werden. Der Austausch der Modellkomponenten wird im WLPSizer mit Hilfe der Bibliotheken realisiert (siehe Abbildung 2.23).

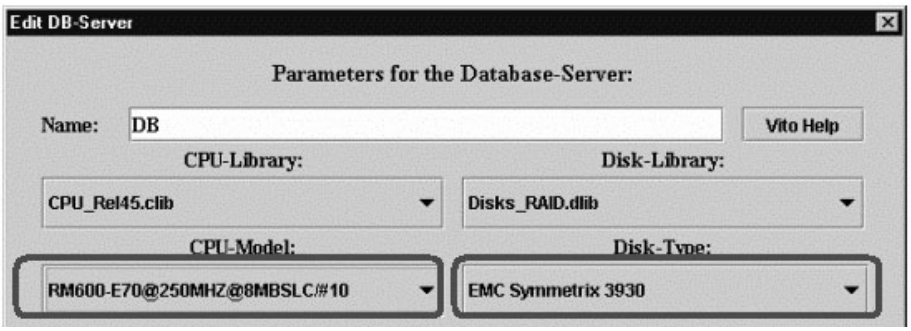


Abbildung 2.23 Vertikaler Upgrade am Beispiel des DB-Servers

- *Horizontaler Upgrade*

  Das Modell wird durch weitere Komponenten ergänzt. Es werden zum Beispiel weitere Applikationsserver in das Modell eingefügt, die für eine verbesserte Lastverteilung genutzt werden können.

2.6.3. Zukünftige Laststeigerungen

Die Modellierung zukünftiger Laststeigerungen kann in drei Arbeitsschritte unterteilt werden:

- *Lastprognose*

  Unter Lastprognose versteht man die Beschreibung der zukünftigen Last, wie zum Beispiel eine allgemeine Durchsatzsteigerung von 100% (DS/h) oder die Beschreibung des Lastprofils einer zukünftigen Business Component mit zu erwartendenden Durchsatz- und Ressourcen-Anforderungen.

- *Erstellung des Lastmodells*

  Das Lastmodell wird durch Modifikation bzw. Definition neuer Workloadprofile erstellt bzw. durch Veränderung der in den Workloads definierten Durchsätze. Der WLPSizer bietet hierzu die Funktion Experiment zur Erhöhung der Durchsätze aller Workloads mittels Eingabe eines Faktors.

- *Modellexperimente*

  Modellexperimente dienen zum Beispiel der Untersuchung von einzuhaltenden Dienstgüten, wie zum Beispiel „Antwortzeit < 2 Sekunden für Dialogschritte der Komplexitätsklasse 3“. Weiterhin können notwendige Upgradevarianten betrachtet werden für zum Beispiel die Erhöhung der CPU-Anzahl oder die Hinzunahme weiterer Applikationsserver.

Die Ergebnisse der Modellexperimente können dann wie in Abbildung 2.24 dargestellt werden. Für einen Server werden unter verschiedenen Lastszenarien (Z-Achse) die Antwortzeiten für die Dialogschritte vom Tasktyp Dialog der Komplexitätsklassen 2 und 3 (Y-Achse) für verschiedene Konfigurationsmöglichkeiten (X-Achse) dargestellt.

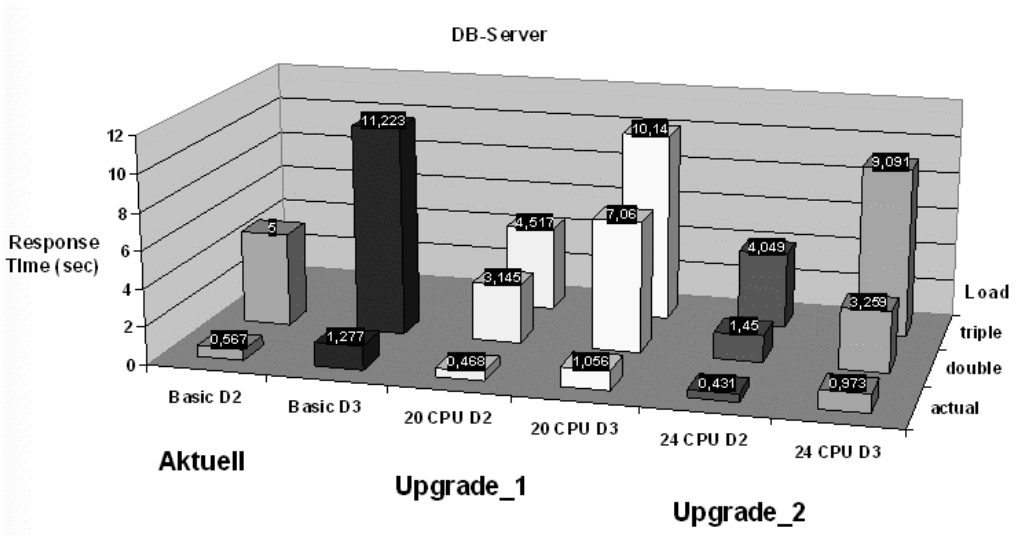


Abbildung 2.24 Prognoseergebnisse für Dialogschritte der Gruppen D2 und D3

2.6.4. Fallstudien

Bislang wurden in dieser Arbeit Konzepte und Methoden zur Leistungsprognose für SAP R/3-Systeme und das dafür verwendete Modellierungswerkzeug WLPSizer vorgestellt. Anhand typischer Problemstellungen wurden die Vorgehensweisen zur Leistungsprognose anhand des Werkzeugs WLPSizer demonstriert. Es folgen abschließend drei aus der Praxis entnommene Fallstudien, die jeweils im Rahmen einer R/3-Kapazitätsplanung behandelt wurden.

Fall 1: Konfigurationsalternativen für einen Anstieg der Benutzerzahl

In diesem Projekt wurde ein System mit vier Application-Servern und einer Datenbank sowie 11 R/3-Instanzen untersucht, das einen R/3-Modulmix (SD, CO, FI, MM, PP, PM, HR) und das von einer kleinen jedoch stetig wachsenden Benutzerzahl verwendete SAP-Modul CATS (Cross Application Time Sheet) zur modulübergreifenden Zeiterfassung der Mitarbeiter, betreibt. Das System besitzt 4000 Anwender, wobei im Durchschnitt 1000 „logged-in user“ existieren und 165 „concurrent user“. Als „logged-in user“ werden die im System angemeldeten Anwender bezeichnet, die jedoch nicht zwingend aktiv sein müssen, d. h. die Verarbeitung von Transaktionen und Dialogschritten verursachen. Die „concurrent user“ sind laut Definition Benutzer, die gleichzeitig aktiv sind, wobei gleichzeitig bedeutet, dass die Benutzer innerhalb eines 10 Sekunden-Intervalls mindestens einen Dialogschritt ausführen. Das verwendete Monitoring-Programm myAMC.LNI liefert dann einen Mittelwert der „concurrent user“ pro Minute.

Die Aufgabenstellung des Projekts bestand aus der Analyse der folgenden beiden Lastszenarien:

- Anstieg der Benutzerzahl des gesamten Modulmixes von 1000 logged-in users auf 2000, 3000, 4000 oder 5000 logged-in users. Es sollten hierzu die anhand der CPU-Auslastungen ermittelten Hochlastzeitfenster verwendet werden.
- Anstieg der Benutzerzahl des SAP-Moduls CATS von 100 concurrent users auf 1000, 2000 und 3000 concurrent users. Die Anforderung bestand somit in der Modellierung von einer Stunde mit den genannten concurrent users zusätzlich zum existierenden R/3-Modul-Mix (siehe Abbildung 2.25). Diese Szenarien stellen Extremsituationen dar. In der Regel

sind in einem System deutlich weniger concurrent users (mindestens Faktor 10 geringer) vorhanden.

Es sollte untersucht werden, ob die bestehende Hardware ausreichende Ressourcen für die beschriebenen zwei Lastszenarien zur Verfügung stellt. Bei unzureichender Performance des existierenden Systems sollten System-Upgrades vorgeschlagen werden, welche die Last bewältigen können. Es sollte hierbei von einer maximalen Antwortzeit der Dialogschritte vom Tasktyp Dialog und Komplexitätsklasse 3 von 1,5 bis 2 Sekunden ausgegangen werden.

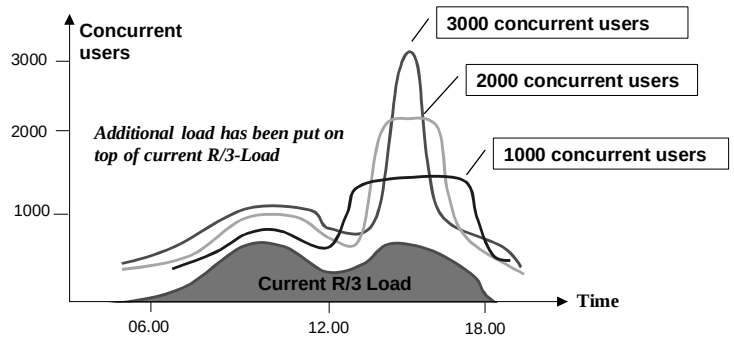


Abbildung 2.25 Steigende Benutzerzahl einer Business Component

Sowohl Betriebssystem- und Datenbank- als auch R/3-Daten wurden gemessen und ausgewertet. Anhand der CPU-Auslastungskurven der Betriebssystemdaten wurden die Hochlastzeitfenster ausgewählt. Es wurden für diese Zeitfenster die entsprechenden R/3-Daten als Workloadprofile verdichtet und in das Modell eingegeben. Nach Erstellung des Basismodells wurden die gewünschten Lasterhöhungen für den gesamten Modul-Mix in das Modell eingetragen. Dazu konnten zum einen die Zahl der User oder die Durchsätze der Workloads erhöht werden.

Für die Betrachtung der CATS-Last wurden die R/3-Daten (R/3-Statistik-Sätze) nach den CATS-Transaktionen gefiltert und analysiert. Anhand der vorliegenden Daten und der Information des Kunden über die bereits existierende Zahl der CATS-User wurde ein Profil für die CATS-Last definiert und in das Modell eingetragen. Bei gleichbleibender Last des Modul-Mixes konnte der Durchsatz der CATS-Last somit erhöht werden.

Je nach Auslastung der Konfigurationskomponenten mussten diese durch leistungsstärkere ersetzt bzw. es mussten weitere Komponenten hinzugefügt werden. Beim Hinzufügen z. B. weiterer Applikationsserver war darauf zu achten, dass die Workloads auf die einzelnen Server sinnvoll verteilt wurden. Zum Teil mussten hierzu einzelne Workloadprofile doppelt, jedoch mit angepasstem (z. B. halbiertem) Durchsatz, eingefügt werden, so dass diese auf mehrere Server verteilt werden konnten.

Das Ergebnis des Projekts umfasste die vom Modell berechneten Konfigurationen für die vom Kunden gewünschten Benutzerzahlen. Eine Modellkonfiguration wurde für eine vorgegebene Benutzerzahl als ausreichend dimensioniert betrachtet, wenn ausgewählte Gruppen von Dialogschritten, wie z. B. D2 und D3 aller Workloadprofile, die Dienstgüte „gut“ besaßen (siehe auch Abbildung 2.24). Die Auswahl der zu betrachtenden Dialogschritte orientierte sich hierbei an dem konkreten Projektziel und den real vorhandenen R/3-Daten. So sollte bei diesem System primär die Dialoglast analysiert werden. Die Hauptlast wurde von den Benutzern durch die Komplexitätsklassen 2 und 3 verursacht. Somit wurden diese Komplexitätsklassen für die Dimensionierung der Konfiguration herangezogen. Dialogschritte der Komplexitätsklasse 1 sollten in einem ausreichend dimensionierten System immer die Dienstgüte „gut“ besitzen.

Fall 2: Konfigurationsvorschlag für das Business Information Warehouse (BW) von SAP

Die Aufgabe des Projekts bestand darin auf Basis von realen Messdaten zu untersuchen, ob die im Sizing ermittelte Systemkonfiguration für das Business Information Warehouse die zu erwartende Last mit gewünschter Dienstgüte verarbeiten kann und es sollten gegebenenfalls Systemalternativen vorgeschlagen werden. Das Projekt startete noch während der Implementierungsphase des BW-Systems, wobei ein Großteil des Systems bereits installiert war.

Das Business Information Warehouse ist ein auf der R/3-Basis aufsetzendes eigenständiges R/3-System. Die Hauptaufgabe des BW-Systems ist die Erstellung von Reports auf Basis von aktuellen Unternehmensdaten. Hierzu müssen die Daten aus den produktiven R/3-Systemen in das BW-System „hochgeladen“, aggregegiert und den BW-Benutzern als Reports zur Verfügung gestellt werden. Eine detaillierte Beschreibung des BW-Systems von SAP befindet sich in [43]. Die Daten eines solchen Systems werden in Form von InfoCubes verwaltet. Ein InfoCube, bestehend aus einer Menge von verschiedenen Datenbanktabellen, unterstützt multidimensionales „slicing“ und „dicing“ für OLAP-Analysen. Eine Benutzeranfrage (Query) greift auf eine Untermenge eines oder mehrerer InfoCubes zu. Bei häufig auftretenden Querys werden InfoCube-Aggregate definiert, die eine Untermenge der InfoCubes bilden und die von der Query geforderten Daten bereits in der entsprechenden Verdichtung beinhalten, so dass eine schnelle Verarbeitung der Query garantiert ist.

Die Definition von InfoCube-Aggregaten hat zur Folge, dass Daten mit jedem weiteren Aggregat mehrfach in der Datenbank vorliegen. Es können somit nicht für alle Queries vorgefertigte Aggregate definiert werden, da ansonsten die Speicheranforderungen an das IO-System zu hoch sind. Der Platzbedarf eines BW-Systems liegt bereits heute im TeraByte-Bereich. Eine weitere Grenze für die Anzahl der InfoCubes und deren Aggregate stellen die Daten-Uploads mit den nachfolgenden Aktualisierungen dar. Das Business Warehouse muss tagesaktuelle Daten liefern. Das bedeutet, dass in zyklischen Abständen von allen produktiven R/3-Systemen die Daten extrahiert und in das BW-System hochgeladen werden müssen (Upload bzw. Load Phase). Häufig geschieht dies über das Operational Data Store (ODS) in das die Daten von den R/3-Systemen geladen werden, von dem aus dann die InfoCubes aktualisiert werden. Mit Erhöhung der Anzahl der InfoCubes sowie deren Aggregate steigt dementsprechend auch der Zeitbedarf zur Aktualisierung der Datenbestände, so dass die maximal zur Verfügung stehenden Zeitfenster für den Daten-Upload und der zugehörigen InfoCube-Aktualisierung nur eine begrenzte Anzahl von InfoCubes und Aggregaten in einem BW-System ermöglichen.

In Anlehnung an den BW-Benchmark [87] wurden in diesem Projekt die beiden Lastszenarien „Daten-Upload“ für das Hochladen der Daten in das ODS und die Aktualisierung der InfoCubes und deren Aggregate sowie die „Query-Verarbeitung“, d. h. die Zugriffe der Anwender auf das BW-System, betrachtet. Für beide Lastszenarien wurden Messungen durchgeführt, auf deren Basis Modelle mit den zukünftig zu erwartenden Lasten erstellt wurden, welche die hierzu notwendigen Systemkonfigurationen prognostizierten.

Das Szenario „Daten-Upload“ umfasst das Hochladen der benötigten R/3-Daten aus verschiedenen R/3-Systemen und die nachfolgende Aktualisierung der Datenbestände des BW-Systems. Der Upload wird im BW-System in Form von Datenpaketen beschrieben, die wiederum aus einzelnen Datensätzen bestehen. Gemessen wurde in diesem Fall ein Daten-Upload bestehend aus 65 Datenpaketen mit jeweils 30 000 Datensätzen, der vom BW-System innerhalb einer Stunde verarbeitet wurde. Aus R/3-Sicht wurden hierzu 45 Dialogschritte verwendet, die dann als Workloadprofil in das Modell eingingen. Mit Hilfe des Modells sollte dann untersucht werden, wieviele Datenpakete pro Stunde in das BW-System hochgeladen werden können, wobei die Systemkonfiguration dem Sizing-Resultat entsprach. Anhand der im Modell resultierenden Systemauslastung wurde die maximale Anzahl an zu verarbeitenden Datenpaketen pro Stunde ermittelt.

Für das Szenario „Query-Verarbeitung“ wurden von Seiten des Kunden drei verschiedene Anfragen (Queries) vorgegeben, die jeweils unterschiedlichen Komplexitäten entsprachen. Die

Komplexitätsdefinition basierte dabei auf der zuzugreifenden Datenmenge und dem Zugriffsziel, d. h. ODS, Aggregat oder InfoCube. Weiterhin wurden für jede Query jeweils zwei unterschiedliche Varianten definiert, die sich in der Anzahl der verwendeten Navigationsschritte unterschieden. Ein Navigationsschritt bedeutet hierbei, dass im erstellten Report weitere Detailinformationen abgefragt werden, die vom BW-System nachgliefert werden müssen.

Alle drei Queries wurden auf R/3- und auf Betriebssystem-Ebene gemessen, so dass jeweils ein Workloadprofil erzeugt werden konnte. Auf Basis dieser Profile wurde ein Modell erstellt, das mit den Durchsatzvorgaben des Kunden parametrisiert wurde. Anhand der im Modell resultierenden Antwortzeiten pro Query und der für die Systemkomponenten resultierenden Auslastungen, wurde zum einen festgestellt, dass die im Sizing ermittelte Systemkonfiguration für die geforderte Last mit den zugehörigen Dienstgüten nicht ausreichende Kapazitäten aufwies und es wurde somit zum anderen eine der Lastanforderung entsprechende Systemkonfiguration mittels Modellierung vorgeschlagen.

Fall 3: Modellierung auf Basis von Sizing-Input

In diesem Projekt wurde auf Basis der Kundenangaben ein Sizing erstellt. Für die Kerntransaktionen wurden Messungen auf R/3- sowie auf Betriebssystem-Ebene in einem Testsystem durchgeführt. Mit Hilfe der Messdaten konnte für jede Transaktion ein Workloadprofil erstellt werden, das anschließend in einem WLPSizer-Modell verwendet wurde. Das Mengengerüst des Sizings in Form von Durchsätzen diente der Parametrisierung des Modells. Die Modellergebnisse konnten im Folgenden zur Überprüfung der Sizing-Ergebnisse herangezogen werden. Somit kann parallel zur Implementierung und Installation des R/3-Systems das Modell mit weiteren Messungen verbessert werden, wodurch eine sukzessive Überprüfung der Sizing-Ergebnisse stattfinden kann.

Hiermit soll veranschaulicht werden, dass mit Hilfe der R/3-Kapazitätsplanung bereits in frühen Phasen der R/3-Einführung ein kontinuierlicher Prozess der Messung und Überprüfung der Prognosen beginnen kann und sollte. Je früher Diskrepanzen in der geplanten Hardware-Dimensionierung und der tatsächlichen Konfiguration aufgrund der realen Last gefunden werden, desto kostengünstiger wirkt es sich für die betreffende Firma aus. Zum Teil werden durch das begleitende Monitoring und Messen auch ressourcenintensive Transaktionen entdeckt, die während der Entwicklungsphase noch weiter untersucht werden können.

Existiert für einen Kunden noch kein Testsystem, so kann auf Basis der Sizing-Ergebnisse ein Modell erstellt werden. Die Sizing-Ergebnisse liefern SAPS-Werte, die in Form von Workloadprofilen in das Modell eingegeben werden können. Zum einen kann hierzu ein den SD-Benchmark beschreibendes Workloadprofil verwendet werden. Da die Leistungskennzahl SAPS einem Durchsatzmaß entspricht (siehe 2.2.1.3.), kann der entsprechende SAPS-Wert mittels dem Workload-Durchsatz beschrieben werden. Zum anderen können mittels eines transaktionsbasierten Sizing SAPS-Werte pro Transaktion angegeben werden. Da SAPS-Werte in DBSU-Angaben umgerechnet werden können, kann jeder Transaktion ein DBSU-Verbrauch zugeordnet und dementsprechend in eine Komplexitätsklasse eingeordnet werden. Es ist somit möglich, Workloadprofile zusammengesetzt aus einzelnen Transaktionen zu definieren. Das Modell kann dann mit zunehmender Installation und Implementation des R/3-Systems mit neu gewonnenen Messdaten bereichert werden.

KAPITEL

3

IO-LASTCHARAKTERISIERUNG FÜR SAP R/3

In diesem Kapitel wird ein Konzept zur IO-Lastcharakterisierung für SAP R/3 dargestellt. Es werden Zusammenhänge von R/3-Performance-Daten und IO-Lasten anhand verschiedener IO-Lastszenarien untersucht. Ziel ist es hierbei, eine auf der IO-Lastcharakterisierung beruhende IO-Prognose mit Hilfe des in Kapitel 4 und 5 zu beschreibenden IO-Modells zu entwickeln. Zuvor werden bereits existierende Methoden und Vorgehensweisen zum IO-Sizing und zur IO-Prognose im Anwendungsfall SAP R/3 vorgestellt.

3.1. IO-Sizing und -Prognose für SAP R/3

Es existiert eine Vielzahl an Publikationen, die sich mit den Themen IO-Messung und IO-Modellierung beschäftigen (siehe Kapitel 4 und 5), wohingegen Veröffentlichungen zur Beschreibung und Analyse der IO-Lasten und IO-Leistungsprognose im Anwendungsfall SAP R/3 mit Ausnahme derjenigen von Mißbach und Hoffmann [72] nicht existieren. In [72] wird ein Ansatz zum IO-Sizing (Dimensionierung von IO-Systemen) für SAP R/3-Systeme mit der Zielsetzung beschrieben, dass in Abhängigkeit der SAPS-Leistung eines Systems dessen IO-Aktivität bestimmt werden kann. Nachfolgend wird dieses Konzept zum IO-Sizing sowie in der Praxis verwendete Verfahren zur Dimensionierung von Speichersubsystem dargestellt.

3.1.1. IO-Sizing

In [72] wird ein Konzept zur Dimensionierung von Speichersubsystemen auf Basis von Sizing-Ergebnissen in Form von SAPS-Werten vorgestellt. Das Ziel ist hierbei, Kennzahlen zu definieren, die eine Abhängigkeit der IO-Aktivität von den SAPS-Leistungen quantifizieren. Es wäre somit möglich, anhand der SAPS-Werte die erforderlichen IO-Durchsätze zu berechnen und ein entsprechend performantes Speichersubsystem für das R/3-System einzuplanen, das dann mit der erforderlichen Anzahl Festplatten und IO-Kanälen ausgestattet wird. Mißbach und Hoffmann unterscheiden im folgenden zwischen Netto-IO-Anforderungen (Frontend IOPS) und Brutto-IO-Anforderungen (Backend IOPS), wobei die Bezeichnung IOPS stellvertretend für IOs pro Sekunde steht:

- *Frontend IOPS:* Die Frontend IOPS sind durch die aus Applikationssicht generierten Zugriffe auf das IO-System definiert.
- *Backend IOPS:* Die Backend IOPS beschreiben die ausgehend von den Frontend IOPS resultierenden IO-Durchsätze auf die einzelnen Festplatten bei der Betrachtung von RAID-Systemen. Diese Anzahl ist abhängig vom verwendeten RAID-Level. Für RAID-Level 1 würden z. B. zusätzlich zu den Frontend IOPS weitere IO-Operationen zur Spiegelung der Daten generiert werden. In Abhängigkeit des RAID-Levels sind unterschiedliche Zuschläge auf die Frontend IOPS zu veranschlagen, wie z. B. 20 % für Raid-1 und 80 % für Raid-5 [72].

In [72] wurden IO-Kennzahlen in Form von IOPS pro SAPS zum einen für den SD-Benchmark und zum anderen für die Last von Produktivsystemen ermittelt. Für den SD-Benchmark wurden die Releases 3.1x und 4.0B mit den Datenbanksystemen Oracle und DB2 sowie dem Betriebssystem Windows NT untersucht. Es ergab sich für das Release 3.1x ein Wert zwischen 0,37 (mit 30 Platten) und 0,40 (mit 58 Platten) IOPS pro SAPS. Es ist zu vermuten, dass durch die Erhöhung der Festplattenanzahl ebenso der Parallelitätsgrad zur IO-Verarbeitung angestiegen ist und entsprechend auch der IO-Durchsatz. Die Messungen unter Release 4.0B ergaben 0,28 IOPS pro SAPS, wobei bei diesen Messungen die Festplattenanzahl nahezu konstant blieb.

Bei der Interpretation der Ergebnisse für den SD-Benchmark ist zu beachten, dass dieser nur eine sehr geringe IO-Last generiert und diese hauptsächlich aus lesenden und zufällig verteilten IO-Zugriffen besteht. Sequentielle Zugriffe sind beim SD-Benchmark nicht vorhanden und schreibende Zugriffe werden einzig beim Database-Checkpoint generiert [72], der während des gesamten Benchmarks nur einmal durchgeführt wird.

Neben dem SD-Benchmark wurden von Mißbach und Hoffmann auch Produktivsysteme hinsichtlich ihrer IO-Aktivität analysiert. Es ergaben sich hierbei Werte zwischen 0,5 und 0,6 IOPS pro SAPS (Relase 3.1 bis 4.5B, Oracle, 8 KB Blockgröße). Als Faustregel wird in [72] somit für ein System mit 10.000 SAPS und 100 % CPU-Auslastung ein Speichersubsystem mit einer Verarbeitungskapazität zwischen 5.000 und 6.000 IOPS empfohlen.

Die hier vorgestellten Kennzahlen für eine Beschreibung der IO-Aktivität sind nach Mißbach und Hoffmann nur als sehr grobe Abschätzung anzusehen, da die IO-Aktivität durch eine Vielzahl an Faktoren beeinflusst wird, wie z. B. Anwendungs- und DB-Versionen, Ausrichtung nach OLAP (Online Analytical Processing, z. B. Business Information Warehouse) oder OLTP (Online Transactional Processing, z. B. SD-Benchmark).

3.1.1.1. IO-Sizing in der Praxis

Die derzeit in der Praxis verwendeten Sizing-Tools (siehe z. B. [42]) liefern neben den Leistungsangaben der Server in SAPS auch die Speicherplatzbedarfe für das Speichersubsystem. Es werden hingegen keine für die IO-Performance relevanten Kennzahlen, wie z. B. IO-Durchsatzanforderungen beschrieben. Somit wird anhand der vom Sizing-Werkzeug ermittelten Speicherkapazität das dazu notwendige Speichersubsystem ausgewählt. Mit der fortschreitenden Entwicklung der Festplattenkapazitäten besteht hierbei die Gefahr, dass sich die R/3-Daten auf immer weniger Festplatten verteilen, da sich die Festplattenkapazitäten im Gegensatz zu den Speicheranforderungen von R/3 nach Hennesey und Patterson [44] pro Jahr verdoppeln. Mit sinkender Festplattenanzahl verringert sich jedoch auch der Parallelitätsgrad zur IO-Verarbeitung und damit auch die Performance in Form von IO-Durchsätzen. Dieses Phänomen wurde auch bei den Ergebnissen von Mißbach und Hoffmann für den SD-Benchmark ersichtlich. Ebenso wie die Zahl der Festplatten ist auch die Zahl der IO-Kanäle ein entscheidender Performance-Faktor, der jedoch anhand der Speicheranforderungen nicht entschieden werden kann. Es wird somit deutlich, dass eine stärkere Berücksichtigung der Performance von Speichersubsystemen im

Sizing-Prozess notwendig ist. Der in [72] beschriebene Ansatz zur IO-Lastcharakterisierung auf Basis von SAPS-Werten bietet hierfür eine gute Basis.

3.2. IO-Lastcharakterisierung für SAP R/3

In diesem Abschnitt wird ein Konzept zur IO-Lastcharakterisierung beschrieben, das im Rahmen der R/3-Kapazitätsplanung und insbesondere als Basis für die IO-Performance-Prognose angewendet werden soll. Das Ziel ist es hierbei, in Abhängigkeit der R/3-Last die IO-Aktivität zu beschreiben. Betrachtet man ein R/3-System, so resultiert die IO-Last aus der Datenbankaktivität, die aufgrund von im R/3-System angestoßenen Transaktionen verursacht wird. Nach Kapitel 2 existiert mit den Database Service Units (DBSU) eine Maßeinheit zur Beschreibung der Datenbankaktivität pro R/3-Dialogschritt.

In Anlehnung an das Konzept von [72] wird nachfolgend für die R/3-Kapazitätsplanung eine Abhängigkeit zwischen der IO-Aktivität und dem Parameter DBSU untersucht. Die Untersuchung wird anhand von drei verschiedenen Szenarien auf Basis realer Messungen durchgeführt. Die beiden Extrema bilden der SD-Benchmark als IO-armes Lastszenario und eine Komponente „Large Tables“ der Toolbox SSQJ, die ein extrem hohes IO-Aufkommen besitzt. Schließlich wird ein Produktivsystem hinsichtlich der IO-Aktivität analysiert. Für jedes Szenario werden Messdaten auf Basis von SAP R/3 (myAMC.LNI), dem Betriebssystem (z. B. IOSTAT, SAR) und der Datenbank (Oracle: UTLB/ESTAT) gemessen. Eine detaillierte Auflistung der Messergebnisse befindet sich in Anhang A, so dass in den folgenden Abschnitten nur auf die wesentlichen Parameter eingegangen wird.

3.2.1. Szenario 1: SD-Benchmark

In den folgenden Abschnitten wird detailliert auf zwei SD-Benchmarks eingegangen, die jeweils auf dem gleichen System jedoch mit unterschiedlicher Anzahl (120 und 150) simulierter SD-User durchgeführt werden. In Abschnitt 3.2.1.5. wird ein Weltrekord-SD-Benchmark mit 23.000 SD-Usern beschrieben und im anschließenden Abschnitt ein Vergleich der drei Benchmarks vorgenommen.

Die zwei SD-Benchmarks mit 120 und 150 SD-Usern werden auf einem Zentralserversystem (PRIMEPOWER M1000, 4 CPUs) durchgeführt. Das System basiert auf dem Betriebssystem Suse Linux 7.2 und dem R/3 Release 4.6c SR2 sowie der Datenbank Oracle 8i Enterprise Edition 8.1.7.0.0.

Die Benchmarks werden von IOSTAT, SAR, VMSTAT und einem Perl-Skript IOPL, das direkt auf die IO-Statistiken im Kernel zugreift, überwacht. Gleichzeitig wird die Messung mit myAMC.LNI, das die vollständigen SAP-Statistiksätze abgreift (in Form der V2-Records) und dem UTLB/ESTAT von Oracle, das die Oracle-Performance-Statistiken sammelt, begleitet.

In der in Anhang A befindlichen Tabelle wird zwischen Gesamtlaufzeit und Sampling-Intervall (Hochlastphase) unterschieden. Als Gesamtlaufzeit wird hierbei diejenige Zeit bezeichnet, die der gesamte Benchmark benötigt. Die Hochlastphase beschreibt hingegen nur den Zeitraum, zu dem alle SD-User parallel aktiv sind, d. h. die Anfangs- und Endphase in denen die SD-User wenige Sekunden zeitversetzt gestartet bzw. beendet werden, sind nicht berücksichtigt. Es folgt eine Beschreibung der verschiedenen Messergebnisse auf Ebene des Betriebssystems, der Datenbank und des R/3-Systems jeweils für die Gesamtlaufzeit.

3.2.1.1. Auslastungsprofile

Betrachtet man die CPU-Auslastung der 120 und 150 SD-User-Benchmarks (siehe Abbildung 3.1 und Abbildung 3.2), so erkennt man die für einen SAP-Benchmark typische Kurve, d. h. die Auslastung steigt bis knapp 100 % an bis alle SD-User aktiv sind. Ab diesem Zeitpunkt beginnt die Hochlastphase, die für einen definierten Zeitraum (z. B. 15 Minuten) anhält. In der Anlaufphase des Benchmarks werden die SD-User zeitversetzt gestartet, so dass nicht alle SD-User zeitgleich die gleichen Transaktionen ausführen.

Man kann anhand der Auslastungskurven erkennen, dass bei 120 SD-Usern eine niedrigere Auslastung vorliegt als beim 150 SD-User-Benchmark. Sowohl an den Auslastungskurven als auch an den mittleren Antwortzeiten pro Dialogschritt (656 ms für 120 SD-User und 2.913 ms für 150 SD-User) kann man deutlich erkennen, dass das System bei 150 SD-Usern überlastet ist. Per Definition des Benchmarks sollte die mittlere Antwortzeit pro Dialogschritt unterhalb von zwei Sekunden liegen.

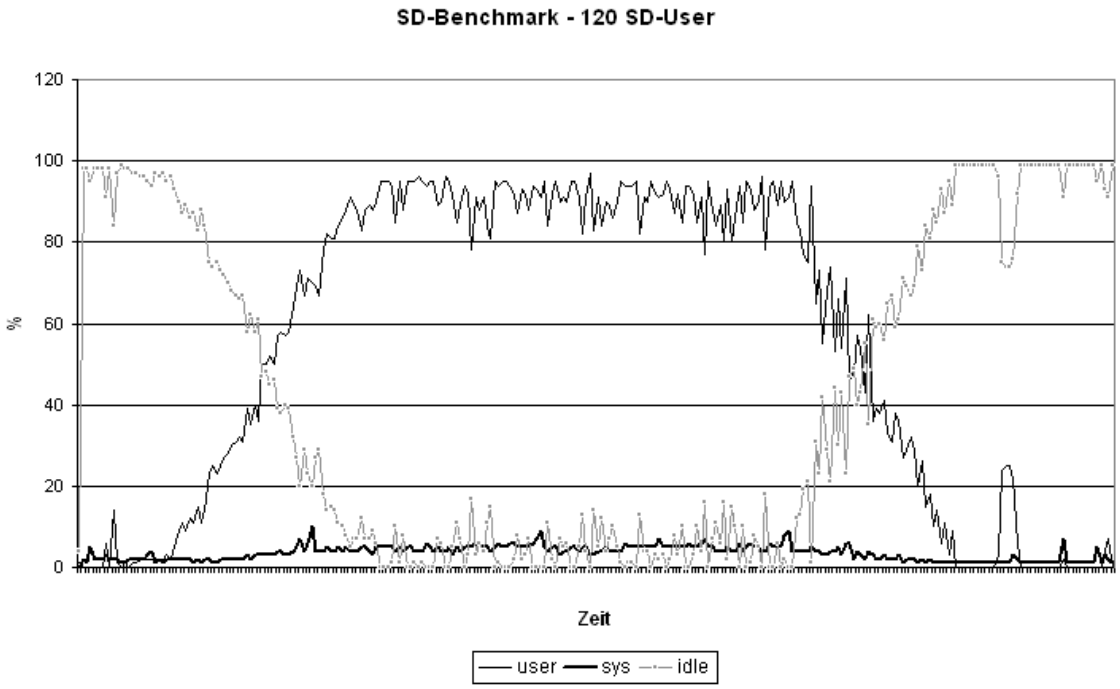


Abbildung 3.1 Auslastungskurve für den 120 SD-User-Benchmark über das Gesamtintervall von 19 Minuten mit einer Hochlastphase (Sampling-Intervall) von 10 Minuten und 30 Sekunden

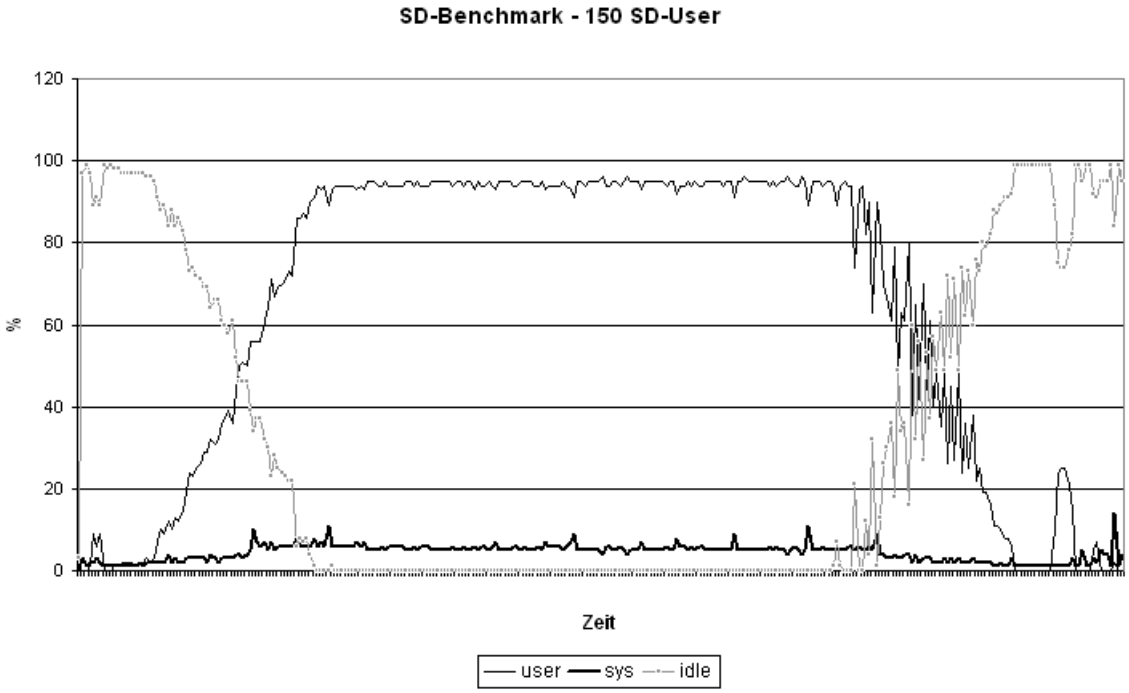


Abbildung 3.2 Auslastungskurve für den 150 SD-User-Benchmark über das Gesamtintervall von 22 Minuten mit einer Hochlastphase (Sampling-Intervall) von 11 Minuten und 30 Sekunden

3.2.1.2. Oracle-Messung

Bei Erhöhung der Anzahl der SD-User steigt die Anzahl der Datenbankzugriffe von 324.295 (120 SD-User) auf 415.306 (150 SD-User) an. Auch die für Oracle "physikalischen" IO-Zugriffe erhöhen sich: 15.382 Writes (15.382 Blocks) und 101 Reads (101 Blocks) für 120 SD-User werden zu 17.888 Writes (17.888 Blocks) und 4.759 Reads (4.759 Blocks) für 150 SD-User. Physikalisch bedeutet, dass Oracle einen IO-Befehl an das Betriebssystem absetzt. Die identische Anzahl von Lese- bzw. Schreiboperationen und Blöcken bedeutet, dass keine Multi-Block-Operationen von der Datenbank durchgeführt, d. h. mehrere Blöcke von einer IO-Operation gleichzeitig verarbeitet werden. In den Analysen wird, wie auch unter UTLB/ESTAT angegeben, von einer Blockgröße von 8 KB ausgegangen.

Die aus den Oracle-Daten berechneten Cache-Hit-Ratios betragen: 99,994 % für 120 SD-User und 99,720 % für 150 SD-User, d. h. die hieraus resultierenden Cache-Miss-Rates erhöhen sich um Faktor 45. Somit ist auch der Anstieg der Leseoperationen um Faktor 47 von 101 IOs auf 4.759 IOs zu erklären. Neben den gemessenen IO-Zugriffen sind auch die geschriebenen Redo Blocks von entscheidender Bedeutung. Hierbei sind für 120 SD-User 305.258 Blocks (2.442.064 KB) und für 150 SD-User 382.800 Blocks (3.062.400 KB) geschrieben worden. Es ist anhand der Oracle-Statisitiken nicht zu erkennen, wie viele physikalische IO-Operationen an das Betriebssystem zur Verarbeitung der Redo Blocks gesendet wurden.

3.2.1.3. Betriebssystemmessung

Die Systemmonitordaten von IOPL, SAR und IOSTAT ergeben für das Gesamtintervall der SD-Benchmarks: 64.170 IOs und 356.684 KB Datentransfer für 120 SD-User sowie 111.234 IOs

und 482.686 KB Datentransfer für 150 SD-User. SAR und IOSTAT liefern hierbei nahezu identische Ausgaben, wohingegen IOPL geringfügig kleinere Werte ausgibt. IOPL liefert 62.545 IOs (4.650 Reads, 57.895 Writes) und 354.645 KB Datentransfer (20.025 KB für Reads, 334.620 KB für Writes) für 120 SD-User sowie 109.095 IOs (18.360 Reads, 90.735 Writes) und 479.900 KB Datentransfer (80.025 KB für Reads, 399.875 KB für Writes) für 150 SD-User. Zu erkennen ist, dass bei einer Erhöhung der SD-User-Anzahl um 25% sich eine Steigerung der Lesezugriffe sowie der hierfür transferierten Datenmengen um Faktor 4 ergibt. Die Schreibzugriffe erhöhen sich hingegen nur um Faktor 1,6.

3.2.1.4. myAMC.LNI-Messung

Ein Vergleich der beiden SD-Benchmarks mit 120 und 150 SD-Usern auf Basis der R/3-Statistiken zeigt, dass durch die Erhöhung der SD-User-Anzahl die Datenbankaktivität im SAP-System z. B. in Form des Parameters DB-Calls (120 SD-User: 385.368, 150 SD-User: 489.965) ansteigt. Den R/3-Statistiken ist zu entnehmen, dass die Zahl der Dialogschritte sich von 7.000 auf 8.000 erhöht und die von den Dialogschritten verbrauchten Database Service Units von 8 Mio. auf 10,2 Mio. anwachsen. Der durchschnittliche DBSU-Verbrauch pro Dialogschitt bleibt hingegen nahezu konstant (120 SD-User: 599 DBSU/DS; 150 SD-User: 611 DBSU/DS für die Gesamtlaufzeit). Somit steigen sowohl der DBSU-Verbrauch als auch die DB-Calls entsprechend der Erhöhung der SD-User-Anzahl um 25 % an.

3.2.1.5. 23.000 SD-User-Benchmark

Am 8. Dezember 2000 wurde von Fujitsu Siemens Computers in München eine neue Bestmarke im High-End-SAP-Umfeld erreicht. Mit einer 3-stufigen System-Architektur wurde mit dem R/3-Modul SD (Sales and Distribution) und 23.000 SD-Usern (117.680 SAPS) ein neuer Benchmark-Weltrekord erreicht. Die Systemkonfiguration bestand aus einem Datenbankserver PrimePower 2000 mit 64 CPUs (450 MHz) und 64 GB Hauptspeicher sowie 160 Application-Servern mit jeweils 4 CPUs (700 MHz) und 4 GB Hauptspeicher. Das an den Datenbankserver angeschlossene Speichersystem bestand aus zwei EMC Symmetrix 8730 (36 GB Platten), das einen Plattenspeicherplatz von 1.344 GB bereitstellte. Für den 23 000 SD-User-Benchmark standen die SAR-Daten und das White Paper [137] sowie Datenblätter für die Hardware- und Software-Informationen zur Verfügung.

Im Benchmark wurden 7.061.000 Dialogschritte pro Stunde verarbeitet. Im Vergleich hierzu wurden bei den beiden Benchmark-Messungen mit 120 bzw. 150 SD-Usern 40.588 bzw. 41.744 Dialogschritte pro Stunde verarbeitet. Das System erzielte eine Leistung von 117.680 SAPS und verarbeitete dabei insgesamt 1.188 Mio. DBSU. Auf Betriebssystemebene wurde der Benchmark mit dem Monitoring-Werkzeug SAR überwacht. Anhand dieser Messdaten ist bekannt, dass während des Benchmarks 1.290.805 IOs verarbeitet und 19.202.440 KB transferiert wurden.

3.2.1.6. Vergleich der SD-Benchmarks und Analyse der Korrelation von R/3- und IO-Aktivität

Vergleicht man die Messdaten der SD-Benchmarks mit 120 und 150 SD-Usern auf den Ebenen SAP, Datenbank und Betriebssystem, so ist ein direkter Zusammenhang jeweils zwischen den drei verschiedenen Messebenen pro Benchmark nicht zu erkennen. Betrachtet man z. B. für den 120 SD-User-Benchmark die transferierten Datenmengen auf Ebene des Betriebssystems, so werden 334.620 KB (57.895 IOs) geschrieben und 20.025 KB (4.650 IOs) gelesen. Die Oracle-Statistik liefert hierzu: 123.056 KB (15.382 IOs) geschrieben und 808 KB (101 IOs) gelesen.

Die Statistiken für den Zugriff auf die Datenbank ergeben somit nicht die tatsächlich auf Betriebssystemebene transferierten Daten. Ein wesentlicher Anteil der noch zusätzlich bewegten Daten auf Betriebssystemebene wird durch das Datenbank-Logging verursacht (2.442.064 KB, 305.258 Blocks). Hierfür existieren aber keine Angaben über die "physikalischen" Zugriffe der Datenbank auf das IO-System. Für den SD-Benchmark mit 150 SD-Usern sind vergleichbare Ergebnisse vorhanden. Ein Zusammenhang zwischen gemessenen IO-Zugriffen auf Datenbank- und Betriebssystemebene ist somit nicht zu erkennen.

Wie bereits beschrieben, soll anhand der Messungen ein möglicher Zusammenhang zwischen der Datenbankaktivität DBSU und den auf Betriebssystemebene gemessenen IO-Zugriffen sowie den transferierten Datenmengen in KB untersucht werden. Die Idee ist hierbei, die Anzahl der IOs bzw. KB pro DBSU zu berechnen. Anhand der Messdaten wurden diese Werte ermittelt (siehe Anhang A). Aufgrund der im Gesamtintervall gemessenen Werte ergibt sich nur eine minimale Unterscheidung der IO-Aktivität pro DBSU zwischen den beiden Benchmarks. Es handelt sich also im Fall der 120 und 150 SD-User-Benchmarks um einen "vernünftig" gewählten Parameter IOs/DBSU bzw. KB/DBSU.

Um neben der vergleichenden Untersuchung der IO-Aktivität pro DBSU beider Benchmarks zu analysieren, in wie weit die Konstante IOs/DBSU bzw. KB/DBSU über mehrere Messintervalle variiert, wurde im Sampling-Intervall pro Minute die Konstante IOs/DBSU berechnet. Es wurde hierzu pro Minute der Quotient von gemessenen DBSU und IOs gebildet. Der DBSU-Verbrauch wurde hierbei anhand der verarbeiteten Dialogschritte der R/3-Statistiksätze ermittelt und die IOs mit Hilfe des verwendeten IO-Monitoring-Programms, das Mittelwerte für Messintervalle von 5 Sekunden lieferte. Abbildung 3.3 stellt die zugehörige Kurve dar, die sich in einem Intervall mit den Grenzen 0,004 und 0,01 befindet und im Mittel einen Wert von 0,0076 IOs/DBSU besitzt. Die Abweichung zum Mittelwert beträgt zwischen 30% und 50%. In Anbetracht der sehr kurzen Zeitintervalle, die zur Verdichtung der Messdaten gewählt wurden, sind diese Schwankungen tolerierbar.

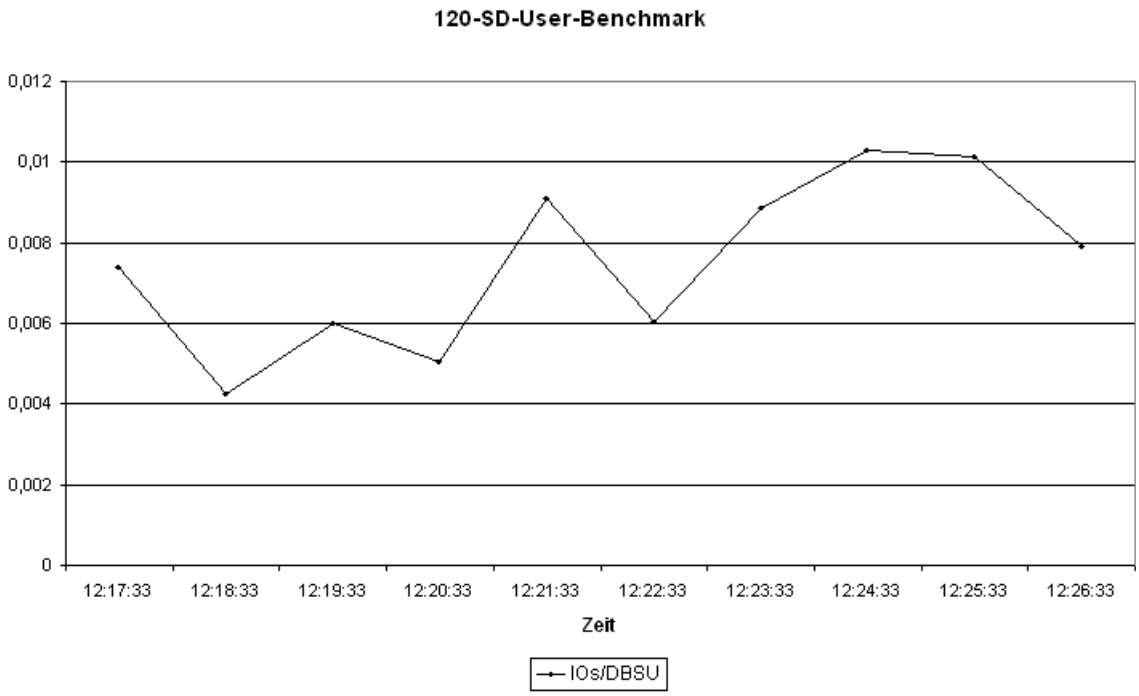


Abbildung 3.3 IOs/DBSU pro Minute für den 120 SD-User-Benchmark über das Sampling-Intervall

Vergleicht man die bereits beschriebenen Ergebnisse mit dem 23.000 SD User-Benchmark zur weitergehenden Verifikation der IO-Lastcharakterisierung mittels IOs/DBSU und KB/DBSU, so erkennt man, dass sich für diesen sehr viel niedrigere Werte ergeben (siehe Anhang A). Im Vergleich zum 150 SD-User-Benchmark liegen diese Werte für den Datentransfer um Faktor 3 bzw. für die verarbeiteten IOs um Faktor 10 niedriger. Auch die im 23.000 SD User-Benchmark gemessene Datenbankaktivität pro Dialogschritt in DBSU ist im Vergleich zum 150 SD User-Benchmark um Faktor 3,7 niedriger.

Die beschriebene Differenz zwischen dem 23.000 SD-User-Benchmark und den 120 und 150 SD-User-Benchmarks kann durch verschiedene Faktoren verursacht worden sein. Ein grundlegender Unterschied ist die Architektur der R/3-Systeme. Bei dem 120- und 150-SD-User-Benchmark handelt es sich um ein Zentralserversystem und beim 23.000 SD-User-Benchmark um ein 3-stufiges System. Auch die Software-Releases sind unterschiedlich. Weiterhin ist zu beachten, dass für die beiden Zentralserver-Benchmarks die Systeme nur gering für den SD-Benchmark optimiert wurden. Für den 23.000 SD User-Benchmark wurde das System hingegen optimal parametrisiert und konfiguriert, so dass ein maximaler Durchsatz erreicht werden konnte.

3.2.2. Szenario 2: SSQJ (Large Table)

SSQJ ist eine Toolbox bestehend aus verschiedenen ABAP-Programmen, die zum Testen und zur Performance-Analyse eines SAP-Systems verwendet werden können (siehe [90]). Entwickelt wurde dieses Werkzeug bei der SAP AG von der Abteilung R/3 Advanced Technology Group (ATG). Das Testen von Datenbanken und Optimizern war die Grundidee zur Implementierung der Toolbox SSQJ. Inzwischen besitzt diese eine sehr viel umfangreichere Sammlung an Testverfahren, wie z. B. die Komponente des TPC-D-Benchmarks, der im Rahmen einer Diplomarbeit im Jahr 2000 an der Universität Passau als eine SSQJ-Komponente implementiert wurde (siehe [143]).

SSQJ beinhaltet über 3.400 Testfälle, die in drei Hauptgruppen unterteilt werden können: ABAP, SQL und „Large Table“. Im vorliegenden Fall wird zum Testen des IO-Systems nur die SSQJ-Komponente „Large Table“ ausgewählt. In dieser Komponente werden "extrem teure" Datenbanktransaktionen (wie z. B. Full Table Scans) auf sehr großen Tabellen ausgeführt.

Installiert ist die SSQJ-Toolbox auf einem Zentralserversystem, Primepower M1000 mit 4 Prozessoren (330 MHz). Der Server ist über vier Fibre Channel-Kanäle an eine Fujitsu Storage Box angebunden. Das System läuft unter dem Betriebssystem Solaris mit SAP Release 4.6D und der Datenbank Oracle 8i Enterprise Edition Release 8.1.6.3.0.

Überwacht wird der SSQJ-Durchlauf von IOSTAT, SAR und VMSTAT. Parallel zu den Betriebssystemmonitoren wird mit myAMC.LNI das R/3-System überwacht, das die vollständigen SAP-Statistiksätze in eine Datenbank speichert (V2-Records) und schließlich wird mit dem UTLB/ESTAT-Skript von Oracle die Performance der R/3-Datenbank gemessen.

3.2.2.1. Auslastungsprofil

Die in der Abbildung 3.4 dargestellte Kurve beschreibt die Auslastung des Systems während der Durchführung des SSQJ-Szenarios "Large Table". Man kann erkennen, dass in der Anfangsphase des Tests eine sehr hohe Auslastung von ca. 80 % bis hin zu 100 % vorliegt. Im weiteren Verlauf des Tests sinkt die Auslastung hingegen auf durchschnittlich 50 % mit 40 % User- und 10% System-Anteil.

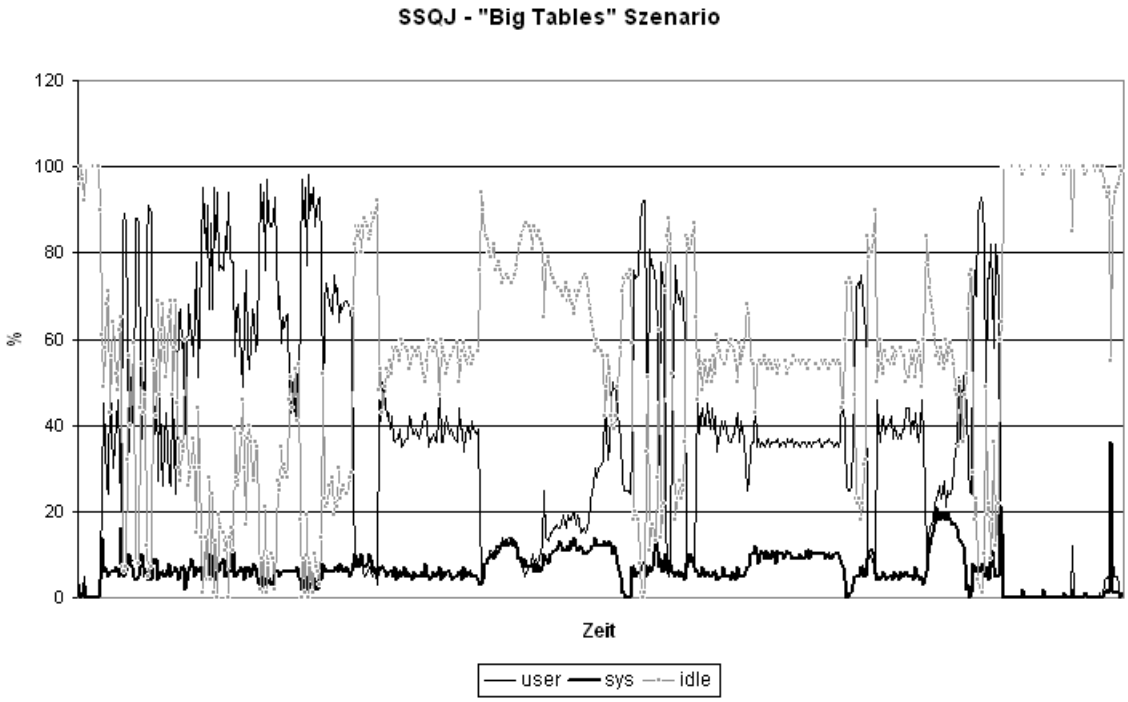


Abbildung 3.4 Auslastungsprofil für das SSQJ-Szenario „Large Table“

3.2.2.2. myAMC.LNI-Messung

Im Vergleich zu den SD-Benchmarks ist in diesem Szenario auffällig, dass nur wenige Dialogschritte (529 DS) generiert werden. Die Datenbankaktivität pro Dialogschritt in Form von DBSU/DS ist im Vergleich zum 150 SD-User-Benchmark mit ca. 600 DBSU/DS hingegen auf ca. 2.800 DBSU/DS angestiegen. Dieses Verhalten ist typisch für Anwendungen im Bereich OLAP, wie z. B. beim Business Information Warehouse (BW). In einem BW-System werden von den Usern Reports angefordert, die mit nur wenigen Dialogschritten in den R/3-Statistik-Sätzen beschrieben werden, jedoch zu deren Erstellung große Mengen an historischen Daten bearbeitet werden müssen. Dies resultiert somit in einer hohen Datenbankaktivität. Der SD-Benchmark ist im Gegensatz hierzu ein klassischer Vertreter der OLTP-Anwendungen, d. h. die Hauptlast wird durch die Interaktion der User mit dem System (Dialoglast) verursacht.

3.2.2.3. Oracle-Messung

Vergleicht man die Oracle-Statistiken für den SSQJ-Test mit den Messdaten für die SD-Benchmarks, so ist direkt die sehr niedrige Anzahl an User Calls von ca. 50.000 im Verhältnis zu ca. 415.000 beim 150-SD-User-Benchmark festzustellen. Begründet liegt dies in der geringen Anzahl an abgesetzten Dialogschritten (siehe Anhang A). Zudem fällt auf, dass die Differenz zwischen DB-Calls und User-Calls erheblich größer ist, als beim SD-Benchmark. In diesem Fall existieren mehr User-Calls (50.000) als R/3-DB-Calls (33.500). Im Normalfall ist dieses Verhältnis genau umgekehrt.

Im Gegensatz zu den User-Calls ist die Anzahl der Lese- und Schreiboperationen in Summe bei SSQJ um ein Vielfaches höher als die beim SD-Benchmark gemessene Anzahl, so dass das Verhältnis der Lese-/Schreiboperationen pro User-Call drastisch gestiegen ist. Vergleicht man die Messdaten von SSQJ mit denen des 150-SD-User-Benchmarks, so ergibt sich eine Steigerung

der ca. 18.000 Schreiboperationen (18.000 Blöcke) vom SD-Benchmark auf 164.000 Operationen (1,8 Mio. Blöcke) sowie eine Erhöhung der ca. 4.800 Leseoperationen (4.800 Blöcke) auf 2 Mio. Operationen (10,7 Mio. Blöcke) beim SSQJ. Vergleicht man weiterhin das Verhältnis von Operationen zu Blöcken, so wird beim SD-Benchmark pro Operation nur ein Block verarbeitet, wohingegen bei SSQJ pro Operation mehrere Blöcke verarbeitet werden (Multi-Block-Operationen). Weiterhin werden beim SSQJ zwei Checkpoints zur Datensynchronisation verwendet (SD-Benchmark: 1 Checkpoint). Auf Grund der Definition des Testszenarios „Large Table“ beim SSQJ liegt die Cache-Hit-Ratio nur bei 18,49 %, d. h. nahezu 80 % der Datenbankzugriffe können nicht vom Cache der Datenbank bedient werden und müssen somit auf den Sekundärspeicher zugreifen.

3.2.2.4. Betriebssystemmessung

Nach IOSTAT werden während des SSQJ-Tests ca. 2 Mio. Reads (86 GB) und ca. 200.000 Writes (15 GB) generiert. Vergleichbare Resultate liefert der Systemmonitor SAR. Vergleicht man die Systemmonitordaten mit den Oracle-Statistiken, so erhält man nach Angabe von IOSTAT: 86 GB gelesene und 15 GB geschriebene Daten. Oracle hingegen beziffert in seinen Statistiken 85,6 GB gelesene und 14,8 GB geschriebene Daten. Nimmt man dann noch die geschriebenen Redo Blocks (466.720 KB, 58.340 Blocks) mit hinzu, erhält man ein nahezu identisches Ergebnis mit den Systemmonitordaten. Eine vergleichbar gute Übereinstimmung der Messdaten erhält man bei der Betrachtung der IO-Operationen auf Datenbank- und Betriebssystemebene: 2.091.247 Lesezugriffe und 200.475 Schreibzugriffe auf Betriebssystemebene sowie 2.086.005 Lesezugriffe und 163.760 Schreibzugriffe auf Datenbankebene.

3.2.2.5. Analyse der Korrelation von R/3- und IO-Aktivität

Analog zu den SD-Benchmarks werden nachfolgend die Konstanten KB/DBSU und IOs/DBSU für SSQJ analysiert. Die Konstanten liegen auf Grund der beschriebenen Statistiken und Messdaten um ein Vielfaches höher als die für den SD-Benchmark (siehe Anhang A). Der wesentliche Grund hierfür ist der bereits beschriebene unterschiedliche Anwendungstyp von SSQJ gegenüber dem SD-Benchmark. SSQJ kann als Vertreter von OLAP-Anwendungen angesehen werden, d. h. die Systemlast wird nur durch eine geringe Zahl an Dialogschritten verursacht, die aber im Gegensatz zu dialogorientierten Anwendungen (OLTP), wie z. B. den SD-Benchmarks, eine wesentlich höhere Last generieren.

Bei der Analyse der Messdaten des SSQJ-Tests ist zu beachten, dass dieser Test darauf ausgerichtet ist, eine hohe IO-Aktivität zu produzieren. Die Messergebnisse sollten somit nicht als typische Messgrößen für OLAP-Anwendungen angesehen werden. Vielmehr kann dieser Test als "Worst-Case"-Szenario für eine OLAP-Anwendung interpretiert werden.

3.2.3. Szenario 3: R/3-Produktivdaten

Bislang wurden mit dem SD-Benchmark und der SSQJ-Toolbox Anwendungsszenarien betrachtet, die auf Basis von Lastgeneratoren in Form von ABAP-Anwendungen oder Skripten synthetische Lasten generieren. Es folgt in dem nun dritten und letzten Szenario die Analyse von IO-Messdaten eines produktiven R/3-Systems.

Das Produktivsystem besteht aus 18 Applikationsservern (Primergy-870, 4 CPUs) und einem Datenbankserver (RM600/E70, 24 CPUs), der an eine EMC Symmetrix angeschlossen ist. Als Betriebssysteme werden Windows NT für die Applikationsserver und Reliant Unix für die

Datenbank verwendet. Als Datenbank ist Oracle 8.0.4 installiert sowie SAP R/3 mit Release 4.0B und den folgenden Business Components: CO, FI, FI-AA, MM, SD.

Das System wird auf Ebene von SAP R/3 mit Hilfe des myAMC.LNI und auf Ebene des Betriebssystems mit SAR überwacht. Für einen typischen Wochenarbeitstag werden für den Zeitraum von 08:00 bis 17:00 Uhr die Datenbankaktivität in Form von DBSU, die Zahl der verarbeiteten Dialogschritte des R/3-Systems und IO-Zugriffe auf Betriebssystemebene sowie die gelesenen bzw. geschriebenen Datenmengen in KB jeweils pro Stunde verdichtet. Es resultieren die in Tabelle 3.1 dargestellten Ergebnisse:

Zeitintervall	DBSU	DS	KB (SAR)	IOs (SAR)	KB/DBSU	IOs/DBSU
08:00-09:00	214.164.641	85.940	70.508.244	6.407.652	0,3292	0,0299
09:00-10:00	177.894.219	107.564	77.983.604	7.118.844	0,4383	0,0400
10:00-11:00	207.828.180	108.204	85.421.423	8.427.447	0,4110	0,0405
11:00-12:00	196.954.630	111.527	93.644.961	9.282.516	0,4754	0,0471
12:00-13:00	186.626.322	92.167	97.663.017	9.600.051	0,5233	0,0514
13:00-14:00	145.720.250	71.622	80.404.958	7.446.582	0,5517	0,0511
14:00-15:00	209.648.081	76.135	79.034.061	7.590.384	0,3769	0,0362
15:00-16:00	182.674.822	68.132	73.435.434	7.347.885	0,4020	0,0402
16:00-17:00	171.044.777	61.718	89.594.499	8.626.434	0,5238	0,0504

Tabelle 3.1 IO-Messdaten für ein Produktivsystem

Zusätzlich zu den Messdaten werden für jedes Stundenintervall die Parameter KB/DBSU und IOs/DBSU, wie bereits in den Szenarien 1 und 2 beschrieben, berechnet. In Abbildung 3.5 werden diese Parameter graphisch dargestellt.

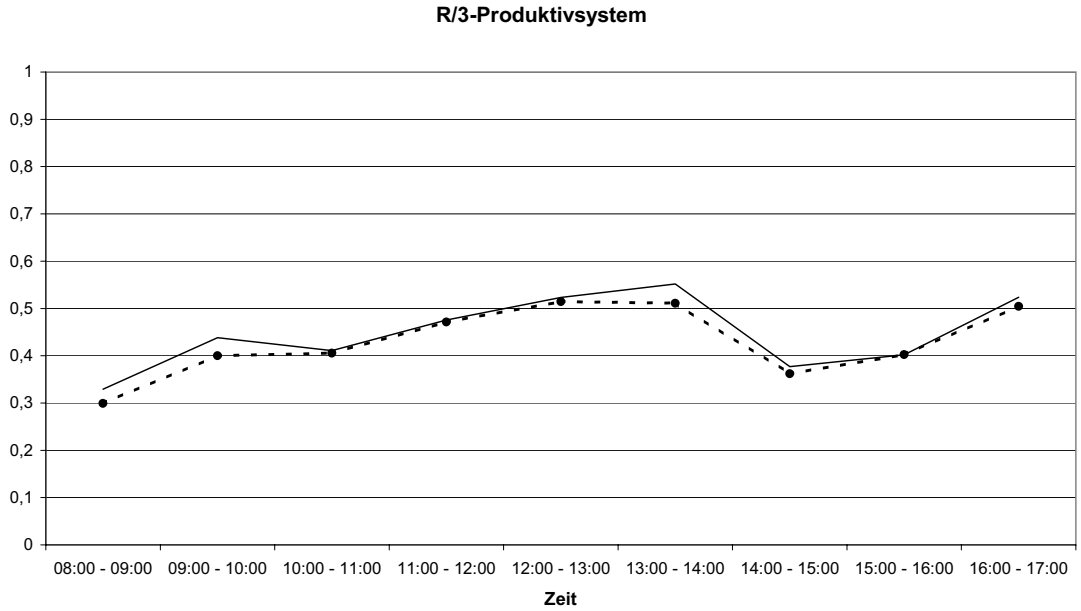


Abbildung 3.5 IOs/DBSU und KB/DBSU pro Stunde für das Produktivsystem

Betrachtet man die Konstanten IOs/DBSU und KB/DBSU, so befinden sich die Konstanten in einem Bereich von 0,3 bis knapp über 0,5. Vergleicht man dieses Ergebnis mit der Untersuchung des SD-Benchmarks in Abbildung 3.3, so existiert in diesem System eine relativ konstante Abhängigkeit von generierten IOs bzw. transferierten KB pro DBSU.

3.2.4. Vergleich der Szenarien

Das Ziel dieser Untersuchung war es, Zusammenhänge zwischen der Last eines SAP R/3-Systems und der IO-Aktivität im Speichersubsystem zu analysieren und eine hierfür geeignete Form der IO-Lastcharakterisierung zu definieren. Hierzu wurden Messungen mit dem SD-Benchmark und der Toolbox SSQJ vorgenommen, sowie Projektdaten eines R/3-Produktivsystems analysiert. Der SD-Benchmark wurde hierbei als Vertreter einer typischen OLTP-Anwendung ausgewählt und die Komponente "Large Table" der Toolbox SSQJ auf Grund ihres Lastmusters als Repräsentant einer OLAP-Anwendung. Für das ausgewählte Szenario der SSQJ-Toolbox ist zu beachten, dass diese mittels Verarbeitung sehr großer Tabellen, ein überdurchschnittlich hohes Aufkommen an IO-Zugriffen generiert, so dass dieses Szenario für die Beschreibung der IO-Aktivität von OLAP-Anwendungen als "Worst-Case“ zu deklarieren ist.

Im Rahmen der Untersuchung wurden die von den Dialogschritten verursachten Datenbankaktivitäten auf Ebene von SAP R/3 in Form von Database Service Units (DBSU) und die auf Betriebssystemebene gemessenen IO-Zugriffe sowie Datentransfers analysiert. Es wurde hierbei, die Abhängigkeit zwischen der IO-Aktivität und den Database Service Units untersucht, indem die folgenden Werte berechnet wurden: transferierte Datenmengen pro DBSU (KB/DBSU) und generierte IOs pro DBSU (IOs/DBSU). Anhand der beschriebenen Ergebnisse konnten insbesondere für die SD-Benchmarks keine adäquaten Korrelationen zwischen der DB- sowie R/3- und Betriebssystemebene definiert werden, so dass die Datenbankebene zur IO-Lastcharakterisierung nicht berücksichtigt wurde.

Vergleicht man die erzielten Ergebnisse der verschiedenen Szenarien untereinander, so ist festzustellen, dass z. B. für die Bewertung der IOs/DBSU für die SD-Benchmarks mit 120 und 150 SD-Usern (0,007 IOs/DBSU) und der SSQJ-Toolbox Differenzen (1,5 IOs/DBSU) um den Faktor 1000 auftraten. Es ist jedoch zu beachten, dass hierbei zwei extrem unterschiedliche Anwendungstypen, d. h. OLTP und OLAP, miteinander verglichen werden, wobei der SD-Benchmark stellvertretend für ein IO-armes und SSQJ für ein IO-intensives Lastaufkommen stehen, so dass die beschriebenen Werte als Minimum und Maximum auftretender IO-Aktivität interpretiert werden können. Anhand des Ergebnisses für das Produktivsystem von 0,047 IOs/DBSU ist zu erkennen, dass die IO-Kennzahlen realer Systeme im Mittelfeld dieser beiden Extrema liegen. Aufgrund der beschriebenen Ergebnisse kann somit eine hohe Abhängigkeit der IO-Aktivität von dem im R/3-System verwendeten Anwendungstyp (OLAP oder OLTP) festgestellt werden. Es existiert somit keine konstante Abhängigkeit der IO-Aktivität von den DBSU für beliebige Anwendungstypen. Werden hingegen Anwendungen gleichen Typs betrachtet, so stimmen auch die Werte IOs/DBSU und KB/DBSU überein, wie anhand der Ergebnisse für die SD-Benchmarks mit 120 und 150 SD-Usern zu erkennen ist.

Abschließend wurde der Zusammenhang zwischen der IO-Aktivität und den DBSU für verschiedene Zeitfenster einer Messung anhand der Szenarien SD-Benchmark und Produktivsystem untersucht. Die Ergebnisse dieser Untersuchung (Abbildung 3.3, Abbildung 3.5) zeigen, dass sich innerhalb einer Messung eine nahezu konstante Abhängigkeit zwischen IO-Aktivität und DBSU ergibt.

3.3. Zusammenfassung

Das Ziel dieses Kapitels war es, Methoden zur IO-Lastcharakterisierung für SAP R/3 zu entwikkeln, die dann weiterführend zur IO-Modellierung im Rahmen der R/3-Kapazitätsplanung angewendet werden sollen. In Anlehnung an das in [72] dargestellte Konzept wurde in diesem Kapitel ein Verfahren vorgestellt und evaluiert, das eine Korrelation von IO-Aktivitäten (IO-Zugriffe, Datentransfers) auf Ebene des Betriebssystems und Datenbankaktivitäten (DBSU) auf Ebene von SAP R/3 ermöglicht. Es hat sich gezeigt, dass eine derartiger Zusammenhang exi-

stiert, jedoch in starker Abhängigkeit des Anwendungstyps (OLAP, OLTP) steht. Der Vorteil des vorgestellten Konzepts gegenüber dem in [72], welches die IO-Aktivitäten nur in Relation zu SAPS-Werten, d. h. SD-Benchmark-Resultaten, betrachtet, ist die Definition der IO-Lasten in Abhängigkeit der real gemessenen Last. Die Verwendung des Ansatzes von [72] bietet sich somit für die Dimensionierung von IO-Systemen im Rahmen eines Sizing-Prozesses an, wohingegen sich die in diesem Kapitel dargestellte IO-Lastcharakterisierung für bereits existierende SAP R/3-Systeme weitaus besser eignet.

Von Mißbach und Hoffmann [72] wurde zwischen Frontend- und Backend-IOs unterschieden. Im Rahmen einer R/3-Kapazitätsplanung ist eine derartige Unterscheidung mit Hilfe der vom Betriebssystem angebotenen Standardwerkzeuge nicht möglich (siehe Kapitel 4). Die Backend-IOs, d. h. die auf den einzelnen Festplatten resultierenden IO-Zugriffe, können mit Werkzeugen, wie zum Beispiel SAR oder IOSTAT nicht gemessen werden, da diese nur auf Betriebssystemebene IO-Aktivitäten messen können. Es existieren hingegen Produkte, wie z. B. Navisphere der Firma EMC^2, welche Messungen bis auf Backend-Ebene ermöglichen, d. h. die IO-Verarbeitung kann angefangen von der Applikationsebene bis hin zu den einzelnen Festplatten verfolgt werden. In der Regel liegen derartige Monitoring-Programme aus Kostengründen im Rahmen einer Kapazitätsplanung jedoch nicht vor, so dass nur auf Ebene der Frontend-IOs Messdaten erhoben werden können und für die Backend-IOs abhängig vom verwendeten IO-System und RAID-Level theoretische Zuschläge mit eingerechnet werden müssen.

Aufgrund der beschriebenen Abhängigkeit der IO-Aktivität von der Datenbankaktivität DBSU können für die Modellierung von R/3-Systemen auf Basis der in der Messung gewonnenen Daten die Parameter IOs/DBSU und KB/DBSU berechnet und im Modell berücksichtigt werden. In Kapitel 2 Abschnitt 2.5.2. wird für den WLPSizer eine derartige Konstante mit IODBSU beschrieben, welche die Anzahl der IO-Zugriffe pro DBSU vorgibt.

KAPITEL

4

BENCHMARKING UND MONITORING VON IO-SYSTEMEN

Ein weiterer Schwerpunkt dieser Arbeit ist das Benchmarking, Monitoring sowie die Modellierung von IO-Systemen. Ziel ist hierbei die Erarbeitung eines analytischen Festplattenmodells, das unter Anwendung der in Kapitel 3 beschriebenen IO-Lastcharakterisierung in die SAP R/3-Kapazitätsplanung und das dafür verwendete Modellierungswerkzeug WLPSizer integriert werden soll. In diesem Kapitel werden die benötigten Grundlagen für IO-Architekturen und Festplatten-Technologien beschrieben sowie das Benchmarking und Monitoring von IO-Systemen anhand von verschiedenen Lastszenarien demonstriert. Die bei den Messungen gewonnenen Ergebnisse sollen dann für das in Kapitel 5 vorzustellende IO-Modell zur Validation und Kalibrierung verwendet werden.

4.1. Festplatten-Technologie und IO-Architekturen

Speichersysteme und (wie in diesem Fall) Festplatten werden in zahlreichen Publikationen und Konferenzbeiträgen beschrieben. Einen guten Überblick geben Tanenbaum und Goodman in [115] oder Silberschatz, Peterson und Galvin in [107] sowie Hennesey und Patterson in [44]. Eine Einführung in die Plattentechnologie und der damit verbundenen Grundbegriffe wird ebenso von Ruemmler und Wilkes in [84] gegeben, wobei deren Arbeit als Schwerpunkt die Simulation von Festplatten besitzt, sowie von Hospodor und Hoagland in [47] und Shriver in [99].

Zur Beschreibung einer Festplatte kann man zwischen der darin verwendeten Mechanik und dem installierten Controller unterscheiden. Die Mechanik beinhaltet zum einen die Komponenten zur Speicherung der Daten, d. h. Platten und Schreib-/Leseköpfe und zum anderen die Positionierungseinheiten, d. h. Plattenarme zur Bewegung der Schreib-/Leseköpfe und ein sogenanntes „Track-Following System“, das zur korrekten Platzierung der Köpfe verhilft. Der Festplatten-Controller enthält hingegen einen Mikroprozessor, einen Cache, eine Queue und eine Schnittstelle zum Bussystem. Die Hauptaufgabe des Controllers liegt darin, eingehende Kommandos zu interpretieren und auszuführen, so dass Daten gespeichert oder gelesen werden können. Er muss hierzu die logischen Adressen des Betriebssystems (LBAs) auf die physikalischen Adressen (Zylinder, Tracks und Sektoren der Platten) abbilden.

4.1.1. Festplattenmechanik, Positionierungs- und Transferzeiten

Wie in Abbildung 4.1 dargestellt, besteht eine Festplatte aus einer festen Anzahl an Platten (Platters), die mittels einer Spindel befestigt sind und sich somit rotieren lassen. Nach [44] bestehen derzeitige Festplatten aus 1-12 Platten und deren Umdrehungsgeschwindigkeit liegt bei 3.600-15.000 Umdrehungen pro Minute (RPM). Zur Speicherung der Daten sind die Platten auf beiden Seiten beschichtet. Jede Plattenoberfläche wird in Form von mehreren konzentrischen Kreisen (Tracks) unterteilt, wobei derzeit 5.000-30.000 Tracks pro Plattenseite typisch sind [44]. Die Tracks aller Platten mit konstantem Abstand zur Spindel werden als Zylinder bezeichnet. Ein Track wird weiterhin in Sektoren unterteilt, welche die kleinste Informationseinheit einer Festplatte darstellen. Pro Track existieren derzeit zwischen 100 und 500 Sektoren [44], wobei ein Sektor meistens eine konstante Größe von 512 Byte besitzt. In der Vergangenheit wurde eine feste Anzahl an Sektoren für alle Tracks festgelegt, so dass die inneren Tracks einer Platte die Zahl der Sektoren für die äußeren Tracks vorgaben. Auf Grund dieser Festlegung wurden gerade auf den äußeren Tracks die Zwischenräume zwischen den Sektoren immer größer und somit ging Speicherplatz verloren. Zur Lösung dieses Problems wurde eine Technik mit der Bezeichnung „Zoned Bit Recording“ eingeführt, die eine Gruppierung der Tracks in Abhängigkeit des Abstands zur Spindel ermöglicht, wobei jede Gruppe eine feste Anzahl an Sektoren pro Track besitzt. Somit steigt die Zahl der Sektoren und die damit verbundene Datenmenge pro Track mit zunehmendem Abstand zur Spindel und dadurch auch die Transferrate, die daher auf den äußeren Tracks höher liegt als auf den inneren. Bei derzeitigen Platten existieren zwischen 10 und 20 Zonen.

Der Zugriff auf die Daten erfolgt über die Schreib-/Leseköpfe, die mittels der Plattenarme zum benötigten Track geführt werden. Da alle Plattenarme am gleichen Aktuator befestigt sind, werden bei jeder Bewegung eines Schreib-/Lese-Kopfs alle weiteren Köpfe mitbewegt, so dass sich zu jeder Zeit alle Köpfe im gleichen Zylinder befinden.

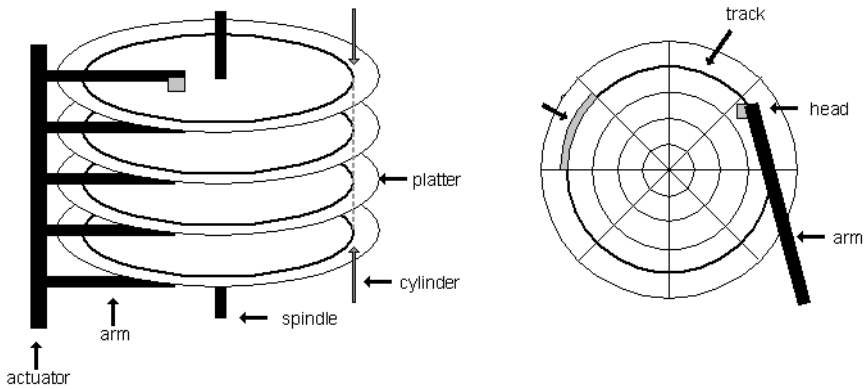


Abbildung 4.1 Die mechanischen Komponenten einer Festplatte

Die Zeit, die zur Positionierung des Kopfes über den gesuchten Track benötigt wird, bezeichnet man als Seek-Time. Ruemmler und Wilkes unterteilen in [84] die Seek-Time in folgende Bestandteile:

- *Speedup:* Beschleunigung des Plattenarms bis dieser die Hälfte der geforderten Distanz oder maximale Geschwindigkeit erreicht hat.
- *Coast:* Plattenarm bewegt sich mit maximaler Geschwindigkeit.
- *Slowdown:* Plattenarm wird gebremst, wenn er sich in der Nähe des benötigten Tracks befindet.

- *Settle:* Platten-Controller justiert die Schreib-/Leseköpfe, so dass die benötigten Sektoren gelesen werden können.

In Abhängigkeit der Seek-Time überwiegt z. B. bei kurzen Seeks die Settle-Time und bei langen Seeks in Abhängigkeit der zu überspringenden Tracks der Coast-Anteil. Hersteller von Festplatten geben in ihren Produktblättern häufig die minimale, maximale und mittlere Seek-Time an, wobei die mittlere derzeit bei 5-12 ms [44] liegt. Nach [139] und [44] wird diese Angabe mittels eines Quotienten bestehend aus dem Zeitbedarf aller möglichen Seek-Times dividiert durch die Anzahl aller möglichen Seeks berechnet. Bei Verwendung dieser Definition entspricht jedoch die real gemessene mittlere Seek-Time abhängig vom Betriebssystem und der Applikation nicht der vom Hersteller angegebenen Seek-Time. Nach [44] wurde ermittelt, dass häufig die reale Seek-Time nur 25 % bis 33 % der Herstellerangabe entspricht.

Nachdem der Plattenarm den Schreib-/Lesekopf über den gesuchten Track positioniert hat und der Seek somit abgeschlossen ist, entsteht eine Wartezeit bis der gesuchte Sektor den Plattenkopf erreicht. Diese Zeit wird als Rotationslatenzzeit bezeichnet (siehe [70]). Ist die Positionierung des Plattenkopfs beendet, beginnt der Datentransfer von der Platte zum Rechner, d. h. zum IO-Controller des Mainboards bzw. in den Hauptspeicher des Rechners. Die Geschwindigkeit des Datentransfers hängt dabei zum einen von der Festplatte und zum andern vom IO-Bussystem ab. Nach [139] kann zwischen den drei Datenübertragungsgeschwindigkeiten Burst Speed, Internal Transfer Speed und Data Speed unterschieden werden:

- *Burst Speed:* Maximale Datenübertragungsgeschwindigkeit zwischen Festplatten-Cache und Hautspeicher des Servers.
- *Internal Transfer Speed:* Geschwindigkeit zur Übertragung der Daten zwischen Platte und Festplatten-Schreib-/Lesekopf.
- *Data Speed:* Geschwindigkeit zur Übertragung der User-Daten unter der Annahme, dass kein mechanischer Seek bzw. keine Rotation notwendig ist. Die Data Speed wird berechnet durch: Data Speed = ((#Sectors per Track) * 512 * rpm) / (60 seconds), wobei rpm die Umdrehungsgeschwindigkeit der Platte und 512 die Sektorgröße in Byte angibt. Ebenso wie die Internal Transfer Speed gibt auch die Data Speed die Geschwindigkeit der Datenübertragung zwischen Platte und Festplatten-Schreib/Lesekopf an.

Die Burst Speed wird somit begrenzt durch die Geschwindigkeit des verwendeten IO-Busses. Die Internal Transfer Speed ist im Vergleich zur Burst Speed wesentlich geringer und beschreibt zusätzlich, mit welcher Geschwindigkeit alle Daten, d. h. auch Fehlerkorrekturdaten, transferiert werden. Im Gegensatz hierzu liefert die Data Speed die Datenübertragungsgeschwindigkeit, die lediglich für die User-Daten benötigt wird.

4.1.2. Festplatten-Controller

Der Controller einer Festplatte ist nach [84] zuständig für den Zugriff auf die Platten, den Betrieb des „Track-Following Systems“, den Datentransfer zwischen Client und Festplatte und die Verwaltung des Festplatten-Cache sowie der Queues. Das Kernstück eines Festplatten-Controllers ist hierbei ein Mikroprozessor, der für die Verarbeitung der eingehenden IO-Anforderungen zuständig ist. In den folgenden zwei Abschnitten werden IO-Bussysteme, die über eine Schnittstelle mit dem Controller verbunden sind sowie mögliche Caching-Strategien einer Festplatte beschrieben.

4.1.3. IO-Bussysteme

Die Kommunikation und der damit verbundene Datentransfer zwischen Computer und IO-System verläuft über ein Bussystem, das sich aus einem Medium und einem Protokoll zusammensetzt. Der wohl älteste und bekannteste Standard hierfür ist das „Small Computer System Interface“ (SCSI). Dieser wurde 1986 basierend auf dem Shugart Associates Systems Interface (SASI) als erster SCSI Standard (SCSI-1, X3.131-1986, Project 375D, ISO/IEC 9316) vom American National Standards Institute (ANSI) verabschiedet. Nachfolgend sind zwei weitere Standards SCSI-2 (ISO/IEC 9316:1995, 2nd Edition) und SCSI-3 entstanden. Die Entwicklung des SCSI-3-Standards begann 1993 und ist bislang nicht abgeschlossen. Eine kurze Einführung in die verschiedenen SCSI-Standards ist z. B. in dem von der SCSI Trade Association verfassten White Paper [94] zu finden.

Ein weiterer IO-Standard insbesondere für den Home-PC-Bereich ist IDE bzw. E-IDE. Ein wesentlicher Unterschied zu den SCSI-Produkten ist der, dass der finanzielle Anschaffungspreis für SCSI bis zu 250 % höher liegt als bei IDE, wobei der SCSI-Standard eine größere Zahl an Optimierungsverfahren (z. B. Tagged Command Queueing) und somit eine bessere Performance liefert. Nach [18] arbeiten IDE-Festplatten bei sequentieller Last 20 % und bei zufällig verteilter Last 44 % langsamer. Die Performance von IDE-Systemen hat sich seit Einführung der IDE-Technologie im Jahre 1989 bis heute (2003) jedoch derartig verbessert und durch den immer stärker ansteigenden Preisunterschied zu den SCSI-Produkten existieren bereits RAID-Systeme mit IDE-Technologie, die intern mit IDE arbeiten und extern über SCSI mit dem Server verbunden sind. Insbesondere bei größeren RAID-Systemen werden jedoch immer noch SCSI-Systeme bevorzugt. Im Rahmen dieser Arbeit wird allein auf den SCSI-Standard eingegangen. Einen Vergleich von IDE und SCSI insbesondere aus Performance-Sicht findet man z. B. in [123] und [18].

SCSI-1 definiert die Grundlagen für die ersten SCSI-Busse. In dieser Festlegung waren Beschreibungen für die Kabellänge, die Signaleigenschaften, die Kommandosprache und die Transfermodi enthalten. Der SCSI-2-Standard ergänzte SCSI-1 mit weiteren Eigenschaften, wie z. B. die Verdopplung der Taktrate (Fast-SCSI) bzw. der Busbreite (Wide-SCSI), die Erhöhung der Anzahl an maximal anschließbaren Geräten, die Einführung verschiedener Signalgebungen (Differential Signaling bzw. High-Voltage Differential) und die Einführung von Tagged Command Queueing. Die Entwicklung von Fast-SCSI hatte als Zielsetzung, die Geschwindigkeit des synchronen Datentransfers zu verbessern und Wide-SCSI definiert die physikalischen und logischen Spezifikationen für den 16- oder 32-Bit SCSI-Bus. Eine wesentliche Performance-Verbesserung wurde durch das Tagged Command Queueing erreicht. Es können hiermit vom Initiator, d. h. dem Host Bus Adapter (HBA), Tags für die zu verschickenden IOs definiert werden, die z. B. die Reihenfolge der Verarbeitung vorgeben. Das Target, typischerweise die Festplatte, kann dann zwischen der von den Tags vorgegebenen Sortierung oder einer eigenen Verarbeitungsreihenfolge wählen. Eine wesentliche Neuerung ist hierbei, dass der Initiator bis zu 256 IOs bzw. Kommandos an das Target senden kann, ohne auf die Beendigung der vorherigen IOs zu warten, d. h. das Target (Festplatte) besitzt eine Queue zur Speicherung der Anfragen. In der Vergangenheit konnte das Target hingegen immer nur einen IO-Befehl annehmen und verarbeiten. Die Pufferung der IOs bietet somit eine Vielzahl an Caching- und Queueing-Strategien, die in Abschnitt 4.1.4 beschrieben werden.

Im Gegensatz zu den beiden vorherigen Standards entspricht SCSI-3 einer Sammlung verschiedener, aber miteinander verwandter Standards. Die Struktur, die Verbindungen und die damit verbundenen Ziele für diese verschiedenen Standards werden mittels eines SCSI-3 Architekturmodells (SAM, X3.270-1996) festgelegt, das vom T10 Technical Committee erstellt wurde. Derzeit, d. h. im Jahr 2003, wird an einer überarbeiteten Fassung SAM-2 gearbeitet. Die verschiedenen Standards von SCSI-3 können in drei generelle Klassen unterteilt werden:

- *Commands:* Standards, die spezielle Kommandosprachen definieren

- *Protocols:* Standards, welche die Regeln für die Kommunikation und den Datenaustausch zwischen den Geräten beschreiben (Transport Layer).
- *Interconnects:* Standards, welche die elektrischen Signalmethoden und Transfermodi festlegen (Physical Layer).

SCSI-1 und SCSI-2 definieren Standards auf Basis der parallelen Datenübertragungsform. Mit SCSI-3 existieren hingegen sowohl parallele als auch serielle Standards zur Datenübertragung. Die Erweiterungen der herkömmlichen SCSI-Standards 1 und 2 werden unter SCSI-3 als „SCSI-3 Parallel Interface (SPI)“ bezeichnet. Es existieren bislang vier verschiedene Versionen von SPI, die jeweils verschiedene neue Eigenschaften und Transfermodi für das herkömmliche SCSI definieren (siehe Tabelle 4.1; [96], [62]).

Wie bereits beschrieben, existieren neben der parallelen Datenverarbeitung unter SCSI-3 auch serielle Verfahren. Die zwei bekanntesten Vertreter dieser Art sind Fibre Channel-Bussysteme und der Serial Bus (IEEE 1394). Sowohl die parallelen als auch die seriellen SCSI-Verfahren basieren auf dem gleichen logischen SCSI-Protokoll, d. h. den gleichen Busphasen (siehe Abschnitt 4.1.3.1). Weiterhin beinhaltet SCSI-3 einen verbesserten synchronen Transfermodus (Fast20). Eine detaillierte Beschreibung des SCSI-3-Standards befindet sich z. B. in [112] und [28].

SCSI-Transfer-Modus	SCSI-Standard	Bustakt [MHz]	Durchsatz [MB/s]	Bus-Breite [Bits]	Anz. Geräte für SE[1], LVD[2] / HVD	max. Buslänge [m] für SE, LVD/ HVD
SCSI-1	SCSI-1	5	5	8	8 / 8	6 / 25
Wide-SCSI	SCSI-2	5	10	16	16 / 16	6 / 25
Fast-SCSI	SCSI-2	10	10	8	8 / 8	3 / 25
Fast-Wide-SCSI	SCSI-2	10	20	16	16 / 16	3 / 25
Ultra-SCSI	SCSI-3 / SPI	20	20	8	8 (4) / 8	1,5 (3) / 25
Wide-Ultra-SCSI	SCSI-3 / SPI	20	40	16	8 (4) / 16	1,5 (3) / 25
Ultra2-SCSI	SCSI-3 / SPI-2	40	40	8	8 (2) / 8	12 (25) / 25
Wide-Ultra2-SCSI	SCSI-3 / SPI-2	40	80	16	16 (2) / 16	12 (25) / 25
Ultra3-SCSI	SCSI-3 / SPI-3	40 (DT[3])	160	16	16 (2)	12 (25)
Ultra160-SCSI	SCSI-3 / SPI-3	40 (DT)	160	16	16 (2)	12 (25)
Ultra320-SCSI[4]	SCSI-3 / SPI-4	80 (DT)	320	16	16 (2)	12 (25)

Tabelle 4.1 SCSI-Protokolle und ihre Transfermodi

1. Signaleigenschaften: Single Ended (SE), Low Voltage Differential (LVD), High Voltage Differential (HVD)
2. ab Ultra2-SCSI wird nur noch LVD zu HVD angegeben
3. Double Transition Clocking
4. keine endgültigen Angaben

SCSI-Busse mit den in Tabelle 4.1 dargestellten Eigenschaften sind parallele halb-duplex bidirektionale Busse, an denen verschiedene Geräte, wie zum Beispiel Festplatten, Scanner, CD-ROM-Laufwerke, angeschlossen werden können (daisy chaining). Die Geräte können jeweils SCSI-Befehle verschicken (Initiators) oder SCSI-Befehle entgegen nehmen (Targets). Die somit möglichen Konfigurationen, nach ANSI spezifiziert, sind wie folgt:

- *Single Initiator/Single Target*, d. h. beispielsweise ein Host Bus Adapter (HBA) wird mit einer Festplatte verbunden, wobei es sich um die einfachste SCSI-Konfiguration handelt.
- *Single Initiator/Multiple Target*, d. h. ein HBA kontrolliert z. B. mehrere Festplatten, ein CD-ROM-Laufwerk und einen Scanner. Dies ist die gebräuchlichste SCSI-Konfiguration.
- *Multiple Initiator/Multiple Target*, d. h. mehrere Server mit jeweils einem HBA sind an einem SCSI-Bus mit mehreren Geräten (Festplatten, CD-ROM etc.) angeschlossen. Diese Konfiguration ist meistens nur in industriell eingesetzten Systemen vorzufinden, um z. B. das Ausfallrisiko des HBAs zu minimieren.

Eine weitere nicht nach ANSI spezifizierte Form der SCSI-Konfiguration ist das Disk-Array:

- *Embedded SCSI:* Disk Arrays, d. h. im Target existiert ein interner SCSI-Bus, der eine Anzahl an Festplatten verbindet. Der Initiator kann über eine konfigurierbare Anzahl an LUNs (Logical Units) auf das Disk Array zugreifen.

4.1.3.1. SCSI-Busphasen zur IO-Verarbeitung

Die Verarbeitung von SCSI-Befehlen kann in fünf Busphasen unterteilt werden, mit Hilfe derer der Datentransfer über den Bus kontrolliert wird:

- *Bus Free Phase:* Der SCSI-Bus ist frei und steht zur Verfügung. Diese Phase kennzeichnet jeweils den Beginn und das Ende eines SCSI-Befehls.
- *Arbitration Phase:* Geräte konkurrieren um den Zugriff auf den Bus (Priorität wird anhand der SCSI-ID ermittelt). Das „gewinnende“ Gerät wählt zum ersten oder wiederholten Mal sein Target aus.
- *Selection Phase:* Das Gerät mit Zugriff auf den Bus wird Initiator und wählt ein Target aus.
- *Reselection Phase:* Das Gerät mit Zugriff auf den Bus wird Target und wählt zum wiederholten Male den Initiator aus, um den vorher gestoppten Auftrag fortzuführen.
- *Information Transfer Phase:* Phase für den Transfer der Daten über den SCSI-Bus. Diese Phase kann in die folgenden vier Sektionen aufgeteilt werden:
  - *Command Phase:* Initiator verschickt SCSI-Befehl zum Target
  - *Data In/Out Phase:* Datentransfer vom Initiator zum Target oder vice versa
  - *Status Phase:* Das Target versendet ein Status-Byte für die Daten-Übetragung zur Signalisierung eines erfolgreichen Datentransfers.
  - *Message In/Out Phase:* Kontrolle der SCSI-Bus-Operationen während der Selection Phase bzw. Reselection Phase

Die Phasen Selection und Reselection sind für die Verarbeitung der IOs von besonderer Bedeutung. Das Target kann selbstständig entscheiden, ob in Abhängigkeit der Größe und Komplexität der IOs während der IO-Verarbeitung der Bus blockiert oder freigegeben wird. Ist z. B. abzusehen, dass für einen IO-Zugriff ein größerer Seek mit einem Zeitbedarf von mindestens 50 ms vorgenommen werden muss, so kann der Bus freigegeben werden und nach Erreichen des gesuchten Sektors mit Hilfe der Reselection-Phase neu „angeworben“ werden.

Der Datentransfer der Information Transfer Phase, also Data In/Out Phase, kann ab SCSI-2 mittels zwei verschiedener Transfermodi, asynchroner und synchroner Datentransfer, vollzogen werden (siehe Abbildung 4.2 und Abbildung 4.3, [33]). Alle weiteren Busphasen werden nur im asynchronen Transfermodus betrieben.

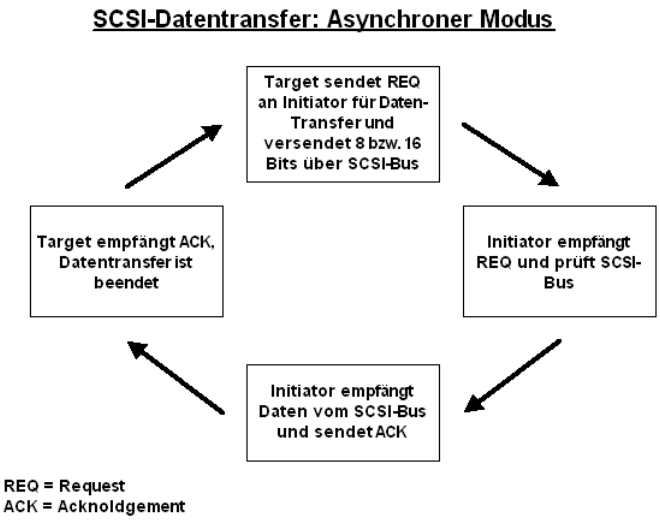


Abbildung 4.2 SCSI-Datentransfermodi: Asynchroner Modus (siehe [33])

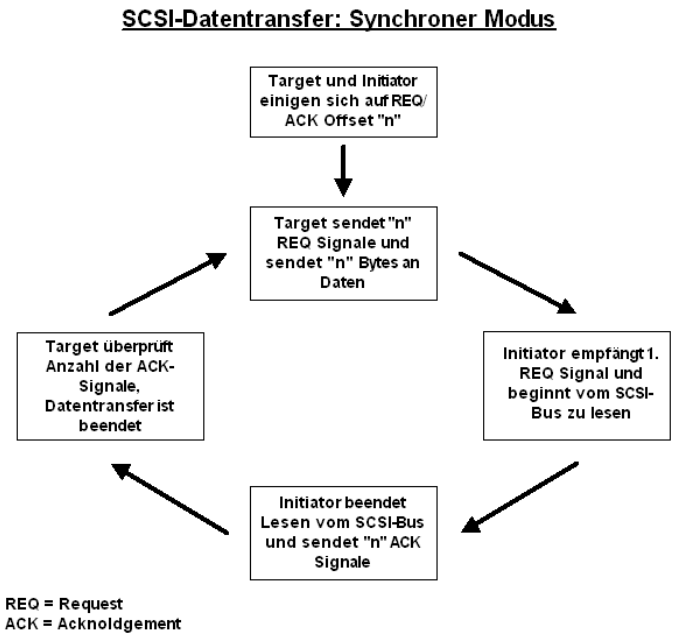


Abbildung 4.3 SCSI-Datentransfermodi: Synchroner Modus (siehe [33])

Wie in der Abbildung zu erkennen, werden im asynchronen Transfermodus für z. B. einen 8-Bit-SCSI-Bus jeweils ein Request und ein direkt nachfolgendes Acknowledgement für ein 8-Bit-Datenpaket benötigt (REQ/ACK Handshake). Im Gegensatz hierzu einigen sich im synchronen Transfermodus Initiator und Target auf einen Offset-Wert n, der die Anzahl der möglichen zu übertragenden Datenpakete angibt. Es werden dann n aufeinanderfolgende Requests und nachfolgende Datenpakete mit jeweils 8 Bits an den Initiator gesendet. Dieser sendet im Anschluss n Acknowledgements an das Target für korrekt eingegangene Datenpakete. Hierdurch wird im synchronen Datentransfermodus eine höhere Datenübertragungsgeschwindigkeit erreicht. Es wird im Vergleich zum asynchronen Modus von einer doppelten Transfergeschwindigkeit aus-

gegangen. In den Produktblättern der Festplatten werden häufig Angaben für beide Transfermodi angegeben.

Wie in Abbildung 4.2 und Abbildung 4.3 dargestellt, müssen für jede IO-Operation die beschriebenen Phasen durchlaufen werden. Die Performance zur Verarbeitung des IO-Zugriffs ist somit vornehmlich von den folgenden Faktoren abhängig:

- *Übertragungsmodus:* Wie bereits beschrieben, kann in der Information Transfer Phase zwischen dem asynchronen und synchronen Transfermodus gewählt werden. Hierbei ist ein Geschwindigkeitsvorteil von nahezu 100 % zu erwarten.
- *IO-Größe:* Jede IO-Operation durchläuft die bereits beschriebenen unterschiedlichen Phasen der Verarbeitung unabhängig davon, wie viele Daten verarbeitet werden sollen. Der mit der IO-Verarbeitung verbundene Overhead wird somit bei steigender Größe der IOs verkleinert [139]. Zur Minimierung des Overheads existieren weitere Verfahren, wie z. B. SCSI-Packetization, wobei mehrere SCSI-Phasen zu Paketen (Packets oder Information Units) zusammengefasst werden, die dann gesendet werden.
- *Anzahl der an den SCSI-Bus angeschlossenen Geräte:* Bei zunehmender Anzahl der an den SCSI-Bus angeschlossenen Geräte steigt die Auslastung des Busses und die damit verbundenen Verzögerungen zum Datentransfer. Eine Analyse mit nachfolgender Modellierung von mehreren Festplatten an einem SCSI-Bus wurde von Barve, Shriver und Gibbons in [3] durchgeführt.

4.1.4. Caching- und Queueing-Strategien

Derzeitige SCSI-Festplatten bestehen aus einem Cache zur Speicherung der IO-Daten und einer Queue für die in diesem Fall zu verarbeitenden SCSI-Commands. Die Queue wird seit der Einführung von SCSI-2 und der darin enthaltenen Funktion Tagged Command Queueing benötigt, da somit eine Festplatte bis zu 256 SCSI-Commands entgegen nehmen kann. Sowohl die Queue als auch der Cache ermöglichen den Einsatz von Optimierungsstrategien, die nachfolgend kurz beschrieben werden. Eine detaillierte Beschreibung der Caching und Queueing-Strategien befindet sich u. a. in [99] und [107].

4.1.4.1. Caching-Strategien

Der Cache einer Festplatte dient der Optimierung von Lese- und Schreiboperationen. Die Verwaltung des Cache wird vom Festplatten-Controller durchgeführt. Zur Optimierung der IO-Zugriffe unter Verwendung des Cache existieren in Abhängigkeit der Lastsituation verschiedene Optimierungsstrategien. Es können hierbei nach [28] vier Basisalgorithmen definiert werden (je zwei für lesende und schreibende IO-Operationen), die nachfolgend beschrieben werden.

- *Caching-Algorithmen für Lesezugriffe:* Lesezugriffe können optimiert werden, indem die gesuchten Daten sich bereits im Festplatten-Cache befinden, so dass nicht mehr auf das sehr viel langsamere Festplattenmedium zugegriffen werden muss. Die Schwierigkeit hierbei ist die Vorhersage der Daten, die in den Cache geschrieben werden sollen bzw. auf die in Zukunft zugegriffen wird. Es existieren hierzu die folgenden zwei Algorithmen:
  - *Last Recently Used (LRU) Caching:* Bei diesem Algorithmus werden die Daten, auf die in der jüngsten Vergangenheit zugegriffen wurde, in den Cache geschrieben. Es wird hierbei davon ausgegangen, dass die IO-Last sog. Hot-Spots (Lokalitäten) aufweist, d. h. es existieren Datenmengen auf die mehrfach zugegriffen wird, wie z. B. bei Datenbanken die Indexdaten. Wesentlich für die Effizienz dieses Algorithmus ist die Erkennung derartiger Hot-Spots.

- *Read-Ahead Caching:* Dieser Algorithmus basiert im Gegensatz zum LRU-Verfahren auf der Annahme, dass Daten sequentiell angefordert werden. Es werden somit jeweils die sequentiell nachfolgenden Daten des IO-Streams in den Cache geschrieben (Prefetching). Im Gegensatz zum LRU-Verfahren wird davon ausgegangen, dass bereits angeforderte Daten nicht wieder benötigt werden. Es existieren für das Read-Ahead Caching verschiedene Strategien, welche definieren, wie viele Daten in den Cache geladen werden sollen (z. B. immer bis zum Ende des Tracks, partial track buffering), wann Read-Ahead durchgeführt wird (z. B. in Abhängigkeit der Plattenauslastung) und ab welcher Position der Platte das Einlesen der Daten in den Cache beginnen soll (z. B. ab dem Zeitpunkt und der Position, wo der Schreib-/Lesekopf sich auf dem richtigen Track befindet, on-arrival read-ahead). Eine Beschreibung dieser verschiedenen Ausprägungen wird in [99] und [84] dargestellt. Damit bei einer IO-Last mit mehreren parallelen IO-Zugriffen auch mehrere Read-Aheads parallel durchgeführt werden können, ist der Cache einer Festplatte in mehrere Segmente unterteilt (Segmentierung). Jedem sequentiellen IO-Stream kann dann ein Segment zugewiesen werden, so dass aus Sicht des Clients jeder IO-Stream seinen eigenen Cache besitzt.

- *Caching-Algorithmen für Schreibzugriffe:* Das Caching von Schreibzugriffen besitzt die Hauptaufgabe, verschiedene Versionen von Daten zu verwalten, d. h. existiert ein Datensatz sowohl auf der Festplatte als auch im Festplatten-Cache und wird nur der Datensatz im Cache modifiziert, so existieren zwei verschiedene Versionen des Datensatzes. Caching-Algorithmen für Schreibzugriffe besitzen somit die Zielsetzung, derartige „korrumpierte Datensätze“ zu vermeiden bzw. Anwendungen nur Zugriff auf aktuellste Datensätze zu erlauben. Es existieren hierzu die folgenden zwei Algorithmen:

  - *Write-Through Caching:* Bei Schreibzugriffen auf die Festplatte werden bei diesem Algorithmus die Daten direkt auf das Plattenmedium geschrieben. Eine Kopie der zu schreibenden Daten kann hierbei im Cache gespeichert werden, wobei diese dann bei Änderungen immer mit den Daten auf dem Plattenmedium synchronisiert werden muss. Es handelt sich hierbei um den konservativsten, aber auch sichersten Schreibalgorithmus.

  - *Write-Back Caching:* Dieser Algorithmus wird auch als Write-Behind-Cache bezeichnet, da sobald die Daten von den IO-Zugriffen in den Cache geschrieben wurden, diese ein entsprechendes Acknowledgement bekommen, so dass die IO-Prozesse direkt für weitere Aufgaben zur Verfügung stehen. Der Vorteil dieses Verfahrens ist, dass aus Sicht der Applikationen eine sehr viel höhere Schreib-Performance vorliegt, da die Schreibzugriffe von der Festplatte direkt als bearbeitet zurückgesendet werden. Weiterhin besitzt die Festplatte die Möglichkeit mehrere Schreibzugriffe zu einer Gruppe zusammenzufassen und auf das Medium zu schreiben oder auch mittels verschiedener Queueing-Strategien die Schreibzugriffe zur Minimierung der Zugriffszeiten zu sortieren (siehe Abschnitt 4.1.4.2.). Dies führt zu erheblichen Performance-Vorteilen im Vergleich zu einzelnen Schreibzugriffen. Der Nachteil dieses Verfahrens ist, dass z. B. bei einem Systemabsturz oder einem Ausfall des Cache alle Daten verloren sind.

4.1.4.2. Queueing-Strategien

Werden in einem System mehrere IOs gleichzeitig bzw. in relativ kurzen Abständen an eine Festplatte gesendet, so befinden sich diese bei Unterstützung von Tagged Command Queueing in der Queue der Festplatte. Jeder IO entspricht einem Lese- oder Schreibzugriff, wobei die entsprechenden Daten zufällig verteilt oder sequentiell hintereinander auf dem Plattenmedium liegen können. Bei zufällig verteilten Daten besitzt der Festplatten-Controller die Möglichkeit die IO-Zugriffe derart zu sortieren, dass Zugriffszeiten in Form von Seek- und Latenzzeiten minimiert werden. Zur Sortierung der IOs existieren eine Vielzahl an unterschiedlichen Strategien, die in die folgenden drei Kategorien eingeordnet werden können:

- *First In, First Out:* Die Festplatte verarbeitet die IO-Zugriffe in der Reihenfolge, in der sie generiert wurden, wobei die Verteilung der angeforderten Daten auf der Platte hierbei nicht beachtet wird.
- *Seek Time Optimization:* Der Festplatten-Controller analysiert alle sich in der Queue befindlichen IO-Zugriffe und sortiert diese anhand der Zylindernummern, die angefordert werden, relativ zu der augenblicklichen Position des Schreib-/Lesekopfes. Dieses Verfahren wird als SSTF (Shortest Seek Time First) oder Elevator-Verfahren bezeichnet, wobei dem Elevator-Verfahren das Prinzip eines Fahrstuhls unterliegt, d. h. der Schreib-/Lesekopf arbeitet zunächst alle Aufträge in einer Richtung ab im Gegensatz zum SSTF-Verfahren, bei dem nur der geringste Abstand der Zylinder relevant ist und somit der Fall eintreten kann, dass äußere Zylinder weniger häufig angefahren werden. Für das Elevator-Verfahren können weiterhin SCAN und C-SCAN sowie LOOK und C-LOOK unterschieden werden. Der SCAN-Algorithmus durchläuft immer alle Zylinder bis zum letzten im Gegensatz zum LOOK, der immer überprüft, ob in der zu durchlaufenden Richtung weitere IO-Zugriffe vorhanden sind. Ist dies nicht der Fall, wird beim LOOK-Verfahren bereits vor Erreichen des letzten Zylinders die Richtung geändert. Die Verfahren C-SCAN und C-LOOK bedeuten hingegen, dass die Platte immer nur in einer Richtung durchlaufen wird, d. h. beim Drehen der Richtung läuft der Schreib-/Lesekopf zunächst wieder an den Anfang zurück und beginnt erst dann wieder die Arbeit.
- *Access Time (Seek and Latency) Optimization:* Die zuvor beschriebenen Verfahren zur Optimierung der Seek-Time berücksichtigen nicht die Latenzzeit, d. h. bei Zugriffen auf den gleichen oder benachbarten Zylinder wird die Lage der Sektoren in dem entsprechenden Track nicht berücksichtigt. Verfahren, die sowohl die Seek-Time als auch die Latenzzeit optimieren, werden als „multiple command reordering“- oder „multiple command optimization“- bzw. „Shortest Positioning Time First“-Verfahren (SPTF) bezeichnet. Beispiele hierfür sind der STF- (Shortest Time First, [98]) sowie der SATF-Algorithmus (Shortest Access Time First, [51]).

In Abhängigkeit der zu erwartenden oder vorhandenen Lastsituation muss das entsprechende Scheduling-Verfahren ausgewählt werden. In [107] wird die Wahl des Disk-Scheduling-Verfahrens diskutiert. Eine detaillierte Beschreibung der verschiedenen Scheduling-Verfahren befindet sich in [140].

4.1.5. Verarbeitungspfad einer IO-Operation im System

Bei der Verarbeitung von IO-Operationen werden in einem Computersystem eine Vielzahl von verschiedenen Schichten durchlaufen. Die oberste Schicht ist hierbei die Ebene der Applikation, welche die Anfragen generiert. Die unterste Schicht ist die Ebene des IO-Systems, d. h. in diesem Fall die Festplatte oder z. B. ein RAID-System. Die verschiedenen Ebenen und die jeweils damit verbundenen Aktivitäten werden in diesem Abschnitt beschrieben. Als Beispiel wird in [139] der Verarbeitungspfad einer IO-Operation mit den jeweiligen Aktivitäten dargestellt.

In der Abbildung wird als Beispiel die Verarbeitung eines Leseprozesses dargestellt, der durch einen User oder eine Applikation generiert wurde. Die Verarbeitung des Leseprozesses erfolgt zunächst auf Ebene des Betriebssystems, indem ein IO-Control-Block, der aus einem READ-Command, einer Adresse (LBA), einer Angabe zur Anzahl der zu lesenden Blöcke und der Hauptspeicheradresse besteht, erzeugt wird. Über einen Interrupt wird der vom Betriebssystem bereitgestellte Device Driver des IO-Systems angestoßen, der die Kommunikation mit dem Host Bus Adapter (HBA) übernimmt. Der IO-Control-Block wird nun vom Device Driver über den Systembus an den HBA gesendet, der, falls vorhanden, Speicherplatz im Cache des HBA für die zu lesenden Daten reserviert und den SCSI-Controller Chip für den Zugriff auf das IO-System (z. B. Festplatte oder RAID-System) instruiert. Der SCSI-Controller muss nun als SCSI-Initia-

tor, wie bereits in Abschnitt 4.1.3.1. beschrieben, gemäß den SCSI-Busphasen auf das Recht des SCSI-Buszugriffs warten (Arbitration-Phase). Wird ihm das Recht erteilt, so schickt der SCSI-Controller den Read-Befehl an das IO-System (SCSI-Target). Anschließend beendet der SCSI-Controller den Kontakt zum IO-System, gibt den SCSI-Bus wieder frei und wartet auf die nächste IO-Operation.

Das IO-System verarbeitet den Lesezugriff durch Positionierung des Schreib-/Lese-Kopfes über den angeforderten Track (Seek) und die Drehung der Platters bis die gesuchten Sektoren sich unter dem Plattenkopf befinden (Rotationslatenz). Sobald sich die Daten unter dem Platten-Kopf befinden, beginnt der Datentransfer zum Festplatten-Cache. In Abhängigkeit der vom Festplatten-Controller verwendeten Scheduling- und Caching-Strategie kann die IO-Verarbeitung auch in leicht abgewandelter Form erfolgen.

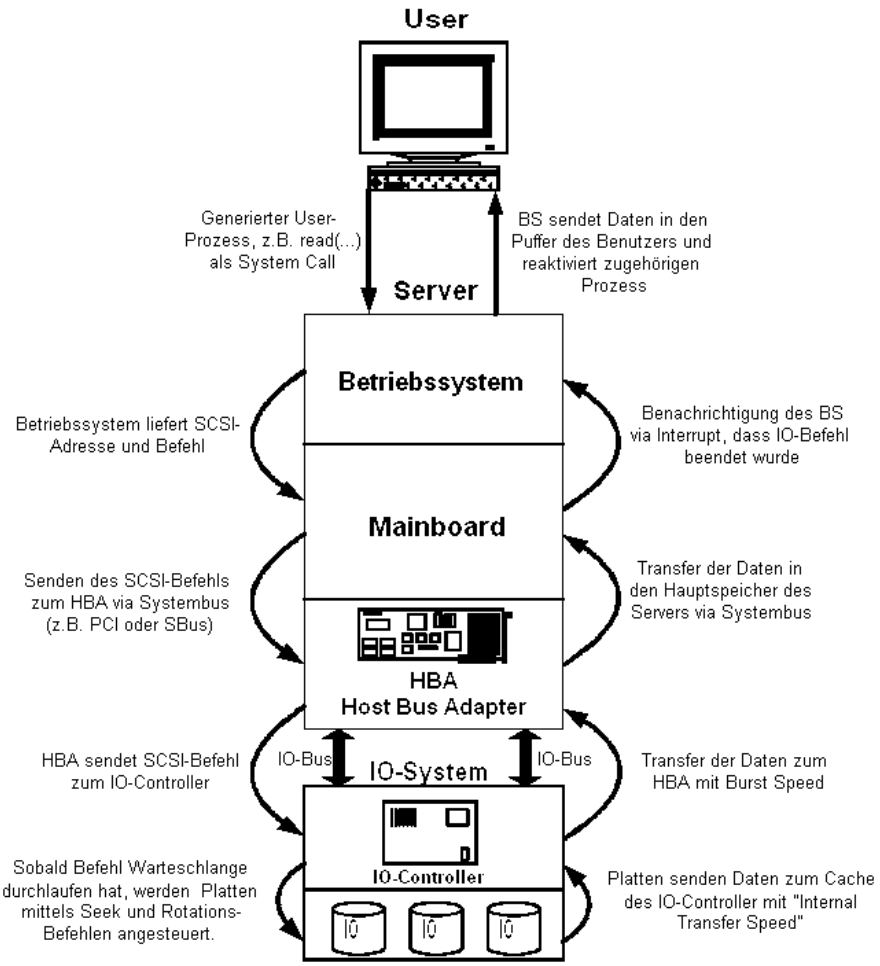


Abbildung 4.4 Pfad einer IO-Operation (siehe [139])

Nachdem die Festplatte die angeforderten Daten in den Platten-Cache geschrieben hat, kann auch die Festplatte als SCSI-Inititator agieren und somit nach der Arbitration-Phase die gelesenen Daten zum HBA schicken. Der HBA sendet die Daten dann via PCI-Datentransfer (memory write transfer) in den Server-Hauptspeicher, wobei die Zieladresse hierzu im IO-Control-Block gespeichert ist. Nachdem dieser Transfer beendet wurde, wird vom HBA ein Interrupt erzeugt, der dem Device Driver des Betriebssystems einen „Completion-Status“ übermittelt. Das Betriebssystem besitzt somit die Information, dass die Leseoperation beendet wurde.

4.1.6. Anwendungsbeispiel: IO-Architektur von Solaris

Die IO-Architektur von Solaris kann, wie in Abbildung 4.5 dargestellt, in drei Bereiche unterteilt werden. Auf oberster Ebene befinden sich die Applikationen, die mittels der in den APIs der Programmiersprachen zur Verfügung gestellten IO-Operationen auf das IO-System zugreifen. Unterschieden werden die IO-Operationen hauptsächlich in der Verarbeitungsstrategie und der zugehörigen IO-Pfade. Es existieren unter Solaris vier Gruppen von IO-Befehlen: System IO, Standard IO, Asynchronous IO und Memory Mapping IO.

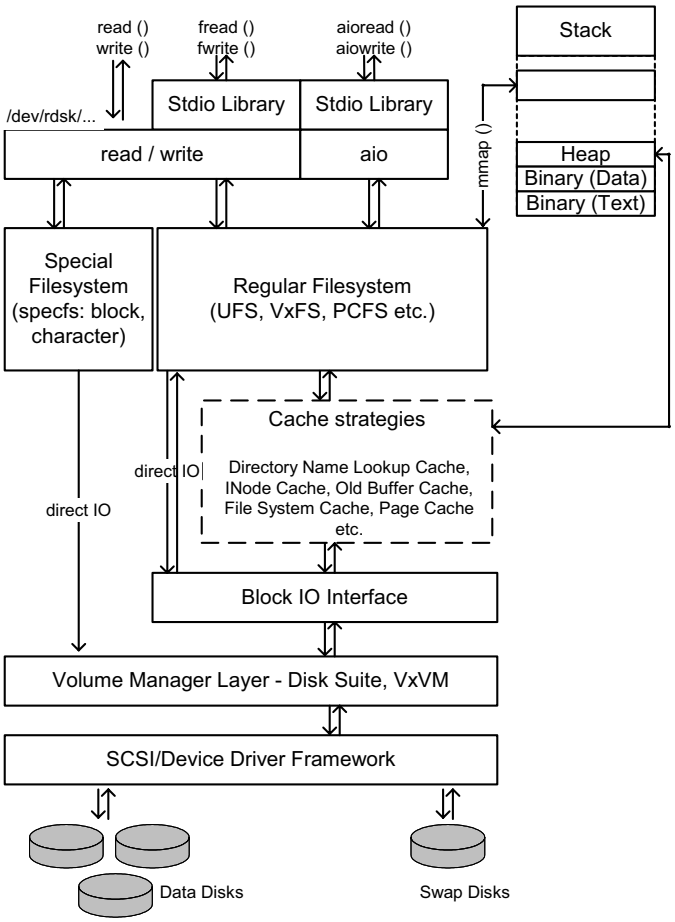


Abbildung 4.5 IO-Architektur von Solaris (siehe [67])

Die mittlere Ebene umfasst den Bereich der Dateisysteme, d. h. der Bereich, auf den die API-IO-Operationen direkt zugreifen. Das Dateisystem entspricht somit der Schicht zwischen Applikation und Speichersubsystem. Zu unterscheiden sind hierbei die Regular Filesystems, d. h. beispielsweise das Universal File System (UFS), von den Special-Filesystems, die entweder einem Character oder Block Device entsprechen können. SCSI-Festplatten werden typischerweise als Block-Device verwaltet, so dass aus Sicht des Betriebssystem-Kernels die SCSI-Festplatte einer Menge von beliebig adressierbaren logischen Blöcken entspricht. Die Special-Filesystems werden häufig auch als Raw Devices bezeichnet, da sie einen nahezu direkten Zugriffspfad zum IO-System bieten. Als Character Device werden z. B. Bandlaufwerke verwaltet.

Der dritte Bereich und somit die unterste Ebene ist die Device- und Hardware-Schicht. Auf dieser Ebene befinden sich die Device Driver, welche die Verbindung zur Hardware ermöglichen.

SCSI-Geräte (Festplatten, CD-ROMs etc.) werden unter Solaris mit dem SCSI-Device Driver (SD) angesteuert. Der SCSI-Device Driver ist für die Umsetzung der IO-Operationen ausgehend von den Applikationen in entsprechende SCSI-Commands zuständig sowie für die Kommunikation mit der Hardware, wie in diesem Fall via SCSI-Bus.

Unter Solaris existieren für die Verwaltung der IO-Operationen zwei Warteschlangen (Active und Wait Queue). Die Active Queue beinhaltet alle IO-Aufträge, die sich entweder auf dem IO-Bus oder in der Festplatte befinden und die Wait Queue enthält alle IO-Aufträge seitens des Servers auf Ebene des Device Drivers. Mit Hilfe der SCSI-2-Eigenschaft Tagged Command Queueing kann eine Festplatte bis zu 256 SCSI-Commands puffern. Sobald die Queue der Festplatte keine IO-Aufträge mehr annehmen kann, füllt sich die Warteschlange des Device Drivers. Im Fall von Solaris, kann auch in der Queue des Device Drivers ein Scheduling-Algorithmus ausgeführt werden, der, wie für die Festplatte beschrieben, eine Sortierung der IO-Zugriffe durchführt. Für Solaris stehen hierbei zwei Algorithmen zur Auswahl: FIFO und der Elevator-Algorithmus. Weitere Informationen befinden sich in Abschnitt 4.2.1. zur Beschreibung des IO-Monitorings.

Weiterführende Informationen zur IO-Architektur von Solaris befinden sich in [114], [139] und [67].

4.2. IO-Monitoring und -Benchmarking

Das Ziel dieses Kapitels ist die IO-Messung und -Analyse, die im nachfolgenden fünften Kapitel als Grundlage zur Validation des IO-Modells genutzt wird. Es werden hierfür IO-Lastszenarien definiert und mittels IO-Benchmarks und IO-Monitoring-Werkzeugen durchgeführt und gemessen. Wie bereits beschrieben, wurde sich für das Benchmarking und Monitoring im Rahmen dieser Arbeit auf frei verfügbare bzw. im Betriebssystem bereits enthaltene Werkzeuge beschränkt. Ein Überblick über existierende und die tatsächlich verwendeten Werkzeuge wird in den folgenden Abschnitten gegeben. Hierbei werden ausschließlich Werkzeuge für oder von Solaris (Version 2.8) betrachtet, die jedoch zum größtenteils auch für andere UNIX-Systeme existieren. MS Windows besitzt zur IO-Messung beispielsweise den Windows Systemmonitor, der in [32] detailliert beschrieben wird.

4.2.1. IO-Monitoring

Für das IO-Monitoring werden in den folgenden Abschnitten die Standard-Monitoring-Werkzeuge SAR, VMSTAT und IOSTAT von Solaris und das frei erhältliche SE Performance Toolkit, das mit dem Skript SIOSTAT eine Erweiterung zum Standard-Werkzeug IOSTAT anbietet, beschrieben.

Die Werkzeuge SAR, IOSTAT und VMSTAT befinden sich im Installationsumfang von Solaris. Zusätzlich zu den Standard-Monitoren existiert ein von Adrian Cockcroft und Richard Pettit entwickeltes SE Performance Toolkit, das mittels einer Skriptsprache die Definition eigener Monitoring-Programme erlaubt, wie z. B. das SIOSTAT. Das SE Performance Toolkit wird kostenlos auf der Homepage http://www.setoolkit.com angeboten und von Cockcroft und Pettit in [20] detailliert beschrieben. Der Unterschied zwischen diesen Werkzeugen ist der Detaillierungsgrad und im Hinblick auf das Skript SIOSTAT auch die Interpretation der IO-Statistiken. Im folgenden werden die unterschiedlichen Werkzeuge vorgestellt und verglichen. Zuvor wird die Kernel Statistics Facility-Datei „KSTAT“ beschrieben, welche die Definition der zum IO-Monitoring benötigten Zähler des Betriebssystems (Counter) beinhaltet und auf welcher jedes der IO-Monitoring-Werkzeuge basiert.

4.2.1.1. Kernel Statistics Facility Datei KSTAT

Alle IO-Aktivitäten, d. h. Zugriffe auf die Speichersubsysteme, wie in diesem Fall die Festplatte, werden unter Solaris mittels der Datei KSTAT (Kernel Disk Information Statistics) protokolliert und beschrieben. Die hierzu verwendeten Parameter werden in Tabelle 4.2 dargestellt [114].

Counter	Beschreibung
nread	Gelesene Datenmenge in Bytes
nwritten	Geschriebene Datenmenge in Bytes
reads	Anzahl der Leseoperationen
writes	Anzahl der Schreiboperationen
wtime	kumulative Wartezeit (pre-service)
wlentime	kumulatives Produkt von length und time
wlastupdate	letzter Zeitpunkt, an dem die Wait-Queue geändert wurde
rtime	kumulative Laufzeit (service)
rlentime	kumulatives Produkt von length und time
rlastupdate	letzter Zeitpunkt, an dem die Active-Queue geändert wurde
wcnt	Anzahl der Elemente im Wartestatus
rcnt	Anzahl der Elemente im Aktivstatus

Tabelle 4.2 Counter der Datei KSTAT

Wie bereits in Abschnitt 4.1.6. beschrieben und in Tabelle 4.2 dargestellt, werden für die IO-Statistiken zwei Listen, Active Queue und Wait Queue, verwaltet. Die Wait Queue beinhaltet alle IO-Aufträge, die vom Betriebssystem generiert wurden, jedoch deren Verarbeitung noch nicht begonnen hat, da z. B. die Festplatte die IO-Zugriffe nicht annimmt, wenn sich beispielweise bereits 256 IO-Zugriffe im Festplatten-Cache befinden (im Fall von Tagged Command Queueing). Die Active Queue verwaltet alle Aufträge, die augenblicklich verarbeitet werden bzw. sich bereits im Cache der Festplatte befinden. Solaris interpretiert alle IO-Aufträge, die vom Rechner an das IO-System gesendet werden als aktiv, d. h. sobald sich die Aufträge auf dem IO-Bus befinden werden sie als Bestandteil der Active Queue interpretiert.

Wie in Tabelle 4.2 beschrieben, werden als IO-Aktivität die Zahl der Lese- und Schreibzugriffe und die damit verbundenen Datenmengen bezeichnet. Weiterhin wird getrennt nach Active und Wait Queue sowohl die aktive Zeit in der Queue (rtime, wtime) als auch die Warteschlangenlängen (wcnt, rcnt) und deren Änderungen (wlentime, rlentime) protokolliert.

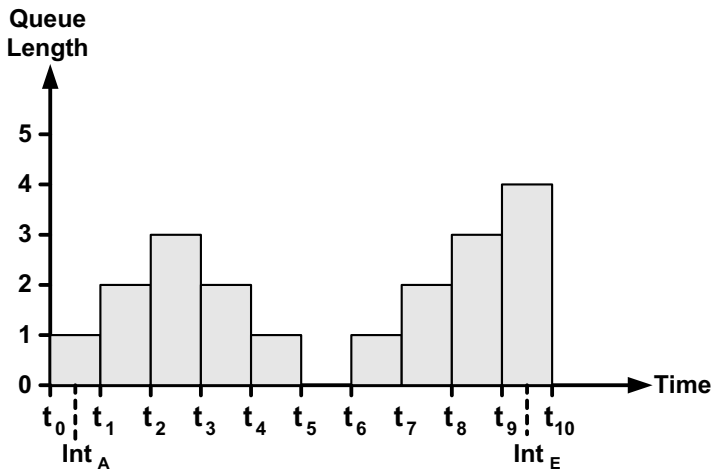


Abbildung 4.6 Darstellung von wlentime der Datei KSTAT

Die Parameter rtime und wtime werden als laufende Summen der aktiven Zeit berechnet, wobei aktive Zeit bedeutet, dass sich mindestens ein Auftrag in der Queue befindet. Die Parameter rlentime und wlentime werden als laufende Summe über das Produkt der Warteschlangenlänge und der bis dahin vergangenen Zeit seit der letzten Änderung der Warteschlangenlänge ermittelt. Das dabei verwendete Produkt entspricht einer Riemannsumme aufgetragen mit der Warteschlangenlänge auf der Y-Achse und der Zeit auf der X-Achse (siehe Abbildung 4.6). Bei jeder Änderung der Warteschlangenlänge (Auftrag tritt in die Queue ein bzw. verlässt sie) wird die vergangene Zeit seit der letzten Änderung zur aktiven Zeit dazu addiert, solange die Warteschlangenlänge ungleich 0 ist. Die in Abbildung 4.6 dargestellten Parameter t_i symbolisieren jeweils den Zeitpunkt eines Austritts bzw. Eintritts eines Auftrags in die Warteschlange. In Abbildung 4.6 sind zwei Parameter Int_A und Int_E abgebildet, die das gesamte Betrachtungsintervall von z. B. IOSTAT darstellen. Der Zeitstempel wlastupdate bzw. rlastupdate würde im Fall von Int_A somit t_0 liefern.

4.2.1.2. Standardwerkzeuge für das IO-Monitoring

In diesem Abschnitt werden die unter Solaris verfügbaren Monitoring-Werkzeuge SAR, IOSTAT und VMSTAT beschrieben. Für jedes Programm werden die Möglichkeiten der IO-Messung aufgezeigt und die wesentlichen Unterschiede zwischen diesen Werkzeugen dargestellt.

- **VMSTAT:** Das Werkzeug VMSTAT bietet Informationen über Virtual Memory, Memory, Paging, System Calls, CPU-Auslastung und die Disk-Activities. Eine Beispielausgabe wird in Tabelle 4.3 dargestellt. Es können optional weitere Statistiken über Caching, Interrupts, Paging Activity, System Events und Swapping dargestellt werden. Eine Beschreibung des Werkzeugs befindet sich in [114] und [67]**.** Die zur IO-Messung relevante Spalte ist mit „disk“ bezeichnet. Hier werden pro Festplatte im System die gemessenen IO-Zugriffe dargestellt. Die IO-Zugriffe werden hierbei nicht in schreibende und lesende Zugriffe unterschieden und es werden auch nicht die verschiedenen Partitionen einer Festplatte berücksichtigt.

procs			Memory		page							disk			faults			cpu		
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	f0	s3	s5	in	sy	cs	us	sy	id
0	0	0	689.588	187.340	0	0	0	0	0	0	0	0	0	0	11	43	97	20	4	76

Tabelle 4.3 Beispielausgabe von VMSTAT

- **SAR:** Der System Activity Reporter (SAR) liefert eine IO-Statistik aggregiert nach IO-System (z. B. Festplatte) und zugehöriger Partition, definiert als Device. Es werden somit pro Zeitstempel und Device die folgenden Informationen dargestellt:
  - *time:* Zeitstempel der Messung
  - *device:* Name der Festplatte und zugehöriger Partition
  - *%busy:* Auslastung des IO-Systems, d. h. prozentualer Zeitanteil, in dem das IO-System aktiv war (sich mindestens ein IO-Auftrag im System befand)
  - *avque:* mittlere Anzahl wartender Aufträge
  - *r+w/s:* Anzahl der lesenden und schreibenden Operationen pro Sekunde
  - *blks/s:* Transferierte Blöcke pro Sekunde mit 1 Block = 512 Byte
  - *avwait:* mittlere Wartezeit in ms
  - *avserv:* mittlere Bedienzeit in ms

Eine Beispielausgabe wird in Tabelle 4.4 dargestellt. Im Vergleich zum VMSTAT werden beim SAR wesentlich detailliertere Informationen zur IO-Aktivität geliefert. Die Messdaten werden pro Partition der Festplatte erhoben und es wird zwischen wartenden und in der Verarbeitung befindlichen IOs unterschieden. Desweiteren werden pro IO-Zugriff Warte- und Bedienzeiten sowie die Auslastung des IO-Systems ausgegeben. Zur Interpretation der Parameter avwait und avserv wird auf den Abschnitt 4.2.1.3. verwiesen.

time	device	%busy	avque	r+w/s	blks/s	avwait	avserv
01:00:00	sd50	1	0,6	7	104	76,3	8,9
	sd50,a	0	0	0	0	0	0
	sd50,b	0	0	0	0	0	0
	sd50,c	0	0	0	0	0	0
	sd50,d	1	0,6	6	100	80,8	9,3
	sd50,e	0	0	0	2	6,1	1,9
	sd50,f	0	0	0	0	4,4	37,6
	sd50,g	0	0	0	2	1,2	2
	sd50,h	0	0	0	0	0	0

Tabelle 4.4 Beispielausgabe der IO-Statistiken mit SAR

- **IOSTAT:** Das Monitoring-Werkzeug IOSTAT liefert im Vergleich zu den bisherigen Programmen die detaillierteste Beschreibung der IO-Aktivität. Wie bereits für SAR dargestellt, liefert auch IOSTAT eine nach Festplatte und Partition aufgeteilte Sicht (siehe Tabelle 4.5). Es werden die Anzahl der IO-Operationen getrennt nach Lese- und Schreiboperationen sowie Angaben über Bedien- und Antwortzeiten sowie Auslastung des IO-Systems angegeben. Die in Tabelle 4.5 dargestellten Parameter können wie folgt beschrieben werden:
  - *device:* Name der Festplatte und zugehörige Partition
  - *r/s:* Anzahl der Leseoperationen pro Sekunde
  - *w/s:* Anzahl der Schreiboperationen pro Sekunde
  - *kr/s:* Gelesene Datenmenge in KB pro Sekunde
  - *kw/s:* Geschriebene Datenmenge in KB pro Sekunde
  - *wait:* Mittlere Anzahl wartender Operationen (Warteschlangenlänge)
  - *actv:* Mittlere Anzahl an Operationen, die bedient werden
  - *svc_t:* Mittlere Bedienzeit pro IO
  - *%w:* Auslastung der Wait-Queue (prozentualer Anteil der Zeit, in dem Operationen auf ihre Verarbeitung warten)
  - *%b:* Auslastung der Active Queue (prozentualer Anteil der Zeit, in dem die Platte aktiv ist und Operationen verarbeitet)

device	r/s	w/s	Kr/s	Kw/s	wait	actv	svc_t	%w	%b
sd50,a	5,9	9,0	45,7	73,7	0,7	0,2	57,8	0	6
sd50,b	0,1	0,4	8,0	9,9	0	0	12,5	0	0
sd50,c	0,8	0,2	6,8	1,3	0	0	13,3	0	1
sd50,d	5,6	1,9	31,1	14,5	0	0,1	13,2	0	5

Tabelle 4.5 Beispielausgabe von IOSTAT

Im Vergleich zum SAR existieren im IOSTAT zusätzliche Angaben zur Wait und Active Queue (%w und actv) sowie eine Unterscheidung des IO-Durchsatzes in lesende und schreibende Zugriffe. Wie bereits für SAR beschrieben wird für die Interpretation der Bedienzeit svc_t auf den Abschnitt 4.2.1.3. verwiesen.

4.2.1.3. IO-Monitoring mit dem SE Performance Toolkit

Das Monitoring-Werkzeug SE Performance Toolkit von Pettit und Cockcroft dient der Performance-Überwachung von Solaris-Rechnern [20]. Das Werkzeug basiert auf der SymbEL Language, eine an der Programmiersprache C orientierte Skriptsprache. Das Toolkit beinhaltet neben vielen weiteren das Skript SIOSTAT, das eine Erweiterung zu dem beschriebenen Werkzeug IOSTAT darstellt. Neben einigen wenigen zusätzlichen Parametern (siehe Tabelle 4.7) ist eine der wesentlichen Neuerungen gegenüber den bisher beschriebenen Monitoring-Programmen die Berechnung der Bedien- und Antwortzeit pro IO-Zugriff.

Die IO-Aktivitäten unter Solaris werden mit Hilfe der Active und Wait Queue beschrieben. Wie bereits dargestellt, beinhaltet die Active Queue alle IO-Aufträge, die sich entweder auf dem IO-Bus oder in der Festplatte befinden. Festplatten der früheren Generation ohne Unterstützung von Tagged Command Queueing konnten keine IO-Zugriffe puffern und somit befand sich zu jedem Zeitpunkt nur ein IO-Auftrag in der Festplatte. Aufgrund dessen konnte die Zeit eines Auftrags in der Active Queue als Bedienzeit definiert werden. Derzeitige Festplatten unterstützen hingegen Tagged Command Queueing und können somit bis zu 256 IO-Zugriffe puffern. Bei mehr als einem IO-Zugriff in der Festplatte entstehen somit Wartezeiten für die gepufferten IO-Aufträge und die Verweilzeit in der Active Queue entspricht nicht mehr der reinen Bedienzeit sondern der Antwortzeit inklusive aufgetretener Wartezeiten. Die bereits beschriebenen Standard-Monitoring-Werkzeuge SAR und IOSTAT berücksichtigen das durch Tagged Command Queueing resultierende Festplattenverhalten nicht und interpretieren weiterhin die Verweilzeit eines IO-Auftrags in der Active Queue als Bedienzeit.

Aus diesem Grund modifizierten Cockcroft und Pettit die Berechnung der Bedien- und Antwortzeit pro IO-Zugriff in der Form, dass anhand der IO-Statistiken der Datei KSTAT die genannten Werte unter Anwendung der Gesetze Little's Law und Utilization Law (Operationale Analyse, [52]) ermittelt werden, so dass auch für die Active Queue zwischen Warte- und Bedienzeit unterschieden wird. In Tabelle 4.6 wird als Beispiel eine Ausgabe des SIOSTAT-Skripts dargestellt. Die Parameter werden in Tabelle 4.7 beschrieben.

5 Sek.	throughput				wait queue				active queue			
disk	**r/s**	**w/s**	**Kr/s**	**Kw/s**	**qlen**	**res_t**	**svc_t**	**%ut**	**qlen**	**res_t**	**svc_t**	**%ut**
sd50	1,0	411,2	8,0	19750,8	113,77	276,01	2,43	100	16	38,81	2,43	100
sd50,a	0	0	0	0	0	0	0	0	0	0	0	0
sd50,b	0,6	0	4,8	0	0,97	1618,34	1618,34	97	0,02	38,31	38,31	2
sd50,c	0,4	412,2	3,2	19782,8	112,81	273,4	2,42	100	15,98	38,72	2,42	100

Tabelle 4.6 Beispielausgabe von SIOSTAT

Die Ausgabe von SIOSTAT beschreibt die IO-Statistiken von einem SCSI-Device mit dem Namen „sd50" für ein Zeitintervall von 5 Sekunden. Ein Device kann eine Festplatte oder z. B. ein RAID-System repräsentieren. Die erste Zeile „sd50" beschreibt die IO-Aktivität summiert über alle Partitionen. Die folgenden Zeilen entsprechen den verschiedenen Partitionen, die auf dem Device konfiguriert sind. Abschließend werden die für SIOSTAT verwendeten Rechenvorschriften für die in Tabelle 4.7 dargestellten Parameter beschrieben [20].

	Parameter	Einheit	Beschreibung
Throughput	disk	---	Device bzw. Partition
	r/s	Anzahl / Sek.	Anzahl der Lesezugriffe pro Sekunde
	w/s	Anzahl / Sek.	Anzahl der Schreibzugriffe pro Sekunde
	Kr/s	KB / Sek.	gelesene Datenmenge pro Sekunde
	Kw/s	KB / Sek.	geschriebene Datenmenge pro Sekunde
Wait Queue	qlen	Anzahl	Anzahl der Aufträge in der Queue
	res_t	ms	Antwortzeit pro IO
	svc_t	ms	Bedienzeit pro IO
	%ut	%	Auslastung der Queue
Active Queue	qlen	Anzahl	Anzahl der Aufträge in der Queue
	res_t	ms	Antwortzeit pro IO
	svc_t	ms	Bedienzeit pro IO
	%ut	%	Auslastung der Queue

Tabelle 4.7 Parameter von SIOSTAT

Berechnung der Parameter von SIOSTAT

Als Basis für die Berechnung der Parameter von SIOSTAT dienen die Daten der Datei KSTAT (siehe Tabelle 4.2). Es werden zunächst die folgenden beiden Parameter definiert:

$$\text{etime [ms]} = \text{timestamp [end]} - \text{timestamp [start]}$$

$$\text{hr\_etime [ms]} = \text{wlastupdate [end]} - \text{wlastupdate [start]}$$

Hierbei wird der Parameter wlastupdate der KSTAT entnommen. Die Angaben Start und End bezeichnen jeweils einen Snap-Shot der IO-Statistiken, der am Anfang und Ende des Zeitintervalls erstellt wird. Der Parameter etime (elapsed time) beschreibt das gesamte Zeitintervall, das betrachtet wird. Mit hr_etime (high resolution elapsed time) wird die Zeitspanne bezeichnet, in der sich mindestens ein Auftrag in der Active Queue befindet. Die nun folgenden im Skript SIOSTAT implementierten Formeln sind in den Dokumenten [19] und [20] beschrieben.

Es werden die Durchsätze in Form von IOs pro Sekunde, Reads per Second (RpS) und Writes per Second (WpS), und für die Datenmenge in KB pro Sekunde, KB Read per Second (KBRpS) und KB Written per Second (KBWpS), berechnet. Hierzu werden die Parameter reads, writes, nread und nwrite aus der KSTAT verwendet. Da diese Parameter als fortlaufende Summen berechnet werden, müssen die Differenzen zum Start- und Endezeitpunkt des Intervalls gebildet werden, die mit [start] und [end] bezeichnet werden:

$$\text{RpS} = \frac{\text{reads[end]} - \text{reads[start]}}{\text{etime}} \; ; \; \text{WpS} = \frac{\text{writes[end]} - \text{writes[start]}}{\text{etime}}$$

$$\text{KBRpS} = \frac{\text{nread[end]} - \text{nread[start]}}{\text{etime}} \; ; \; \text{KBWpS} = \frac{(\text{nwrites})\text{[end]} - (\text{nwrites})\text{[start]}}{\text{etime}}$$

Es folgen die Parameter für die Wait Queue. Die Werte wlentime und wtime sind der KSTAT entnommen. Für eine eindeutige Schreibweise der Formeln, wurde für den Ausdruck %ut in Tabelle 4.7 die Bezeichnung util verwendet:

$$qlen = \frac{wlentime[end] - wlentime[start]}{hr\_etime}\ ;\ res\_t\,[ms] = \frac{qlen \bullet 1000}{RpS + WpS}$$

$$util\,[\%] = \frac{(wtime[end] - wtime[start]) \bullet 100}{hr\_etime}\ ;\ svc\_t\,[ms] = util \bullet \frac{10}{RpS + WpS}$$

Vergleichbare Berechnungen müssen für die Active Queue vorgenommen werden, wobei der Parameter wlentime durch rlentime, wtime durch rtime sowie qlen und ut durch die entsprechenden Werte der Active Queue ersetzt werden müssen.

Die Berechnungen der Antwortzeit und Bedienzeit für die beiden Warteschlangen bilden die wesentlichen Unterschiede zu den Monitoring-Werkzeugen SAR und IOSTAT. Für die Berechnung der Antwortzeit pro IO wurde Little's Law ($E[N]=\lambda*E[V]$) und für die Berechnung der Bedienzeit das Utilization Law ($U=E[B]*\lambda$) verwendet, wobei E[N] die mittlere Population, λ den Durchsatz, E[V] die mittlere Verweildauer pro Auftrag und U die Auslastung bezeichnet.

4.2.2. IO-Benchmarking

Das in Kapitel 5 vorzustellende IO-Modell wird anhand von realen Messdaten validiert, die mittels eines IO-Benchmarks generiert werden. Der im Rahmen dieser Arbeit verwendete sowie an der Universität Essen entwickelte IO-Benchmark DBench wird nachfolgend beschrieben, wobei zuvor ein kurzer Überblick über das IO-Benchmarking und bereits existierende Benchmarks gegeben wird.

4.2.2.1. IO-Benchmarks - Ein Überblick

Das Ziel von IO-Benchmarks ist die Ermittlung von Leistungsmaßen in Form von beispielsweise Durchsätzen, Antwortzeiten und Auslastungen für eine vorgegebene Last. Mit Hilfe eines IO-Benchmarks wird somit die Leistungsfähigkeit eines IO-Systems analysiert, die dann gegebenenfalls für den Vergleich mit weiteren IO-Systemen verwendet werden kann. Nach Chen und Patterson [17] können IO-Benchmarks in sogenannte Applikations- und Synthetik-Benchmarks unterschieden werden. Applikations-Benchmarks repräsentieren hierbei die durch Standardanwendungen (z. B. Datenbanken und Compiler) erzeugten Lasten eines speziellen Anwendungstyps, wie z. B. die TPC-Benchmarks OLTP-Lasten (On-Line Transaction Processing) abbilden. Hingegen werden bei Synthetik-Benchmarks synthetische Lasten definiert und vom Benchmark direkt über Lese- und Schreibanweisungen erzeugt. Der Vorteil von Synthetik-Benchmarks liegt somit in der detaillierten Analyse bestimmter IO-Lasten, die jedoch im Gegensatz zu den Applikations-Benchmarks keine realen Lasten widerspiegeln.

Ein Vergleich verschiedener IO-Benchmarks befindet sich z. B. in [17], [142], [44] und [64]. Es folgt eine Beschreibung ausgewählter IO-Benchmarks:

- *Bonnie:* Misst die IO-Performance basierend auf einer Datei für verschiedene einfache Workload-Szenarien, wie z. B. sequentielles Lesen und Schreiben pro Zeichen, sequentielles Lesen und Schreiben pro Block bzw. Lesen und Schreiben im Wechsel, zufällig verteilte Zugriffe mit drei parallelen IOs. Die Größe der Datei kann beliebig gesetzt werden. Für die Implementierung von Bonnie wurde die Programmiersprache C

verwendet, so dass eine Einschränkung in der Adressierung von Partitionen mit einer Größe von über 2 GB existiert.

- *Bonnie++:* Hierbei handelt es sich um eine Weiterentwicklung von Bonnie in der Programmiersprache C++. Einer der wesentlichen Vorteile von Bonnie++ ist die erweiterte Addressierungsmöglichkeit, so dass auf Partitionen mit einer Größe von über 2 GB zugegriffen werden kann.
- *IOStone:* Synthetischer IO-Benchmark basierend auf System-Traces von Unix Minicomputern, Workstations und IBM Mainframes [80]. Mit Hilfe von 400 Dateien werden Lese- und Schreibzugriffe in Anlehnung an die IO-Lastbeschreibungen in [79] generiert.
- *IOTest:* Ein UNIX-basiertes Softwarepaket bestehend aus verschiedenen Werkzeugen zur Performance-Messung und Zuverlässigkeitsüberprüfung [108]. Die Performance-Messungen werden auf Ebene des Raw Device ausgeführt, wobei ausschließlich Messungen mit zufällig verteilten IO-Zugriffen angeboten werden, die wie folgt parametrisiert werden können: Angabe des prozentualen Anteils an Lese- bzw. Schreibzugriffen, IO-Größe und Anzahl der parallelen IO-Zugriffe.
- *IOZone:* Ein Benchmark zur Messung auf Dateisystemebene. IOZone unterstützt eine Vielzahl an Dateioperationen, wie z. B. Lesen, Schreiben, Wiederholtes Lesen und Schreiben, fread und fwrite.
- *H2Bench:* Festplatten-Benchmark der Fachzeitschrift „Magazin für Computertechnik“ (c’t). H2Bench besteht aus den vier Komponenten: Messung der Interface-Transferrate, Zonenmessung, HDBench-Messung und Messung der mittleren Zugriffszeit [7].
- *SQLBench:* Der SQLBench ist ein auf einem Datenbanksystem aufsetzendes Benchmark-Programm, das datenbankspezifische Lasten generiert [82].

Im Rahmen dieser Arbeit soll allein die Performance des IO-Systems analysiert werden, so dass die IO-Messungen auf Raw Device-Ebene durchgeführt werden müssen, d. h. Einflüsse, wie z. B. verursacht durch das Dateisystem, werden nicht betrachtet. Es wird somit ein Synthetik-Benchmark benötigt, der Messungen auf Raw Device-Ebene durchführen kann und eine für verschiedene IO-Lastszenarien ausreichende Parametrisierung erlaubt. Viele der beschriebenen Benchmarks arbeiten auf Dateisystemebene und unterstützen zum Teil nur einfache Lastszenarien, welche die in dieser Arbeit zu betrachtenden IO-Lasten nicht abbilden können. Der Benchmark IOTest misst auf Raw Device-Ebene, unterstützt jedoch keine sequentiellen Lasten. Eine weiterführende kritische Sicht zu bestehenden Benchmarks wird in [17] gegeben. Es wurde somit für diese Arbeit ein an der Universität Essen entwickelter IO-Benchmark mit der Bezeichnung DBench [132] verwendet, der die beschriebenen Anforderungen erfüllt.

4.2.2.2. Benchmark DBench

DBench ist ein an der Universität Essen entwickelter Benchmark zur Performance-Messung von IO-Systemen [132]. Das Programm ist in der Programmiersprache C geschrieben und generiert mittels der Standardbefehle Read, Write und Seek IO-Aufträge. Der Zugriff kann dabei auf Ebene des Dateisystems oder des Raw Device erfolgen. DBench bietet die folgenden Funktionen zur IO-Lastbeschreibung:

- *Synchron/Asynchron:* Die IO-Verarbeitung kann synchron oder asynchron erfolgen, d. h. im synchronen Fall können mehrere IO-Prozesse parallel generiert werden, die jeweils zu jedem Zeitpunkt immer nur einen IO-Zugriff bearbeiten. Der asynchrone Fall bedeutet hingegen, dass ein IO-Prozess eine vorgegebene Anzahl an parallelen IO-Zugriffen generiert.

- *IO-Size:* Für alle zu generierenden IO-Zugriffe wird mit der IO-Size eine feste IO-Größe vorgegeben.
- *Burst-Probability:* Als Burst wird beim DBench eine Gruppe von sequentiellen IO-Zugriffen mit einer mittleren Anzahl an IOs (Burst-Length) definiert. Mit der Burst-Probability kann nun angegeben werden, mit welcher Wahrscheinlichkeit beim nächsten IO-Zugriff ein Burst auftritt.
- *Read-Write-Probability:* Hiermit wird die Wahrscheinlichkeit für die Zugriffsart des zu generierenden IO-Zugriffs definiert, ob Daten gelesen oder geschrieben werden. Die Bursts stellen hierbei eine Ausnahme dar, denn innerhalb eines Bursts kann die Zugriffsart nicht gewechselt werden.
- *IO-Count:* Der IO-Count definiert die Gesamtanzahl der für einen Benchmark zu generierenden IO-Zugriffe.
- *Arrival-Delay:* Hiermit wird der Abstand zwischen zwei IO-Zugriffen eines IO-Prozesses definiert. Der zeitliche Abstand kann hierbei deterministisch oder negativ exponentiell verteilt sein.
- *Process-Delay:* Bei synchroner IO-Verarbeitung wird mit dem Process-Delay die Verzögerungszeit zwischen zwei IO-Prozessen definiert. Die Zeit kann hierbei deterministisch oder negativ exponentiell verteilt sein.

Die IO-Last wird unterschieden in sequentielle bzw. zufällig verteilte (Burst-Probability) sowie lesende bzw. schreibende IO-Zugriffe (Read-Write-Probability). Beide Parameter werden als prozentuale Wahrscheinlichkeiten angegeben, so dass bei jedem IO-Zugriff mit der jeweiligen Wahrscheinlichkeit neu entschieden wird, ob es sich hierbei um einen lesenden oder schreibenden Aufruf handelt und ob ein Burst mit vorgegebener Länge begonnen oder zu einem zufällig ausgewählten Sektor gesprungen wird. Als Einschränkung ist zu beachten, dass innerhalb eines Bursts keine Wechsel zwischen Lesen und Schreiben erlaubt sind. Die Auswahl eines Sektors bei zufällig verteilten IO-Zugriffen erfolgt durch Ziehung einer gleichverteilten Zufallszahl, die im Intervall von 0 bis maximale Sektornummer der ausgewählten Partition liegen muss. Wird bei der Lastspezifikation eine 100 % sequentielle Last gefordert, so beginnt DBench immer beim ersten Sektor der Partition.

4.3. IO-Lastszenarien und Messergebnisse

In den nun folgenden Abschnitten werden die zur Validation des in Kapitel 5 zu beschreibenden Festplattenmodells verwendeten IO-Lastszenarien und die dabei erzielten Messergebnisse beschrieben.

4.3.1. IO-Lastszenarien

Zur Validiation des Festplattenmodells sollen Ergebnisse realer Messungen verwendet werden. Notwendig hierfür ist die Definition verschiedener IO-Lastszenarien. Es werden zunächst die folgenden fünf Lastcharakteristiken definiert, die eine für das IO-Modell ausreichende Beschreibung von IO-Lasten bieten:

- *Synchrone oder asynchrone IO-Verarbeitung:* Synchron bedeutet, dass die IO-Zugriffe nacheinander generiert und asynchron, dass die IO-Zugriffe parallel vom IO-Prozess versendet werden können. Diese Begriffe beziehen sich auf die einzelnen IO-Zugriffe und

besitzen somit eine andere Bedeutung als die bereits beschriebene synchrone bzw. asynchrone Verarbeitung der IO-Prozesse für den IO-Benchmark DBench.

- *Parallelitätsgrad der IO-Last:* Der Parallelitätsgrad wird durch die Anzahl der parallelen IO-Zugriffe definiert. Im Fall von synchroner IO-Verarbeitung kann Parallelität nur durch parallel laufende IO-Prozesse erreicht werden.
- *Mittlerer Ankunftsabstand für IO-Zugriffe und IO-Prozesse:* Der Ankunftsabstand kann z. B. deterministisch oder negativ exponentiell verteilt sein.
- *Lesende oder schreibende IO-Zugriffe*
- *Zufällig verteilte (Random) oder sequentielle (Sequential) IO-Zugriffe:* Bei Random IOs wird von gleichverteilten Zugriffen über das gesamte Medium der zu analysierenden Partition ausgegangen. Sequentielle IO-Zugriffe sind aufeinanderfolgende IOs auf Basis der Nummerierung der logischen Blöcke.

Es folgt nun die Definition von vier IO-Lastszenarien, die im weiteren Verlauf der Arbeit zur Validation des in Kapitel 5 zu beschreibenden Festplattenmodells verwendet werden. Die Definition der vier Lastzenarien basiert auf den bereits beschriebenen fünf Lastcharakteristiken:

- *Simple Load:* Einfache Lastmuster mit 100 % lesenden oder schreibenden IO-Zugriffen, wobei die Zugriffe selbst 100 % zufällig verteilt oder sequentiell sind. Die IO-Zugriffe werden mit einem IO-Prozess synchron verarbeitet, d. h. keine parallele IO-Verarbeitung. Variiert wird bei Simple Load für jede Messung die IO-Größe.
- *Sequential/Random Load-Mix:* Lastmuster mit 100 % lesenden oder schreibenden IO-Zugriffen. Es findet keine parallele IO-Verarbeitung statt, d. h. synchron. Die IO-Größe ist auf 8 KB festgelegt. Die Last entspricht einem Mix aus sequentiellen und zufällig verteilten IO-Zugriffen.
- *Read/Write Load-Mix:* Lastmuster mit 100 % zufällig verteilten oder sequentiellen IO-Zugriffen. Es findet keine parallele IO-Verarbeitung statt, d. h. synchron. Die IO-Größe ist auf 8 KB festgelegt. Die Last entspricht einem Mix aus lesenden und schreibenden IO-Zugriffen.
- *Multi IO Load:* Lastmuster mit 100 % zufällig verteilten IO-Zugriffen, wobei die Art des Zugriffs lesend oder schreibend ist. Die IO-Größe ist auf 8 KB festgelegt. Die IO-Zugriffe werden asynchron verarbeitet, wobei in diesem Lastszenario die Maximalanzahl an parallelen IO-Zugriffen für jede Messung variiert wird.

Bei den beschriebenen IO-Lastszenarien wurde jeweils mit Ausnahme der Simple Load die IO-Größe auf 8 KB festgelegt, da diese häufig als Standardgröße für IO-Lasten verwendet wird ([72], [83]). Weiterhin wurde in allen vier Szenarien kein mittlerer Ankunftsabstand für die IO-Zugriffe (also 0 ms) vorgegeben, d. h. die IO-Zugriffe werden direkt ohne Verzögerung vom Benchmark erstellt. Eine Analyse von IO-Lasten mit verschiedenen Ankunftsabständen der IO-Zugriffe wird in Kapitel 5 Abschnitt 5.4. beschrieben.

Die IO-Lastszenarien wurden derart definiert, dass sie jeweils mit Ausnahme der Ankunftsabstände zwischen den IO-Zugriffen eine bestimmte IO-Last-Charakteristik abbilden. Man könnte diese Lastszenarien somit auch als Basisszenarien bezeichnen, da real gemessene IO-Lasten eine Komposition der vier Basisszenarien darstellen. Es folgt eine Beschreibung der Messergebnisse für die vier Lastszenarien. In Kapitel 5 wird hingegen auch die Mixtur der vier verschiedenen Basisszenarien analysiert.

4.3.2. Ergebnisse der Messungen

Für jedes der in Abschnitt 4.3.1. beschriebenen IO-Lastszenarien wurden Benchmark-Messungen mit Hilfe des IO-Benchmarks DBench durchgeführt, die mittels des Skripts SIOSTAT vom SE Performance Toolkit überwacht wurden. Bei der Analyse der Messergebnisse wurde die Bedienzeit pro IO-Zugriff analysiert und zwar in Abhängigkeit der jeweiligen Lastsituation. Die hierbei erzielten Ergebnisse werden in den folgenden Abschnitten beschrieben. Es folgt zunächst eine Beschreibung des für die Benchmarks verwendeten Testsystems.

4.3.2.1. Konfiguration des Testsystems für IO-Benchmarks

Das Testsystem bestand aus einem Pentium III (500 MHz) mit 256 MB Hauptspeicher und einem SCSI-Host Bus Adapter (HBA) Adaptec SCSI-Card 29160 sowie zwei daran angeschlossen SCSI-Festplatten IBM DNES-309170W und IBM DDYS-T36950N. Als Betriebssystem wurde Solaris 2.8 für Intel verwendet. Der HBA unterstützte den Anschluss von LVD SCSI-Geräten (insbesondere Ultra-160) sowie Ultra-SCSI und Fast/Wide-SCSI-Geräten. Für beide Gerätetypen bietet der HBA einen separaten Anschluss, so dass beide Gerätegruppen unabhängig voneinander arbeiten konnten. Die Festplatte IBM DNES-309170W (Wide-SCSI) wurde als Systemplatte an den 68-Pin Single Ended-Connector angeschlossen. Als Benchmark-Festplatte wurde die DDYS-T36950N an dem 68-Pin Low Voltage Differential-Connector verwendet. Die wichtigsten Herstellerangaben der Benchmark-Festplatte werden in Tabelle 4.8 beschrieben. Weiterhin unterstützt die Festplatte Funktionen wie z. B. Read-Ahead und Write-Behind-Caching-Strategien sowie Tagged Command Queueing. Der in Tabelle 4.8 beschriebene Command Overhead definiert sich als Zeitverbrauch angefangen vom letzten Byte der Command Phase bis hin zum ersten Byte der Data Phase ohne Berücksichtigung von Seek Time, Latency Delay und Initiator Delay (mit reconnections).

Plattenparameter: IBM DDYS-T36950N					
Konfiguration			**Mechanical Seek Time**		
Schnittstelle		Ultra-160 SCSI	*Avg. Seek Time [ms] (inkl. Settling)*		
Kapazität	GB	36,7	Read	Typical: 4,9	Max: 5,9
Sektorgröße	Byte	512	Write	Typical: 5,9	Max: 6,9
Logische Datenblöcke	Anzahl	71.703.918.080	*Full Stroke Seek [ms]*		
Recording Zones	Anzahl	11	Read	Typical: 10,5	Max: 11,5
Datenköpfe	Anzahl	12	Write	Typical: 11,5	Max: 12,5
Platten	Anzahl	6	*Cylinder Switch Time (cylinder skew)*		
Performance			Cylinder Skew	ms	1,6
Cache	MB	4[1]	*Head Switch Time (head skew)*		
Rotationsgeschwindigkeit	RPM	10.000	Cylinder Skew	ms	1,6 -> 1,1
mittlere Latenzzeit	ms	2,99	**Data Transfer Speed**		
Medium-Transferzeit	MB/s	35-56,5	*Disk Buffer Transfer (Zone 0)*		
Schnittstellen-Transferzeit	MB/s	160	ad hoc	MB/s	43,0
stetige Datentransferrate	MB/s	21,7-36,1	konstant	MB/s	36,1
mittlere Seek-Time	ms	4,9	*Disk Buffer transfer (Zone 11)*		
Track-to-Track Seek Time	ms	0,5	ad hoc	MB/s	26,0
Full Track Seek Time	ms	10,5	konstant	MB/s	22,1

Tabelle 4.8 Spezifikationen und Performance-Daten der Festplatte IBM DDYS-T36950N

Plattenparameter: IBM DDYS-T36950N		
Command Overhead		
Cache not hit	µsec	400
Cache hit	µsec	30

Tabelle 4.8 Spezifikationen und Performance-Daten der Festplatte IBM DDYS-T36950N

1. 512 KB belegt durch Firmware

Weitere Festplattenparameter, die mittels SCSI-Mode-Pages ausgelesen werden können, werden in Kapitel 5 Abschnitt 5.2.1. beschrieben. Die Festplatte wurde bei den nachfolgend beschriebenen Messungen mit den Standardeinstellungen des Herstellers betrieben, d. h. mit aktiviertem Read- und Write-Cache (Read Ahead, Back-to-Back Write (Write-Back-Caching)), sowie Tagged Command Queueing.

4.3.2.2. Simple Load

Die Ergebnisse für Simple Load werden unterteilt in lesende und schreibende Zugriffe (siehe Abbildung 4.7 und Abbildung 4.8). Für beide Fälle werden Messergebnisse, unterschieden nach den Zugriffsarten zufällig verteilt (Random) und sequentiell (Sequential), dargestellt. Für den Fall Sequential wird die Gesamtzahl der IO-Zugriffe derart gewählt, dass auf alle Blöcke der Festplatte ein Zugriff erfolgt. Somit werden alle Zonen (Recording Zones, in diesem Fall 11 Zonen) der Festplatte berücksichtigt. Die Abbildungen zeigen die Abhängigkeit der Bedienzeit pro IO-Zugriff in ms (Y-Achse) von der IO-Größe in KB (X-Achse).

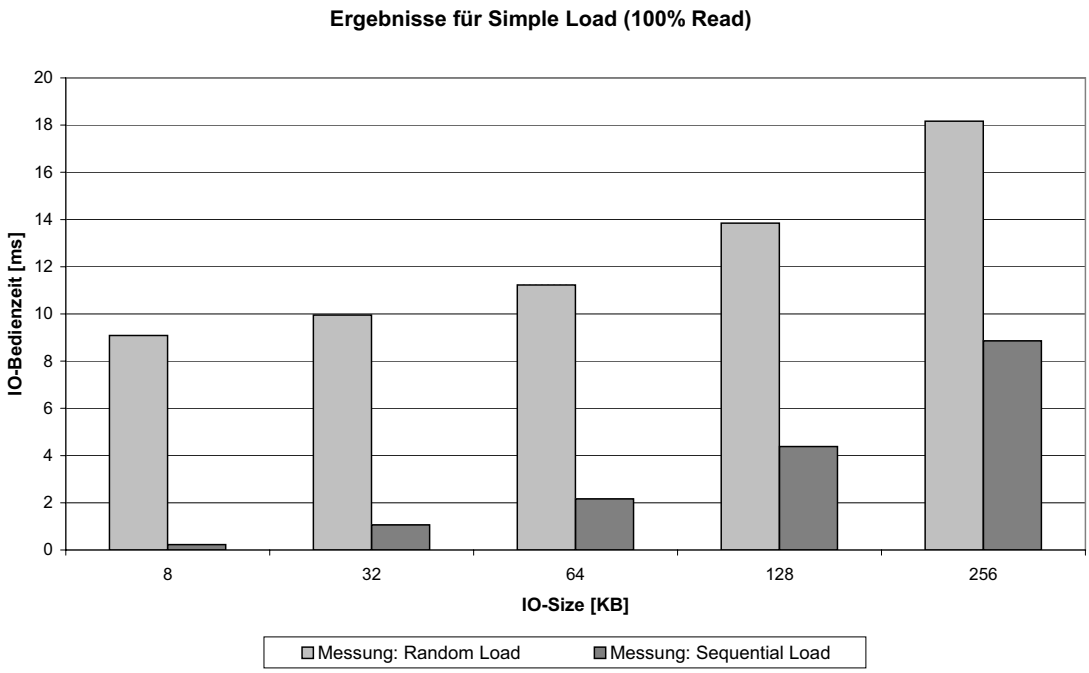


Abbildung 4.7 Messergebnisse für Simple Load mit 100 % lesenden IO-Zugriffen

Die in Abbildung 4.7 dargestellten Ergebnisse für 100 % Lesezugriffe lassen unter Berücksichtigung der Skalierung der X-Achse einen nahezu linearen Zusammenhang zwischen den verschiedenen IO-Größen erkennen. Vergleicht man die Zugriffsarten Random und Sequential miteinander, so erkennt man, dass sich diese um einen nahezu konstanten Betrag von 9 ms unter-

scheiden. Dieser Wert beinhaltet die Zugriffszeiten Seek- und Latency-Time der Festplatte, die für das Lastszenario Random auftreten. Da die Zugriffe über das Medium gleichverteilt werden, existieren für jeden Zugriff vergleichbare Positionierungszeiten.

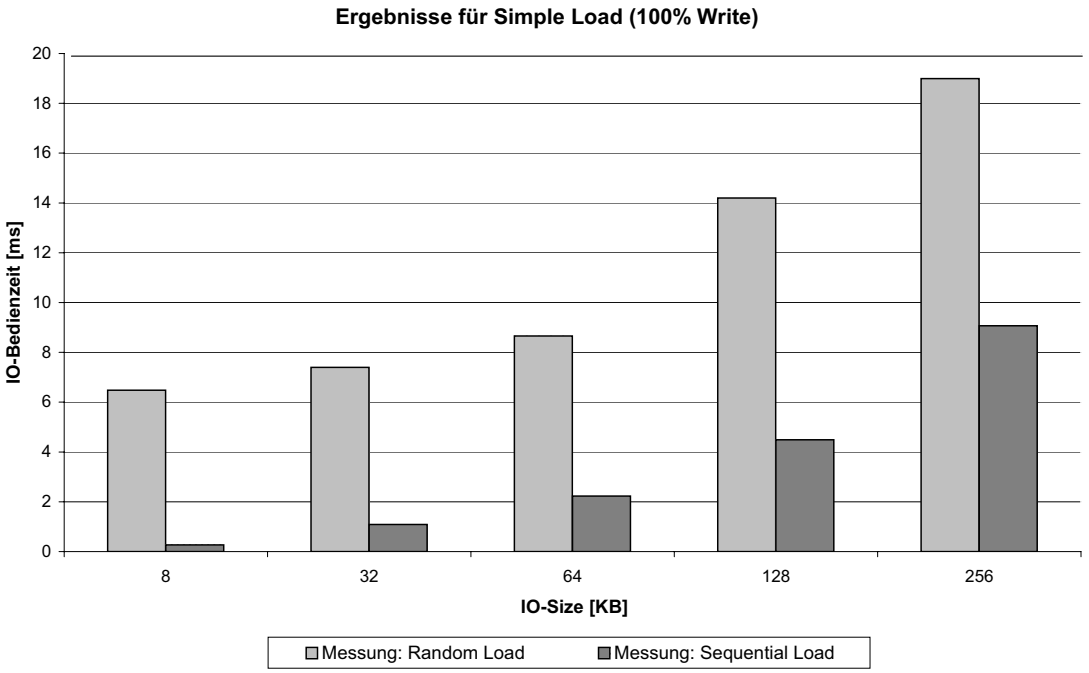


Abbildung 4.8 Messergebnisse für Simple Load mit 100% schreibenden IO-Zugriffen

In Abbildung 4.8 werden die Messergebnisse für Schreibzugriffe dargestellt. Für den sequentiellen Zugriff sind die Bedienzeiten pro IO mit denen für lesende Zugriffe nahezu identisch. Im Fall Random unterscheiden sie sich hingegen für fast alle IO-Größen. Die Ursache liegt hierbei in der Verarbeitungsstrategie der Festplatte bei aktiviertem Write-Cache. Die zu schreibenden Daten werden hierbei in den Cache der Festplatte transferiert und der zugehörige IO-Prozess erhält danach den Status, dass der IO-Zugriff verarbeitet wurde und wird somit frei gegeben, so dass der nächste IO-Auftrag verarbeitet werden kann. Das Schreiben der Daten auf das physikalische Medium der Festplatte geschieht dann in Abhängigkeit der im Festplatten-Controller implementierten Caching-Strategie im Hintergrund. Es sammeln sich somit eine Vielzahl an Schreibzugriffen in der Queue der Festplatte, die dann mittels des aktivierten Scheduling-Algorithmus der Festplatte optimiert werden können. Erreichen die IO-Zugriffe hingegen eine Größe von 128 KB ($PreFetch_{Max}$, siehe Kapitel 5 Abschnitt 5.2.1.), so werden von der Platte keine Optimierungsverfahren mehr eingesetzt. Somit ist auch der überdurchschnittliche Anstieg der Bedienzeit bei 128 KB im Fall Random zu erklären. Die Bedienzeiten gleichen sich dann ab 128 KB den Bedienzeiten der Lesezugriffe an.

4.3.2.3. Sequential/Random Load-Mix

Die Messergebnisse für das Lastszenario Sequential/Random Load-Mix werden in Abbildung 4.9 dargestellt. In diesem Szenario wird zwischen 100 % Read und 100 % Write unterschieden. Die Abbildung stellt die Abhängigkeit der Bedienzeit pro IO-Zugriff in ms (Y-Achse) von der

Burst-Probability in % (X-Achse) dar, d. h. die Wahrscheinlichkeit, dass ein Burst (Segment mit einer konstanten Anzahl von sequentiell aufeinander folgenden IO-Zugriffen) erfolgt.

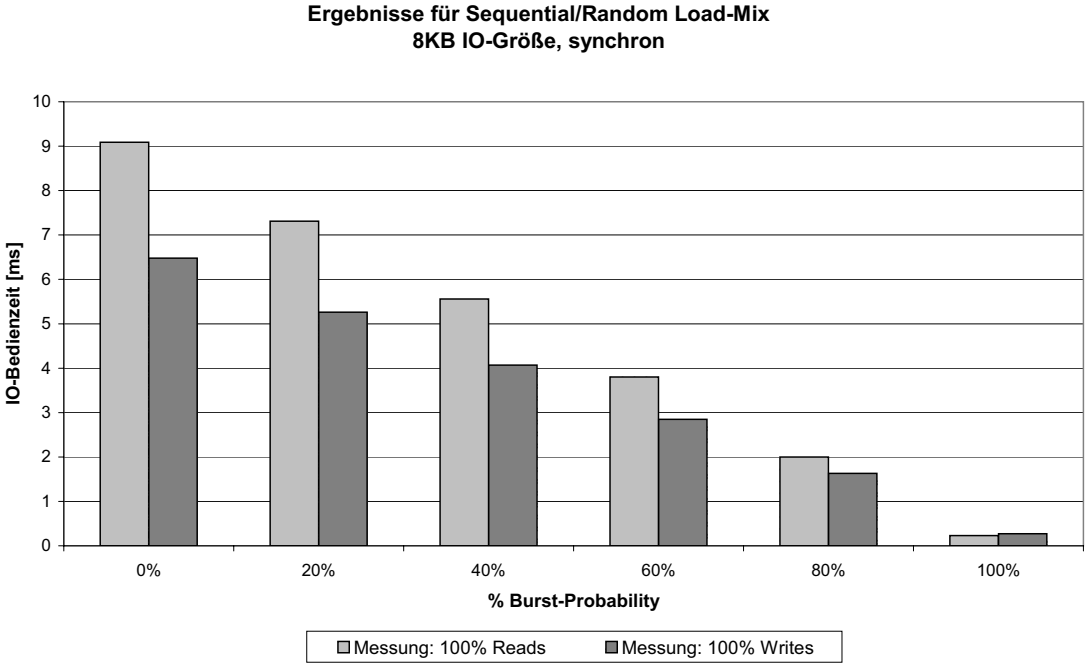


Abbildung 4.9 Messergebnisse für Sequential/Random Load-Mix

Anhand der Ergebnisse ist ein linearer Zusammenhang zwischen der Bedienzeit pro IO-Zugriff und der Burst-Probability zu erkennen. Signifikant ist der Unterschied zwischen lesenden und schreibenden IO-Zugriffen, der durch die Verarbeitungsstrategie der Schreibzugriffe bei aktiviertem Schreib-Cache zu erklären ist (siehe auch Simple Load). Mit zunehmendem Anteil an sequentiellen IO-Zugriffen gleichen sich die Bedienzeiten für lesende und schreibende IO-Zugriffe an, da im sequentiellen Fall insbesondere die Caching-Strategie der Schreibzugriffe keinen größeren Performance-Gewinn bewirkt, da die Daten direkt geschrieben bzw. gelesen werden können.

4.3.2.4. Read/Write Load-Mix

In Abbildung 4.10 werden die Messergebnisse für das Lastszenario Read/Write Load-Mix dargestellt. In diesem Szenario wird die Read-Probability variiert, d. h. die prozentuale Wahrscheinlichkeit für Lesezugriffe (0 % Read-Probability entspricht somit 100 % Schreibzugriffe). Unterschieden wird hierbei zwischen den beiden Zugriffsarten 100 % Sequential und 100 % Random. Die Abbildung beschreibt die Abhängigkeit der Bedienzeit pro IO-Zugriff (Y-Achse) zur Read-Probability (X-Achse) in %.

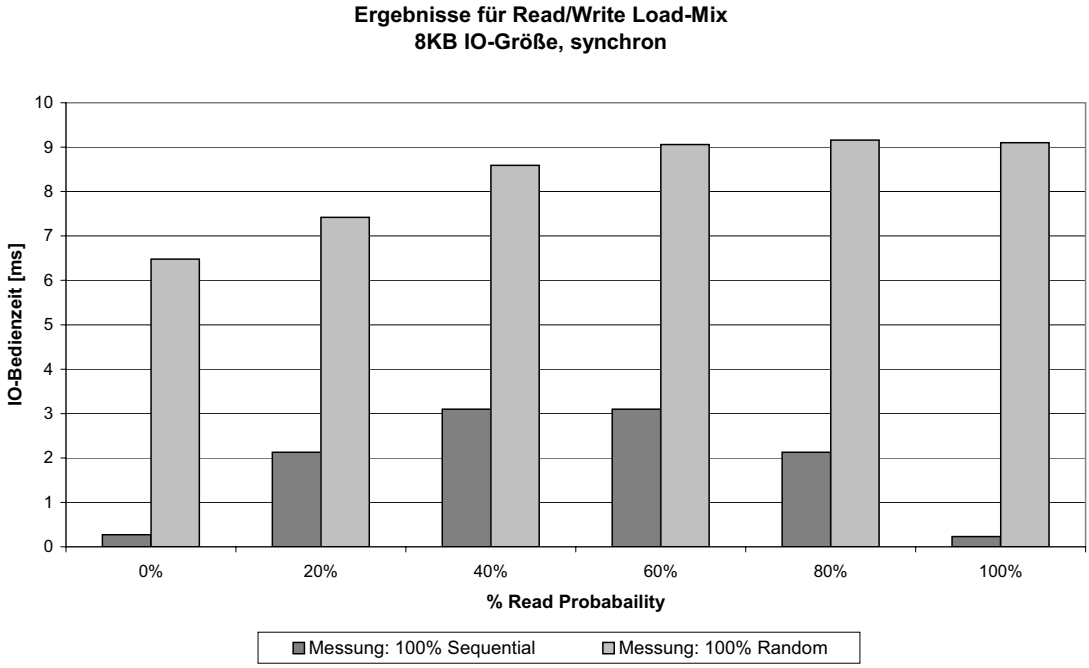


Abbildung 4.10 Messergebnisse für Read/Write Load-Mix

Im Fall 100 % Random ist mit zunehmendem Anteil an Lesezugriffen ein Anstieg der Bedienzeit zu erkennen, da die Festplatte unterschiedliche Verarbeitungsstrategien für Lese- und Schreibzugriffe verwendet (siehe auch Simple Load). Ab 50 % Read-Probability ist eine nahezu konstante Bedienzeit von ca. 9 ms zu verzeichnen. Dies kann damit erkärt werden, dass ab 50 % nahezu jeder zweite IO-Zugriff einem Lesezugriff entspricht und somit eine Optimierung der Schreibzugriffe nicht mehr möglich ist.

Für den Fall 100 % Sequential ist ein gleichmäßiger Overhead durch den Wechsel von Lese- auf Schreibzugriff zu erkennen. Sowohl für einen erhöhten Anteil an schreibenden als auch an lesenden Zugriffen ist ein identischer Overhead festzustellen.

4.3.2.5. Multi IO Load

In dem vierten und letzten Szenario wird bei einer IO-Größe von 8 KB und 100 % zufällig verteiltem Zugriff eine asynchrone IO-Verarbeitung durchgeführt. Hierbei wird die Zahl der maximal parallel zu verarbeitenden IO-Zugriffe variiert. Unterschieden wird in diesem Lastszenario zwischen lesenden und schreibenden IO-Zugriffen. Die erzielten Ergebnisse werden in den Abbildung 4.11, Abbildung 4.12 und Abbildung 4.13 dargestellt.

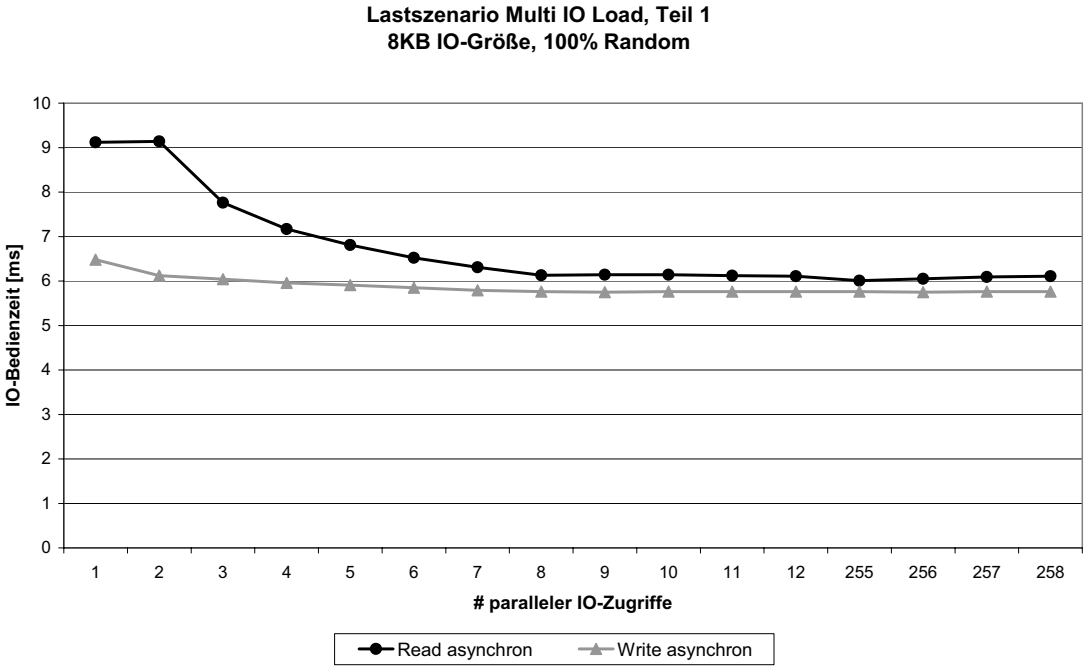


Abbildung 4.11 Messergebnisse für Multi IO Load, Teil 1

Betrachtet man die Ergebnisse in Abbildung 4.11, so erkennt man, dass eine Optimierung der IO-Bedienzeit für lesende und schreibende IO-Zugriffe vom Festplatten-Controller nur bis zu acht parallelen IO-Zugriffen vorgenommen wird. Ab acht parallelen IO-Zugriffen stagniert die Bedienzeit. Die unterschiedlichen Verläufe der Bedienzeitkurven für Lese- und Schreibzugriffe sind durch die verschiedenen Caching-Strategien zu erklären, die bereits im Abschnitt Simple Load beschrieben wurden. Die Lesezugriffe können erst ab drei parallellen IO-Zugriffen optimiert werden, da der erste IO-Zugriff direkt verarbeitet wird und dann mindestens zwei weitere IO-Zugriffe benötigt werden, die optimiert bzw. sortiert werden können. Bei den Schreibzugriffen ist zu berücksichtigen, dass durch die Back-To-Back-Caching-Strategie sich bereits eine Vielzahl an IO-Zugriffen im Cache befanden und somit durch die Erhöhung der parallelen IO-Zugriffe keine mit den Lesezugriffen vergleichbare Optimierung zu erwarten war und somit kein vergleichbarer Kurvenverlauf.

Weiterhin ist zu bemerken, dass sich auch ab acht parallelen IO-Zugriffen die Bedienzeit der Schreibzugriffe unterhalb der Verarbeitungszeit der Lesezugriffe befindet. Es ist anzunehmen, dass dieses Phänomen ebenso in den verschiedenen Caching-Strategien begründet liegt.

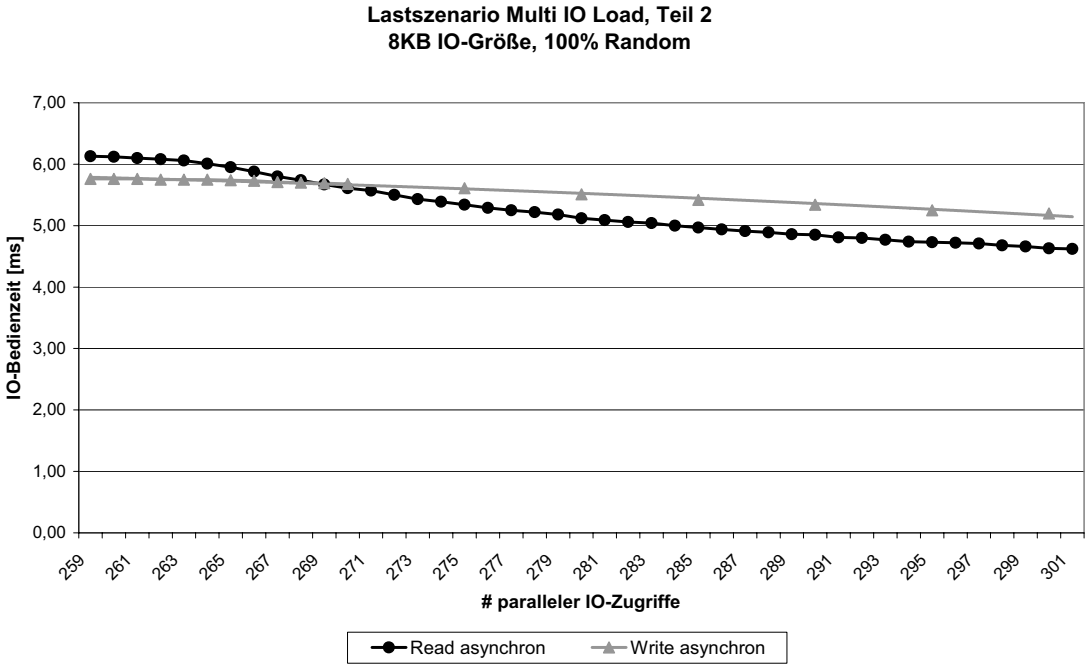


Abbildung 4.12 Messergebnisse für Multi IO Load, Teil 2

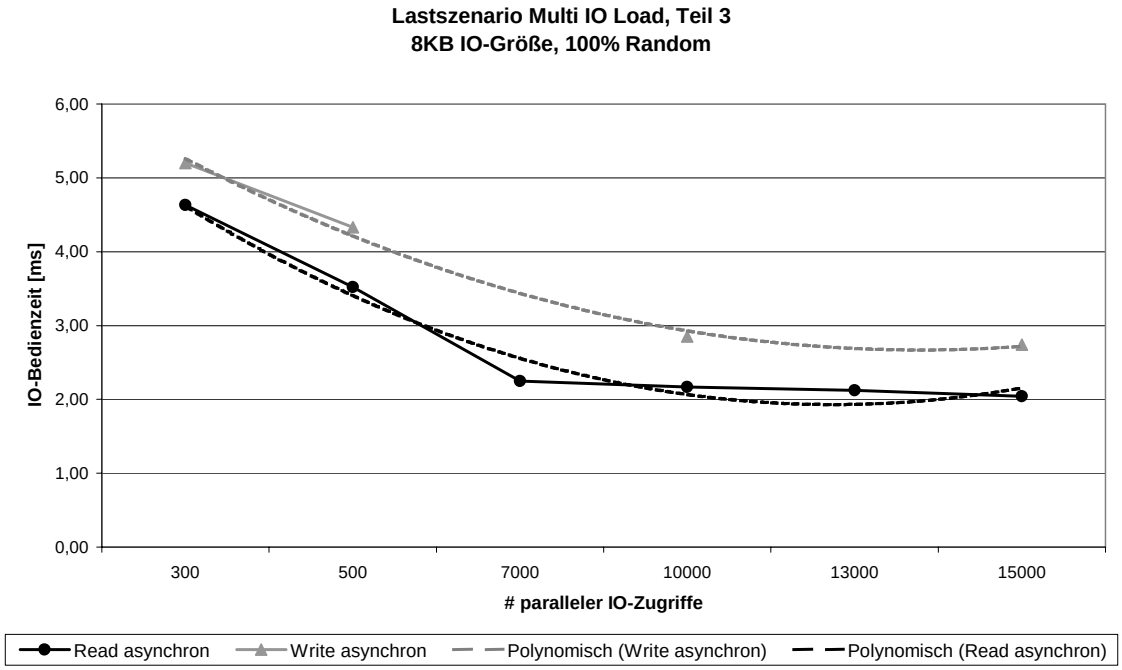


Abbildung 4.13 Messergebnisse für Multi IO Load, Teil 3

Der bei Erhöhung der parallelen IO-Zugriffe weitere Verlauf der Bedienzeit ist in den Abbildungen 4.11 und 4.12 dargestellt. Es ist zu beachten, dass ab 264 parallelen IO-Zugriffen die Bedienzeit sowohl für Lese- als auch für Schreibzugriffe weiter sinkt. Dies kann damit erklärt werden, dass die Festplatte (Active Queue) maximal 256 IO-Zugriffe puffern kann (Tagged Command Queueing). Alle weiteren IO-Zugriffe werden im Cache des Device Drivers von Solaris (Wait Queue) gepuffert. Der Device Driver sortiert in der Standardeinstellung von Solaris die IO-Zugriffe nach dem Elevator-Prinzip bzgl. der Positionierungszeit. Die Festplatte optimiert bis zu maximal acht parallele IO-Zugriffe. Ab 264 parallelen IO-Zugriffen ist anhand der IO-Bedienzeitkurve in Abbildung 4.12 zu erkennen, dass die Optimierung der IO-Aufträge weitergeführt wird. Mit 264 IO-Zugriffen liegen 256 IOs in der Active Queue und 8 IO-Zugriffe in der Wait Queue. Mit 265 IOs wird somit eine Sequenz von 9 sortierten IOs in die Active Queue gesendet, so dass eine verbesserte IO-Zugriffszeit erreicht wird. Es ist somit anzunehmen, dass der Festplatten-Controller maximal acht IO-Aufträge sortieren kann und bereits vom Device Driver sortierte IO-Aufträge in die Active Queue gesendet bekommt, so dass die Festplatte diese ohne weitere Optimierung direkt verarbeiten kann.

Das Festplattenverhalten ist somit teilweise erklärbar, jedoch sind z. B. die von der Festplatte verwendeten Sortieralgorithmen nicht bekannt. Auch ist nicht verständlich, warum die Festplatte nur bis zum achten parallelen IO-Zugriff die Bedienzeit optimiert. Um die beschriebenen Performance-Charakteristiken der Festplatte im Modell zu berücksichtigen, müssten detailliertere Festplattenbeschreibungen verfügbar sein. Derartige Beschreibungen sind jedoch nicht frei erhältlich. Zur Berücksichtigung des beschriebenen Festplattenverhaltens im zu erarbeitenden IO-Modell, wird in Kapitel 5 eine IO-Bibliothek eingeführt, die das im Szenario Multi IO Load beschriebene Festplattenverhalten abbildet. Hierzu werden durch Skalierungsfaktoren die in Abhängigkeit der parallelen IO-Zugriffe resultierenden Bedienzeitveränderungen beschrieben. Die Skalierungsfaktoren werden somit wie folgt berechnet:

$$\text{Skalierung für x parallele IO-Zugriffe} = \frac{\text{Bedienzeit für x parallele IO-Zugriffe}}{\text{Bedienzeit für einen IO-Zugriff ohne Parallelität}}$$

Wird z. B. eine Bedienzeit von 9 ms ohne Parallelität gemessen und weiterhin eine Bedienzeit von 6 ms für 5 parallele IO-Zugriffe, so resultiert ein Skalierungsfaktor von 0,667, d. h. die Bediengeschwindigkeit verringert sich um 44 %. In Kapitel 5 werden die für das Festplattenmodell verwendeten Skalierungsfaktoren beschrieben.

KAPITEL

5

MODELLIERUNG VON IO-SYSTEMEN

In diesem Kapitel wird die Erstellung eines analytischen Festplattenmodells beschrieben, das Caching-Strategien für Lese- und Schreibzugriffe sowie Tagged Command Queueing unterstützt. Nach Möglichkeit sollen bereits existierende Algorithmen zur Festplattenmodellierung verwendet werden, wobei insbesondere auf die Arbeiten von Menascé [70] und Shriver [99] Bezug genommen wird. Das Modell wird anhand der in Kapitel 4 beschriebenen IO-Lastszenarien validiert. Der Detaillierungsgrad des Modells wird durch die Verwendung von Monitoring-Werkzeugen wie in diesem Fall SIOSTAT des SE Performance Toolkit sowie frei erhältlichen Hardware-Beschreibungen zur Festplattenparametrisierung bestimmt.

Es werden in diesem Kapitel nach einer kurzen Einführung in die bisherigen Arbeiten zur analytischen Modellierung von Festplatten zum einen das erarbeitete Festplattenmodell und zum anderen die damit erzielten Modellergebnisse dargestellt. Das Festplattenmodell unterteilt sich in die folgenden zwei Abschnitte: Berechnung der Bedienzeit pro IO-Zugriff sowie Einbindung der berechneten Bedienzeit in ein Warteschlangenmodell zur Ermittlung der Leistungsmaße Antwortzeit pro IO-Zugriff, Auslastung der Festplatte sowie Warteschlangenlänge für die mittels Tagged Command Queueing gepufferten IO-Zugriffe. Die Bedienzeitberechnung wird hierbei auf Basis der Last-, Festplatten- und Betriebssystem-Beschreibung ermittelt. Verglichen werden die berechneten Bedienzeiten dann mit den in Kapitel 4 beschriebenen Messergebnissen für die definierten Lastszenarien. Weiterhin werden neue Lastszenarien durch die Mixtur der bereits existierenden Lastdefinitionen gebildet und hierfür zum Teil notwendige Modellerweiterungen beschrieben.

Die Antwortzeit pro IO-Zugriff und die Auslastung der Festplatte werden in einem weiteren Schritt mit Hilfe der berechneten Bedienzeit unter Verwendung eines M/G/1-Modells gelöst. Abschließend wird ein Vorgehen zur Integration des erarbeiteten Festplattenmodells in das Modellierungswerkzeug WLPSizer beschrieben.

5.1. Festplattenmodellierung - Stand der Technik

Die Modellierung von Festplatten wird in der Literatur sowohl unter Verwendung analytischer als auch simulativer Verfahren beschrieben. Das Einsatzgebiet der Festplatten-Simulation lag in der Vergangenheit häufig in der detaillierten Analyse von Scheduling-Strategien oder Caching-Algorithmen. Als Simulationswerkzeuge werden zum größten Teil sog. „trace-driven simulators“ verwendet, d. h. sie arbeiten auf Basis einer Liste von Events mit zugehörigen Zeitstempeln, die anhand von realen Messungen ermittelt wurden. Die in vielen Arbeiten verwendeten

Simulationswerkzeuge für Festplatten sind Pantheon [135] und DiskSim [34]. Eine der bekanntesten Publikationen zur Modellierung von Festplatten mittels Simulation stammt von Ruemmler und Wilkes [84]. In Anlehnung an das in [84] erarbeitete Simulationsmodell, wird in [59] ein Modell für mehrere Festplatten an einem oder mehreren SCSI-Bussen präsentiert. Eine an der Universität Essen erstellte Diplomarbeit [75] beschreibt die Simulation von Festplatten mit Hilfe des Simulationswerkzeugs JavaDemos [66] unter Verwendung der in Kapitel 4 beschriebenen Lastszenarien und des IO-Benchmarks DBench. Als Ergebnis der Arbeit resultierte ein Simulationsmodell, das sehr detailliert die Funktionsweise einer Festplatte abbildete und somit für bestimmte IO-Lastszenarien, verglichen mit der Messung, nahezu identische Bedienzeiten, Antwortzeiten und Durchsätze lieferte.

Wie bereits in der Motivation beschrieben, wird in dieser Arbeit die analytische Modellierung von Festplatten betrachtet. Schon in den 70'er Jahren wurden die ersten analytischen Festplattenmodelle entwickelt (siehe z. B. [134]). Zu dieser Zeit besaßen Festplatten keine On-Board-Controller oder Cache-Speicher, so dass aus heutiger Sicht relativ einfache Modelle ohne komplexe Bedienstrategien zur Abbildung einer Festplatte ausreichten. In den 80'er Jahren wurden Disk Arrays mit der sogenannten RAID-Architektur (Redundant Arrays of Inexpensive Disks) entwickelt, so dass der Schwerpunkt der Forschung sich auf die Modellierung von Disk Arrays konzentrierte und die Festplattenmodelle größtenteils nicht weiter entwickelt wurden. Die jüngste Arbeit im Gebiet der Festplattenmodellierung, die ein analytisches Modell für Festplatten mit On-Board-Controller und Cache und der damit verbundenen Unterstützung der verschiedenen Caching- und Queueing-Strategien beschreibt, stammt von Shriver ([99], [100]).

Im Bereich der analytischen Festplattenmodellierung beschäftigen sich eine Vielzahl von Arbeiten mit der Approximation der Seek-Time ([44], [38], [84]), der Berechnung der Cache-Hit-Ratio sowie der Analyse verschiedener Caching-Strategien ([15], [109], [99]) und der Ermittlung des Zeitverbrauchs in der Festplattenwarteschlange unter Verwendung verschiedener Queueing-Strategien ([133], [46], [116], [21]). Einen guten Überblick über diese Arbeiten erhält man in [100]. Unter Verwendung der in den verschiedenen Publikationen erzielten Ergebnisse und Algorithmen beschreibt Shriver in ihrer Dissertation ein analytisches Festplattenmodell, das Read-Ahead und Tagged Command Queueing unterstützt. Das Caching von Schreibzugriffen wird in ihrer Arbeit nicht betrachtet. Die Validation ihres Modells führt sie mit Hilfe von Simulationsergebnissen durch (Pantheon Disk Simulator). Wie bereits in der Motivation dieses Kapitels beschrieben, ist nach Angaben von Shriver eine direkte Validation ihres Modells auf Basis von realen Messdaten für beliebige Festplatten nicht möglich, da hierfür viele Festplatten nicht die notwendigen Messdaten liefern.

Neben Shriver beschreibt auch Menascé sowohl in [71] als auch in [70] ein analytisches Festplattenmodell, das die Bedienzeitberechnung pro IO-Zugriff beschreibt. Das von ihm in [70] dargestellte Modell und die damit verbundene IO-Lastcharakterisierung basieren hierzu zum Teil auf den Algorithmen von Shriver, wie z. B. die Berechnung der Rotationslatenzzeit für sequentielle Zugriffe. Wie auch bei Shriver, werden Schreibzugriffe im Modell von Menascé ebenfalls nicht berücksichtigt.

Neben der Analyse einzelner Festplatten werden in weiteren Arbeiten, wie z. B. von Lazowska [63] sowie Barve, Shriver und Gibbons [3], insbesondere die Zugriffspfade zu den Festplatten (IO-Busse), und der Einfluss mehrerer Festplatten auf einen IO-Bus untersucht.

Sowohl für die Simulation als auch für die analytische Modellierung ist die Güte der Modellergebnisse abhängig von den verwendeten Eingabeparametern. Für die Festplattenmodellierung kann zwischen Festplattenparametern (Eigenschaften der Festplatte, wie z. B. Seek-Time, Rotationslatenzzeit) und der IO-Lastbeschreibung unterschieden werden. Für die Festplattenparameter existieren die Herstellerangaben und SCSI-Mode-Pages. Weitergehend wird in [141] die Ermittlung von Festplattenparametern via IO-Messungen beschrieben. Als Voraussetzungen werden jedoch direkte „low-Level" SCSI-Zugriffsmöglichkeiten und hoch auflösende Timer im System benötigt sowie ein Device Driver Development Environment oder eine „offene" IO-

Card. Für die IO-Lastcharakterisierung existieren Konzepte von Ruemmler und Wilkes in [83], Shriver in [99] sowie Menascé in [70]. Angewendet wurden diese Konzepte bereits für die in Kapitel 4 beschriebenen IO-Lastszenarien, d. h. beispielsweise für die Definition der sequentiellen Lastanteile.

5.2. Festplattenmodell zur Berechnung der Bedienzeit

Es folgt eine Beschreibung des Festplattenmodells zur Berechnung der Bedienzeit pro IO-Zugriff. Diese wird aus verschiedenen Modell-Eingabeparametern, welche die IO-Last, die Festplatte und die Systemparameter des verwendeten Rechners beschreiben, ermittelt.

Unter Solaris existieren beim IO-Monitoring eine Wait Queue und eine Active Queue. In der Wait Queue befinden sich die Aufträge vom Device Driver und in der Active Queue die Aufträge der Festplatte. Für Solaris befindet sich ein IO-Auftrag in der Active Queue, sobald er den Rechner verlässt und sich somit auf dem (in diesem Fall) SCSI-Medium befindet. Das Skript SIOSTAT (SE Performance Toolkit) liefert Bedienzeiten, Antwortzeiten, Warteschlangenlängen und Auslastungen für beide Queues. Die für das in diesem Kapitel zu beschreibende Modell relevante Bedienzeit ist die der Active Queue, d. h. die aktive Zeit eines IO-Auftrags in der Festplatte und auf dem SCSI-Medium. Da Modell und Messung vergleichbar sein sollen, wird die Definition der Bedienzeit für das Modell der von SIOSTAT angeglichen. Die weiteren Parameter, Antwortzeiten und Auslastungen, der Queues von SIOSTAT werden nachfolgend in Abschnitt 5.10. für die Warteschlangenmodellierung verwendet.

5.2.1. Modell-Eingabeparameter

Die Eingabeparameter des Modells lassen sich in vier Gruppen unterteilen: Festplatte, Workload, SCSI-Mode-Pages und SCSI Device Driver (siehe Tabelle 5.1). Die Spalte Testsystem beschreibt die tatsächlich für die Festplatte und das System definierten Parameter, wobei die Workload-Parameter hingegen nur ein Beispiel für ein beliebiges Lastszenario darstellen sollen. Die Angaben über die betrachtete Festplatte befinden sich in Kapitel 4 (Abschnitt 4.3.2.1.) bzw. in [49]. Die Parameter zur Workload-Charakterisierung wurden in starker Anlehnung an den IO-Benchmark DBench definiert, der in Kapitel 4 beschrieben wurde. Jede Gruppe der in Tabelle 5.1 dargestellten Modell-Eingabeparameter wird in den folgenden Abschnitten weitergehend beschrieben.

Bezeichnung	Beschreibung	Einheit	Testsystem
Festplatten-Parameter			
SeekTi	Mittlere Seek-Time für Random Load	ms	4,9
DiskSpeed	Rotationsgeschwindigkeit	RPM	10.000
$TransRate_{Inst}$	Maximale Transferrate pro Track	MB/s	34,5
$TransRate_{Sust}$	Stetige Transferrate inkl. Overheads (z. B. Head Switches)	MB/s	29
$Ovhd_{Ctrl}$	IO-Verarbeitungszeit am Platten-Controller	ms	0,03 ... 0,4
BusSpeed	IO-Bus-Transferzeit	MB/s	110
CacheSize	Cache-Größe exklusive Firmware	Byte	3.682.304

Tabelle 5.1 Modell-Eingabeparameter

Bezeichnung	Beschreibung	Einheit	Testsystem
Workload-Parameter			
TotalIOs	Gesamtanzahl der IO-Zugriffe im Benchmark	Anzahl	100.000
IOSize	Größe eines IO-Zugriffs	Byte	8192
IO_{Paral}	Maximale Anzahl paralleler IO-Zugriffe	Anzahl	3
IODelay	Mittlerer Abstand zwischen den IO-Zugriffen (deterministisch oder neg. exp. verteilt)	ms	0
$Burst_{Prob}$	Wahrscheinlichkeit, dass ein Burst (Segment sequentiell zusammenhängender IOs) auftritt, sonst wird ein zufällig verteilter Zugriff (Random) erzeugt; im Fall Random werden die IO-Zugriffe über alle logischen Blöcke der Festplatte gleichverteilt	---	0,20
$RunLength_{Seq}$	Mittlere Anzahl der IO-Zugriffe in einem Burst	Anzahl	3
$Read_{Prob}$	Wahrscheinlichkeit, dass ein Lesezugriff (Read) auftritt, sonst wird ein Schreibzugriff (Write) erzeugt	---	0,50
SCSI-Mode-Parameter			
$PreFetch_{Max}$	Maximale Anzahl an Bytes, die bei jedem IO zusätzlich in den Cache geschrieben wird, falls sich kein weiterer IO in der Festplatten-Queue befindet	Byte	131.072
Cache Segmentation	Segmentierung des Cache (Anzahl und Größe der Segmente)	---	14 x 256 KB
Read Ahead	Read Ahead aktiviert oder deaktiviert	On/Off	On
Tagged Command Queueing	Tagged Command Queueing aktiviert oder deaktiviert	On/Off	On
Write Cache	Write Cache aktiviert oder deaktiviert; bei aktiviertem Write Cache wird Back-to-Back Write-Caching-Strategie verwendet	On/Off	On
SCSI-Device-Driver-Parameter			
$IOSize_{Max}$	Maximale Größe der IO-Zugriffe	Byte	1.048.576
SortAlg	Verwendeter Sortier-Algorithmus für Device Driver Queue (FCFS oder Elevator bei Solaris)	---	Elevator

Tabelle 5.1 Modell-Eingabeparameter

5.2.1.1. Festplattenparameter

Unter der Rubrik Festplattenparameter werden alle vom Hersteller frei erhältlichen Informationen über Hardware-Ausstattung und die Performance der Festplatte beschrieben. Die in Tabelle 5.1 dargestellte Unterscheidung der Medium-Transferrate in Instantaneous und Sustained basiert auf der folgenden Definition (siehe [49]):

- **Instantaneous Transfer Rate:** Unter der Instantaneous Transfer Rate versteht man die maximal mögliche Daten-Transferrate ohne Overhead-Zeiten durch z. B. Kopf/Zylinder-Wechselzeiten. Die Festplatte ist mittels Zone Bit Recording in 11 verschiedene Zonen unterteilt. Die Transferrate muss somit in Abhängigkeit der Zonen definiert werden. Im Modell wird die mittlere Transferrate über alle Zonen verwendet.
- **Sustained Transfer Rate:** Die Sustained Transfer Rate beschreibt die Transferrate zwischen Platten-Medium (Platters) und Platten-Cache unter Berücksichtigung der Kopf/Zylinder-Wechselzeit. Die Definition der Transferrate ist auch hier abhängig von der zu betrachtenden Zone, so dass im Modell der Mittelwert über alle Zonen verwendet wird.

Der Parameter $Ovhd_{Ctrl}$ basiert auf der Definition eines „Command Overhead“ von IBM. Nach [49] unterscheidet IBM hierbei zwischen einem Cache Hit (0,03 ms) und einem Cache Miss (0,4 ms), wobei die jeweiligen Zeiten die Overhead-Zeiten darstellen. Der Parameter BusSpeed beschreibt die Geschwindigkeit des SCSI-Bus, welche dem Standard entsprechend 160 MB/s beträgt. Nach [139] ist jedoch von einem ca. 30 %-igen Overhead auszugehen, so dass als Eingabeparameter 110 MB/s angenommen wird. Alle weiteren nicht beschriebenen Parameter können in Kapitel 4 oder in [70] bzw. [84] nachgelesen werden.

5.2.1.2. Workload-Parameter

Die Workload-Parameter beschreiben den Zugriffstyp (lesend oder schreibend) und in welcher Art auf die Festplatte zugegriffen wird (zufällig oder sequentiell). In dem in Tabelle 5.1 dargestellten Beispiel werden insgesamt 100.000 IO-Zugriffe mit einer IO-Größe von 8 KB generiert. Die IOs werden asynchron erzeugt, jedoch mit der Beschränkung, dass immer nur maximal drei IO-Aufträge parallel existieren dürfen. Der mittlere Zwischenankunftsabstand zwischen zwei IO-Zugriffen beträgt 0 ms. Der sequentielle Anteil der Last wird durch Bursts definiert, d. h. die Last enthält Segmente sequentiell aufeinander folgender IO-Zugriffe, die eine mittlere Länge von $RunLength_{Seq}$ besitzen. Der Parameter $Burst_{Prob}$ gibt an, mit welcher Wahrscheinlichkeit Bursts generiert werden. Innerhalb eines Burst darf dann nicht mehr zwischen lesendem und schreibendem Zugriff gewechselt werden. Zuletzt gibt der Parameter $Read_{Prob}$ die Wahrscheinlichkeit mit der ein Lesezugriff generiert wird an. Wie bereits beschrieben, wurde die Worklod-Charakterisierung in starker Anlehnung an die Parametrisierung des IO-Benchmarks DBench definiert.

5.2.1.3. SCSI-Mode-Parameters

Die SCSI-Mode-Pages beschreiben die für die Festplatte gesetzten Parameter. Über diese Mode-Pages werden Funktionen wie z. B. Read Ahead Caching oder Tagged Command Queueing gesteuert und somit das Festplattenverhalten beeinflußt. Die in Tabelle 5.1 angegebene Caching-Strategie Back-to-Back-Writing bei aktiviertem Write Cache bedeutet, dass ein Schreibzugriff bereits nach dem Schreiben der Daten in den Cache aus Sicht des Betriebssystems beendet ist und somit ein neuer IO-Auftrag generiert werden kann (siehe auch Kapitel 4).

5.2.1.4. SCSI-Device-Driver-Parameter

Abschließend werden in Tabelle 5.1 die Parameter des SCSI Device Driver angegeben, die wie in diesem Fall die maximale Größe eines IO-Zugriffs und den eingestellten Sortieralgorithmus für die Device Driver-Queue vorgeben. Die Queue des Device Driver entspricht dann nach Definition von Solaris der Wait Queue (siehe auch Kapitel 4).

5.2.2. Berechnung der Bedienzeit

In diesem Abschnitt wird dargestellt, wie anhand der zuvor beschriebenen Modell-Eingabeparameter die Bedienzeit pro IO berechnet wird. Diese entspricht der Bedienzeit in der Active Queue (siehe SIOSTAT), d. h. die aktive Zeit eines IO-Auftrags auf dem SCSI-Bus sowie in der Festplatte.

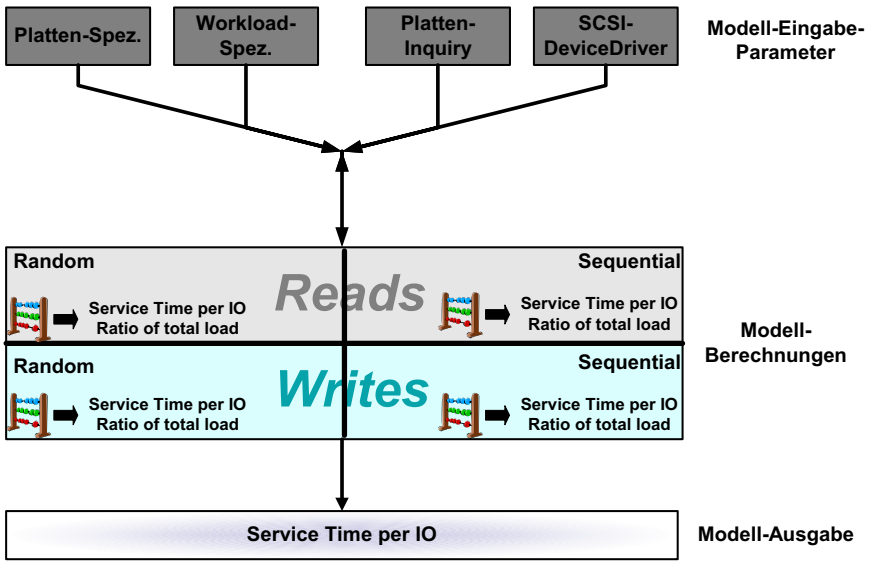


Abbildung 5.1 IO-Modell (siehe [131])

Die Berechnung der Bedienzeit wird, wie in Abbildung 5.1 dargestellt, in zwei Fälle unterschieden: Lesender und schreibender Zugriff. Es werden somit anhand der Lastbeschreibungen die IOs in lesende und schreibende Zugriffe unterteilt, da jeweils verschiedene Bedienstrategien von der Festplatte verwendet werden (siehe Kapitel 4) und somit verschiedene Bedienzeiten resultieren können. Diese Unterscheidung in lesende und schreibende Zugriffe wird von Menascé und Shriver nicht vorgenommen, da diese, wie bereits beschrieben, nur die Bedienstrategien für lesende Zugriffe betrachten. Weiterhin wird für jeden lesenden und schreibenden Zugriff nochmals zwischen sequentiellem (Sequential) und zufälligem (Random) Zugriff unterschieden. Man erhält somit, wie in Abbildung 5.1 dargestellt, vier verschiedene Bedienzeiten, die zum Schluss zu einer Gesamtbedienzeit vereint werden, da auch das Monitoring-Skript SIOSTAT nur eine Bedienzeit für alle IO-Zugriffe liefert.

Nachfolgend werden für jeden dieser Fälle die entsprechenden Algorithmen zur Berechnung der Bedienzeit beschrieben. Zunächst werden jedoch Hilfsparameter berechnet, die direkt aus den Eingabeparametern abgeleitet werden können und die für die nachfolgenden Berechnungen benötigt werden:

- **Anzahl aller Bursts (TotalBursts) und prozentualer Anteil von Random IOs an der Gesamtlast ($\mathbf{Rand_{Ratio}}$)** berechnet anhand der vorgegebenen Parameter $RunLength_{Seq}$ und $Burst_{Prob}$:

$$TotalBursts = \frac{TotalIOs \bullet Burst_{Prob}}{Burst_{Prob} \bullet RunLength_{Seq} + 1 - Burst_{Prob}}$$

$$Rand_{Ratio}\,[\%] = 1 - \left(\frac{TotalBursts \bullet RunLength_{Seq}}{TotalIOs}\right)$$

- **Transferzeiten (TT)** vom Platten-Medium in den Platten-Cache für die vorgegebene IO-Größe (unterschieden nach Instantaneous und Sustained Transferraten):

$$TT_{Inst}\,[ms] = \frac{IOSize\,[Byte]}{TransRate_{Inst}\,[MB/s]} \bullet 1000$$

$$TT_{Sust}\,[ms] = \frac{IOSize\,[Byte]}{TransRate_{Sust}\,[MB/s]} \bullet 1000$$

  Der Faktor 1000 dient jeweils zur Berechnung der Transferzeiten in Millisekunden.

- **Read und Write RunLength** als durchschnittliche Anzahl von aufeinander folgenden IOs gleichen Typs, d. h. Read oder Write:

$$RunLength_{Read} = \frac{1}{1 - Read_{Prob}}\text{ , falls } Read_{Prob} < 1\text{, sonst } RunLength_{Read} = TotalIOs$$

$$RunLength_{Write} = \frac{1}{Read_{Prob}}\text{ , falls } Read_{Prob} > 0\text{, sonst } RunLength_{Write} = TotalIOs$$

  Die Fälle $Read_{Prob} = 0$ bzw. $Read_{Prob} = 1$ stellen Sonderfälle dar, denn dann existieren nur IO-Zugriffe eines Typs und somit ist die Gesamtzahl aller generierten IO-Zugriffe (TotalIOs) gleich der jeweiligen RunLength.

- **Disk Controller Overhead ($Ovhd_{Ctrl}$) und SCSI-Bus-Transferzeit (TT_{Bus})** berechnet auf Basis der IO-Lastbeschreibung:

  Für sequentielle Last: $Ovhd_{Ctrl}\,[ms] = \frac{maxOvhd_{Ctrl}\,[ms]}{RunLength_{Seq}}$; mit $maxOvhd_{Ctrl} = 0{,}4$ ms

  Für zufällig verteilte Last: $Ovhd_{Ctrl} = maxOvhd_{Ctrl}\,[ms]$; mit $maxOvhd_{Ctrl} = 0{,}4$ ms

  Die Angaben min und max für $Ovhd_{Ctrl}$ bedeuten, dass die in Tabelle 5.1 angegebenen minimalen und maximalen Werte verwendet werden. Es wird hierbei der Controller Overhead in Abhängigkeit der Last definiert, jedoch kann dieser minimal nur 0,03 ms betragen.

  Für sequentielle Last ergibt sich die Definition der Disk Controller Overhead-Zeit, da bei einem Burst der erste IO immer einen zufällig verteilten Zugriff darstellt, der demnach mit hoher Wahrscheinlichkeit einen Cache-Miss produziert und somit einen maximalen Overhead. Die nachfolgenden IOs hingegen können bei Read-Ahead aus dem Cache gelesen werden, so dass keine weiteren Overhead-Zeiten entstehen. Betrachtet man eine zufällig verteilte IO-Last, so existieren keine Bursts und somit entsteht bei jedem IO-Zugriff ein Cache-Miss und damit die maximale Overhead-Zeit.

  Abschließend wird die SCSI-Bus-Transferzeit pro IO-Zugriff definiert:

$$TT_{Bus}\,[ms] = \frac{IOSize\,[Byte]}{BusSpeed\,[MB/s]} \bullet 1000$$

  Der Faktor 1000 dient zur Berechnung der Transferzeit in Millisekunden.

5.2.2.1. Bedienzeit für lesende Zugriffe

Es folgt in diesem Abschnitt die Bedienzeitberechnung für lesende Zugriffe, unterteilt in die Fälle Random (zufällig verteilte IO-Zugriffe) und Sequential Load (Last beinhaltet Segmente (Bursts) mit sequentiell aufeinanderfolgenden IO-Zugriffen). Für beide Fälle wird jeweils eine separate Bedienzeit berechnet.

Bedienzeit für lesende Zugriffe bei Random Load

Zur Berechnung der Bedienzeit für Random Load werden, wie in [99] und [70] beschrieben, die folgenden Parameter ermittelt:

- *Disk Cache Miss Probability [%] = $Cache_{Rand}$ = 100 %*, d. h. kein IO-Zugriff wird vom Cache bedient, so dass in diesem Fall der Cache keinen Einfluss auf die Performance der Festplatte besitzt
- *Mittlere Seek Time [ms] = $SeekTi_{Rand}$ = SeekTi [ms]*
- *Mittlere Latenzzeit [ms] = $Latency_{Rand}$ = 60000 / (2 • DiskSpeed [RPM])*, d. h. es wird davon ausgegangen, dass jeder IO-Zugriff im Durchschnitt eine halbe Plattenumdrehung warten muss

Die beiden nachfolgenden Parameter stellen den Einfluss paralleler IO-Verarbeitung auf die Bedienzeit der Festplatte dar. Wie bereits beschrieben, sind für dieses Lastszenario keine Informationen über die Bedienstrategie und Sortieralgorithmen der Festplatte bekannt. In dieser Arbeit wird somit die Arbeitsweise der Festplatte mittels gemessener Skalierungsfaktoren beschrieben, d. h. mittels Benchmarking wurden die Bedienzeiten in Abhängigkeit der parallelen IO-Zugriffe ermittelt und dann als Skalierungsfaktoren (Quotient aus der Bedienzeit n paralleler IOs und der Bedienzeit eines IOs ohne Parallelität) in einer IO-Bibliothek gespeichert (siehe Kapitel 4). Somit erhält man den folgenden Parameter:

- Skalierung der Bedienzeit in Abhängigkeit paralleler IOs = $Skal_{Read}$ (der IO-Bibliothek entnommen)

Ein vergleichbares Vorgehen zur Bestimmung von Modelleingabeparametern wird auch in [141] beschrieben, wo mit Hilfe von Messung und Benchmarking Modell-Eingabeparameter für die Simulation von Festplatten gewonnen werden. Wie bereits in Abschnitt 5.1. beschrieben, ist jedoch der wesentliche Unterschied, dass in [141] u. a. „low-level“-Zugriffsmöglichkeiten benötigt und verwendet werden, aber im Rahmen dieser Arbeit ausschließlich das Monitoring-Werkzeug SIOSTAT, dass auf gleicher Ebene wie die Standardwerkzeuge IOSTAT oder VMSTAT arbeitet, verwendet wird. Somit werden auch die Skalierungsfaktoren der IO-Bibliothek, wie bereits in Kapitel 4 beschrieben, mit Hilfe des SIOSTAT-Skripts und des IO-Benchmarks DBench ermittelt.

Abschließend wird die Bedienzeit für den Fall Random Load wie folgt berechnet:

$$\begin{aligned} ServiceTime_{ReadRand}\,[ms] = (&SeekTi_{Rand}\,[ms] + Latency_{Rand}\,[ms] + TT_{Inst}\,[ms] + \\ &Ovhd_{Ctrl}\,[ms] + TT_{Bus}\,[ms] + K\,[ms]) \bullet Skal_{Read} \end{aligned}$$

Für Random Load wird die Instantaneous Transferrate verwendet, d. h. die pro Track maximal mögliche Geschwindigkeit, da in diesem Fall im Gegensatz zur Sequential Load nur auf wenige Sektoren eines Tracks zugegriffen wird (in Abhängigkeit der IO-Größe) und somit die Wahrscheinlichkeit für z. B. durch Head-Switches verursachte Overhead-Zeiten sehr gering ist. Die Konstante K steht für einen eventuell notwendigen konstanten Kalibrierungswert.

Bedienzeit für lesende Zugriffe bei Sequential Load

Unter der Lastbeschreibung Sequential Load wird verstanden, dass sich ein IO innerhalb eines Bursts befindet, d. h. innerhalb eines Segments mit IO-Zugriffen, die sequentiell auf die Sektoren einer Festplatte zugreifen. Es ist hierbei zu beachten, dass nach Definition der erste IO-Zugriff eines Bursts immer einem zufällig verteilten IO-Zugriff entspricht, da sich ansonsten der IO-Zugriff innerhalb eines Bursts befindet. Zur Berechnung der Bedienzeit für Sequential Load werden, wie in [99] und [70] beschrieben, die folgenden Parameter berechnet:

- *Disk Cache Miss Probability [%] = $Cache_{Seq}$ = 1 / ($RunLength_{Seq}$),*

  falls IOSize [Byte] < $PreFetch_{Max}$ [Byte], sonst $Cache_{Seq}$ = 100 %, d. h. der prozentuale Anteil an IO-Zugriffen, dessen Daten sich nicht im Cache der Festplatte befinden. Ist die IO-Größe größer als $PreFetch_{Max}$, so wird von der Festplatte kein Read-Ahead durchgeführt und somit befinden sich auch die Daten für den nachfolgenden IO-Zugriff nicht mehr im Cache.

- *Mittlere Seek Time [ms] = $SeekTi_{Seq}$ = SeekTi [ms] / $RunLength_{Seq}$*

- *Mittlere Latenzzeit [ms] = $Latency_{Seq}$ = 60000 / (2 • DiskSpeed [RPM])*

  Die mittlere Latenzzeit wird bei Menascé [70] und Shriver [99] in Abhängigkeit der Festplattenauslastung definiert, d. h.:

$$\text{Mittlere Latenzzeit} = \frac{\frac{1}{2} + (\text{RunLength}_{Seq} - 1) \bullet \left[\frac{(1 + U_d)}{2}\right]}{\text{RunLength}_{Seq}} \bullet \frac{60}{\text{DiskSpeed}}$$

  mit U_d = Auslastung der Festplatte (die Auslastung liegt zwischen 0 und 1)

  Vorausgesetzt wird hierbei, dass mit zunehmender Auslastung der Festplatte, die Zahl der an direkt aufeinander folgenden IO-Aufträge anwächst. Weiterhin wird angenommen, dass in diesem Fall der nachfolgende IO durch den Controller-Overhead derart verzögert wird, dass der Schreib-/Lese-Kopf bereits den gesuchten Sektor passiert hat und somit der nachfolgende IO-Zugriff eine vollständige Umdrehung der Festplatte abwarten muss. Dieses von Shriver und Menascé beschriebene Festplattenverhalten ist anhand von Messungen mit Hilfe des in dieser Arbeit verwendeten Testsystems nicht festgestellt worden. Zu vermuten ist, dass derzeitige Festplatten Optimierungsverfahren und eine verbesserte Blockadressierung verwenden, so dass gerade bei sequentiellen Zugriffen keine Latenzzeiten zwischen zwei IOs auftreten bzw. eine effizientere SCSI-Command-Verarbeitung vorhanden ist, wie z. B. durch das in Kapitel 4 beschriebene SCSI-Packetization. Somit wurde bei dem hier zu beschreibenden Modell keine Abhängigkeit der Latenzzeit von der Plattenauslastung definiert und es ergibt sich daher eine identische Berechnung der Latenzzeit, wie bereits für Random Load beschrieben.

Zusätzlich zu den Parametern von Shriver und Menascé wird, wie auch in Kapitel 4 anhand der Messergebnisse beschrieben, ein Overhead zur Beschreibung der Last mit einem Mix aus Lese- und Schreiboperationen eingeführt. Es wird hierbei angenommen, dass beim Wechsel von einem Lese- auf einen Schreibzugriff der Festplattenkopf bzw. der Festplatten-Controller eine Verzögerung verursacht, so dass im sequentiellen Fall der nachfolgende IO eine vollständige Umdrehung warten muss. Dieser zusätzliche Zeitaufwand verteilt sich dann auf das Segment mit Länge $RunLength_{Read}$.

- Overhead für Read-Write-Mix [ms] = $OvhdR_{ReadProb} = \frac{2 \bullet \text{Latency}_{Seq}\,[\text{ms}]}{\text{RunLength}_{Read}}$,

  falls $0 < Read_{Prob} < 1$

  Der beschriebene Overhead wird nur für den Wechsel von einem lesenden auf einen schreibenden Zugriff berücksichtigt, da hierbei nach dem lesenden Zugriff zunächst der

schreibende Zugriff generiert werden muss. Hingegen entsteht beim Wechsel von einem Schreib- auf einen Lesezugriff aufgrund der Bedienstrategie Back-to-Back-Writing keine Verzögerungszeit, da während der Bedienung des Schreibzugriffs bereits der Lesezugriff generiert wird, denn bereits nach dem Schreiben der Daten in den Festplatten-Cache, kann der nachfolgende IO-Zugriff generiert werden.

Die Bedienzeit für den Fall Sequential Load wird wie folgt berechnet:

$$ServiceTime_{ReadSeq}\,[ms] = SeekTi_{Seq}\,[ms] + Cache_{Seq}\,[\%] \bullet Latency_{Seq}\,[ms] + TT_{Sust}\,[ms] + Ovhd_{Ctrl}\,[ms] + Ovhd_{ReadProb}\,[ms]$$

Im Gegensatz zur Random Load wird bei sequentieller Last die Sustained Transferrate verwendet, da hierbei auf die Sektoren der Festplatte sequentiell zugegriffen wird und somit auch im Gegensatz zur Random Load eine größere Datenmenge in Folge gelesen wird, so dass eine höhere Wahrscheinlichkeit für z. B. einen Track- oder Head-Switch existiert und bedingt hierdurch auch die entsprechenden Overhead-Zeiten entstehen, die in der Sustained Transferrate enthalten sind. Ein weiterer Unterschied zur Random Load ist die fehlende Berücksichtigung der Bus-Transfer-Zeit TT_{Bus}. Begründet liegt dies in der Caching-Strategie Read Ahead der Festplatte, die bei sequentieller Last in hohem Maße Verwendung findet. Die Daten befinden sich in Abhängigkeit der Ankunftsrate der IOs bereits vollständig bzw. teilweise im Cache. Dadurch bedingt laufen das Read Ahead-Caching und der Bus-Transfer parallel, so dass für die Bedienzeit die SCSI-Bus-Transferzeit entfällt.

Für den Controller Overhead muss ergänzt werden, dass im Fall 100 % sequentieller Zugriff anhand von Messungen nachgewiesen wurde, dass die Bedienzeit nahezu der Transferzeit entspricht und somit der Controller Overhead das Modellergebnis verschlechtert. Daher wird im Fall 100 % sequentieller IO-Last der Controller Overhead nicht berücksichtigt.

5.2.2.2. Bedienzeit für schreibende Zugriffe

Wie bereits für lesende Zugriffe beschrieben, wird auch für die schreibenden zur Bedienzeitberechnung zwischen den Fällen Random (zufällig verteilte IOs) und Sequential Load (Last beinhaltet Segmente (Bursts) mit sequentiell aufeinanderfolgenden IOs) unterschieden. Für beide Fälle resultiert somit ebenfalls eine separate Bedienzeit.

Bedienzeit für schreibende Zugriffe bei Random Load

Schreibzugriffe werden in diesem Modell mit aktiviertem Schreib-Cache betrachtet. Die Verarbeitung der Schreibzugriffe erfolgt somit unter Verwendung der Back-to-Back Write-Caching-Strategie, d. h. der IO-Prozess wird nach dem Schreiben der Daten in den Cache wieder frei gegeben. Aufgrund der fehlenden Kenntnis über die Bedienstrategie der Festplatte bei mehreren Aufträgen in der Queue wird für die Bedienzeit bei Random Load eine durch Messung ermittelte Bedienzeit ($ServiceTime_{WriteBench}$) zugrunde gelegt, die für das folgende Lastszenario ermittelt wurde:

- 100 % Schreiben, IO-Größe 8 KB, keine Parallelität, 100 % Random (Lastszenario für $ServiceTime_{WriteBench}$)

Weiterhin steht, wie bereits für lesende Zugriffe beschrieben, eine IO-Bibliothek mit den Skalierungsfaktoren (Quotient aus der Bedienzeit n paralleler IOs und der Bedienzeit eines IOs ohne Parallelität) für schreibende Zugriffe bei paralleler IO-Verarbeitung zur Verfügung:

- Skalierung der Bedienzeit = $Skal_{Write}$ (der IO-Bibliothek entnommen)

Die Bedienzeit für den Fall Random Load wird dann wie folgt berechnet:

$ServiceTime_{WriteRand}$ [ms] =

$$\left(ServiceTime_{WriteBench}\ [ms] - \left(\frac{8\ [KB] \bullet 1000}{TransRate_{Inst}\ [MB/s]} + \frac{8\ [KB] \bullet 1000}{BusSpeed\ [MB/s]}\right) + TT_{Bus}\ [ms] + TT_{Inst}\ [ms]\right) \bullet Skal_{Write}$$

falls IOSize < $PreFetch_{Max}$, sonst $ServiceTime_{WriteRand} = ServiceTime_{ReadRand}$

Die Bedienzeit wird, wie bereits beschrieben, anhand des gemessenen Zeitverbrauchs $ServiceTime_{WriteBench}$ ermittelt, so dass zunächst Medium- und Bus-Transferzeit für eine IO-Größe von 8 KB subtrahiert und dann für die gewünschte IO-Größe addiert werden. Es hat sich weiterhin anhand von Messungen gezeigt, dass sobald die IO-Größe den Wert $PreFetch_{Max}$ übersteigt, sich die Bedienzeiten für lesende und schreibende Zugriffe angleichen. Wie bereits für die Lesezugriffe beschrieben, werden ab einer derartigen IO-Größe die Optimierungsverfahren der Festplatte nicht weiter verwendet, d. h. in diesem Fall das Back-to-Back-Writing deaktiviert.

Bedienzeit für schreibende Zugriffe bei Sequential Load

Auf Basis der bereits beschriebenen Bedienzeit für Random Load wird nachfolgend die Bedienzeit für Sequential Load berechnet. Es wird hierbei davon ausgegangen, dass die Schreibzugriffe als zusammenhängendes Segment (Burst) angeordnet sind. Man definiert wie auch bereits für die lesenden Zugriffe, dass ein derartiges Segment immer mit einem zufällig verteilten IO-Zugriff beginnt. Alle nachfolgenden Schreibzugriffe werden dann mit der Sustained Transferrate bedient. Bei einem Lastmix mit Lese- und Schreibzugriffen muss wie auch bei den Lesezugriffen ein Overhead für den Wechsel vom Lese- auf den Schreibzugriff berücksichtigt werden:

- Overhead für Read-Write-Mix [ms] = $OvhdW_{ReadProb} = \frac{2 \bullet Latency_{Rand}\ [ms]}{RunLength_{Write}}$,
  falls $0 < Read_{Prob} < 1$

Eine detaillierte Beschreibung zur Wahl der Sustained Transferrate und des Read-Write-Overheads befindet sich in Abschnitt 5.2.2.1. (Sequential Load).

Die Bedienzeit für den Fall Sequential Load wird dann wie folgt berechnet:

$ServiceTime_{WriteSeq}$ [ms] = $Ovhd_{ReadRatio}$ +

$$\frac{ServiceTime_{WriteRand}\ [ms] + (RunLength_{Seq} - 1) \bullet (TT_{Sust}\ [ms] + Ovhd_{Ctrl}\ [ms])}{RunLength_{Seq}}$$

Anhand von Messungen wurde festgestellt, dass im Fall Burst-Probability = 1, wie auch bereits für die Lesezugriffe dargestellt, die Bedienzeit nahezu der Transferzeit entspricht, so dass in diesem Fall die Overhead-Zeit des Controllers zur Berechnung der Bedienzeit nicht berücksichtigt wird.

5.2.2.3. Berechnung der Gesamtbedienzeit

Die Berechnung der Gesamtbedienzeit setzt sich somit aus den vorher errechneten Bedienzeiten für Lese- und Schreibzugriffe zusammen:

$$ServiceTime_{Disk}\ [ms] =$$

$$Read_{Ratio} \bullet (Rand_{Ratio} \bullet ServiceTime_{ReadRand} + (1 - Rand_{Ratio}) \bullet ServiceTime_{ReadSeq}) +$$
$$(1 - Read_{Ratio}) \bullet (Rand_{Ratio} \bullet ServiceTime_{WriteRand} + (1 - Rand_{Ratio}) \bullet ServiceTime_{WriteSeq})$$

$$\text{mit } Read_{Ratio}\ [\%] = Read_{Prob} \bullet 100$$

Die Gesamtbedienzeit setzt sich aus den vier verschiedenen Bedienzeiten für Reads und Writes, jeweils aufgegliedert nach Sequential und Random Load, zusammen. Die einzelnen Bedienzeiten werden gewichtet mit den prozentualen Anteilen $Rand_{Ratio}$ und $Read_{Ratio}$. Die somit resultierende Bedienzeit ist mit derjenigen bei der IO-Messung von SIOSTAT gelieferten vergleichbar. Im nachfolgenden Abschnitt werden vom Modell errechnete und anhand von Messungen ermittelte Bedienzeiten verglichen.

5.3. Bedienzeit-Modellierung und -Messung im Vergleich

Das bislang beschriebene Festplattenmodell zur Bedienzeitberechnung wird in den folgenden Abschnitten anhand der in Kapitel 4 definierten IO-Lastszenarien validiert. Es werden hierzu die Lastszenarien Simple Load, Sequential/Random Load-Mix und Read/Write Load-Mix verwendet (siehe auch Kapitel 4). In einem weiteren Schritt werden aus diesen drei Basisszenarien neue Lastkompositionen zusammengestellt, indem die Basislasten miteinander kombiniert werden. Die in den folgenden Tabellen dargestellten Bedienzeiten sind jeweils als Bedienzeiten pro IO-Zugriff zu interpretieren. Der für die Bedienzeitberechnung beschriebene Kalibrierungswert K wurde für alle nachfolgenden Szenarien wie folgt definiert:

K = 0,5 ms für Lesezugriffe (Random Load)

Der Kalibrierungswert wurde auf Basis der in den realen Messungen gewonnenen Erkenntnisse ermittelt und dient der Angleichung des Modells an die Messergebnisse.

5.3.1. Simple Load

Das IO-Lastszenario Simple Load besitzt die folgenden Charakteristiken: 100 % lesende oder 100 % schreibende IO-Zugriffe, wobei Zugriffe entweder 100 % zufällig verteilt oder 100 % sequentiell sind; keine parallele IO-Verarbeitung; Variation der IO-Größe. Es folgen zwei Tabellen, Tabelle 5.2 für lesende und Tabelle 5.3 für schreibende IO-Zugriffe, die jeweils nach sequentiellen (Sequential Load) und zufällig verteilten (Random Load) IO-Lasten unterteilt sind und die Modellergebnisse sowie deren Abweichung zu den Benchmark-Messungen beschreiben. Eine Darstellung der Benchmark-Messungen erfolgte bereits in Kapitel 4.

IO-Size [KB]	Bedienzeit Messung [ms]	Bedienzeit Modell [ms]	Fehler[1] [%]
100 % Read, Random Load			
8	9,09	9,1	0,11 %
32	9,95	9,99	0,40 %
64	11,23	11,18	-0,45 %
128	13,85	13,56	-2,09 %
256	18,17	18,32	0,83 %
100 % Read, Sequential Load			
8	0,23	0,27	17,39 %
32	1,06	1,08	1,89 %
64	2,16	2,16	0,00 %
128	4,38	4,31	-1,60 %
256	8,86	8,62	-2,71 %

Tabelle 5.2 Modellergebnisse für Simple Load mit 100 % Lesezugriffen

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

IO-Size [KB]	Bedienzeit Messung [ms]	Bedienzeit Modell [ms]	Fehler[1] [%]
100 % Write, Random Load			
8	6,48	6,48	0,00 %
32	7,40	7,37	-0,41 %
64	8,66	8,56	-1,15 %
128	14,20	14,26	0,42 %
256	18,99	19,02	0,16 %
100 % Write, Sequential Load			
8	0,27	0,27	0,00 %
32	1,09	1,08	-0,92 %
64	2,23	2,16	-3,14 %
128	4,49	4,31	-4,01 %
256	9,07	8,62	-4,96 %

Tabelle 5.3 Modellergebnisse für Simple Load mit 100 % Schreibzugriffen

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Anhand der in Tabelle 5.2 und Tabelle 5.3 dargestellten Ergebnisse ist zu erkennen, dass für die verschiedenen IO-Größen eine maximale Abweichung der Modellergebnisse von den Messergebnissen von 4,96 % (prozentualer Fehler in Betrag) existiert. Die Ausnahme bildet hierbei der Fall 100 % Read bei 100 % sequentieller Last mit einer IO-Größe von 8 KB. Hierbei liegt der Fehler bei 17,39 %, ist jedoch hinsichtlich der Größenordnung der absoluten Werte in diesem Fall tolerierbar.

5.3.2. Sequential/Random Load-Mix

Das IO-Lastszenario Sequential/Random Load-Mix besitzt die folgenden Charakteristiken: 100 % lesende oder 100 % schreibende IO-Zugriffe; IO-Größe von 8 KB; keine parallele IO-Verarbeitung sowie ein Mix aus sequentiellen und zufällig verteilten IO-Zugriffen. Es folgt somit in Tabelle 5.4 eine Unterteilung nach lesenden und schreibenden IO-Zugriffen, welche die Modellergebnisse sowie deren Abweichung zu den Benchmark-Messungen darstellt. Eine Beschreibung der Benchmark-Messungen erfolgte bereits in Kapitel 4. Die Spalte $Burst_{Prob}$ der Tabelle 5.4 stellt die Burst-Probability dar, d. h. die Wahrscheinlichkeit, dass ein Burst entsteht. Die Spalte $RunLength_{Seq}$ gibt die Länge eines Bursts an, also die mittlere Anzahl der in einem Burst enthaltenen IO-Zugriffe.

$Burst_{Prob}$ [%]	$RunLength_{Seq}$	Bedienzeit Messung [ms]	Bedienzeit Modell [ms]	Fehler[1] [%]
100 % Read, 8 KB IO-Größe				
0	0	9,09	9,10	0,11 %
0,20	2	7,31	7,33	0,27 %
0,40	3	5,56	5,56	0,00 %
0,60	4	3,80	3,79	-0,26 %
0,80	6	2,00	2,03	1,50 %
1	TotalIOs	0,23	0,27	17,39 %
100 % Write, 8 KB IO-Größe				
0	0	6,48	6,48	0,00 %
0,20	2	5,26	5,27	0,19 %
0,40	3	4,07	4,06	-0,25 %
0,60	4	2,85	2,82	-1,05 %
0,80	6	1,63	1,56	-4,29 %
1	TotalIOs	0,27	0,27	0,00 %

Tabelle 5.4 Modellergebnisse für Random/Sequential Load-Mix

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Wie in Tabelle 5.4 dargestellt, beträgt der maximale Fehler zwischen realer Messung und Modell ca. 4,29 % (prozentualer Fehler in Betrag). Die Interpretation des Fehlers wurde bereits für Simple Load beschrieben. Die Ausnahme bildet hierbei wieder der Fall 100 % sequentielles Lesen mit 17,39 % Abweichung zur realen Messung. Diese Lastkomposition ist identisch mit Simple Load und wurde somit bereits in Abschnitt 5.3.1. beschrieben.

5.3.3. Read/Write Load-Mix

Das IO-Lastszenario Read/Write Load-Mix besitzt die folgenden Charakteristiken: 100 % zufällig verteilte oder 100 % sequentielle IO-Zugriffe; IO-Größe von 8 KB; keine parallele IO-Verarbeitung sowie ein Mix aus lesenden und schreibenden IO-Zugriffen. Es folgt somit Tabelle 5.5 unterteilt nach sequentiellen und zufällig verteilten IO-Zugriffen, welche die Modellergebnisse sowie deren Abweichung zu den Benchmark-Messungen darstellt. Eine Beschreibung der Benchmark-Messungen erfolgte bereits in Kapitel 4. Die Spalte $Read_{Prob}$ der Tabelle 5.5 stellt die Read-Probability dar, d. h. die Wahrscheinlichkeit, dass ein Lesezugriff entsteht.

$Read_{Prob}$ [KB]	Bedienzeit Messung [ms]	Bedienzeit Modell [ms]	Fehler[1] [%]
100 % Sequential Load, 8 KB IO-Größe			
0	0,27	0,27	0,00 %
0,20	2,13	2,19	2,82 %
0,40	3,10	3,15	1,61 %
0,60	3,23	3,15	-2,48 %
0,80	2,13	2,19	2,82 %
1	0,23	0,27	17,39 %
100 % Random Load, 8 KB IO-Größe			
0	6,48	6,48	0,00 %
0,20	7,42	7,27	-2,02 %
0,40	8,59	8,70	1,28 %
0,60	9,06	9,10	0,44 %
0,80	9,16	9,10	-0,66 %
1	9,10	9,10	0,00 %

Tabelle 5.5 Modellergebnisse für Read/Write Load-Mix

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Bei Betrachtung der in Tabelle 5.5 dargestellten Ergebnisse erkennt man einen maximalen Fehler zwischen realer Messung und Modell von 2,82 % (prozentualer Fehler in Betrag). Die Interpretation des Fehlers wurde bereits für Simple Load beschrieben. Die Ausnahme bildet hierbei wieder der Fall 100 % sequentielles Lesen mit 17,39 % Abweichung zur realen Messung. Diese Lastkomposition ist identisch mit Simple Load und wurde somit bereits in Abschnitt 5.3.1. beschrieben.

5.3.4. Load-Mix-Szenarien

Die in den bisherigen Abschnitten dargestellten Ergebnisse basieren auf den in Kapitel 4 definierten Basislastszenarien Simple Load, Sequential/Random Load-Mix und Read/Write Load-Mix, wobei jedes Lastszenario im Einzelfall betrachtet wurde. Die Lastmuster besitzen bislang eine sehr einfache Struktur, wodurch sich auch die relativ geringen Abweichungen von ca. 3 % bis 5 % zwischen Modell und Messung erklären lassen. Wie bereits in Kapitel 4 beschrieben, setzen sich reale IO-Lasten aus einer Kombination der definierten Basislastszenarien (Load-Mix) zusammen. Derartige Lastzusammenstellungen werden in den nun folgenden Abschnitten beschrieben, wobei die folgenden Gruppierungen von IO-Lasten betrachtet werden:

- Analyse der Szenarien Simple Load, Sequential/Random Load-Mix und Read/Write Load-Mix bei gleichzeitiger paralleler IO-Verarbeitung
- Betrachtung von Sequential/Random Load-Mix bei gleichzeitiger Verarbeitung von Read/Write Load-Mix und parallelen IOs, d. h. ein Szenario mit allen Basislastszenarien parallel

5.3.4.1. Simple Load und parallele IO-Verarbeitung

Für das Szenario Simple Load muss zwischen sequentiellen und zufällig verteilten sowie lesenden und schreibenden IO-Zugriffen unterschieden werden. Der Einfluss von paralleler IO-Verarbeitung für zufällig verteilte IO-Zugriffe (Random) wird sowohl für den Fall Lesen als auch für

den Fall Schreiben bereits in dem Basislastszenario Multi IO Load in Kapitel 4 (Abschnitt 4.3.2.5.) beschrieben.

Wird die Anzahl der parallelen IOs im sequentiellen Fall erhöht, so ergeben sich nur minimale Abweichungen in der Bedienzeit zwischen Modell und Messung, die im Durchschnitt unter 1 % liegen. Tendenziell liegen die Bedienzeiten hierbei aufgrund der parallelen IO-Verarbeitung höher als im nicht-parallelen Fall. Begründet liegt dies sicherlich in den Overhead-Zeiten, die durch die parallele IO-Verarbeitung entstanden sind.

5.3.4.2. Read/Write Load-Mix und parallele IO-Verarbeitung

In der nun folgenden Untersuchung wird das Basislastszenario Read/Write Load-Mix unter Verwendung paralleler IO-Verarbeitung analysiert. Es wird zwischen den Fällen 100 % sequentieller und 100 % zufällig verteilter IO-Zugriffe unterschieden.

100 % sequentielle IO-Zugriffe

In dem zu betrachtenden Lastszenario werden zum einen die Read-Probability und zum anderen die Zahl der parallelen IOs variiert. Eine detaillierte Beschreibung des Read/Write Load-Mix befindet sich in Kapitel 4. Die erzielten Ergebnisse für diese Lastkomposition werden in Tabelle 5.6 dargestellt, wobei die Modellergebnisse mit Hilfe der noch folgenden Modellmodifikation errechnet wurden.

Wie anhand von Tabelle 5.6 zu erkennen, sinken die Bedienzeiten pro IO-Zugriff bei zunehmender Zahl der parallelen IOs. Dieses Festplattenverhalten ist mit den bisher beschriebenen Modellalgorithmen nicht abbildbar. Da es sich um 100 % sequentielle Last handelt, werden die Skalierungsfaktoren der IO-Bibliothek zur Berechnung der Bedienzeit nicht verwendet, so dass diese nicht zur Modellierung dieses Szenarios herangezogen werden können. Wie bereits für das Basislastszenario Read/Write Load-Mix in Kapitel 4 Abschnitt 4.3.2.4. dargestellt, werden bei diesem Szenario durch das Modell Overhead-Zeiten berücksichtigt, die durch den Wechsel von lesenden auf schreibende Zugriffe entstehen. Diese Overheads werden als $OvhdR_{ReadProb}$ und $OvhdW_{ReadProb}$ bezeichnet und basieren, wie bereits beschrieben, auf der Annahme, dass durch den Wechsel von einem lesenden auf einen schreibenden IO-Zugriff eine Verzögerungszeit entsteht, so dass die Festplatte eine vollständige Rotation warten muss, d. h. $2 \bullet Latency_{Rand}$. Dieser Zeitverbrauch wird dann durch die entsprechende Anzahl der aufeinanderfolgenden Lese- bzw. Schreibzugriffe (RunLength) dividiert, so dass man die Bedienzeit pro IO-Zugriff erhält. Eine detaillierte Beschreibung der Overhead-Zeiten befindet sich in den Abschnitten 5.2.2.1. und 5.2.2.2. zur Berechnung der Bedienzeit für Sequential Load.

Die Grundidee zur Modellierung des Read/Write Load-Mix unter Verwendung paralleler IO-Verarbeitung ist, dass bei steigender Parallelität eine zunehmende Anzahl an Wechseln zwischen Lese- und Schreibzugriffen ohne Overhead vollzogen werden können. Betrachtet man einen Lesezugriff, so kann bei nicht paralleler IO-Verarbeitung zunächst nur der Lesezugriff bearbeitet werden und erst nach Beendigung dieses Zugriffs ein neuer IO-Auftrag an die Festplatte gesendet werden, wie z. B. ein Schreibzugriff. Durch diesen Wechsel entstehen Verzögerungszeiten und somit die beschriebenen Overheads. Bei paralleler IO-Verarbeitung hingegen existieren bereits eine bestimmte Anzahl an IOs in der Queue der Festplatte, so dass direkt im Anschluss an den zu bearbeitenden IO-Zugriff, die IO-Verarbeitung weitergeführt werden kann. Aus Sicht des Modells müssen somit bei Erhöhung der parallelen IOs die Parameter $RunLength_{Read}$ und $RunLength_{Write}$ ansteigen.

IO-Lasten mit $0 < Read_{Prob} < 1$ bestehen aus Segmenten mit einer bestimmten Anzahl an Lese- ($RunLength_{Read}$) und Schreibzugriffen ($RunLength_{Write}$), wobei jedes Segment aus zwei Übergängen zwischen Lesen und Schreiben besteht. Unter Verwendung dieser Definition existieren

bei einer $Read_{Prob}$ von 0,5 die größte Anzahl an Übergängen in Relation zur gesamten Last, da hierbei die Segmente die kleinste Größe (zwei IO-Zugriffe) besitzen. Verwendet man eine $Read_{Prob} \neq 0{,}5$ ergeben sich größere Segmente und somit eine geringere Anzahl an Übergängen. Die dabei entstehende Differenz an Übergängen im Vergleich zur $Read_{Prob}$ von 0,5 wird in den nachfolgenden Berechnungen als Parameter Diff bezeichnet und wie folgt berechnet:

$$Diff = \left[\frac{TotalIOs}{2} - \frac{TotalIOs \bullet 2}{RunLength_{Read} + RunLength_{Write}}\right] \bullet \frac{2}{TotalIOs}$$

Die Parameter $RunLength_{Read}$ und $RunLength_{Write}$ werden nun derart angepasst, dass zusätzlich zur bisherigen Berechnung eine bestimmte Anzahl an IOs abhängig von den parallelen IOs und der Read-Probability addiert wird, d. h. $IO_{Paral} \bullet Read_{Prob}$. Dieser Zuschlag wird bei einer $Read_{Prob}$ von 0,5 vollständig berücksichtigt. Bei einer $Read_{Prob} \neq 0{,}5$ hat sich anhand von Messungen jedoch gezeigt, dass in Abhängigkeit der Segmentgröße für Lese- und Schreibzugriffe (RunLength) und der damit verbundenen Anzahl an Wechseln zwischen Lesen und Schreiben nicht der vollständige Betrag von $IO_{Paral} \bullet Read_{Prob}$ benötigt wird, sondern ein Betrag abzüglich der bereits beschriebenen Differenz Diff, so dass sich die folgenden Berechnungsformeln für $RunLength_{Read}$ und $RunLength_{Write}$ ergeben:

$$RunLength_{Read} = \frac{1}{1 - Read_{Prob}} + IO_{Paral} \bullet Read_{Prob} \bullet (1 - Diff)$$

$$RunLength_{Write} = \frac{1}{Read_{Prob}} + IO_{Paral} \bullet (1\text{-}Read_{Prob}) \bullet (1 - Diff)$$

Die durch Messungen erzielten Bedienzeiten pro IO und die durch Modifikation des IO-Modells errechneten Bedienzeiten werden in Tabelle 5.6 dargestellt. Die Abkürzung Mess steht für Messung und Mod für Modell. Der dargestellte Fehler beschreibt jeweils den Durchschnitt über die resultierenden Fehler (in Betrag) der Messungen für eine Read-Probability mit unterschiedlicher Anzahl paralleler IO-Zugriffe (Diese Abkürzungen gelten ebenfalls für die Tabellen 5.7 bis 5.15.).

$Read_{Prob}$	**Paral-IOs 1; 2; 3**		**Paral-IOs 4**		**Paral-IOs 5**		**Paral-IOs 6**		**Paral-IOs 7**		**Paral-IOs 8**		**Fehler[1]**
	Mess	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mittel*
0,20	2,14	2,20	1,99	2,01	1,93	1,86	1,81	1,73	1,67	1,62	1,56	1,53	2,80 %
0,40	3,07	3,16	2,72	3,00	2,46	2,36	2,19	2,1	1,93	1,9	1,73	1,74	2,39 %
0,50	3,2	3,28	2,84	2,68	2,48	2,28	2,18	1,99	1,90	1,78	1,72	1,61	6,26 %
0,60	3,07	3,16	2,71	2,69	2,36	2,36	2,09	2,10	1,85	1,90	1,67	1,75	1,94 %
0,80	2,13	2,2	2,01	2,00	1,78	1,86	1,62	1,73	1,49	1,62	1,39	1,53	5,56 %

Tabelle 5.6 Bedienzeiten in ms für Variable Read/Write Load bei 100 % sequentieller IO-Verarbeitung

1. Mittlere Fehler über alle Messungen für eine Read-Probability (in Betrag)

Anhand der Messdaten kann man erkennen, dass zum einen eine Optimierung der Bedienzeit erst ab 4 parallelen IOs beginnt und nur bis zu 8 parallelen IOs vollzogen wird. Eine Optimierung der IOs bis zu 8 parallelen IOs stellt den gleichen Effekt dar, wie bereits für das Szenario Multi-IO-Load dargestellt. Weitere Messungen haben gezeigt, dass bei über 256 parallelen IOs keine weiteren Optimierungen durchgeführt werden bzw. durchführbar sind. Es ist zu vermuten, dass bei 100 % sequentieller Last eine minimale Bedienzeit bereits bei 8 parallelen IOs erreicht wird. Die Abweichung zwischen Modell und Messung liegt in diesem Lastszenario bei maximal 6,26 % (in Betrag).

Betrachtet man die Fälle mit 2 und 3 parallelen IOs, so kann angenommen werden, dass sich hierbei aufgrund von Overhead-Zeiten auf Seiten des HBA (Host Bus Adapters) oder des Betriebssystems nicht genügend Aufträge in der Queue befinden. Führt man Messungen mit einer IO-Größe von 32 KB durch, so beginnt eine Optimierung bereits ab 3 parallelen IOs, da das Betriebssystem oder der HBA durch die erhöhte Bedienzeit pro IO mehr Zeit besitzt, IO-Aufträge zur Festplatte zu senden.

100 % zufällig verteilte IO-Zugriffe

Wie bereits für die 100 % sequentiellen IO-Zugriffe beschrieben, werden in diesem Lastszenario zum einen die Read-Probability und zum anderen die Zahl der parallelen IOs variiert. Die erzielten Ergebnisse für diese Lastkomposition werden in Tabelle 5.7 dargestellt, wobei die Modellergebnisse mit Hilfe der noch folgenden Modellmodifikation errechnet wurden.

$Read_{Prob}$	Paral-IOs 1		Paral-IOs 3		Paral-IOs 5		Paral-IOs 7		Paral-IOs 8		Paral-IOs 10		Fehler[1]
	Mess	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mittel*
0,20	7,54	7,27	6,73	6,69	6,36	6,59	6,13	6,52	6,05	6,48	6,06	6,48	4,70 %
0,40	8,62	8,70	7,18	7,35	6,66	7,26	6,29	6,77	6,17	6,60	6,22	6,60	5,50 %
0,60	9,09	9,11	7,45	8,02	6,67	7,09	6,37	6,67	6,26	6,53	6,28	6,53	4,53 %
0,80	9,12	9,10	7,54	8,14	6,66	7,02	6,22	6,56	6,06	6,39	6,07	6,39	4,96 %

Tabelle 5.7 Bedienzeiten in ms für Read/Write Load-Mix bei 100 % zufällig verteilten IO-Zugriffen

1. Mittlere Fehler über alle Messungen für eine Read-Probability (in Betrag)

In dem hier zu betrachtenden Lastszenario mit 100 % zufällig verteilten IOs kann man anhand der Messdaten erkennen, dass die Bedienzeiten mit steigender Anzahl paralleler IOs optimiert werden. Die Optimierung wird von der Festplatte mittels implementierter Queueing-Strategien (siehe Kapitel 4) erreicht, die auch im Abschnitt Multi IO-Load beschrieben wurden. Die Modellierung dieses Lastszenarios wird somit über die IO-Bibliothek bestehend aus den Skalierungsfaktoren für Lese- und Schreibzugriffe durchgeführt. Die Modifikation des Modells zur Berücksichtigung paralleler IO-Verarbeitung besteht nun darin, dass mit Hilfe der Read-Probability die Zahl der parallelen Lese- und Schreibzugriffe berechnet wird und anhand derer die entsprechenden Skalierungsfaktoren der IO-Bibliothek entnommen werden. Die Zahl der parallelen Lese- ($Read_{Paral}$) und Schreibzugriffe ($Write_{Paral}$) werden dann wie folgt berechnet:

$$Read_{Paral} = IO_{Paral} \bullet Read_{Prob} \quad \text{und} \quad Write_{Paral} = IO_{Paral} \bullet (1 - Read_{Prob})$$

Wie auch bereits für Multi IO Load beschrieben, beendet die Festplatte die Optimierung der Festplattenzugriffe ab 8 parallelen IO-Zugriffen. Diese Eigenschaft, die auch in der IO-Bibliothek festgehalten ist, muss bei der Modellierung des zu betrachtenden Szenarios beachtet werden, d. h. die Bedienzeit wird beendet, wenn die Summe der parallelen Lese- und Schreibzugriffe 8 IO-Zugriffe erreicht hat. Die Optimierung beginnt wieder ab 256 parallelen IO-Zugriffen. In Tabelle 5.8 wird die Bedienzeit pro IO für 280 parallele IOs dargestellt.

$Read_{Prob}$	Paral.-IOs 280		Fehler[1]
	Mess	*Mod*	
0	5,56	5,51	-0,90 %
0,20	5,55	5,71	2,88 %
0,40	5,49	5,78	5,28 %
0,60	5,38	5,65	5,02 %
0,80	5,32	5,41	1,69 %
1	5,17	5,11	-1,16 %
Mittelwert (in Betrag):			2,82 %

Tabelle 5.8 Bedienzeiten in ms für Read/Write Load-Mix - Random

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Die dargestellten Ergebnisse wurden ebenfalls mit Hilfe der Skalierungsfaktoren für Lese- und Schreibzugriffe der IO-Bibliothek berechnet. Insgesamt erhält man in diesem Lastszenario (Tabelle 5.7 und Tabelle 5.8) eine maximale Abweichung zwischen Modell und Messung von 5,5 %.

5.3.4.3. Sequential/Random Load-Mix und parallele IO-Verarbeitung

Die folgende Untersuchung beschreibt das Basislastszenario Sequential/Random Load-Mix unter Verwendung paralleler IO-Verarbeitung. Es wird zwischen den Fällen 100 % lesende und 100 % schreibende IO-Zugriffe unterschieden.

100 % Lesezugriffe

In dem zu betrachtenden Lastszenario werden die Burst-Probability sowie die Zahl der parallelen IOs variiert. Die Bursts besitzen hierbei eine mittlere Anzahl von drei IO-Zugriffen ($RunLength_{Seq}$). Eine detaillierte Beschreibung des Sequential/Random Load-Mix befindet sich in Kapitel 4.

Die erzielten Ergebnisse für dieses Lastszenario werden in Tabelle 5.9 dargestellt, wobei die Modellergebnisse mit Hilfe der noch folgenden Modellmodifikation errechneten wurden. In Tabelle 5.10 wird die Bedienzeit pro IO für über 256 parallele IOs dargestellt. In diesem Fall wurden 300 parallele IOs generiert. Man kann anhand der gemessenen Bedienzeit erkennen, dass mit zunehmendem sequentiellen Anteil der IO-Last sowie steigendem Parallelitätsgrad der IO-Verarbeitung die Bedienzeiten pro IO-Zugriff sinken. Zum einen liegt dies in der geringen Positionierungszeit bei sequentiellen Zugriffen und zum anderen in der Optimierung der zufällig verteilten IO-Zugriffe hinsichtlich ihrer Position auf der Festplatte. Diese Optimierung erfolgt im Modell mit Hilfe der Skalierungsfaktoren der IO-Bibliothek. Hierzu muss berechnet werden, wie viele der parallelen IOs (IO_{Paral}) sequentiell ($SeqIO_{Paral}$) bzw. zufällig verteilt ($RandIO_{Paral}$) sind:

$$SeqIO_{Paral} = IO_{Paral} \bullet IO_{Seq} \quad \text{und} \quad RandIO_{Paral} = IO_{Paral} - SeqIO_{Paral}$$

$$\text{mit } IO_{Seq}[\%] = \frac{TotalBursts \bullet (RunLength_{Seq} - 1)}{TotalIOs} \quad \text{und } IO_{Paral} < 8 + RunLength_{Seq}$$

Der Parameter IO_{Seq} gibt den prozentualen Anteil der sequentiellen IOs an, wobei hierbei berücksichtigt wird, dass der erste IO-Zugriff eines Burst-Segments immer aus einem zufällig verteilten IO-Zugriff besteht.

Die Berechnung der Skalierungsfaktoren unterteilt sich in zwei Fälle. Zum einen können, wie beschrieben, die Skalierungsfaktoren für 2 bis ($8+RunLength_{Seq}$) parallele IOs berechnet werden. Wie bereits für das Lastszenario Multi IO Load beschrieben, stagniert die Bedienzeit ab 8 parallelen IOs, d. h. bei einem Lastszenario mit Burst-Segmenten bestehend aus drei IO-Zugriffen nach ca. 11 parallelen IOs und die Optimierung beginnt erst wieder ab 256 parallelen IOs. Es werden somit für die Fälle über 256 parallele IOs die Skalierungsfaktoren wie folgt berechnet:

$$RandIO_{Paral} = 256 + (IO_{Paral} - 256) \bullet (1 - IO_{Seq}) \quad \text{für } IO_{Paral} > 256$$

$Burst_{Prob}$	Paral-IOs 1		Paral-IOs 3		Paral-IOs 5		Paral-IOs 7		Paral-IOs 8		Paral-IOs 10		Fehler[1]
	Mess	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mittel*
0,20	6,83	6,57	6,38	6,44	5,72	5,46	5,24	5,06	5,07	4,92	4,91	4,71	3,29 %
0,40	5,35	5,17	5,28	5,17	4,93	4,68	4,36	4,29	4,37	4,19	4,36	4,10	3,70 %
0,60	4,47	4,27	4,48	4,27	4,28	4,15	4,02	3,77	3,86	3,68	3,85	3,57	5,06 %
0,80	3,82	3,66	3,81	3,66	3,82	3,66	3,68	3,41	3,55	3,30	3,40	3,17	5,58 %

Tabelle 5.9 Bedienzeiten in ms für Sequential/Random Load-Mix bei 100 % Lesezugriffen

1. Mittlere Fehler über alle Messungen für eine Burst-Probability (in Betrag)

Die Skalierungsfaktoren beeinflussen im Modell die Berechnung der Bedienzeit für Random-IOs. Die Bedienzeit für sequentielle Zugriffe wird im bisher beschriebenen Modell derart definiert, dass die mittlere Bedienzeit pro IO in einem Burst ermittelt wird. Ein Burst besteht hierbei aus einer mit $RunLength_{Seq}$ spezifizierten mittleren Anzahl an IO-Zugriffen, wobei der erste Zugriff einem zufällig verteilten IO entspricht. Dieser erste IO wird jedoch durch die Parallelität der Zugriffe mit optimiert. Es wird hierbei angenommen, dass die Festplatte SSTF-Verfahren (Shortest Seek Time First) anwendet ohne Berücksichtigung der Latenzzeit, so dass zur Berechnung der Bedienzeit für sequentielle Zugriffe die SeekTime mit dem soeben beschriebenen Skalierungsfaktor multipliziert wird:

$$SeekTi_{Seq}\,[ms] = \frac{Skal_{Read} \bullet (SeekTi_{Rand} + K)}{RunLength_{Seq}}$$

Wie bereits beschrieben, existiert für die Random-IOs bei den Lesezugriffen ein aus Messungen ermittelter Kalibrierungswert (K = 0,5 ms), der auch hierbei mit berücksichtigt wird.

$Burst_{Prob}$	Paral-IOs 300		Fehler[1]
	Mess	*Mod*	
0,20	3,88	3,81	-1,80%
0,40	3,38	3,30	-2,37%
0,60	2,94	2,96	0,68%
0,80	2,72	2,71	-0,37%
Mittelwert (in Betrag):			1,30%

Tabelle 5.10 Bedienzeiten in ms für Sequential/Random Load-Mix 100% Lesen

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

In diesem Lastszenario (Tabelle 5.9 und Tabelle 5.10) resultiert eine maximale Abweichung von 5,5 % zwischen Modell und Messung.

100% Schreibzugriffe

Wie bereits für die 100% lesenden IO-Zugriffe beschrieben, werden in diesem Lastszenario zum einen die Burst-Probability ($RunLength_{Seq}$ = 3) und zum anderen die Zahl der parallelen IOs variiert. Zur Modellierung der Schreibzugriffe bei variierendem Burst-Anteil und steigender Anzahl paralleler IO-Zugriffe werden die bereits für die Schreibzugriffe beschriebenen Methoden zur Berechnung der Skalierungsfaktoren verwendet. Es ergeben sich somit die in Tabelle 5.11 und Tabelle 5.12 dargestellten Mess- und Modell-Ergebnisse.

$Burst_{Prob}$	**Paral-IOs 1**		**Paral-IOs 3**		**Paral-IOs 5**		**Paral-IOs 7**		**Paral-IOs 8**		**Paral-IOs 10**		**Fehler[1]**
	Mess	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mess*	*Mod*	*Mittel*
0,20	4,77	4,74	5,28	4,48	4,99	4,40	4,83	4,34	4,76	4,33	4,76	4,25	9,58%
inkl. Overhead		4,74		4,98		4,90		4,84		4,83		4,75	1,87%
0,40	3,83	3,78	4,84	3,65	4,51	3,54	4,30	3,60	4,20	3,48	4,20	3,44	16,87%
inkl. Overhead		3,78		4,65		4,54		4,60		4,48		4,44	4,21%
0,60	3,20	3,17	4,41	3,11	4,10	2,99	3,91	2,96	3,81	2,94	3,81	2,92	21,33%
inkl. Overhead		3,17		4,11		3,99		3,96		3,94		3,92	3,41%
0,80	2,79	2,74	4,06	2,72	3,75	2,61	3,59	2,58	3,49	2,57	3,50	2,54	24,52%
inkl. Overhead		2,74		3,72		3,61		3,58		3,57		3,54	3,16 %

Tabelle 5.11 Bedienzeiten in ms für Sequential/Random Load-Mix bei 100 % Schreibzugriffen

1. Mittlere Fehler über alle Messungen für eine Burst-Probability (in Betrag)

Bei der Betrachtung der Messergebnisse ist insbesondere der Übergang von einem einzelnen IO-Zugriff zu drei parallelen IOs hinsichtlich der Bedienzeit auffällig. Wie anhand der Daten zu erkennen ist, steigt die Bedienzeit hierbei an, und zwar um einen konstanten Sockelbetrag, der sich in Abhängigkeit des sequentiellen Anteils der IO-Last ergibt. Modelliert man die Schreibzugriffe ohne Berücksichtigung des Overheads, so ergeben sich mittlere Abweichungen zwischen Modell und Messung von 10% bis 25 %. Berücksichtigt man hingegen die Overheads, indem ab drei parallelen IOs ein konstanter Sockelbetrag zu der modellierten Bedienzeit addiert wird (Output-Kalibrierung, [29]), so ergeben sich die in den Zeilen „inkl. Overhead“ der Tabelle 5.11 dargestellten Ergebnisse, d. h. Abweichungen zwischen 2% und 3,5 %. Die Overhead-Zeiten ergaben sich hierbei derart, dass für $Burst_{Prob}$ = 0,2 ein Overhead von 0,5 ms und für alle weiteren Fälle eine Overhead-Zeit von 1 ms verwendet wurde.

Die Overhead-Zeiten treten nur in der beschriebenen Lastzusammensetzung auf. Ein vergleichbarer Effekt wurde bereits in in Kapitel 4, Abschnitt 4.3.2. beschrieben, wo die Bedienzeit bei 100 % sequentieller Last für 100 % schreibende Zugriffe jedoch nicht signifikant anstieg.

$Burst_{Prob}$	Paral-IOs 300		Fehler[1]
	Mess	*Mod*	
0,20	4,20	3,96	-5,71 %
0,40	3,59	3,23	-10,03 %
0,60	3,18	2,76	-13,21 %
0,80	2,87	2,42	-15,86 %
Mittelwert (in Betrag):			11,16 %

Tabelle 5.12 Bedienzeiten in ms für Sequential/Random Load-Mix, 100 % Schreiben

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

In Tabelle 5.12 sind die Bedienzeiten des Modells ohne Overheadzeiten berechnet worden. Bei Berücksichtigung der Overheadzeiten, würden die Modellergebnisse über den Messergebnissen liegen und dann einen mittleren Fehler von 15,36 % ergeben.

5.3.4.4. Read/Write und Sequential/Random Load-Mix sowie parallele IO-Verarbeitung

Es folgt nun das letzte Lastszenario, in dem alle Basislasten parallelel betrieben werden, d. h. sowohl die Read- als auch die Burst-Probability werden zwischen 0 und 1 variiert und zusätzlich wird eine parallele IO-Verarbeitung durchgeführt. Hierzu wurden drei Szenarien definiert, die jeweils unterschiedliche Parametrisierungen der Read- und Burst-Probability unter paralleler IO-Verarbeitung beschreiben.

Szenario 1: Burst-Probability = 0,80; $RunLength_{Seq}$ = 2; Read-Probability = 0,20

In diesem Szenario werden mit einer Wahrscheinlichkeit von 0,80 Bursts generiert, die eine Größe von zwei IO-Zugriffen im Mittel besitzen, sowie werden mit einer Wahrscheinlichkeit von 0,20 Lesezugriffe erzeugt. Wie bereits beschrieben, ist hierbei zu beachten, dass in einem Burst kein Wechsel zwischen Lesen und Schreiben durchgeführt wird.

Bei der Modellierung der beschriebenen Last müssen zum einen die Overhead-Zeiten für den Read-Write-Mix und zum anderen die Berechnungsvorschriften für die parallelen IOs, unterteilt nach lesenden und schreibenden Zugriffen, zur Auswahl der Skalierungsfaktoren anhand der IO-Bibliothek wie folgt modifiziert werden.

Die Overhead-Zeiten für den Read-Write-Mix geben den Zeitbedarf zum Wechsel zwischen einem lesenden und einem schreibenden sequentiellen IO-Zugriff an (siehe Abschnitte 5.2.2.1. und 5.2.2.2.), wobei diese Definition nur IO-Lasten ohne Bursts und zwar bei 100 % sequentieller Verteilung der IO-Zugriffe repräsentiert. Betrachtet man hingegen eine Burst-Probability $\neq$ 0 % und Burst-Probability $\neq$ 100 %, so muss zum einen die RunLength zur Beschreibung der aufeinanderfolgenden IO-Zugriffe gleichen Typs (lesend oder schreibend) bei Auftreten eines Bursts um dessen Länge ($RunLength_{Seq}$) vergrößert werden. Zum anderen wird die Wahrscheinlichkeit, dass nach einem Burst ein weiteres Segment anderen Typs (lesend oder schreibend) entsteht, berücksichtigt, da nur in diesem Fall die Overhead-Zeit entsteht. Es folgt somit die Berechnung der Overhead-Zeit:

Für sequentiell lesende IO-Zugriffe:

$$OvhdR_{ReadProb} = \frac{2 \bullet Latency_{Rand} \bullet ((1 - Read_{Prob}) \bullet Burst_{Prob})}{RunLength_{Read} \bullet (1 + Burst_{Prob} \bullet RunLength_{Seq})}$$

Für sequentiell schreibende IO-Zugriffe:

$$OvhdW_{ReadProb} = \frac{2 \bullet Latency_{Rand} \bullet (Read_{Prob} \bullet Burst_{Prob})}{RunLength_{Write} \bullet (1 + Burst_{Prob} \bullet RunLength_{Seq})}$$

Der Ausdruck $(1\text{-}Read_{Prob}) \bullet Burst_{Prob}$ bzw. $Read_{Prob} \bullet Burst_{Prob}$ gibt hierbei die Wahrscheinlichkeit an, dass nach dem Burst ein neues Segment entgegengesetzten Typs (d. h. schreibend oder lesend) entsteht. Diese Berechnung der Overheads ist, wie bereits beschrieben, jedoch nur für den nicht 100 % sequentiellen Fall anwendbar, da ansonsten keine Bursts existieren. Weiterhin berücksichtigt der Ausdruck $(1+Burst_{Prob} \bullet RunLength_{Seq})$ für die Definition der $RunLength_{Read}$ bzw. $RunLength_{Write}$ die Wahrscheinlichkeit eines Bursts, bei dessen Auftreten sich die RunLength um die Länge des Bursts ($RunLength_{Seq}$) verlängert.

Neben der Betrachtung der Overhead-Zeiten wird ebenfalls eine Berechnung der parallelen und zufällig verteilten IO-Zugriffe (IO_{Rand}), die für die Spezifikation der Skalierungsfaktoren anhand der IO-Bibliothek notwendig sind, benötigt. Die Berechnung der parallelen IO-Zugriffe wird unterteilt in lesende und schreibende Operationen:

- *Lesend:* $IO_{Rand} = IO_{Paral} \bullet Read_{Rand}$

  $$\text{mit } Read_{Rand} = \frac{Bursts_{Read} + IO_{Read}}{TotalIOs} \text{ und } IO_{Paral} \leq 256$$

  Der Parameter $Read_{Rand}$ beschreibt den prozentualen Anteil an zufällig verteilten lesenden IO-Zugriffen (Random-IOs) an der Gesamtlast. Die Random-IOs ergeben sich zum einen aus den Bursts ($Bursts_{Read}$), da jeder Burst mit einem Random-IO beginnt, und zum anderen aus den separaten Random-IOs (IO_{Read}), die zwischen den Bursts liegen. Der Parameter $Bursts_{Read}$ gibt hierbei die Zahl der Bursts mit lesenden IO-Zugriffen an, so dass dieser berechnet wird durch:

  $$Bursts_{Read} = Read_{Prob} \bullet TotalBursts$$

  Alle weiteren Random-IOs werden durch den Wert IO_{Read} beschrieben, der wie folgt berechnet wird:

  $$IO_{Read} = Read_{Prob} \bullet (TotalIOs - RunLength_{Seq} \bullet (Bursts_{Read} + Bursts_{Write}))$$

  Der Parameter $Bursts_{Write}$ wird in der folgenden Beschreibung für Schreibzugriffe definiert. Übersteigt die Zahl der parallelen IOs den Wert 256, d. h. die maximale Anzahl an IO-Zugriffen, welche durch die Festplatte gepuffert werden kann, so wird die Zahl der parallelen und zufällig verteilten IO-Zugriffe wie folgt berechnet:

  $$IO_{Rand} = (IO_{Paral} - 256) \bullet Read_{Rand} + 256 \quad \text{für } IO_{Paral} > 256$$

- *Schreibend:* $IO_{Rand} = IO_{Paral} \bullet Write_{Rand}$

$$\text{mit } Write_{Rand} = \frac{Bursts_{Write} + IO_{Write}}{TotalIOs} \text{ und } IO_{Paral} \leq 256$$

In gleicher Form wie bereits für die lesenden IO-Zugriffe beschrieben, werden die parallelen zufällig verteilten und schreibenden IO-Zugriffe berechnet, deren prozentualer Anteil an der Gesamtlast durch den Parameter $Write_{Rand}$ beschrieben wird. Der Parameter $Bursts_{Write}$ gibt hierbei die Zahl der Bursts mit schreibenden IOs an und wird berechnet durch:

$$Bursts_{Write} = (1 - Read_{Prob}) \bullet (AnzBursts)$$

Wie bereits für die Schreibzugriffe beschrieben ergibt sich dann der Parameter IO_{Read}:

$$IO_{Read} = (1 - Read_{Prob}) \bullet (TotalIOs - BurstLength \bullet (ReadBursts + WriteBursts))$$

Ebenso vergleichbar mit den Lesezugriffen wird die Zahl der Random-IOs bei über 256 parallele IOs berechnet:

$$IO_{Rand} = (IO_{Paral} - 256) \bullet Write_{Rand} + 256 \quad \text{für } IO_{Paral} > 256$$

In Tabelle 5.13 werden die Mess- und Modellergebnisse für dieses Lastszenario dargestellt. Die Modellergebnisse wurden hierbei unter Berücksichtigung der beschriebenen Modellmodifikationen berechnet.

IO_{Paral}	Szenario 1		Fehler[1]
	Mess	*Mod*	
1	4,24	4,34	2,36 %
5	4,74	4,13	-12,87 %
mit Ovhd.	4,74	4,93	4,01 %

Tabelle 5.13 Bedienzeiten in ms für Load-Mix, Szenario 1

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Wie bereits in Abschnitt 5.3.4.3. (Schreibzugriffe) beschrieben, entsteht bei variierendem sequentiellen Anteil ein Overhead, der mit Hilfe der Methode der Output-Kalibrierung berücksichtigt wurde. Bei einer Burst-Probability von 80 % existiert somit ein Overhead von 1 ms bei 100 % Schreibzugriffen. Da der Overhead nur bei Schreibzugriffen auftritt, wird dieser mit (1 - $Read_{Prob}$) multipliziert, d. h. der Wahrscheinlichkeit für einen Schreibzugriff, so dass in diesem Fall eine Overheadzeit von 0,8 ms resultiert.

Szenario 2: Burst-Probability = 0,20; $RunLength_{Seq}$ = 2; Read-Probability = 0,80

Mit Hilfe der für das Szenario 1 beschriebenen Algorithmen wurden die in Tabelle 5.14 dargestellten Modellergebnisse berechnet.

IO_{Paral}	Szenario 2		Fehler[1]
	Mess	*Mod*	
1	7,81	7,678	-1,69 %
5	6,20	6,28	1,29 %
mit Ovhd.	6,20	6,38	2,90 %

Tabelle 5.14 Bedienzeiten in ms für Load-Mix, Szenario 2

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Der Overhead wird gemäß der Beschreibung für Szenario 1 errechnet. Da in diesem Fall nur eine Burst-Probability von 0,20 vorliegt, beträgt der Overhead nur 0,5 ms bei 100 % Schreibzugriffen (siehe 5.3.4.3.), d. h. hier 0,1 ms.

Szenario 3: Burst-Probability = 0,40; $RunLength_{Seq}$ = 2; Read-Probability = 0,40

Mit Hilfe der für das Szenario 1 beschriebenen Algorithmen wurden die in Tabelle 5.15 dargestellten Modell-Ergebnisse berechnet.

IO_{Paral}	Szenario 3		Fehler[1]
	Mess	*Mod*	
1	6,14	5,88	4,14 %
5	5,53	5,45	-1,32 %
mit Ovhd.	5,53	6,05	9,40 %

Tabelle 5.15 Bedienzeiten in ms für Load-Mix, Szenario 3

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Der Overhead beträgt nach Abschnitt 5.3.4.3. hierbei 0,6 ms. Betrachtet man die Modellergebnisse, so ist klar erkennbar, dass der Overhead zu hoch gewählt wurde. Wählt man einen Overhead von 0,75 ms (für 100 % Schreibzugriffe), so erhält man in diesem Fall einen zusätzlichen Zeitaufwand von 0,45 ms und ein daraus resultierenden Fehler von 6,6 %. Die Wahl von 0,75 ms basiert auf der Berechnung, dass bei einer Burst-Probability von 0,20 eine Overhead-Zeit von 0,5 ms und bei einer Burst-Probability von 0,60 ein Overhead von 1 ms verwendet wird, so dass 0,75 ms einen Mittelwert darstellt. Weiterhin erkennt man, dass auch in Abschnitt 5.3.4.3. mit einer Overhead-Zeit von 1 ms ein in Relation zu den anderen Szenarien höherer Fehler mit zu hohen Modellbedienzeiten auftritt, so dass die Verwendung einer niedrigeren Overhead-Zeit plausibel erscheint.

5.3.5. Zusammenfassung

In diesem Kapitel wurden für verschiedene IO-Lastszenarien Bedienzeiten, die anhand von realen Messungen sowie mittels des in Abschnitt 5.2. beschriebenen Festplattenmodells berechnet wurden, verglichen. Die IO-Lastszenarien konnten hierbei unterteilt werden in die Basislasten, die in Kapitel 4 definiert wurden, d. h. Simple Load, Sequential/Random Load-Mix und Read/Write Load-Mix sowie eine Kombination der verschiedenen Basislasten. Insbesondere bei

der Betrachtung der kombinierten IO-Lasten haben sich Modellerweiterungen bzw. -modifikationen ergeben, die jedoch keinen Einfluss auf die Modellierung der Basislasten besitzen.

Bei der Betrachtung der aufgetretenen Fehler, d. h. der Abweichungen zwischen Modell und Messung, kann man erkennen, dass für die Basislasten eine relativ geringe Differenz von 3 % bis maximal 5 % aufgetreten ist. Im Vergleich hierzu traten bei der Kombination der IO-Lastszenarien teilweise Fehler zwischen 5 % und maximal 8 % auf. Insbesondere in Abschnitt 5.3.4.3. bei der Betrachtung des Sequential/Random Load-Mix und paralleler IO-Verarbeitung im Fall von 100 % schreibenden IO-Zugriffen wurden Abweichungen von über 20 % festgestellt, die jedoch mit Hilfe der Output-Kalibrierung auf 5 % gesenkt werden konnten. Bei der Betrachtung von über 256 parallelen IO-Zugriffen wurden im Durchschnitt Fehler von ca. 11 % festgestellt.

Wie bereits in Kapitel 4 beschrieben, stellen die Basislasten die Grundbausteine für IO-Lasten dar, die im Einzelfall eine relativ geringe Komplexität aufweisen und somit auch für die Modellierung einen geringen Fehler besitzen. Kombiniert man hingegen die verschiedenen Basislasten, so erhält man auf der einen Seite einen höheren Realitätsgrad für die IO-Last, auf der anderen Seite aber auch eine höhere Komplexität und somit einen höheren Fehler. Zur Bewertung der dargestellten Fehler kann man die von Lazowska [63] beschriebenen Toleranzen zur Validierung von Auslastungen verwenden (0 % - 10 %), da die Definition der Auslastung auf dem Durchsatz und der Bedienzeit einer Last beruht und für die beschriebene Modellierung der Durchsatz vorgegeben wurde. Aufgrund dessen kann man die beschriebenen Modellergebnisse im Vergleich zu den realen Messungen als gute Approximation der Bedienzeit bezeichnen.

5.4. Erstellung eines Warteschlangenmodells mit Hilfe der Festplatten-Bedienzeit

In den bisherigen Abschnitten wurden Algorithmen zur Berechnung der Bedienzeit pro IO-Zugriff anhand von Festplattenparametern und Workload-Charakterisierungen für verschiedene IO-Lastszenarien dargestellt. Wie bereits in Kapitel 4 beschrieben, wurden in den betrachteten Lastszenarien Ankunftsströme der IO-Zugriffe ohne Zwischenankunftsabstand betrachtet, d. h. in Abhängigkeit der Parallelität der IO-Verarbeitung wird nach Beendigung eines IO-Zugriffs der nachfolgende IO unmittelbar erzeugt. In dem nun folgenden Abschnitt dieses Kapitels soll ein auf den berechneten Bedienzeiten basierendes analytisches Warteschlangenmodell präsentiert werden, das unter Verwendung unterschiedlicher Ankunftsströme der IO-Zugriffe die Leistungsmaße Antwortzeiten, Auslastungen und Warteschlangenlängen (Populationen) berechnet. Zur Analyse der verschiedenen Ankunftsströme werden mit dem IO-Benchmark DBench (siehe Kapitel 4) Messungen mit verschiedenen Zwischenankunftsabständen für die einzelnen IO-Zugriffe unter Verwendung der Negativ-Exponential-Verteilung durchgeführt, wobei hierzu verschiedene Lastszenarien betrachtet werden. Die Lastszenarien unterscheiden sich hierbei in der Komplexität der Lastzusammenstellung.

5.4.1. Warteschlangenmodellierung und Messung im Vergleich

Für die nun folgenden Lastszenarien wird zum Benchmarking und Monitoring das in Kapitel 4 beschriebene Testsystem verwendet. Ziel ist es, vergleichbare IO-Lastszenarien, wie bereits in Kapitel 4 beschrieben, zu definieren, die zusätzlich Zwischenankunftsabstände für die IO-Zugriffe berücksichtigen. Die IO-Größe soll daher 8 KB betragen, wodurch eine Bedienzeit von maximal 9 ms pro IO-Zugriff resultiert (siehe Kapitel 4). Die Zwischenankunftsabstände müssten somit für Festplattenauslastungen im Bereich von 30 % bis 60 % in Abhängigkeit des Lastszenarios überwiegend zwischen 1 ms und 20 ms liegen.

Das Testsystem basiert auf einer Intel-Architektur (Pentium III) und Solaris 2.8 und besitzt somit nach [114] eine minimale Zeitauflösung von 10 ms. Zeiten unter 10 ms werden vom System automatisch auf 0 ms abgerundet. Lastszenarien mit einer IO-Größe von 8 KB und negativ-exponential verteilten Zwischenankunftsabständen sind somit aufgrund der unzureichenden Zeitauflösung nicht realisierbar. Solaris bietet für SPARC-Architekturen einen auf Threads basierenden High Resolution-Timer an, der eine Genauigkeit von unter 1 ms garantiert. Im Rahmen dieser Arbeit stand ein entsprechender Rechner nicht zur Verfügung, so dass zur Lösung des Problems die Bedienzeiten der IO-Zugriffe mittels Erhöhung der IO-Größe angehoben wurden. Ab einer IO-Größe von 512 KB liegen die Bedienzeiten bei 50 ms und somit der Hauptanteil der Zwischenakunftsabstände zwischen 10 ms und 100 ms, so dass für die Negativ-Exponential-Verteilung ein im Hinblick auf die Zeitauflösung des Testsystems ausreichend großes Intervall vorliegt.

Es folgt eine Beschreibung der IO-Lastszenarien und der damit verbundenen Messdaten sowie der zugehörigen Modellergebnisse. Zur Modellierung der verschiedenen Ankunftsströme werden für die Messungen unterschiedliche Zwischenankunftsabstände vorgegeben, so dass sich in jeder Messung ein entsprechender Durchsatz an IO-Zugriffen ergibt. Die dabei erzielten Durchsätze werden für die Modellparametrisierung verwendet. Aufgrund der Negativ-Exponential-Verteilung der Zwischenankunftsabstände sowie der in Kapitel 4 beschriebenen Abhängigkeit der Bedienzeit von der IO-Lastcharaktierisierung, wird zur Modellierung der Festplatte ein M/G/1-Modell verwendet. Es werden hierzu die folgenden zwei IO-Lastszenarien betrachtet:

- **Lastszenario 1:** Burst-Probability = 0
  $\text{RunLength}_{Seq} = 1$
  Read-Probability = 1
  IO-Größe = 512 KB

  In dem beschrieben Lastszenario 1 mit Burst-Probability = 0 und $\text{RunLength}_{Seq} = 1$ wird eine 100 % Random Load generiert, d. h. alle IO-Zugriffe werden über alle logischen Blöcke der Festplatte gleichverteilt. Weiterhin werden nur Lesezugriffe (Read-Probability = 1) mit einer Größe von 512 KB erzeugt und zwischen den IO-Zugriffen existiert jeweils eine mittlere Wartezeit (Delay), die der Negativ-Exponential-Verteilung unterliegt. Die erzielten Messergebnisse werden in Tabelle 5.16 dargestellt. Die Bezeichnung „Delay“ steht hierbei für den im Benchmark geforderten mittleren Abstand zwischen zwei IO-Zugriffen, „Durchs.“ für den mit dem vorgegebenen Delay erzielten Durchsatz, „RespTi“ für die Antwortzeit und „SvcTi“ für die Bedienzeit jeweils pro IO-Zugriff sowie „Queue“ für die Zahl der IO-Aufträge sowohl in der Active Queue als auch in der Wait Queue (siehe Kapitel 4) und „Util.“ für die Auslastung der Festplatte. Für die Modellierung wird mit „VK“ der Variationskoeffizient für die Bedienzeit SvcTi beschrieben.

  Vergleicht man die in Tabelle 5.16 dargestellten Durchsätze und die in dem Benchmark geforderten Delays, so ist zu erkennen, dass die tatsächlich erreichten Durchsätze minimal unter den gewünschten Werten (Durchsatz = 1/Delay) liegen. Betrachtet man weiterhin die Bedienzeiten pro IO-Zugriff, so liegen diese nahezu konstant bei 54,6 ms. Verwendet man für das hier zu betrachtende Lastszenario das bereits in Abschnitt 5.2. beschriebene Festplattenmodell zur Bedienzeitberechnung, so ergibt sich eine Bedienzeit von 27,84 ms pro IO-Zugriff. Die gemessene Bedienzeit entspricht somit nahezu dem Doppelten der vom Festplattenmodell errechneten Zeit. Es ist daher anzunehmen, dass aufgrund der in diesem Lastszenario gewählten IO-Größe ein nicht typisches Festplattenverhalten resultiert.

Messung						Modell (M/G/1)			Fehler[1]
Delay [ms]	Durchs. [IO/s]	RespTi [s]	SvcTi [s]	Queue [Anz.]	Util. [%]	RespTi [s]	Queue [Anz.]	VK	Fehler RespTi [%]
90	10,7	0,112	0,0548	1,22	59 %	0,114	1,22	0,72993	1,79 %
100	9,6	0,101	0,0549	0,99	53 %	0,103	0,99	0,76199	1,98 %
110	8,8	0,095	0,0548	0,84	48 %	0,096	0,84	0,77205	1,05 %
120	8,1	0,090	0,0546	0,74	44 %	0,091	0,74	0,83930	1,11 %
130	7,4	0,085	0,0544	0,64	40 %	0,087	0,64	0,88159	2,35 %
140	6,9	0,082	0,0543	0,58	38 %	0,084	0,58	0,92081	2,44 %

Tabelle 5.16 Modellergebnisse für Lastszenario 1 mit verschiedenen VK und SvcTi

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Anhand von verschiedenen Messungen konnte festgestellt werden, dass bei aktiviertem Cache zu Beginn einer Messung mit Lastszenario 1 die vom Festplattenmodell berechnete Bedienzeit existiert, jedoch im weiteren Verlauf des Benchmarks die doppelte Bedienzeit pro IO-Zugriff entsteht. Es ist daher sehr wahrscheinlich, dass z. B. die Caching-Strategien der Festplatte ab einer IO-Größe von 512 KB nicht mehr eingesetzt werden können. Aufgrund des nicht erklärbaren Festplattenverhaltens wurde in diesem Lastszenario der Festplatten-Cache deaktiviert, so dass bei der Messung direkt von Beginn an die in Tabelle 5.16 dargestellten Bedienzeiten resultieren. Aufgrund des deaktivierten Cache wird in dem hier zu betrachtenden Lastszenario keine Lastzusammenstellung mit Schreibzugriffen bertrachtet, da ohne Cache die Schreibzugriffe hinsichtlich ihrer Verarbeitungsstrategie und der daraus resultierenden Bedienzeit identisch mit den Lesezugriffen sind.

Das Festplattenverhalten und die resultierenden hohen Bedienzeiten pro IO-Zugriff werden durch IO-Größen ab 512 KB verursacht. Das dargestellte Lastszenario ist daher mit dem Festplattenmodell aus Abschnitt 5.2. zur Bedienzeitberechnung nicht abbildbar. Es werden somit für das M/G/1-Modell die mit der Messung ermittelten Bedienzeiten und Durchsätze verwendet, so dass die Auslastungen der Festplatte vom Modell immer exakt berechnet werden und somit nicht explizit als Modellergebnisse beschrieben werden.

Die mit dem M/G/1-Modell erzielten Modellergebnisse werden in Tabelle 5.16 dargestellt. Jede Messreihe wurde mittels des Variationskoeffizienten anhand der Queue-Länge (Population) kalibriert. Für die Antwortzeit resultiert somit ein sehr geringer Fehler, wobei die prozentuale Abweichung zwischen Modell und Messung unter 3 % liegt.

Bisher wurde für jede Modifikation des Zwischenankunftsabstands der IO-Zugriffe (Delay) das Modell mit Hilfe des Variationskoeffizienten kalibriert. Da ein derartiges Vorgehen einen sehr hohen Kalibrierungsaufwand besitzt, ist ein Variationskoeffizient für alle Messreihen des Lastszenarios wünschenswert. Es wird hierzu der Mittelwert aller Werte VK gebildet. Weiterhin wird auch der Mittelwert über alle Bedienzeiten berechnet. Es wird somit die Bedienzeit SvcTi im Mittel auf 0,0546 s und der Variationskoeffizient VK = 0,81761 gesetzt. Unter Verwendung dieser Mittelwerte wird das M/G/1-Modell für alle Messungen wiederholt gelöst und es resultieren die in Tabelle 5.17 dargestellten Ergebnisse.

Messung				Modell		Fehler[1]	
Durchs. [IO/s]	**RespTi [s]**	**Queue [Anz.]**	**Util. [%]**	**RespTi [s]**	**Queue [Anz.]**	**Fehler RespTi [%]**	**Fehler Queue [%]**
10,7	0,112	1,22	59 %	0,119	1,27	6,25 %	4,10 %
9,6	0,101	0,99	53 %	0,105	1,01	3,96 %	2,02 %
8,8	0,095	0,84	48 %	0,097	0,85	2,11 %	1,19 %
8,1	0,090	0,74	44 %	0,091	0,73	1,11 %	-1,35 %
7,4	0,085	0,64	40 %	0,085	0,63	0,00 %	-1,56 %
6,9	0,082	0,58	38 %	0,082	0,57	0,00 %	-1,72 %
					Mittelwert (in Betrag):	2,24 %	1,99 %

Tabelle 5.17 Modellergebnisse für Lastszenario 1 mit mittlerem VK und SvcTi

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Der in Tabelle 5.17 dargestellte Fehler beschreibt die prozentuale Abweichung der Antwortzeit und der Queue zwischen Modell und Messung. Der Fehler für die Antwortzeit beträgt im Mittel 2,24 % und maximal 6,25 %. Nach Lazowska [63] sollte der Fehler hinsichtlich der Antwortzeit für Modelle mit nur einer Lastkette unter 20 % liegen. Betrachtet man weiterhin die Modellergebnisse von Shriver unter Verwendung von Poisson-Ankunftsströmen und einer nahezu identischen Lastzusammensetzung [100], so liegen diese im Mittel bei 15 % und maximal bei 28 %. Das M/G/1-Modell liegt somit sowohl unter der 20 %-Schranke von Lazowska als auch unter den 15 % von Shriver. Es ist hierbei zu beachten, dass sowohl die Durchsätze als auch die Bedienzeiten für die Modellparametrisierung der Messung entnommen worden sind, da ab einer IO-Größe von 512 KB ein nicht typisches Festplattenverhalten resultierte.

- **Lastszenario 2:** Burst-Probability = 0,10
  $RunLength_{Seq}$ = 3
  Read-Probability = 1
  IO-Größe = 512 KB

Für Lastszenario 2 wird mit Burst-Probability = 0,10 und $RunLength_{Seq}$ = 3 eine IO-Last generiert, die aus Bursts mit drei zusammenhängenden IO-Zugriffen besteht, die mit einer Wahrscheinlichkeit von 10 % erzeugt werden. Weiterhin existieren nur Lesezugriffe (Read-Probability = 1) mit einer Größe von 512 KB und zwischen den IO-Zugriffen existiert jeweils eine mittlere Wartezeit (Delay), die, wie bereits für Lastszenario 1 beschrieben, der Negativ-Exponential-Verteilung unterliegt. Zu beachten ist hierbei, dass für IO-Zugriffe in einem Burst keine Delays erzeugt werden. Die erzielten Messergebnisse werden in Tabelle 5.18 dargestellt. Die Bezeichnungen der Tabelle können dem Abschnitt Lastszenario 1 entnommen werden.

Die in diesem Lastszenario erzielten Durchsätze sind nicht wie bei Lastszenario 1 mit den geforderten Delays (Durchsatz = 1/Delay) vergleichbar, da in diesem Lastszenario zwischen in einem Burst befindlichen IO-Zugriffen kein Delay erzeugt wird. Wie bereits in Lastszenario 1 liegen auch in diesem Fall die Bedienzeiten pro IO-Zugriff vergleichbar hoch und es wurde ohne Cache gemessen, so dass eine Lastzusammenstellung mit Schreibzugriffen keine Veränderung der Bedienzeiten ergeben hätte. Eine Messung mit aktivier-

tem Cache liefert vergleichbare Ergebnisse wie in Lastszenario 1. Aufgrund der Burst-Anteile an der Last hat sich die Bedienzeit um ca. 4 ms verringert. Vergleicht man diesen Wert mit der errechneten Bedienzeit des Festplattenmodells in Abschnitt 5.2. (25,92 ms), so ist auch in diesem Fall ein Faktor 2 zwischen Messung und Modell zu erkennen. In Tabelle 5.18 werden weiterhin die Modellergebnisse für das M/G/1-Modell dargestellt. Für jede Messreihe wurde anhand der Population das Modell mit Hilfe des Variationskoeffizienten kalibriert. Aufgrund der komplexeren Laststruktur steigen die Abweichungen zwischen Messung und Modell im Vergleich zu Lastszenario 1 deutlich an und es resultiert ein maximaler Fehler von 8,03 %.

Messung						Modell			Fehler[1]
Delay [ms]	**Durchs. [IO/s]**	**RespTi [s]**	**SvcTi [s]**	**Queue [Anz.]**	**Util. [%]**	**RespTi [s]**	**Queue [Anz.]**	**VK**	**Fehler RespTi [%]**
110	11,6	0,177	0,0503	2,20	58 %	0,190	2,20	1,72172	7,34 %
120	10,8	0,167	0,0503	1,90	54 %	0,176	1,90	1,78926	5,39 %
130	9,8	0,151	0,0501	1,59	49 %	0,162	1,59	1,90929	7,28 %
140	9,4	0,146	0,0501	1,47	46 %	0,157	1,47	1,94547	7,53 %
150	8,7	0,137	0,0502	1,29	43 %	0,148	1,29	2,01185	8,03 %
160	8,1	0,131	0,0501	1,14	40 %	0,141	1,14	2,07431	7,63 %

Tabelle 5.18 Modellergebnisse für Lastszenario 2 mit verschiedenen VK und SvcTi

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Auch für dieses Lastszenario wird für alle Messreihen das M/G/1-Modell mit dem mittleren Variationskoeffizienten und der mittleren Bedienzeit neu berechnet, so dass für das Modell eine Bedienzeit SvcTi von 50,18 ms und für den Variationskoeffizienten VK ein Wert von 1,90865 angenommen wird. Berechnet man das M/G/1-Modell mit Hilfe der Mittelwerte VK und SvcTi, so erhält man die in Tabelle 5.19 dargestellten Modellergebnisse.

Messung				Modell		Fehler[1]	
Durchs. [IO/s]	**RespTi [s]**	**Queue [Anz.]**	**Util. [%]**	**RespTi [s]**	**Queue [Anz.]**	**Fehler RespTi [%]**	**Fehler Queue [%]**
11,6	0,177	2,20	58 %	0,212	2,46	19,77 %	11,82 %
10,8	0,167	1,90	54 %	0,188	2,03	12,57 %	6,84 %
9,8	0,151	1,59	49 %	0,163	1,60	7,95 %	0,63 %
9,4	0,146	1,47	46 %	0,154	1,45	5,48 %	-1,36 %
8,7	0,137	1,29	43 %	0,140	1,22	2,19 %	-5,43 %
8,1	0,131	1,14	40 %	0,130	1,05	-0,76 %	-7,89 %
					Mittelwert (in Betrag):	8,12 %	5,66 %

Tabelle 5.19 Modellergebnisse für Lastszenario 2 mit mittlerem VK und SvcTi

1. Prozentuale Abweichung der Modellergebnisse von den Messergebnissen

Die Modellergebnisse besitzen für die berechneten Antwortzeiten einen mittleren Fehler von 8,12 % und für die berechneten Queue-Längen von 5,66 %. Gemessen an den Toleranzgrenzen von Lazowska [63] mit 20 % und Shriver [100] mit 15 % besitzen die Antwortzeiten eine geringe Abweichung zur realen Messung. Wie bereits für Lastszenario 1 beschrieben, ist auch hier zu beachten, dass die Festplatte ein nicht typisches Verarbei-

tungsverhalten aufgrund der IO-Größe von 512 KB aufweist, so dass für das Modell die Durchsätze und Bedienzeiten resultierend aus den Messungen verwendet wurden.

Zusammengefasst weisen die beiden beschriebenen IO-Lastszenarien einen maximalen Fehler von 19,77 % für die Antwortzeit auf. Sowohl im Vergleich zu Shriver als auch zu Lazowska ist die Güte der Modellergebnisse gut, wobei die bereits beschriebenen Einschränkungen zur Modellparametrisierung zu berücksichtigen sind.

5.4.2. Integration des Festplattenmodells in den WLPSizer

In den bisherigen Abschnitten wurde ein Festplattenmodell zur Berechnung der Bedienzeit pro IO-Zugriff und die Einbindung der berechneten Bedienzeit in ein Warteschlangenmodell beschrieben. Abschließend folgt nun die Darstellung eines Konzepts zur Einbindung des Festplattenmodells in das bereits beschriebene Modellierungswerkzeug WLPSizer (siehe Kapitel 2).

Der WLPSizer verwendet zur Modellierung der SAP R/3-Systeme die Klasse der BCMP-Netze, so dass das Warteschlangenmodell zur Abbildung der Festplatte den Voraussetzungen der BCMP-Netze entsprechen muss, d. h. mit negativ-exponential verteilten Zwischenankunftsabständen und Bedienzeiten ([4], [35]). Die BCMP-Netze ermöglichen eine Modellierung unter Verwendung verschiedener Auftragsklassen sowie unterschiedlicher Bedienstrategien. Eine den BCMP-Netzen zugehörige Bedienstrategie ist die lastabhängige Bedienung, d. h. in Abhängigkeit der Auftragsanzahl an der zu betrachtenden Station wird eine entsprechende Bediengeschwindigkeit verwendet. Diese Bedienstrategie stellt somit eine adäquate Abbildung einer Festplatte dar, denn die in Kapitel 4 beschriebenen Optimierungsstrategien einer Festplatte (z. B. SSTF) führen in Abhängigkeit der Anzahl der parallelen IO-Zugriffe eine Reduktion der Zugriffszeiten durch, indem die Zugriffspfade auf die Platten mittels Sortierung der IO-Aufträge optimiert werden, so dass die Zugriffszeit pro IO-Auftrag sinkt und sich somit die Bediengeschwindigkeit der Festplatte erhöht.

Das Modellierungswerkzeug WLPSizer verwendet den Warteschlangenlöser TOTO zur Lösung der Modelle (siehe Kapitel 2). TOTO basiert auf der Klasse der BCMP-Netzwerke und bietet einen Stationstyp „Queue Dependent Station“ (QD-Station) zur lastabhängigen Bedienung von Aufträgen an. Typischerweise werden mit Hilfe der QD-Station Multiprozessor-Server modelliert, jedoch wird in diesem Fall die QD-Station zur Abbildung einer Festplatte verwendet.

Zur Integration des Festplattenmodells in den WLPSizer müssen die in Abschnitt 5.2. beschriebenen Algorithmen in den WLPSizer implementiert werden, so dass dieser sowohl die Spezifikation der IO-Last als auch die Bedienzeitberechnung durchführen kann. Weiterhin wird für die Modellbildung mit TOTO die Festplatte als Queue Dependent Station beschrieben. Die Bediengeschwindigkeit der QD-Station wird mittels sog. Speed-Up-Faktoren definiert, d. h. es werden für verschiedene parallele IO-Auftragsanzahlen die zugehörigen Bediengeschwindigkeiten vorgegeben. Die Speed-Up-Faktoren der QD-Station werden mit Hilfe des in Abschnitt 5.2. beschriebenen Festplattenmodells ermittelt, d. h. es werden für die möglichen Parallelitätsgrade der IO-Zugriffe die Bedienzeiten berechnet. Die jeweilige prozentuale Veränderung der Bedienzeit wird als Speed-Up-Faktor dargestellt, wobei z. B. für eine Bedienzeitreduzierung von 20 % ein Speed-Up-Faktor von 1,2 (Speed-Up-Faktor = 1 entspricht dem Fall ohne Parallelität) gesetzt wird, da die Station die Aufträge zu 20 % schneller verarbeiten kann. Als Beispiel werden in Tabelle 5.20 für das Lastszenario Multi IO Load (100 % Read) aus Kapitel 4 die aus den Messungen resultierenden Bedienzeiten und Speed-Up-Faktoren für die QD-Station beschrieben.

Anz. paral. IOs	SvcTi [ms]	Speed Up Faktor
1	9,12	1
2	9,12	1
3	7,76	1,15
4	7,17	1,21
5	6,81	1,25
6	6,52	1,29

Tabelle 5.20 Speed-Up-Faktoren für die QD-Station (Multi IO Load, 100 % Read)

Die für das WLPSizer-Modell notwendige IO-Lastbeschreibung wird mittels der in Kapitel 2 und 3 beschriebenen Konzepte realisiert. Es wird somit in Relation zur Datenbankaktivität DBSU eine feste Anzahl an IO-Zugriffen definiert. Die IO-Zugriffe besitzen die anhand des Festplattenmodells berechnete Bedienzeit, die eine Lastsituation ohne parallele IO-Zugriffe beschreibt. Die aufgrund von IO-Parallelität resultierende Optimierung der Bedienzeit wird durch die bereits beschriebenen Speed-Up-Faktoren der QD-Station nachgebildet. Eine Unterscheidung zwischen lesenden und schreibenden IO-Zugriffen wird hierbei nicht durchgeführt, da das Monitoring-Werkzeug SIOSTAT zwar hinsichtlich der Durchsätze zwischen lesenden und schreibenden Zugriffen unterscheiden kann, jedoch nicht für die Bedienzeit und Antwortzeit (siehe Kapitel 4).

Eine Integration des Festplattenmodells in den WLPSizer ist somit möglich. Zu beachten ist jedoch, dass für das Warteschlangenmodell die Klasse der BCMP-Netze verwendet wird. Die QD-Station für die Festplatte kann daher nach Kendall mit M/M/QD beschrieben werden, d. h. im Gegensatz zu dem in Abschnitt 5.4.1. dargestellten Modell wird hierbei keine Generelle Verteilung sondern die Negativ-Exponential-Verteilung für die Bedienzeiten angenommen. Der WLPSizer wird somit nicht exakt die gleichen Modellergebnisse, wie in Abschnitt 5.4.1. beschrieben, liefern.

KAPITEL

6

ZUSAMMENFASSUNG UND AUSBLICK

Das Kapitel beschreibt eine Zusammenfassung der in dieser Arbeit erzielten Ergebnisse und liefert einen Ausblick auf weitere mögliche Themen aufbauend auf dieser Arbeit.

6.1. Zusammenfassung der Ergebnisse

Die Arbeit umfasst die beiden Themenschwerpunkte SAP R/3-Kapazitätsplanung in Kapitel 2 und Modellierung von Speichersubsystemen in Kapitel 4 und 5. Es wurde mit Hilfe der in Kapitel 3 dargestellten IO-Lastcharakterisierung ein Konzept vorgestellt, das eine Integration des IO-Modells in die SAP R/3-Kapazitätsplanung und des hierzu verwendeten Modellierungswerkzeugs WLPSizer ermöglicht und somit die Basis für eine IO-Prognose im Anwendungsfall SAP R/3 bietet.

Sowohl ein Vorgehensmodell mit Methodiken und Werkzeugen zur Messung, Analyse und Modellierung für die SAP R/3-Kapazitätsplanung als auch ein analytisches Festplattenmodell mit der beschriebenen IO-Lastcharakterisierung wurden erarbeitet. Das Festplattenmodell besitzt die in Kapitel 5 geschilderten Einschränkungen für die Warteschlangenmodellierung. Die Zielsetzung der Arbeit, Messung und Modellierung von SAP R/3- und Storage-Systemen, in diesem Fall Festplatten, wurde somit erreicht. In den folgenden beiden Abschnitten werden die für beide Themenschwerpunkte bereits existierenden Lösungen und Vorgehensweisen und die in dieser Arbeit erlangten Ergebnisse zusammengefasst dargestellt und gegenübergestellt.

6.1.1. SAP R/3-Kapazitätsplanung

Im Rahmen dieser Arbeit wurden Methoden und Werkzeuge zur Messung, Analyse und Aggregation von R/3-Lasten sowie zur Modellierung von SAP R/3-Systemen entwickelt. Die Anwendung der beschriebenen Methoden und Werkzeuge wurde auf Basis eines definierten Vorgehensmodells zur SAP R/3-Kapazitätsplanung, das sich bereits in zahlreichen erfolgreichen Einsätzen in Industrieprojekten bewährt hat, dargestellt. Anhand von Fallstudien und Beispielen wurden aktuelle Problemstellungen in der SAP R/3-Kapazitätsplanung, wie z. B. Release-Wechsel und Lastzuwächse oder auch Performanceprognosen für beispielsweise das Business Information Warehouse, analysiert und Lösungen auf Basis der beschriebenen Methoden und

Werkzeuge präsentiert. Es folgt zunächst eine zusammenfassende Darstellung der bereits existierenden Arbeiten zur SAP R/3-Kapazitätsplanung und nachfolgend die Gegenüberstellung zu den in dieser Arbeit erzielten Ergebnissen.

Es existieren verschiedene Arbeiten zu den Gebieten Vorgehensmodelle, Daten-Messung, -Charakterisierung und -Analyse sowie Performance-Prognose, die dem Themenbereich der Kapazitätsplanung für den Anwendungsfall SAP R/3 zugeordnet werden können. Im Jahre 1996 wurde von Thompson ein allgemeines Vorgehensmodell für ein „IT Enterprise Resource Management“ (IT ERM) definert (siehe [118] und [117]), das er 1997 zusammen mit Munoz und DeBruhl auf den Fall SAP R/3 anwendete [119]. Das von Thompson deklarierte Ziel von IT ERM sollte die Unterstützung eines Unternehmens in der Planung und Akquirierung von IT-Ressourcen sein. In [117] werden hierfür die in einem Unternehmen zu etablierenden „Enterprise Resource Management Processes“ und die damit zu erreichenden Ziele beschrieben. Zur Umsetzung seines Konzepts hebt Thompson als Basis von IT ERM eine kundenorientierte, hoch automatisierte und integrierte Last-Messungs- und -Prognose-Infrastruktur hervor. In [118] werden die Last-Messung und -Charakterisierung sowie Ansätze zur Performance-Prognose mittels Trendanalysen detailliert beschrieben, wobei hier der Schwerpunkt auf der Lastcharakterisierung liegt.

In [119] werden die im IT ERM-Framework beschriebenen Ansätze zur Workload-Charakterisierung auf den Fall SAP R/3 angewandt. Es wird eine Klassifizierung der Last nach Anwender, Applikation oder Tasktyp vorgeschlagen, die Ressourcenverbräuche (CPU-Zeit im Workprozess, DB-Request-Time, Requested KB) und Antwortzeiten ebenfalls beinhaltet. Es werden mögliche Datenquellen, wie z. B. das CCMS, der R/3-Accounting-Exit und die R/3-Statistik-Sätze für das Data Repository, beschrieben. Einen Schwerpunkt bildet die Untersuchung der Güte der gewonnenen Messdaten. Zusätzlich zu den in [119] beschriebenen Datenquellen im SAP R/3-System werden nach Angaben von Ansfield [1] für die R/3-Kapazitätsplanung weitere Performance-Messdaten, wie z. B. solche aus Sicht der Datenbank und des Betriebssystems, für eine ganzheitliche Sicht auf das System benötigt. Die Schwierigkeit liegt dann in der Korrelation dieser verschiedenen Messdaten.

Bei den oben genannten Arbeiten stehen die Bereiche Vorgehensmodelle und Daten-Messung, -Charakterisierung und -Analyse im Vordergrund. In den Arbeiten von Giacone und Munoz ([36], [37]) sowie von Somin [110] wird hingegen die Performance-Prognose mittels analytischer Modellierung von SAP R/3-Systemen beschrieben. Die Erstellung eines entsprechenden Modells basiert nach [36] auf den Messdaten des R/3-Systems (CCMS), des Betriebssystems und der Datenbank. Die Workload-Charakterisierung wird auf Applikationsebene, d. h. Klassifizierung nach R/3-Modulen, vorgenommen. Mittels Korrelation der CCMS- und der Betriebssystem-Daten (auf Prozessebene) werden die Workload-Beschreibungen pro Applikation erstellt. Ein mit vergleichbarer Workload-Charakterisierung parametrisiertes Modell wird in [110] am Beispiel des SD-Benchmarks beschrieben.

Das in der vorliegenden Arbeit erarbeitete Vorgehensmodell für die R/3-Kapazitätsplanung basiert auf dem im Dokument [122] dargestellten MAPKIT-Vorgehensmodell und dem oben beschriebenen IT ERM-Framework. Für den Bereich der Datenmessung und -analyse wurde ein Konzept zur R/3-Lastbeschreibung dargestellt und erweitert, das zum einen eine im Unterschied zu den bereits beschriebenen Methoden ([36], [119]) differenzierte Lastanalyse und, wie in dieser Arbeit dargestellt, darauf basierende Dienstgütedefinition ermöglicht und zum anderen eine für die Modellierung adäquate Lastbeschreibung liefert. Für die weiteren Ebenen eines R/3-Systems, d. h. Betriebssystem und Datenbank, wurden Werkzeuge und die für die R/3-Kapazitätsplanung erforderlichen Messdaten beschrieben sowie insbesondere für die Datenbank Korrelationen zwischen der R/3- und Datenbankaktivität aufgezeigt.

Mit Hilfe der definierten R/3-Lastcharakterisierung können Messdaten mit beliebigem Detaillierungsgrad beschrieben werden, wie z. B. auf Ebene der Instanzen, der Module, der User oder auch Transaktionen. Aufgrund dessen und des zugehörigen Dienstgütekonzepts können R3-Lasten quantifiziert und bewertet werden, so dass eine gute Basis für die Beschreibung von IT-Leistungen im Bereich der ASP-Dienstleister oder von Garantieleistungen der Hardware-Hersteller gegenüber den Kunden für die verkaufte Leistung gegeben ist. Der in dieser Arbeit beschriebene Ansatz bietet weiterhin eine Grundlage für Kostenmodelle zur Verrechnung von Hardware- oder IT-Leistungen in Form eines „Preises pro User“. Die Idee ist hierbei, einen Festpreis pro verbrauchtem DBSU je User zu definieren, so dass auf Basis des gemessenen DBSU-Verbrauchs Kosten pro User festgelegt bzw. bereits definierte Kosten anhand der Messung verifiziert werden können.

Neben der Datenmessung und -analyse wurde für die R/3-Kapazitätsplanung die Modellierung von R/3-Systemen beschrieben. Mit Hilfe eines analytischen Modells können auf Basis realer Lasten unter Verwendung der beschriebenen Lastcharakterisierung Leistungsmaße, wie Antwortzeiten, Auslastungen und Durchsätze, berechnet werden. Mit Hilfe des Werkzeugs WLPSizer können die Modelle auf Grundlage verschiedener Bibliotheken erstellt sowie die erforderlichen R/3-Lasten automatisch generiert werden. Die Vorgehensweisen zur Modellierung von SAP R/3-Systemen von Giacone und Munoz basieren hingegen auf einer zum Teil manuellen Beschreibung der R/3-Last auf Basis von CCMS-Reports mit einem Detaillierungsniveau auf Applikationsebene. Im Rahmen dieser Arbeit wurde der WLPSizer als Modellierungswerkzeug für SAP R/3 im Vergleich zu seinem Entwicklungsstand in [111] hinsichtlich der Funktionalität, des zugrunde liegenden Modells und der Bedienung wesentlich weiterentwickelt, so dass er für die durchgeführten Projekte erfolgreich eingesetzt werden konnte.

Die beschriebenen Bereiche Datenmessung und -analyse sowie Modellierung der SAP R/3-Kapazitätsplanung bauen direkt aufeinander auf. Die Ergebnisse der Datenmessung können mit Hilfe der beschriebenen Lastcharakterisierung unmittelbar zur Modellierung verwendet werden, so dass ein gleitender Übergang zwischen den Phasen des Vogehensmodells zur R/3-Kapazitätsplanung entsteht. Der direkte Zusammenhang zwischen Messung und Modellierung sowie der hohe Automatisierungsgrad zur Durchführung der beschriebenen Aktivitäten ist hierbei hervorzuheben, da diese Eigenschaften wesentliche Unterschiede zu bereits existierenden Methoden und Verfahren der R/3-Kapazitätsplanung darstellen. Eine derart hoch automatisierte Infrastruktur zur R/3-Kapazitätsplanung wird von Thompson in der Beschreibung von IT ERM als Basis zur Einführung und Etablierung von IT ERM oder in diesem Fall R/3-Kapazitätsplanung gefordert.

6.1.2. Modellierung von Storage-Systemen

Die vorliegende Arbeit umfasst im Bereich der Modellierung von Storage-Systemen die Erarbeitung und Anwendung von Methoden und Werkzeuge zum Benchmarking, Monitoring und für die Modellierung von Festplatten. Das Ziel war es, ein analytisches Festplattenmodell zu erstellen, das anhand von realen Messdaten kalibriert und validiert werden sollte. Es folgt eine Beschreibung der bereits existierenden Arbeiten zur Modellierung von Storage-Systemen und nachfolgend die Gegenüberstellung zu den in dieser Arbeit erzielten Ergebnissen.

In der Literatur wird die Modellierung von Festplatten unter Verwendung von analytischen und simulativen Verfahren beschrieben. Das Einsatzgebiet der Festplattensimulation lag in der Vergangenheit häufig in der Analyse von Scheduling-Strategien oder Caching-Algorithmen. Eine der bekanntesten Publikationen zur Modellierung von Festplatten mittels Simulation stammt von Ruemmler und Wilkes [84]. Diese Arbeit beschreibt ein Simulationsmodell mit einem hohen Detaillierungsniveau der einzelnen internen mechanischen Abläufe und Caching-Strategien

einer Festplatte, so dass entsprechend detaillierte Eingabeparameter und Beschreibungen der Festplatte benötigt werden, die in frei erhältlichen Festplattenbeschreibungen nicht enthalten sind und die auch über Messungen nur mit Zusatzinstrumentierungen des Systems oder gar nicht zu bestimmen sind. Die Simulationsmodelle sind durch die detaillierte Modellabbildung nur für bestimmte Festplatten anwendbar.

Im Bereich der analytischen Festplattenmodellierung beschäftigen sich eine Vielzahl von Arbeiten mit der Approximation der Seek-Time ([44], [38], [84]), der Berechnung der Cache-Hit-Ratio sowie der Analyse verschiedener Caching-Strategien ([15], [109], [99]) und der Ermittlung des Zeitverbrauchs in der Festplattenwarteschlange unter Verwendung verschiedener Queueing-Strategien ([133], [46], [116], [21]). Auf Basis der in den verschiedenen Publikationen erzielten Ergebnisse und Algorithmen wird in der Dissertation von Shriver [99] ein analytisches Festplattenmodell beschrieben, das Read-Ahead und Tagged Command Queueing unterstützt. In ihrer Arbeit werden ausschließlich Lese-Zugriffe betrachtet und die Validation des Modells wird mit Hilfe von Simulationsergbenissen (Pantheon Disk Simulator [135]) durchgeführt. Nach Shriver ist eine Validation des Modells auf Basis von realen Messdaten und eine Anwendung des Modells auf beliebige Festplatten nur mit erhöhtem Aufwand oder gar nicht möglich, da hierzu entweder eine Zusatzinstrumentierung des Systems notwendig ist oder die Festplatten nicht die für das Modell notwendigen Messdaten liefern können. Neben Shriver beschreibt auch Menascé sowohl in [71] als auch in [70] ein analytisches Festplattenmodell, das die Bedienzeit pro IO-Zugriff berechnet. Das von ihm in [70] dargestellte Modell und die damit verbundene IO-Lastcharakterisierung basieren hierzu teilweise auf den Algorithmen von Shriver, wie z. B. die Berechnung der Rotationslatenzzeit für sequentielle Zugriffe.

Bereits in den 70er Jahren wurden die ersten analytischen Festplattenmodelle entwickelt (siehe z. B. [134]), wobei aber die dabei betrachteten Festplatten keine On-Board-Controller oder Cache-Speicher besaßen, so dass aus heutiger Sicht relativ einfache Modelle ohne komplexe Bedienstrategien zur Abbildung einer Festplatte ausreichten. Mit der Einführung der RAID-Architektur in den 80er Jahren verlagerte sich der Schwerpunkt der Forschung im Bereich Storage auf die Modellierung von Disk Arrays. Die jüngsten Arbeiten zur analytischen Modellierung von Festplatten sind die oben genannten Arbeiten von Menascé und Shriver, die als Grundlage für das in dieser Arbeit entwickelte Festplattenmodell dienten.

In der vorliegenden Arbeit wurde zur Modellierung von Speichersubsystemen ein analytisches Festplattenmodell entwickelt, das im Gegensatz zu bisherigen dem Autor bekannten Arbeiten auf Basis von realen Messdaten, unter Verwendung von Standard- bzw. frei erhältlichen IO-Monitoring-Werkzeugen, hinsichtlich der Bedienzeit und Antwortzeit pro IO-Zugriff validiert und kalibriert wurde. Die Parametrisierung des Modells wurde mit frei erhältlichen Hardware-Beschreibungen für die Festplatte, einer in dieser Arbeit definierten IO-Lastcharakterisierung sowie Einflussgrößen des betrachteten Betriebssystems auf die IO-Verarbeitung vorgenommen. Es wurden in Kapitel 4 verschiedene IO-Lastszenarien definiert und mittels eines an der Universität Essen entwickelten IO-Benchmarks generiert. Das Messen der IO-Last wurde hierbei mittels des von Cockcroft entwickelten und frei erhältlichen Skripts SIOSTAT des SE Performance Toolkits [97] durchgeführt.

In Kapitel 4 wurde ein Konzept zur IO-Lastcharakterisierung erarbeitet, das vier verschiedene IO-Lasten (Basislasten) definiert, so dass eine beliebige IO-Last immer einer Kombination dieser Basislasten entspricht. Mit Hilfe des IO-Benchmarks wurden die Basislasten erzeugt und analysiert. In Kapitel 5 wurden für das Festplattenmodell komplexere IO-Lasten untersucht, die einem Mix der verschiedenen IO-Basislasten entsprachen. Für jedes Lastszenario wurden Messungen durchgeführt und für die Kalibrierung und Validierung des Festplattenmodells verwendet. Im Vergleich zu den Festplattenmodellen von Shriver und Menascé hat sich in dieser Arbeit für bestimmte Lastszenarien ein abweichendes Festplattenverhalten ergeben. Anhand der Messergebnisse wurden die Algorithmen für das Festplattenmodell entsprechend erweitert. Für die

Abbildung der parallelen IO-Verarbeitung wurde im Rahmen dieser Arbeit eine IO-Bibliothek definiert, die mit Hilfe von einfachen Benchmark-Messungen ermittelt wurde. Ein derartiges Vorgehen zur Herleitung von Festplattenparametern anhand von Messungen wird in [141] beschrieben, wobei dort direkte „low-Level“ SCSI-Zugriffsmöglichkeiten und hoch auflösende Timer im System benötigt werden sowie ein Device Driver Development Environment oder eine „offene“ IO-Card. Derartige Zusatzinstrumentierungen wurden im Rahmen dieser Arbeit nicht verwendet.

Mittels Kombination der analytischen Modellierung und der auf Basis von einfachen Benchmarkmessungen erzielten Festplatteneigenschaften wurde in dieser Arbeit bei den betrachteten IO-Lastszenarien eine zu Shriver verbesserte Modellgüte erzielt. Zusammengefasst weisen die in Kapitel 5.4. betrachteten IO-Lastszenarien einen maximalen Fehler von 19,77 % für die Antwortzeit auf. Sowohl im Vergleich zu Shriver (maximaler Fehler von 28 %) als auch zu Lazowska, der für Modelle mit nur einer Laskette eine maximale Abweichung der Antwortzeit von 20 % angibt, ist die Güte der Modellergebnisse gut, wobei die bereits beschriebenen Einschränkungen zur Modellparametrisierung zu berücksichtigen sind. Weiterhin ist durch die Verwendung der beschriebenen Monitoring-Werkzeuge sowie der Modellparameter ein realer Einsatz des erarbeiteten Festplattenmodells in Kapazitätsplanungsprojekten möglich. Aus Kostengründen werden in derartigen Projekten nur eine einfache Messumgebung für die Erhebung der Messdaten sowie Standard-Hardwarebeschreibungen zur Verfügung gestellt, wie sie in dieser Arbeit verwendet wurden. Die für das Festplattenmodell verwendeten Eingabeparameter ermöglichen weiterhin, dass das Modell auf eine Vielzahl von Festplatten anwendbar ist, da die Hardware-Beschreibungen der Festplattenhersteller inzwischen ein vergleichbares Detaillierungsniveau besitzen.

Für den Einsatz der Festplattenmodellierung in der Kapazitätsplanung für den Anwendungsfall SAP R/3, war eine IO-Lastcharakterisierung für den Anwendungsfall SAP R/3 notwendig. Bisher existiert eine mögliche Lastcharakterisierung ausschließlich von Mißbach und Hoffmann [72], die jedoch eine Abhängigkeit der IO-Last von dem Leistungsmaß SAPS beschreiben. Eine derartige Lastbeschreibung ist jedoch auch nach Meinung von Mißbach und Hoffmann nur eine sehr grobe Abschätzung, da Faktoren wie z. B. Anwendungs- und Datenbank-Version, Ausrichtung nach OLAP und OLTP hierbei nicht berücksichtigt werden. In Kapitel 3 wurde ein Konzept zur Korrelation von R/3-Lasten auf Dialogschrittebene und IO-Aktivitäten auf Betriebssystemebene erarbeitet. Anhand verschiedener Lastszenarien wurden die IO-Aktivitäten in einem SAP R/3-System analysiert und der Datenbankaktivität DBSU gegenübergestellt. Es wurden Konzepte und Vorgehensweisen zur IO-Lastcharakterisierung im Anwendungsfall SAP R/3 erarbeitet, die anhand von realen Produktivdaten validiert wurden. Mit Hilfe des erarbeiteten Festplattenmodells und der in Kapitel 3 beschriebenen IO-Lastcharakterisierung ist somit eine Leistungsprognose bzw. Modellierung von IO-Systemen im Anwendungsfall SAP R/3 möglich.

6.2. Ausblick

In beiden Bereichen, SAP R/3-Kapazitätsplanung und Modellierung von Speichersubsystemen, sind auf Basis der beschriebenen Ergebnisse die folgenden Weiterentwicklungen und Forschungsarbeiten möglich:

- Die beschriebene Kapazitätsplanung für SAP R/3-Systeme sollte zur Analyse und Modellierung von mySAP.com-Systemen erweitert werden. mySAP.com-Systeme können aus einer Vielzahl von agierenden und insbesondere (via ALE) kommunizierenden R/3-Systemen bestehen, so dass die Verarbeitung von Transaktionen sich über mehrere Systeme erstrecken kann. Weiterhin existieren für mySAP.com Systeme ohne R/3-Basis, wie z. B. der ITS-Server, die für eine Performance-Analyse evtl. mit berücksichtigt werden müssen. Beide

Eigenschaften von mySAP.com-Systemen können mit der derzeitigen SAP R/3-Kapazitätsplanung nicht betrachtet werden.

- Die in Abschnit 6.1.1. beschriebenen Ansätze für Kostenmodelle zur Verrechnung von IT-Leistungen (ASP, Garantieleistungen der Hardwarehersteller) basierend auf der in dieser Arbeit beschriebenen R/3-Lastcharakterisierung sollten konzeptionell beschrieben und anhand von Fallstudien erprobt werden.
- Die für die Modellierung von Festplatten dargestellten Konzepte und Methoden sollten für IO-Systemarchitekturen, wie z. B. RAID, NAS oder SAN, erweitert werden. Insbesondere für RAID-Systeme existieren eine Vielzahl an Publikationen, welche die Performance sowie eine analytische Modellierung derartiger Systeme beschreiben ([16], [81]). Auch wird in [102] eine Erweiterung des in dieser Arbeit beschriebenen Festplattenmodells von Shriver [99] zu einem Disk Array präsentiert. Eine Beschreibung zur Kapazitätsplanung von SAN-Systemen befindet sich in [31]. Die Schwierigkeit bei der Betrachtung weiterer IO-Architekturen ist das IO-Monitoring. Bereits bei der Betrachtung von Festplatten konnten für schreibende Zugriffe mit aktiviertem Schreib-Cache unter bestimmten Lastszenarien keine Bedienzeiten und Antwortzeiten ermittelt werden. Betrachtet man hingegen z. B. RAID-Systeme, so bestehen diese intern aus einer Vielzahl von Festplatten und teilweise auch aus mehreren Controllern und IO-Bussen. Aus Sicht des Betriebssystems und der damit verbundenen Sicht der Standard-Monitoring-Werkzeuge, können die in einem solchen IO-System existierenden Festplatten und die zugehörigen IO-Aktivitäten nicht gemessen werden. Es können somit nur Bedienzeiten, Antwortzeiten und Auslastungen für das Gesamtsystem ermittelt werden. Für eine detaillierte Analyse eines solchen Systems müssen Monitoring-Werkzeuge, wie z. B. EMC Navisphere der Firma EMC^2, verwendet werden, die eine Darstellung der IO-Aktivität bis auf Ebene der Festplatten ermöglichen.

ANHANG

A

MESSDATEN ZU DEN IO-UNTERSUCHUNGEN MIT DEM SD-BENCHMARK UND SSQJ

In der folgenden Tabelle A.1 werden die Ergebnisse der Messungen für die in Kapitel 3 analysierten Szenarien beschrieben. Bei der Beschreibung des Oracle-Output sei auf das Skript UTLB/ESTAT verwiesen.

	120 SD-User	150 SD-User	23.000 SD-User	SSQJ (Large Table)
Hardware				
Server	Zentralserver: 4 x 550 MHz XEON Prozessoren mit 2 MB Cache	Zentralserver: 4 x 550 MHz XEON Prozessoren mit 2 MB Cache	3-tier: DB-Server (PRIMEPOWER 2000, 64 CPUs 450 MHz, 64 GB Hauptspeicher), 160 Appl.-Server (Primergy H400, N400 mit jeweils 4 CPUs 700 MHz, 4 GB Hauptspeicher)	Zentralserver: 4 x 330 MHz Prozessoren
IO-System	6 x 9 GB Platten, RAID-0	6 x 9 GB Platten, RAID-0	2 EMC Symmetrix 8730 mit 36 GB Platten, 2 EMC Connectrix ED-1032, Gesamtspeicherplatz: 1.344 GB	Fujitsu-Siemens-Storage-Box mit 4 FC-Kanälen und 16 GB Cache
Software	Linux, SAP Rel. 4.6c SR2, Oracle8i Enterprise Edition Release 8.1.7.0.0	Linux, SAP Rel. 4.6c SR2, Oracle8i Enterprise Edition Release 8.1.7.0.0	DB-Server: Solaris, Appl.-Server: Linux RedHat 6.1 EE, DB: Oracle 8.1.6, SAP Rel. 6.4B	Solaris, SAP Rel. 4.6D, DB: Oracle8i Enterprise Edition 8.1.6.3.0
Benchmark-Output				
Gesamtlaufzeit	19 Minuten	22 Minuten	12 Minuten	48 Minuten
Sampling-Intervall	10 Minuten 30 Sekunden	11 Minuten 30 Sekunden	---	---
Anzahl DS[1]	7.103	8.001	---	529
Anzahl DS/h	40.589	41.744	7.061.000	659
Mittl. Antwortzeit	656 ms	2.913 ms	1.730 ms	5.177 ms
myAMC.LNI-Output (Gesamtlaufzeit)				
DBSU	8.026.796	10.229.634	1.188.000.000	1.520.340
DB-Calls	385.368	489.965	---	33.489

Tabelle A.1 Messergebnisse für die IO-Lastcharakterisierung

	120 SD-User	150 SD-User	23.000 SD-User	SSQJ (Large Table)
Delete-DB-Calls	2.646	3.306	---	49
Insert-DB-Calls	118.922	153.948	---	484
Update-DB-Calls	46.929	58.678	---	301
Read-DB-Calls	216.871	274.033	---	32.655
DSQLCNT[2]	495.408 KB	617.858 KB	---	1.556.046 KB
myAMC.LNI-Output (Sampling-Intervall)				
DBSU	6.033.490	6.936.189	---	---
Oracle-Output				
User Calls	324.295	415.306	---	49.652
Laufzeit	wie Gesamt-LZ	wie Gesamt-LZ	---	wie Gesamt-LZ
Writes	15.382	17.888	---	163.760
Blocks Written	15.382	17.888	---	1.845.907
Write Time	17.990 ms	25.880 ms	---	2.330 ms
Reads	101	4.759	---	2.086.005
Blocks Read	101	4.759	---	10.698.866
Read Time	470 ms	55 180 ms	---	23.260 ms
Block-Größe	8 KB	8 KB	---	8 KB
Cache-Hit-Ratio[3]	99,9937567 %	99,7201888 %	---	18,49%
CPU-Time	180.760 ms	237.050 ms	---	4.675.170 ms
Redo Blocks written	305.258 (2.442.064 KB)	382.800 (3.062.400 KB))	---	58.340 (466.720 KB)
# Checkpoints	1	1	---	2
Systemmonitor-Output (Gesamtlaufzeit)				
	IOs | KB	**IOs | KB**	**IOs | KB**	**IOs | KB**
IOPL Gesamt	62.545 | 354.645	109.095 | 479.900	--- | ---	--- | ---
IOPL Reads	4.650 | 20.025	18.360 | 80.025	--- | ---	--- | ---
IOPL Writes	57.895 | 334.620	90.735 | 399.875	--- | ---	--- | ---
SAR Gesamt	64.170 | 356.684	111.234 | 482.686	1.290.805 | 19.202.440	2.232.615 | 88.652.180[4]
IOSTAT Gesamt	64.165 | 356.357	111.207 | 482.608	--- | ---	2.291.723 | 101.169.147
IOSTAT Reads	--- | ---	--- | ---	--- | ---	2.091.247 | 86.139.151
IOSTAT Writes	--- | ---	--- | ---	--- | ---	200.475 | 15.029.997
Systemmonitor-Output (Sampling-Intervall)				
IOPL Gesamt	46.085 | 262.695	77.730 | 319.280	--- | ---	--- | ---
Berechnete Werte				
IOs/DBSU (Gesamt-Laufzeit)	0,00779	0,01066	0,00108	1,50737
IOs/DBSU (Sampling-Int.)	0,00763	0,01121	---	---
KB/DBSU (Gesamt-Laufzeit)	0,04418	0,04691	0,01599	66,54376
KB/DBSU (Sampling-Int.)	0,04354	0,04603	---	---

Tabelle A.1 Messergebnisse für die IO-Lastcharakterisierung

1. Anzahl der Dialogschritte für die Messung
2. Daten, die zwischen Applikationsserver und Datenbank transferiert werden
3. Berechnet durch: (1 - physical reads / (consistent gets + db block gets)) * 100
4. unter der Annahme, dass 1 Block = 512 Byte entspricht

LITERATURVERZEICHNIS

[1] S. Ansfield; *„SAP R/3: Facing The Unknown? Capacity Management and SAP R/3“*; In: Proceedings of the UK CMG-Conference; 1996

[2] P. Awoseyi, J. Day: *„Disk Subsystem Performance Analysis in WINTEL Platform“*; Compaq Computer Corporation; 1998

[3] R. Barve, E. Shriver, P. Gibbons; *„Modeling and optimizing I/O throughput of multiple disks on a bus“*; ACM SIGMETRICS; Atlanta, Georgia; 1999

[4] F. Baskett, K. M. Chandy, R. R. Muntz, F. G. Palacios; *„Open, closed and mixed networks of queues with different classes of customers“*; In: Journal of ACM, Vol. 22; 1975

[5] BMC; *„BMC Software Solutions for SAP Environments“*; White Paper; http://www.bmc.com/products/documents/16/77/11677/11677.pdf

[6] BMC; *„PATROL for SAP Solutions“*; White Paper; http://www.bmc.com/products/documents/26/11/12611/12611.pdf

[7] H. Bögeholz; *„H2Bench“*; c't Magazin für Computertechnik; http://www.heise.de/ct/Redaktion/bo/liesmich.asc; 2000

[8] H. Bögeholz; *„Platten Karussell“*; c't Magazin für Computer Technik, Nr. 12; 2001

[9] R. Bordewisch, C. Flüs, R. Grabau, J. Hintelmann, K. Hirsch, H. Ristaus; *„Kapazitätsmanagement gestern & heute“*; In: B. Müller-Clostermann, Kursbuch Kapazitätsmanagement; BoD; ISBN 3-8311-2823-5; 2001

[10] S. Bös; *„Algorithms of the Quick Sizer“*; SAP; White Paper, Version 2.1; 2001

[11] R. Buck-Emden, J. Galimow; *„Die Client/Server-Technologie des SAP-Systems R/3“*; Addison-Wesley; ISBN 3893198709; 1995.

[12] R. Buck-Emden; *„Die Technologie des SAP R/3 Systems“*; Addison-Wesley; ISBN 3827313791; 1998

[13] D. Burleson; *„Oracle8 Tuning“*; Sybex-Verlag; ISBN 3-81557-291-6; 1998

[14] c't; *„Server-Tuning mit dem SE Performance-Toolkit“*; c't Magazin für Computer Technik, Nr. 6; 1999

[15] S. C. Carson, S. Setia; *„Analysis of the periodic update write policy for disk cache“*; IEEE Transactions on Software Engineering, Vol. 18; 1992

[16] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, D. A. Patterson; *„RAID: high-performance, reliable secondary storage“*; ACM Computing Surveys, Vol. 26; 1994

[17] P. M. Chen, D. A. Patterson; *„Storage Performance - Metrics and Benchmarks“;* Computer Science Division, Dept. of EECS; University of California, Berkley; 1993

[18] L. Chung, J. Gray, B. Worthington, R. Horst; *„Windows 2000 Disk IO-Performance“*; Technical Report MS-TR-2000-55; Microsoft Research Advanced Technology Division; 2000

[19] A. Cockcroft; *„How do disks really work?“*; http://www.unixinsider.com/unixinsideronline/swol-06-1996/swol-06-perf_p.html; 1996

[20] A. Cockcroft, R. Pettit; *„Sun Performance and Tuning“*; Prentice Hall PTR; ISBN 0-13095-249-4; 1998

[21] E. G. Coffman, M. Hofri; *„On the expected performance of scanning disks“*; In: SIAM Journal on Computing, Vol. 11; 1982

[22] P. V. Crain, C. D. Hanson; *„Oracle Performance Analysis Using Wait Events“*; In: Proceedings of the CMG-Conference; 1998

[23] P. Dadam; *„Verteilte Datenbanken und Client/Server-Systeme“*; Springer-Verlag; ISBN 3540613994; 1996.

[24] T. Dapper, C. Dietrich, B. Klöppel; *„Windows NT 4.0 im professionellen Einsatz“*; Hanser Fachbuch; ISBN 3-44618-672-7; 1996

[25] J. Doppelhammer, T. Höppler, A. Kemper, D. Kossmann; *„Database Performance in the Real World - TPC-D and SAP R/3“*; In: Proceedings of the ACM SIGMOD international conference on Management of data; 1997

[26] I. Englander; *„The Architecture of Computer Hardware and Systems Software“*; John Wiley & Sons; ISBN 0-47136-209-3; 2000

[27] G. Färber, J. Kirchner; *„mySAP Technology“*; Galileo Press; ISBN 3-89842-266-6; 2002

[28] M. Farley; *„Building Storage Networks“*; McGraw-Hill Osborne Media; ISBN 0-07213-072-5; 2001

[29] D. Ferrari, G. Serazzi, A. Zeigner; *„Measurement and Tuning of Computer Systems“*; Prentice-Hall; ASIN 0135685192; 1983

[30] T. Fischer; *„Kooperation von Monitoring- und Analyse-Tools zur Kapazitätsplanung von Client/Server-Systemen: Entwurf und Implementierung von Software-Komponenten mit Visual Cafe“*; Technischer Bericht; Informatik, Universität Essen; 1999

[31] M. B. Friedman; *„An Introduction to SAN Capacity Planning“*; In: Proceedings of the CMG Conference; 2001

[32] M. Friedman, O. Pentakalos; *„Windows 2000 Performance Guide“*; O'Reilly & Associates; ISBN 1-56592-466-5; 2002

[33] Fujitsu Siemens Computers; *„SCSI Tutorial“*; Internal Paper

[34] G. R. Ganger, B. L. Worthington, Y. N. Patt; *„The DiskSim simulation environment: version 1.0 reference manual“*; Tech. Report CSE-TR-358-98, Department of Electrical Engineering and Computer Science; University of Michigan, Ann Arbor, MI; 1998

[35] E. Gelenbe, G. Pujolle; *„Introduction to Queueing Networks“*; John Wiley & Sons; ISBN 0-47196-294-5; 1987

[36] G. B. Giacone, J. Munoz; *„SAP R/3 In A UNIX Environment: A Case Study For Client/Server Capacity Planning“*; In: Proceedings of the CMG-Conference; 1997

[37] G. B. Giacone, J. Munoz; *„SAP R/3 Tuning and Upgrade Practices“*; In: Proceedings of the CMG-Conference; 1997

[38] C. C. Gotlieb, G. H. MacEwen; *„Performance of movable -head disk storage systems“*; In: Journal of the ACM, Vol. 20; 1973

[39] A. Grummit, T. Foxon; *„Performance Tuning and Capacity Planning for Oracle on Unix - A case study“*; In: Proceedings of the CMG-Conference; 1997

[40] M. Gurry, P. Corrigan; *„Oracle Performance Tuning“*; O'Reilly & Associates; ISBN B00007FYAO; 1996

[41] J. L. Gustafson; *„Reevaluating Amdahl´s Law“*; www.scl.ameslab.gov/Publications/AmdahlsLaw/Amdahls.html; 1988

[42] M. Handschuher; *„R/3 SizingTool - Users Guide“*; Fujitsu Siemens Computers; Internal Paper; 1999

[43] N. Hashmi; *„Business Information Warehouse for SAP“*; Premier Press; ISBN 0-76152-335-9; 2000

[44] J. L. Hennesey; D. A. Patterson; *„Computer Architecture - A quantitative approach“*; Morgan Kaufmann Publishers; ISBN 1558607242; 1996

[45] G. Higginbottom; *„Performance evaluation of communication networks“*; Artech House; ISBN 0-89006-870-4; 1998

[46] M. Hofri; *„Disk scheduling: FCFS vs. SSTF revisited“*; Communications of the ACM, Vol. 23; 1980

[47] A. D. Hospodor, A. S. Hoagland; *„The changing nature of disk controllers“*; Proceedings of the IEEE, 81(4); 1993

[48] G. E. Houtekamer; *„Benchmarking and Modeling: Pick your favorite“*, Consul Risk Management B.V.; Delft, The Netherlands; 1997

[49] IBM; *„Hard disk drive specifications Ultrastar 36LZX“*; Revision 1.0, S31L-8989-00; 1999

[50] Ideas International; *„Benchmarks, Top Performers Lists“*; http://www.ideasinternational.com/benchmark/bench.html

[51] D. M. Jacobson, J. Wilkes; *„Disk Scheduling algorithms based on rotational position“*; HP Laboratories Technical Report HPL-CSP-91-7rev1; 1991

[52] R. K. Jain; *„The Art of Computer Systems Performance Analysis“*; John Wiley & Sons; ISBN 0-47150-336-3; 1991

[53] S. Janssen; *„Carrying out Customer Performance Tests“*; SAP; White Paper, Version 1.1; 2001

[54] S. Janssen; *„Sizing mySAP.com“*; SAP; White Paper, Version 1.1; 2001

[55] G. Keller, T. Teufel; *„SAP R/3 prozeßorientiert anwenden“*; Addison-Wesley; ISBN 3-82731-401-1; 1997

[56] A. Kemper, D. Kossmann, B. Zeller; *„Performance Tuning for SAP R/3“*; In: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering; 1999

[57] L. Kleinrock; *„Queueing Systems - Volume I“*; John Wiley & Sons; ISBN 0-47149-110-1; 1975

[58] L. Kleinrock; *„Queueing Systems - Volume II“*; John Wiley & Sons; ISBN 0-47149-111-X; 1976

[59] D. Kotz, S. B. Toh; *„A Detailed Simulation Model of the HP 97650 Disk Drive“*, Department of Computer Science, Dartmouth College; 1994

[60] C. Kowarschick; *„Anwendungsbezogen: SAP Standard Application Benchmark“*; iX Magazin für professionelle Informationstechnik, Nr. 12; 1999

[61] C. Kowarschick; *„Der SAP Standard Applikations Benchmark“*; http://xware.net/html/artikel/saps.htm

[62] C. M. Kozierok; *„Summary of SCSI Protocols and Transfer Modes“*; http://www.pcguide.com/ref/hdd/if/scsi/summary-c.html

[63] E. D. Lazowska, J. Zahorjan, G. S. Graham, K. C. Sevcik; *„Quantitative System Performance: Computer System Analysis using Queueing Network Models“*; Prentice-Hall; ISBN 0-13-746975-6; 1984

[64] D. J. Lilja; *„Measuring Computer Performance“*; Cambridge University Press; United Kingdom; 2000

[65] C. Loosley, F. Douglas; *„High Performance Client/Server“*; John Wiley & Sons; ISBN 0-47116-269-8; 1998

[66] O. Matthes; *„Eine Umgebung zur Durchführung und Visualisierung von Simulationsexperimenten“*; Diplomarbeit; Wirtschaftsinformatik, Universität Essen; 1999; http://www.cs.uni-essen.de/SysMod/JavaDEMOS/

[67] J. Mauro, R. McDougall; *„Solaris Internals“*; Prentice Hall PTR; ISBN 0-13022-496-0; 2000

[68] G. Meissner; *„SAP die heimliche Software-Macht“*; Wilhelm Heyne Verlag; ISBN 3-45315-548-3; 1999

[69] D. A. Menascé; *„Capacity Planning in Client/Server Environments“*; Tutoriumsunterlagen: Performance Tools; MMB-Konferenz; Heidelberg; 1995

[70] D. A. Menascé; *„Capacity Planning for Web performance: metrics, models, and methods“*; Prentice Hall; ISBN 0-13693-822-1; 1998

[71] D. A. Menascé, V. A. F. Almeida, L. W. Dowdy; *„Capacity Planning and Performance Modeling from Mainframes to Client-Server Systems“*; Prentice Hall; ISBN 0-13789-546-1; 1994

[72] M. Mißbach, U. M. Hoffmann; *„Hardware-Lösungen für SAP-Systeme“*; Galileo Press; ISBN 3-89842-124-4; 2001

[73] B. Müller-Clostermann (ed.); *„Capacity Planning and Performance Evaluation with the Tool VITO “*; Informatik, Universität Essen; 2000

[74] B. Müller-Clostermann, G. Totzauer; *„TOTO Benutzerhandbuch, Version 0.1“*; 1997

[75] M. Musolf; *„Simulation und Leistungsanalyse von Magnetplattensystemen mit Java Demos“*; Diplomarbeit, Wirtschaftsinformatik, Universität Essen; 2002

[76] A. Nagpal; *„ALE, EDI, & IDoc Technologies for SAP“*; Prima Publishing; ISBN 0-7615-1903-3; 1999

[77] G. Oberniedermaier, M. Geiß; *„SAP R/3 Systeme effizient testen“*; Addison-Wesley; ISBN 3-82731-504-2; 1999

[78] Oracle; *„Oracle 8i On Line Generic Documentation“*

[79] J. K. Ousterhout, H. Da Costa und et al.; *„A Trace-Driven Analysis of the UNIX 4.2 BSD File System“*; Operating Systems Review 19, 5; Proceedings of the 10th Symp. on Operating System Principles

[80] A. Park, J. C. Becker; *„IOStone: A synthetic file system benchmark“*; Computer Architecture News 18, 2; 1990

[81] D. A. Patterson, G. Gibson, R. H. Katz; *„A case for redundant arrays of inexpensive disks (RAID)“*; In: H. Boral, P. A. Larson, Proceedings of the SIGMOD International Conference on Management of Data, Chicago; 1988

[82] M. Reindl; *„SQLBench - database load testing prevents bad is-investments“*; http://www.fors.com/eoug97/papers/0503.html; 1997

[83] C. Ruemmler, J. Wilkes; *„UNIX disk access patterns“*; In: Proceedings of the Winter USENIX Technical Conference, 1993

[84] C. Ruemmler, J. Wilkes; *„An Introduction to disk drive modeling“*; IEEE Computer, 27(3); 1994

[85] SAP AG; *„SAP-Kurs SAP50 Basistechnologie“*; Release 4.0A; Materialnr. 50023323; 1998

[86] SAP AG; *„Standard Application Benchmarks Description“*; Version 3.1; 1998

[87] SAP AG; *„SAP Business Information Warehouse - Standard Application Benchmark“*; 2000

[88] SAP AG; *„Standard Application Benchmarks - Published Results“*; Version 4.7; http://www.sap.com/solutions/technology/pdf/50020428.pdf; 2001

[89] SAP AG; *„The SAP Standard Application Benchmark Certified Results“*; http://www.xware.net/sapbench

[90] SAP R/3 Advanced Technology Group; *„SSQJ Documentation“*; Version 2.1; 2000

[91] A. Schätter, G. Totzauer; *„Aufteilung von Rechenkapazität durch Steuerung von Durchsätzen“*; In: H. Beilner (Hrsg.): Proceedings „Messung, Modellierung und Bewertung von Rechensystemen“; Springer Verlag; 1985

[92] T. Schneider; *„SAP R/3-Performance-Optimierung“*; Addison-Wesley; ISBN 3-8273-1400-3; 1999

[93] T. Schneider; *„SAP Performanceoptimierung“*; Galileo Press; ISBN 3-89842-192-9; 2001

[94] G. P. Schulz; *„New Storage Interfaces“*; White Paper; The SCSI Trade Association; 1997

[95] P. Schweitzer; *„Approximate Analysis of Multiclass Closed Networks of Queues“*; In: Proceedings of the International Conference on Stochastic Control and Optimization; Amsterdam; Netherlands; 1979

[96] SCSI Trade Association; *„STA overview of the different SCSI Standards and Cables“*, http://www.scsita.org/terms/scsiterms.html, 2000

[97] SE Performance Toolkit; Homepage; http://www.setoolkit.com

[98] M. Seltzer, P. Chen, J. Ousterhout; *„Disk Scheduling Revisited“*; In: Proceedings of the 1990 Winter Usenix; Washington D.C.; 1990

[99] E. Shriver; *„Performance modeling for realistic storage devices“*; PhD thesis, New York University, Department of Computer Science; 1997

[100] E. Shriver, B. K. Hillyer, A. Silberschatz; *„Performance Analysis of Storage Systems“*; Bell Laboratories, In: G. Haring, C. Lindemann, M. Reiser (eds.), Performance Evaluation, Springer-Verlag, Berlin Heidelberg; 2000

[101] E. Shriver, A. Merchant, J. Wilkes; *„An analytic behavior model for disk drives with readahead caches and request ordering“*; In: Conference Proceedings of the Sigmetrics, Madison, WI; 1998

[102] L. Shriver; *„Disk array modeling“*; Technical Report HPL-SSP-97-7, Storage Systems Program; Hewlett-Packard Laboratories; Palo Alto, CA; 1997

[103] Siemens AG, *„Reliant UNIX 5.45 - Tuning-Leitfaden“*; Internal Paper; 1999

[104] Siemens AG; *„Application Management Center - Overview“*; Siemens AG (I&S IT PS MHM); White Paper; http://www.myAMC.de; 2000

[105] Siemens AG; *„myAMC.LNI User Guide“*; Siemens AG (I&S IT PS MHM); Handbuch; 2000

[106] Siemens AG; *„myAMC.DBMC“*; Handbuch; Siemens (I&S IT PS MHM); 2001

[107] A. Silberschatz, G. Gagne, P. B. Galvin; „*Operating System Concepts*“; John Wiley & Sons; ISBN 0-47141-743-2; 2001

[108] Solid Data Systems; „*IOTest, Performance, Reliability, and Maintenance Test Software*“; Solid Data Systems; http://www.soliddata.com/products/iotest.html

[109] J. A. Solworth, C. U. Orji; „*Write-only disk caches*“; In: Proceedings of the ACM SIGMOD International Conference on Management of Data (H. Garcia-Molina, H. V. Jagadish, eds.), Vol. 19; 1990

[110] Y. Somin; „*Digging Into SAP R/3 for Capacity Planning*“; In: Proceedings of the CMG-Conference; 1999

[111] L. Springmann; „*WLPSizer - Ein Werkzeug zur Kapazitätsplanung von SAP R/3-Systemen*“; Diplomarbeit; Informatik, Universität Essen; 1999

[112] J. D. Stai; „*The SCSI Bench Reference*“; ENDL Publications; ISBN 1-87993-630-5; 1997

[113] T. Stein; „*SAP Sued Over R/3*“; Information Week; August 31, 1998

[114] Sun Microsystems, „*Sun Solaris 2.8 Manual Pages*“; Sun Microsystems; 2000

[115] A. Tanenbaum, J. Goodman; „*Computerarchitektur*“; Addison-Wesley; ISBN 3-82737-016-7; 2001

[116] T. J. Teorey, T. B. Pinkerton; „*A comparative analysis of disk scheduling policies*“; Communications of the ACM, Vol. 15; 1972

[117] G. I. Thompson; „*A Manager´s Framework for Enterprise Resource Management*“; In: Proceedings of the CMG-Conference; 1996

[118] G. I. Thompson; „*The Need For An Enterprise Resource Management Measurement/Forecasting Infrastructure*“; In: Proccedings of CMG-Conference; 1996

[119] G. I. Thompson, J. Munoz, J. K. DeBruhl; „*The Availabilty & Quality of SAP R/3 Workload Data For Performance/Capacity Management Process Requirements*“; In: Proceedings of the CMG-Conference; 1997

[120] G. Totzauer „*Kopplung der Kettendurchsätze in geschlossenen Warteschlangennetzwerken*“; In: U. Herzog and M. Paterok (Hrsg.): Proceedings „Messung, Modellierung und Bewertung von Rechensystemen“; Springer-Verlag; 1987

[121] G. Totzauer; „*Documentation of TOTO*“; Internal project document; University of Essen; 1998

[122] Universität Essen, Siemens; „*Anforderungen und Vorgehensmodell für das Kapazitätsmanagement*“; Studie im BMBF-Verbund MAPKIT; 1999

[123] B. White, W. T. Ng, B. K. Hillyer; „*Performance Comparison of IDE und SCSI Disks*“; Technical Report, Bell Labs; 2001

[124] K. Wilhelm; „*Einsatz von Monitoring und Benchmarking beim R/3 Performance Engineering*“; In: R. Dumke, C. Rautenstrauch (eds.), 1. Workshop Performance Engineering in der Softwareentwicklung (PE2000); 2000

[125] K. Wilhelm; „*Kapazitätsplanung für SAP R/3 - Werkzeugunterstützte Leistungsbewertung und -prognose von SAP R/3-Systemen*“; Technischer Bericht Nr. 2; Informatik, Universität Essen; 2000

[126] K. Wilhelm, M. Paul; „*SAP R/3 Kapazitätsmanagement - Design to Performance*“; In: Conference Proceedings of the CECMG, Berlin; 2000

[127] K. Wilhelm; *„Capacity Planning for SAP - Concepts and tools for performance monitoring and modelling“*; In: CMG Journal of Computer Resource Management, Issue 104; 2001

[128] K. Wilhelm; *„Modellierung und Prognose für SAP R/3 mit dem WLPSizer“*; In: B. Müller-Clostermann (Hrsg.), Kursbuch Kapazitätsmanagement; Books on Demand, www.bod.de; ISBN 3-8311-2823-5; 2001

[129] K. Wilhelm; *„WLPSizer - Werkzeug zur Kapazitätsplanung von SAP R/3-Systemen“*; Handbuch; http://www.cs.uni-essen.de/SysMod/WLPSizer/WLPSizer.htm; Informatik, Universität Essen; 2001

[130] K. Wilhelm, J. Pfister, C. Kowarschick, S. Gradek; *„Monitoring und Benchmarking für das R/3-Performance-Engineering“*; In: B. Müller-Clostermann (Hrsg.), Kursbuch Kapazitätsmanagement; Books on Demand, www.bod.de; ISBN 3-8311-2823-5; 2001

[131] K. Wilhelm; *„Analytische Modellierung von Festplatten mit Read-Ahead Caching und Tagged Command Queueing für die Kapazitätsplanung“*; 2. MMB Workshop, Universität Hamburg; 2002

[132] K. Wilhelm, J. Neumann; *„DBench - IO-Benchmarking“*; Technischer Bericht Nr. 4; Informatik, Universität Essen; 2002

[133] N. C. Wilhelm; *„An anomaly in disk scheduling: a comparison of FCFS and SSTF seek scheduling using an empirical model for disk accesses“*; Communications of the ACM, Vol. 19; 1976

[134] N. C. Wilhelm; *„A general model for the performance of disk systems“*; In: Journal of the ACM, 24(1): 14-31; 1977

[135] J. Wilkes; *„The Pantheon storage-system simulator“*; Tech. Report HPL-SSP-95-14, Storage Systems Program; Hewlett-Packard Laboratories; Palo Alto, CA; 1995

[136] L. Will, F. Hienger, F. Straßenburg, R. Himmer; *„Administration des SAP-Systems R/3“*; Addison-Wesley, ISBN 3-8273-1136-5; 1997

[137] J. Witte; *„Scaling to 100.000 SAPS“*; White Paper; Fujitsu Siemens Computers; 2001

[138] F. Wolf; *„ABAP Performance Workshop“*; dpunkt.verlag; ISBN 3-932588-73-8; 2000

[139] B. L. Wong; *„Configuration and Capacity Planning for Solaris Servers“*; Sun Microsystems Press, Prentice Hall; ISBN 0133499529; 1997

[140] B. L. Worthington, G. R. Ganger, Y. N. Patt; *„Scheduling Algorithms for Modern Disk Drives“*; ACM SIGMETRICS; CA Santa Clara; 1994

[141] B. L. Worthington, G. R. Ganger, Y. N. Patt, J. Wilkes; *„On-Line Extraction of SCSI Disk Drive Parameters“*; ACM SIGMETRICS, Ottawa, Ontario, Canada; 1995

[142] R. Yomtoubian; *„Survey of I/O-Benchmarks“*; CMG Transactions, Issue 94; 1998

[143] B. Zeller; *„Leistungsbewertung von SAP R/3-Installationen“*; Diplomarbeit; Universität Passau, Fakultät für Mathematik und Informatik, Lehrstuhl für Dialogorientierte Systeme; 2000

[144] B. Zeller, A. Kemper; *„Exploiting Advanced Database Optimizing Features for Large-Scale SAP R/3 Installations“*; In: Proceedings of the 28th VLDP Conference; Hong Kong, China; 2002