TECHNISCHE UNIVERSITÄT CHEMNITZ-ZWICKAU
FAKULTÄT FÜR INFORMATIK

Diplomarbeit

Untersuchung von Einflußfaktoren auf SPEC CINT95- und CFP95-Benchmarks

eingereicht von Carsten Mund
geboren am 20.November 1970 in Erlabrunn

Betreuender Hochschullehrer:
Prof. Dr. W. Kalfa
Lehrstuhl Betriebssysteme

Betreuer:
Dipl.-Inform. S. Graupner (LS Betriebssysteme)
Dipl.-Math. M. Ehrig (Universitätsrechenzentrum)

Chemnitz, den 10. Juli 1996

Selbständigkeitserklärung

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbständig und ohne unzulässige fremde Hilfe verfaßt habe. Ich habe für die Erstellung dieser Arbeit keine anderen als die angegebenen Quellen benutzt.

Chemnitz, den 10. Juli 1996

Carsten Mund

Bibliographische Beschreibung

Einflußfaktoren auf SPEC CINT95- und CFP95-Benchmarks.
Mund, Carsten. 1996. 122 Seiten.
Chemnitz, Technische Universität Chemnitz-Zwickau, Fakultät für Informatik,
Diplomarbeit.

Kurzreferat

Die Arbeit untersucht Einflußfaktoren auf die SPEC-CPU-Benchmarks CINT95 und CFP95. Es werden die Meßbedingungen bei Messungen von Rechnerherstellern für SPEC-Veröffentlichungen und bei Messungen im Universitätsrechenzentrum verglichen. Aus den unterschiedlichen Bedingungen werden mögliche Einflußfaktoren abgeleitet, die für die Abweichung der Benchmarkergebnisse unter Lastbedingungen ursächlich sein könnten. Für diese Einflußkomponenten eines Rechnersystems werden charakteristische Lastmaße erarbeitet.
Im experimentellen Teil der Arbeit werden 4 Rechner des Universitätsrechenzentrums mit 4 gezielt ausgewählten SPEC-CPU-Benchmarks untersucht. Über mehrere Wochen hinweg werden dabei Messungen unter verschiedensten Lastverhältnissen durchgeführt, wobei jeweils die Benchmarklaufzeit und die mittleren Werte für die definierten Lastmaße erfaßt werden.
Auf Basis dieser 16 Meßreihen von jeweils 20 bis 60 Messungen werden schließlich Aussagen zu Einflußgrößen auf einzelne Benchmarks, zum generellen Einfluß einzelner Teilsysteme, sowie zum differenzierten Lastverhalten der einzelnen betrachteten UNIX-Rechnersysteme gewonnen.

Inhaltsverzeichnis

1 Leistungsbewertung von Rechensystemen

1.1 Ziele

Heutige Rechensysteme erlauben es einem breitem Benutzerkreis, eine Vielzahl von Diensten zu nutzen, um ein breites Spektrum von Aufgaben mit Hilfe von Computern zu lösen. Mit der Weiterentwicklung der Computertechnik von der reinen Rechenmaschine zum vollwertigen Arbeitsmittel veränderten sich auch die Kriterien zur Bewertung von Systemen. Neben der Rechenleistung sind auch Kosten, Benutzerfreundlichkeit, Ausfallsicherheit und Verfügbarkeit wichtige Maßstäbe für eine Einordnung. Außer Leistung und Preis sind viele dieser Kriterien jedoch von Erwartung des Anwenders bestimmt und damit subjektiv. Dagegen können für die Leistung Maße gefunden werden, die einen Vergleich von Systemen möglich machen.
Die Bewertung der Leistung ist aus verschiedenen Blickwinkeln möglich. Dem Anwender stellt sich die Leistung des Rechners als das von außen beobachtbare Verhalten dar, am deutlichsten repräsentiert durch Abarbeitungs- und Antwortzeiten. Den Betreiber interessiert daneben auch das interne Systemverhalten, wobei besonders die Auslastung von Komponenten und des Gesamtsystems von Interesse ist. Allgemein kann man definieren:

Leistung ist eine Funktion der einzelnen Komponenten eines Rechensystems und der darauf wirkenden Arbeitslast. $(Leistung = f (Komponente, ..., Komponente_n , Arbeitslast))$

Die Arbeitslast ist die Menge an Aufträgen (Anweisungen) und Daten, die während eines Beobachtungszeitraumes vom System bearbeitet beziehungsweise gespeichert werden.

Ohne Kenntnis der Arbeitslast ist es nicht möglich, eine Aussage über die Leistung zu treffen [Lang92].
Ein Rechensystem wird zur besseren Überschaubarkeit und Funktionalitätsbeschreibung in mehrere Schichten eingeteilt. Jede der Schichten bietet mit ihren Komponenten bestimmte Dienste an. Zur Erfüllung der Dienste nutzt eine Schicht die Dienste der darunterliegenden Schicht, die über eine Schichtenschnittstelle verfügbar sind, gleichzeitig bietet sie über eine Schnittstelle ihre Dienste für die nächsthöhere Schicht an [Kalfa90].
Ein Auftrag der Arbeitslast „wandert“ durch das Schichtensystem „nach unten“ und wird dabei von den erforderlichen Systemkomponenten bearbeitet, bis sie in die unterste Schicht (Hardware) transformiert ist, wo im technischen Sinne die Leistung erbracht wird. Diese Leistung wird durch die einzelnen Ebenen des Systems zurück „nach oben“ transformiert, wo sie sich dem Anwender als Ergebnis des Auftrages präsentiert.
Daß sich Leistung für verschiedene Anwendungs- und Personenkreise unterschiedlich darstellt, wurde bereits erwähnt. Grundsätzliche Hauptanwendungsgebiete der Leistungsbewertung sind:

- Hard- und Softwareentwicklung:
  Während der Entwicklung von Produkten muß ständig geprüft werden, ob Entwurfsentscheidungen und folgende Realisierungen die gestellten Anforderungen erfüllen. Dies gilt für Einzelkomponenten und das Zusammenwirken dieser. Daneben erfolgt der Vergleich mit Leistungsmaßen, die von den Konkurrenzprodukten bekannt sind, um Verkaufsargumente für das eigene Produkt zu gewinnen.

- Vergleich von Rechensystemen:
  Für Neuanschaffungen ist die Betrachtung wichtig, wie ein System eine zu erwartende Last bewältigen wird, und ob bestimmte Leistungs- und Qualitätsparameter für diese Last garantiert werden können. Bestenfalls stehen für diese Vergleiche Referenzsysteme für eigene Messungen zur Verfügung, anderenfalls muß man sich auf vergleichbare Leistungsangaben der Hersteller verlassen.

- Kontrolle und Tuning laufender Systeme:
  Hierbei geht es darum, Engpässe beim Betrieb eines Rechensystems zu ermitteln und zu beseitigen. Dieses Anwendungsgebiet ist durch ständige Zyklen von Leistungsbewertung, Konfigurationsänderungen/-erweiterungen und erneute Leistungsbewertung gekennzeichnet. Untersuchungen zu veränderten Arbeitslasten führen zu Konfigurations- oder Entwicklungsentscheidungen bei Hard- und Softwarekomponenten oder zu kompletten Neuanschaffungen mit erforderlichem Systemvergleich.

Nach dieser Anwendungsunterteilung kann man auch bestimmte Personengruppen unterscheiden, die an einer Leistungswertung interessiert sind:

- Hersteller von Hard- und Software
- Vertreiber von Hard- und Software
- Betreiber/Verwalter von Rechensystemen
- Benutzer von Rechensystemen

In Abhängigkeit vom Ziel der Leistungsbewertung und vom Personenkreis, der an der Bewertung interessiert ist, erfolgt die Wahl spezieller Leistungsmaße und die Wahl der Methode der Leistungsbewertung.

1.2 Methoden der Leistungsbewertung

Leistungsuntersuchungen werden mit unterschiedlichen Zielen durchgeführt, weswegen dabei unterschiedliche Methoden angewendet werden. Es wurde bereits herausgestellt, daß die Leistung eines Rechnersystems von den Einzelkomponenten des Systems und der betrachteten Arbeitslast abhängt. Die einfachste Möglichkeit der Leistungsbewertung ist die Angabe von Leistungskenngrößen für die Einzelkomponenten auf Basis der Herstellerangaben. Dieses Vorgehen läßt die Komplexität des Systems im Zusammenwirken der Komponenten jedoch völlig außer acht, woraus Fehleinschätzungen für die Leistungsfähigkeit eines Rechensystems resultieren können. Für komplexe Systeme müssen andere Bewertungsmethoden herangezogen werden.

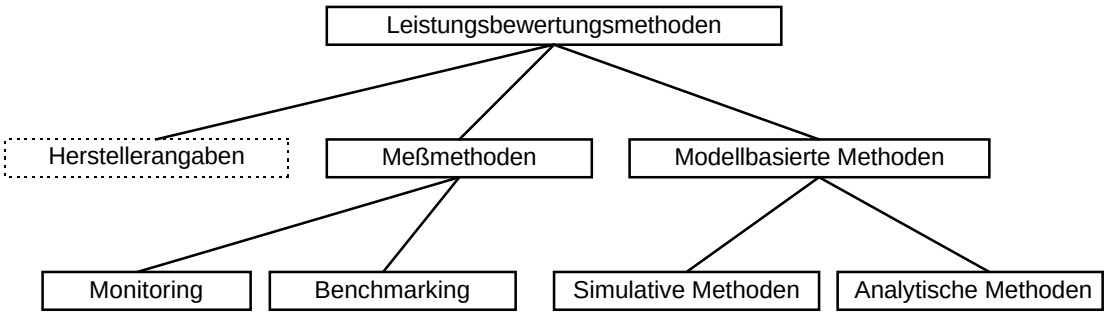


Abbildung 1-1 Methoden der Leistungsbewertung

Meßmethoden können an arbeitenden, verfügbaren Systemen angewendet werden. Dabei wird das System mit einer bestimmten Arbeitslast belegt. Während dieses *Benchmark*-Tests wird das Verhalten des Rechensystems beobachtet, indem Aktivitäten und charakteristische Größen gemessen und aufgezeichnet werden. Dieser Erfassungsvorgang von Systemgrößen und Ereignissen wird als *Monitoring* bezeichnet. Die als *Monitore* bezeichneten Erfassungsgeräte dienen allgemein der Überwachung operierender Systeme und der Erkennung von Engpässen.

Steht das zu bewertende System nicht (vollständig) zur Verfügung, so müssen Verfahren angewendet werden, die auf der **Bildung eines Modells** für das System beruhen. Modelle vereinigen in ihrer Beschreibung die wesentlichen Eigenschaften eines Systems. Für Entwürfe und grundlegende Vergleiche sind sie gut geeignet, da sie eine Modifikation ohne großen Aufwand zulassen. Es besteht jedoch die Gefahr, daß wichtige Eigenschaften der Realwelt im Modell fehlen und dafür enthaltene „unwichtige“ Eigenschaften das Modellverhalten maßgeblich beeinflussen. Die *analytischen Methoden* stellen Beziehungen zwischen Modellkomponenten her und versuchen auf theoretischem Wege, die resultierenden Leistungswerte zu berechnen. Das Problem hierbei ist, daß bei steigender Komplexität die funktionale Beschreibung der Beziehungen immer schwieriger wird. Klassisches Beispiel für die analytische Methode sind Berechnungen zur Warteschlangentheorie. Bei der *Simulationsmethode* untersucht man das Verhalten eines Rechnersystems durch Nachbildung des dynamischen Verhaltens eines Systems, z.B. in einem Simulationsprogramm. Die Leistungsmaße werden experimentell aus der Simulation der Realwelt im Modell gewonnen.

Für diese Arbeit sollen existierende Rechnersysteme der TU Chemnitz-Zwickau mit den SPEC[1]-CPU-Benchmarks praktisch untersucht werden. Daher muß hier auf die Charakteristik der Meßmethoden näher eingegangen werden.

1.2.1 Benchmarks

Benchmarks sind das klassische Mittel zum Vergleich von Rechensystemen und nachfolgend zur Auswahl von Erweiterungen bzw. von völlig neuen Systemen. Daneben werden sie auch zu Vergleichen bei der Entwicklung neuer Hardware(komponenten) und von Systemsoftware herangezogen.
Benchmarks versuchen, die für die Leistungsbewertung wichtige Einflußkomponente *Arbeitslast* zu erfassen, indem sie das System mit einer definierten Arbeitslast belegen. [Lang92] definiert diese Arbeitslast als die Menge aller Aufträge und Daten, die während einer Beobachtungsperiode vom System bearbeitet werden. Typische Leistungsmaße, wie Auslastungen, Verweilzeiten oder Durchsatzraten bestimmter Komponenten, sind direkt von dieser Arbeitslast abhängig und können durch die Benchmarks für die bestimmte Arbeitslast quantifiziert werden. Ein einfaches und das am häufigsten benutzte Leistungsmaß bei einem Benchmarklauf ist die Abarbeitungszeit für den Benchmark. Für den Vergleich unterschiedlicher Systeme werden die Abarbeitungszeiten untereinander oder relativ zu einem Referenzwert verglichen.
In der Praxis ist die aktuelle Arbeitslast eines Systems starken Schwankungen unterworfen. Über längere Beobachtungszeiträume hinweg folgt die Arbeitslast jedoch bestimmten statistischen Gesetzmäßigkeiten, die durch Verteilungsfunktionen beschreibbar sind [Lang92]. Anhand dieser kann eine repräsentative Auswahl aus Aufträgen getroffen werden, die die Arbeitslast beschreiben. Diese charakteristische Auftragsmenge bildet einen Benchmarktest, mit dem Systeme verglichen werden sollen.
Für Auswahl der Auftragsmenge gibt es beim Entwurf von Benchmarks zwei Möglichkeiten, die Bildung von Benchmarks aus Benutzerprogrammen und die Bildung synthetischer Benchmarks.

Benchmarks aus Benutzerprogrammen:

Es kann eine zufällig gewählte Stichprobe aus der Menge aller möglichen Aufträge verwendet werden, selbst wenn keine Auftragseigenschaften bekannt sind. Wichtig ist hier die Wahl einer ausreichenden Stichprobengröße, so daß die typische Auftragsverteilung im System auch in der Stichprobe repräsentiert ist. Sind Auftragseigenschaften bekannt, so ist es möglich, eine Klassifizierung der Aufträge vorzunehmen, beispielsweise nach Nutzungshäufigkeit, Laufzeit oder besonders kritischen Ressourcenforderungen von Anwendungen. Aus jeder Klasse werden dann einige Aufträge ausgewählt, die den Eigenschaften dieser Klasse am meisten entsprechen. Die Anzahl der gewählten Aufträge spiegelt den Klassenanteil an der Gesamtarbeitslast wieder. Der Nachteil dieser Auswahlverfahren besteht in der Orientierung an der Vergangenheit. Aus neuen Anwendungsmöglichkeiten einer neuen Anlage, neuen Systemfunktionen, Dienstprogrammen oder Programmiersprachen können stark veränderte Auftragsprofile resultieren, die in einem vergangenheitsbasierten Benchmark nicht repräsentiert sind.

[1] Standard Performance Evaluation Corporation

Bekannte Beispiele für Benchmarks aus Anwendungsprogrammen sind:

- LINPACK (LINear PACKage, 1976) - Eine Sammlung von Unterprogrammen aus dem Bereich der linearen Algebra. Es handelt sich um FORTRAN-Programme zur Lösung linearer Gleichungssysteme, die viele Fließkommaoperationen erfordern. Ergebnisse sind MFLOPS[2] . Durch hohe Codelokalität ist dieser Benchmark stark durch Caches beeinflußt.
- Stanford Small Programs Benchmark Set (SSPBS) - Sehr kleine Programme zum Vergleich von CISC- und RISC-Prozessoren.
- NAS-Kernel (Numarical Aerodynamical Simulation) - FORTRAN-Programme, die die Vektorleistung von Supercomputern testen (z.B. Matrizenmultiplikation und -inversion)

Die zu untersuchenden SPEC-Benchmarks basieren ebenfalls auf praktischen Anwendungsprogrammen. Probleme bereitet speziell die Portierung solcher Benutzerprogramme. Dies gilt für Maschinen- und Hochsprachenprogramme, sowie auch für die Datenbestände der Anwendungen. Sehr aufwendig ist der Nachweis der Korrektheit für die portierten Programme und der Gleichwertigkeit des erzeugten Codes bei Hochsprachenprogrammen. Deshalb generiert man häufig synthetische Benchmarks.

Benchmarks aus synthetischen Programmen:

Man erstellt relativ kurze Programme, die bestimmte Systemaktivitäten auslösen (Prozessorbefehle verschiedener Klassen, E/A-Aktivitäten, Speicherbelastung) und dabei durch Eingabeparameter gesteuert werden können, was die Kenntnis der realen Arbeitslastverhältnisse voraussetzt. Über die Eingabeparameter erfolgt die Steuerung des Auftragsprofils über die Anzahl von Schleifendurchläufen mit bestimmten Aufträgen und die Größenbeschreibung für die Datenbereiche. Den Gegenpol zu diesem *ressourcenorientierten Ansatz* bildet der *funktionsorientierte Ansatz*, bei dem eine gezielte Nutzung bestimmter Funktionen eines Rechnersystems ausgelöst wird.
Bekannte synthetische Benchmarks sind:

- Whetstone - bestimmt die mittlere Befehlsausführungsrate eines Prozessors, das Resultat sind Whetstone-Befehle pro Sekunde. Der Befehlsmix wurde aus einer Analyse von über ca. 950 wissenschaftlichen Algol-60-Programme bestimmt und später auf andere Programmiersprachen übertragen. Moderne Methoden der Programmiertechnik, wie Unterprogramme und komplexe Datentypen nur wenig integriert. [Curn76]
- Dhrystone - entwickelt 1984 von Reinhold Weicker [Weic84]. Er basiert auf Datensammlungen aus Programmen unterschiedlicher Programmiersprachen, repräsentiert jedoch hauptsächlich Systemprogramme und ist daher für die Untersuchung der Tauglichkeit von Rechnern für kommerzielle und wissenschaftliche Anwendungen nur bedingt tauglich.

Die Vorgehensweise bei der vollständigen Durchführung eines Benchmarktest ist stark vom Testziel abhängig. Nach [Lang92] ist eine Gliederung in 5 wesentliche Schritte sinnvoll:

[2] millions of floating point operations per second

(a) Zielsetzung -

Was soll erreicht werden ?

- Vergleich von Rechensystemen
- Qualitative Diagnose von Softwareeigenschaften
- Test spezieller Hard- und Softwarekomponenten
- Aufspüren von Engpässen und Verbesserungsmöglichkeiten
- Vergleich von Leistungsgrößen bei Workload-Einfluß

(b) Zusammenstellen der Tests -

Hier erfolgt die schon beschriebene Auswahl der repräsentativen Arbeitslast.

(c) Fixierung der Rahmenbedingungen -

Die Benchmarkresultate unterliegen dem Einfluß einer Vielzahl von Bedingungen. Nur die Kenntnis dieser erlaubt eine realistische Interpretation und einen Vergleich verschiedener Messungen. Unbedingt zu beschreibende Eigenschaften sind:

- CPU mit Anzahl, Art (Befehlsvorrat, Befehlsklassen, Taktzeiten)
- Größe und Art des Hauptspeichers
- E/A-Subsystem (Bus, Controller, Harddisks)
- Größen von Pufferspeichern
- Softwareausstattung (Compiler, Dienstprogramme)
- Compilereinstellungen (Optimierung)
- Betriebssystem
- Betriebssystemstatus, aktuelle Systemlast (Workload)
- (Datum, Zeit)

Nur durch diese Beschreibungen ist eine Vergleichbarkeit der Benchmarkergebnisse gegeben und auch eine Reproduzierung des Resultate möglich. Genau diese Reproduzierbarkeit ist Voraussetzung für einen Vergleich verschiedener Systeme.

(d) Ausführung der Tests -

Dabei sollte das System überwacht werden, um unerwartete Ereignisse und Effekte erkennen und im Ergebnis berücksichtigen zu können. Dazu gehört auch die Überprüfung der Korrektheit und Vollständigkeit des Benchmarklaufes. Die erforderliche Konstanz der Rahmenbedingungen ist ebenfalls zu sichern bzw. zu überwachen.

(e) Ergebnisanalyse -

Hier erfolgt die Bewertung der ermittelten Leistungsmaße unter Einbeziehung wichtiger Einflußfaktoren, häufig erfolgt eine Einbeziehung ökonomischer Faktoren, wie Anschaffungs- und Betriebskosten.

Werden Benchmarktests durchgeführt, um Rechnerergebnisse zu vergleichen, so müssen für eine wirkliche Vergleichbarkeit die Rahmenbedingungen bei den Tests identisch sein. Daneben muß der Benchmark gewisse Anforderungen erfüllen:

- Offengelegter Algorithmus
- Offengelegte Implementierung
- Unabhängigkeit gegenüber Optimierungen der Compiler oder Beschreibung der Optimierungen in den Rahmenbedingungen

- Einheitliche Bereitstellung der Eingabewerte bei parametrisierten Benchmarks
- Cache-Unabhängigkeit oder quantitative Beschreibung der Cache-Einflüsse

Besonders der letzte Punkt stellt sich in der Praxis als sehr schwierig dar. Eine Cache-unabhängige Gestaltung von Programmcode und Datenmodellen ist praktisch kaum möglich. Wegen der Hardwarerealisierung der Caches ist eine Quantifizierung des Einflusses kompliziert. Die Überwachung der Trefferrate bei Cache-Zugriffen mittels Hardware-Monitor gestaltet sich sehr aufwendig und ist für den Anwender nahezu unmöglich. Eine Isolierung durch Cacheabschaltung und entsprechende Vergleichsmessungen sind ebenfalls nicht ohne weiteres durchführbar. Parametrisierbare Benchmarks erlauben eine gewisse Quantifizierung des Cache-Einflusses unter der Bedingung, daß alle Rahmenbedingungen völlig identisch sind.
Die meisten Anwender streben jedoch nicht die Entwicklung eigener Benchmarks an. Im Gegenteil, für die Durchführung vergleichender Messungen zu verschiedenen Systemen für Systemerweiterungen und Neuanschaffungen ist die Definition von Standard-Benchmarks sinnvoll. So stellen oder stellten die schon genannten Benchmarks Quasi-Standards dar. Immer wieder hat es von Seiten verschiedener Gremien Standardisierungsversuche gegeben. Ziel war die Einführung allgemein zugänglicher Bibliotheken von Standardprogrammen zur Generierung synthetischer Benchmarks. Diese Versuche scheiterten häufig an zahlreichen Problemen:

- Nichtausschließbarkeit der Bevorteilung von Systemen durch bestimmte Programme
- Zielgerichtete Optimierung bei Neuentwicklung von Produkten durch Hersteller mit direkter Orientierung auf die Benchmarks
- Nichtbestimmbarkeit des Einflusses optimierender Compiler
- Kosten für Unterhaltung, Aktualisierung, Berichtigung der Bibliotheken und die Überwachung der korrekten Anwendung

Resultat dieser Bemühungen sind verschiedene Benchmarksammlungen, sogenannte Suiten, mit denen jeweils bestimmte Funktionen oder Komponenten eines Systems getestet werden können. Bekannte Vertreter sind die verschiedenen SPEC-Suiten, die SSBA[3]-Suite und die TPC[4]-Datenbank-Benchmarks. Häufig werden von Fachzeitschriften eigene Benchmarksysteme für Vergleichstests entwickelt, für die dann aber nur eine bedingte Vergleichbarkeit gegeben ist.
Die SPEC-Benchmarks werden von diversen Rechnerherstellern administriert und weiterentwickelt, woraus sich aus Anwendersicht immer wieder Kritikpunkte ergeben, da die ermittelten Ergebnisse hauptsächlich den Herstellern als Verkaufsargument dienen und speziell die Rahmenbedingungen bei den Herstellermessungen weniger auf realistische, als vielmehr auf optimale Ergebnisse ausgerichtet sind. Genau dieser Punkt lieferte den Ansatz für diese Arbeit. Deshalb wird die Umsetzung der erörterten theoretischen Anforderungen an eine Leistungsmessung mit Benchmarks im nächsten Kapitel am Beispiel der SPECint95- und SPECfp95-Suiten betrachtet und mit Ansätzen anderer Gremien und Organisationen verglichen.
Vorher wird jedoch noch das zweite wichtige Verfahren der Klasse der Meßverfahren in der Leistungbewertung vorgestellt.

[3] Suite Synthetique des Benchmarks de l'AFUU
[4] Transaction Processing Performance Council

1.2.2 Monitore

Monitore sind Meßgeräte, mit denen operierende Rechensysteme beobachtet werden. In Rechensystemen laufen Prozesse ab und erzeugen eine Arbeitslast. Monitore erfassen die von dieser Arbeitslast ausgelösten Aktivitäten im System. Ziel ist die Beschreibung der charakteristischen Eigenschaften dieser Arbeitslast. Monitore sind daher gut zur Datenerfassung während eines Benchmarktests geeignet.
Ein Monitor läßt sich durch Schichten von funktionalen Einheiten beschreiben [Lang92]:

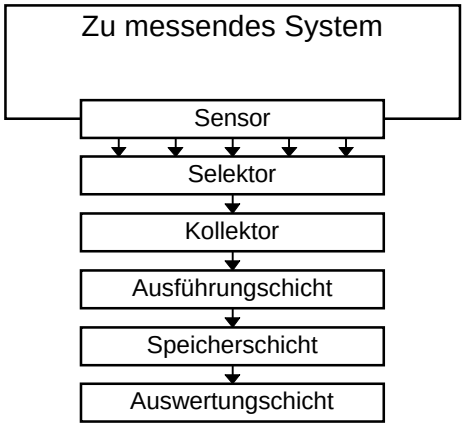


Abbildung 1-2 Schematische Darstellung eines Leistungsmonitors

Der Sensor ermittelt an bestimmten Meßstellen des Systems Meßgrößen und übergibt sie dem Selektor. Der Selektor ist ein Filter mit der Aufgabe, aus der eventuell sehr großen vom Sensor gelieferten Datenmenge die für die eigentlich gestellte Meßaufgabe interessanten Daten zu extrahieren. Der Kollektor sammelt Daten und verdichtet sie, beispielsweise durch Summierung oder Mittelwertbildung. In der Ausführungschicht des Monitors erfolgt die Verarbeitung und Aufbereitung der Rohdaten, um sie in weiterverabeitbarer und gut lesbarer Form in der Speicherschicht abzuspeichern. Die Auswertungsschicht verwendet die gespeicherten Daten, um sie unter statistischen Gesichtspunkten auszuwerten und um Beziehungen zwischen den ermittelten Maßen herzustellen.
Monitore können nach unterschiedlichen Gesichtspunkten in Klassen eingeteilt werden. Eine Möglichkeit ist die Klassifizierung nach der technischen Realisierung der Schichten Sensor-Selektor-Kollektor.

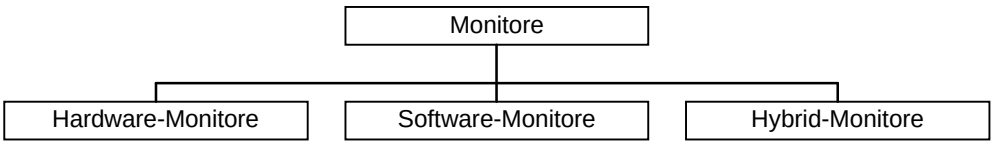


Abbildung 1-3 Monitorklassen nach technischer Realisierung

Hardwaremonitore sind Hardwarebausteine, die direkt am Rechensystem angeschlossen sind und elektrische Impulse registrieren und messen. Durch diese technische Realisierung werden Daten aus der Hardware-Schicht der Rechensystems gewonnen. Damit ist es sehr schwierig, Daten aus höheren Systemschichten mit Hardwaremonitoren zu erfassen. Der Vorteil besteht darin, daß der Monitor selbst

keine Ressourcen des betrachteten Rechensystems verbraucht. Typischerweise werden technische Systemzustände, wie Bus- und Prozessorstatus, sowie Statussignale erfaßt.
Softwaremonitore sind Prozesse, die parallel zum System laufen, sie können Teil des Betriebssystems oder Anwendungsprozesse sein. Die zu messenden Daten werden in den Softwareschichten (Betriebssystemschicht, Anwendungsschicht) oder der Firmwareschicht eines Rechnersystems nach Ablauf bestimmter Zeiten oder beim Eintreffen bestimmter Ereignisse gewonnen, erfaßt und aufbereitet. Typische erfaßte Daten sind die Betriebsmittelforderungen der im System aktiven Prozesse. Problematisch ist bei der Verwendung von Softwaremonitoren, daß diese selbst einen Teil der Systemressourcen für ihre Arbeit verbrauchen. Man bezeichnet diese Beanspruchung der Systemressourcen durch Monitore als Interferenz. Der Grad der Ergebnisbeeinflussung durch das Meßsystem selbst ist praktisch schwer bestimmbar, da hierfür ein interferenzfreies Meßsystem erforderlich ist. Ein solches muß, wie oben erwähnt, als Hardware-Monitor realisiert werden.
Kombinationen beider Monitorsysteme, bei denen die Komponenten Sensor, Selektor, Kollektor nicht eindeutig der Hardware- oder den Softwareschichten zugeordnet werden können, werden als *Hybridmonitore* bezeichnet.

Eine andere Möglichkeit zur Klasseneinteilung von Monitoren ist die Art und Weise, auf die die Messungen des Monitors ausgelöst werden:

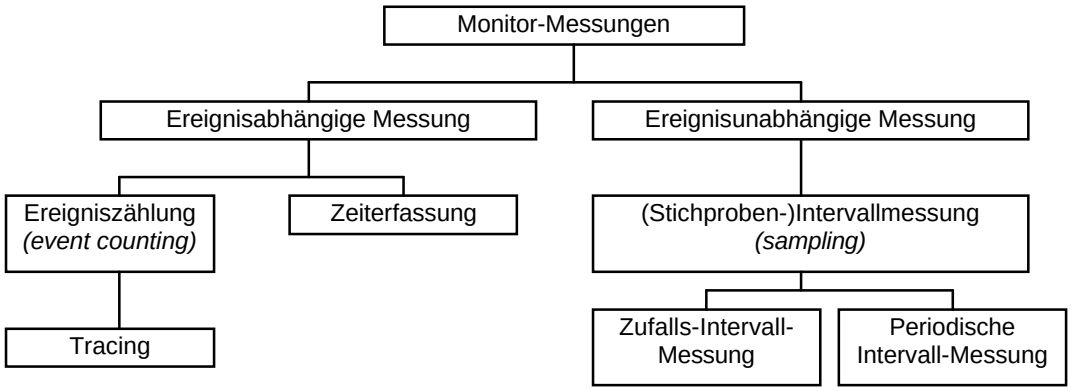


Abbildung 1-4 Monitoring nach Methode der Messungsauslösung

Das Verhalten eines Rechensystems wird durch eine Folge von Systemzuständen beschrieben. Durch bestimmte Ereignisse (*events*) erfolgen Zustandsübergänge zwischen den Systemzuständen. Dieser Vorgang läßt sich funktional beschreiben:

$$Systemzustand_{\,Zeitpunkt\ t} = f\,(Systemzustand_{\,Zeitpunkt\ t-1},\ Ereignis_{\,i})$$

Man unterscheidet beim Monitoring grundsätzlich nach ereignisabhängiger und ereignisunabhängiger Messung.
Ereignisabhängige Monitore werden nur bei bestimmten Ereignissen aktiv, die restliche Zeit sind sie inaktiv. Werden die zu messenden Ereignisse gezählt, bezeichnet man diese Methode als *Ereigniszählung (event counting)* . Man erhält ein Maß für die Häufigkeit der Ereignisse. Auf diese Weise werden beispielsweise verschiedene Fehlerarten im System oder Prozeduraufrufe erfaßt. Eine vollständige Aufzeichnung aller Ereignisse ohne Selektion oder Verdichtung wird auch als *Tracing*

bezeichnet. Dagegen wird bei der *Zeiterfassung* gemessen, wie lange ein Rechensystem in einem bestimmten Zustand verweilt. Diese Art erlaubt es, die Auslastung von Systemkomponenten zu ermitteln, indem man die Zeitintervalle aufaddiert, in denen sich das System im Beobachtungszeitraum in einem bestimmten Zustand befindet. Start und Ende der Zeitabschnitte sind durch die Ereignisse bestimmt, die das System in den oder aus dem interessierenden Zustand überführen.

Ereignisunabhängige Monitore führen in (von Ereignissen) unabhängigen Zeitabständen eine *Stichproben-Intervallmessung (sampling)* durch. Abhängig von der Art der Wahl der Meßintervalle unterscheidet man zwischen *Zufalls-Intervall-Verfahren (Random Verfahren),* bei denen die Beobachtungszeitpunkte zufällig gewählt werden und dem *periodischen Verfahren*, bei dem Messungen in fest definierten Intervallen erfolgen. Dabei dürfen Periodenlänge und das Auftreten von Ereignissen nicht korreliert sein. Die Stichprobenmessungen haben den wesentlichen Vorteil, daß die anfallende Datenmenge begrenzt bleibt. Außerdem sinkt die Interferenz des Monitors mit wachsender Intervallänge. Nach der Messung ist es erforderlich, die Stichprobenergebnisse statistisch auszuwerten (Auswertungsschicht).

Beschränkungen für die Anwendung von Monitoren ergeben sich aus den „kleinsten“ meßbaren Ereignissen jener Systemschicht, in der der Monitor agiert, und der Anzahl der in einer Zeiteinheit registrierbaren Ereignisse (Bandbreite). Für realistische Ergebnisse ist die Definition eines akzeptablen Interferenzgrades wichtig, danach kann entschieden werden, auf welcher Systemschicht das Monitoring erfolgen kann.

Moderne Systeme verfügen über eine Reihe eingebauter Monitoring-Mechanismen, mit denen Systemaktivitäten überwacht werden, und aus denen verschiedene Dienstprogramme Systemdaten bereitstellen können. Dem Anwender stehen damit die Schichten Sensor, Selektor, Kollektor und teilweise auch Ausführungsschicht und Speicherschicht bereits zur Verfügung. Dem Nutzer bleibt die Aufgabe, Ausführungs- und Speicherschicht an seine Bedürfnisse anzupassen und die Auswertungen nach eigener Interessenlage vorzunehmen. Aus der Sicht des Anwenders erfolgt das Monitoring dabei auf dem Abstraktionsniveau einer bestimmten Systemschicht, im einfachsten Fall direkt in der Anwendungsschicht. Stellt die betrachtete Schicht nicht die gewünschten Daten bereit, so muß das Monitoring auf eine niedrigere Abstraktionsebene und damit Systemschicht verlagert werden, im Extremfall bis auf die Hardwareschicht mit einem Hardwaremonitor.

2 Leistungsbewertung mit Benchmarksystemen

2.1 Leistungsbewertung mit SPEC-CPU-Benchmarks

Die **Standard Performance Evaluation Corporation (SPEC)** ist eine 1988 gegründete Organisation, die sich als Ziel gesetzt hat, realitätsnahe Benchmarkprogramme zu entwickeln, die leicht auf viele Rechnerplattformen portiert werden können und einen fairen Vergleich der Leistungsfähigkeit von Rechnern unterschiedlicher Hersteller zulassen. Neben den großen Rechnerherstellern findet man unter den Mitgliedern auch eine Reihe von Softwarehäusern, Verlagen und wissenschaftlichen Einrichtungen. Zu den deutschen Mitgliedern gehört auch die TU Chemnitz-Zwickau als assoziiertes Mitglied. In jedem SPEC-Newsletter [SPEC9x_x] oder in [SPECwww] findet man eine vollständige Liste der SPEC-Mitglieder.
Die SPEC hat Pakete von Benchmarks, sogenannte Suiten, für verschiedene Anwendungsgebiete entwickelt. Neben den Benchmarks zur Bewertung der Prozessorleistung (CINT, CFP), die Gegenstand der Untersuchungen dieser Arbeit sind, werden Pakete zur Bewertung der Fähigkeiten im UNIX-Multiuser-Betrieb (Software Development Multiuser - SDM) und zur Bewertung der Leistung bei der Arbeit im Netzwerk mit NFS (System Benchmark Fileserver - SFS) angeboten. In der Entwicklung befinden sich weitere Benchmarksysteme, u.a. für die Grafik-Leistungsfähigkeit und für WWW-Server. Die SPEC-Benchmarks sind kommerzielle Softwareprodukte, die von Nichtmitgliedern käuflich erworben werden müssen.
Alle diese Benchmarksuiten testen jedoch immer nur spezielle Komponenten oder ein Teilsystem eines komplexen Rechnersystems. Für die Auswahl von Rechensystemen für spezielle Anwendungsgebiete ist dieses Herangehen sehr gut geeignet. Anhand der Benchmarkergebnisse auf dem Zielanwendungsgebiet ist es möglich, das für die geplante Anwendung optimale System auszuwählen. Für die Auswahl eines Computersystems für universelle Anwendungen ist es jedoch notwendig, mehrere Komponenten, beispielsweise Prozessor, Speichersystem, Betriebssystem, Compiler, Netzanbindung ... , in ihrem Zusammenwirken zu betrachten. In dieser Richtung wird bei SPEC die Entwicklung heterogener Benchmarks diskutiert, bisher stehen allerdings noch keine Werkzeuge dieser Art zu Verfügung.
Die Entwicklungsarbeit zu den Benchmarks wird primär von den Hardwareherstellern geleistet. Sie sind auch die Hauptnutzer der Bewertungssysteme, indem sie bereits im Entwicklungsstadium neuer Produkte (Hardware und zugehörige System- und Anwendungssoftware) die Benchmarks anwenden, um Entwurfsentscheidungen zu bestätigen oder zu korrigieren. Die Leistungsfähigkeit neuer Produkte wird mit diesen Benchmarks gemessen und die Werte werden veröffentlicht. In der Folge dienen die ermittelten Leistungswerte als wichtiges Verkaufsargument für die Vertriebsabteilungen der Hersteller. Die Testergebnisse der Hersteller werden im vierteljährlich erscheinenden SPEC-Newsletter veröffentlicht und sind im WorldWideWeb [SPECwww] zugänglich. Eine umfangreiche Sammlung von verschiedenen Benchmarkergebnissen findet man ebenfalls im WWW in [PERFwww].
Speziell die SPEC-Kennzahlen zur Prozessorleistungsfähigkeit (SPECint9x, SPECfp9x) haben sich im Bereich der UNIX-Rechnersysteme zu einem Quasi-Standard entwickelt. Diese Leistungsangaben findet man beinahe bei jedem Rechnertest in diversen Zeitschriften und in den Datenblättern der Hersteller.

Damit ist das ursprüngliche Ziel erreicht, Produkte unterschiedlicher Hersteller vergleichbar zu machen. Allerdings wird davor gewarnt, blind auf die von den Herstellern gemessenen Werte zu vertrauen, ohne die genauen Konfigurationsdetails, Randbedingungen und die Differenzierung der Werte der Einzelbenchmarks zu betrachten. In der Vergangenheit wurden häufig geschönte Leistungsdaten, resultierend aus exotischen Konfigurationen und speziellen, auf die Benchmarks ausgerichteten Optimierungen, veröffentlicht. Viele Fachzeitschriften machen deshalb bei Hardware-Tests eigene Messungen, deren Ergebnisse teilweise erheblich differieren [Siev_iX_96]. Zur Wahrung der Vergleichbarkeit von Ergebnissen aus unterschiedlichen Quellen hat die SPEC sich selbst und allen Nutzern der Benchmarks strenge Richtlinien hinsichtlich der Durchführung und Veröffentlichung von Tests auferlegt, die SPEC Run Rules und die SPEC Report Rules [SPEC_doc]. In diesen Regeln finden sich auch eine ganze Reihe von Bestimmungen, die genau die in Kapitel 1 genannten Anforderungen an Benchmarks [Lang92] umsetzen.
An der TU Chemnitz-Zwickau erfolgen Anwendungen von Benchmarks mit mehreren Zielen:

- Analyse der vorhandenen, laufenden Rechensysteme, um notwendige Erweiterungen dieser Systeme erkennen und begründen zu können
- Untersuchungen an Referenzmaschinen verschiedener Hersteller, um die für die universitären Bedingungen am besten geeigneten Plattformen zu ermitteln, insbesondere ist dabei das Verhalten bei unterschiedlichen Lastverhältnissen von Interesse
- Untersuchungen beim „Interworking“ verschiedener Plattformen innerhalb des Netzwerkes von URZ und der Fachbereichsrechenzentren

Mit diesen Zielen erfolgt speziell eine Auseinandersetzung mit den SPEC-CPU-Suiten und der SPEC-SFS-Suite, da diese grundsätzlich die Voraussetzungen für die Untersuchung von zwei wichtigen Elementen der Realisierung des Campusnetzes bieten. Es geht dabei einerseits um die Untersuchung der Rechenleistungen der für allgemeine Rechenanwendungen zur Verfügung gestellten Compute-Server (und der vorhandenen Referenzmaschinen zur Auswahl weiterer Compute-Server) mit den CPU-Benchmarks. Ein zweiter Punkt ist die Bewertung der Leistungsfähigkeit der vorhandenen Fileserver im Hinblick auf die Leistungsfähigkeit des globalen Universitäts-Filesystems (/uni/global) mit Hilfe des SFS-Benchmarks. Erfahrungen aus den Anwendungen der Testssuiten findet man in [Mund95] zu SPEC-CPU und in [Hofb96] zu SPEC-SFS. Ausgehend von den Erfahrungen in [Mund95] soll in dieser Arbeit das Verhalten verschiedener URZ-Rechner bei unterschiedlichen Lastbedingungen (*workload*) und die Auswirkungen auf die Ergebnisse der SPEC-CPU-Benchmarks auf diesen Rechnern untersucht werden.

2.1.1 Inhalt der SPEC-CPU-Benchmarks CFP95 und CINT95

Grundsätzlich unterscheidet SPEC die CPU-Benchmarks in zwei Benchmarksuiten, denen das gleiche Bewertungsverfahren zugrunde liegt, die das zu untersuchende Rechnersystem jedoch mit unterschiedlichen Profilen hinsichtlich der erzeugten Arbeitslast belegen.
Die **Floating Point Suite CFP95** besteht aus 10 Einzelbenchmarks, die alle aus praktischen numerischen Anwendungen abgeleitet wurden und besonders die Fließkomma-Arithmetik von Rechnersystemen beanspruchen. Alle Benchmarks dieser Suite sind in Fortran unter Verwendung von doppeltgenauen Gleitkommazahlen implementiert. Eine Ausnahme bildet 102.swim, bei dem Gleitkommazahlen einfacher Genauigkeit verwendet werden. Wie für numerische Anwendungen typisch, sind relativ große Datenmengen zu verarbeiten, so daß neben der Prozessorleistungsfähigkeit auch die Leistungsfähigkeit des gesamten Speichersystems einen starken Einfluß auf die Benchmarkergebnisse nimmt. Der Einfluß des Ein-/Ausgabe-Systems sollte durch Parametrisierung weitgehend minimiert werden. Die Tabelle 2-1 zeigt jeweils eine Kurzbeschreibung für jeden Einzelbenchmark und die offiziellen Referenzlaufzeiten, die von SPEC auf einer SPARCstation 10 Model 40 ohne Cache gemessen wurden. Diese Beschreibungen wurden [SPEC95_3] und [Weic_El_96] entnommen.

Benchmark-Name	Anwendungsgebiet	Referenzzeit Sekunden	Beschreibung
101.tomcatv	Flüssigkeitsdynamik	3700	Netzgenerierung um allgemeine geometrische Bereiche
102.swim	Wettervorhersage	8600	Lösung von "Wasser-Verflachungs-Gleichungen" mit finiter Differenzen-Approximation (Fließkommazahlen mit einfacher Genauigkeit)
103.su2cor	Quantenphysik	1400	Berechnung der Massen von Elementarteilchen nach der Quark-Gluon-Theorie
104.hydro2d	Astrophysik	2400	Lösung hydrodynamischer Navier-Stokes-Gleichungen zur Berechnung galaktischer Strömungen
107.mgrid	Elektromagnetismus	2500	Berechnung eines dreidimensionalen Potentialfeldes
110.applu	Flüssigkeitsdynamik	2200	Lösung linearer Gleichungssysteme über Pivots
125.turb3d	Simulation	4100	Simuliert Verwirbelungen in einem dreidimensionalen Raum
141.apsi	Wettervorhersage/ Statistik	2100	Berechnet Statistiken zur Temperatur- und Niederschlagsverteilung
145.fpppp	Chemie	9600	Ableitungen und Berechnungen bei Multi-Elektronen-Systemen
146.wave5	Elekrodynamik	3000	Lösung Maxwell'scher Gleichungen in einem kartesischen Koordinatensystem

Tabelle 2-1 Anwendungsklassen, Kurzbeschreibungen und Referenzzeiten auf SPARCstation 10 Model 40[5] für CFP95

Die **Integer Suite CINT95** setzt sich aus 8 Benchmarks zusammen, die aus praktischen Anwendungen abgeleitet wurden, die jeder Nutzer im UNIX-Umfeld (zumindest teilweise) häufig benutzt. Die Bezeichnung als Integer-Suite führt oftmals zu der irrigen Annahme, daß es sich bei diesen Benchmarks

[5] Offizielle SPEC-Referenzmaschine für CFP95 und CINT95

um spezielle arithmetische Anwendungen handelt. Die Bezeichnung dient jedoch nur der Abgrenzung von den speziellen Fließkomma-Benchmarks. Die Suite CINT95 testet durch verschiedene Anwendungen die gesamte Breite der Prozessorleistungsfähigkeit, mit Ausnahme der Fließkomma-Arithmetik. Dies kommt auch in der Tabelle 2-2 der in CINT95 enthaltenen Anwendungen zum Ausdruck, die ebenfalls aus [SPEC95_3], [Weic_El_96] entnommen wurde.

Benchmark-Name	Anwendungsgebiet	Referenzzeit Sekunden	Beschreibung
099.go	Spielprogramm, künstliche Intelligenz	4600	Der Rechner spielt "GO" mit verschiedenen Feldgrößen gegen sich selbst
124.m88ksim	Simulation	1900	Simuliert den Motorola 88100-Prozessor bei der Abarbeitung eines Speichertestes und dem Dhrystone-Benchmark
126.gcc	Compiler	1700	cc1-Phase des GNU-C-Compilers (Übersetzt vorübersetzten Sourcecode in optimierten SPARC-Maschinencode)
129.compress	Textkompression	1800	Komprimierung eines 16 MByte großen Textfiles mit Lempel-Ziv-Methode
130.li	LISP-Interpreter	1900	Ein in c geschriebener LISP-Interpreter führt ein LISP-Programm aus
132.ijpeg	Bildverarbeitung	2400	Bildkomprimierung mit verschiedenen Parametern
134.perl	Kommandointerpreter	1900	Textverarbeitung und numerische Berechnungen mit Primzahlen in PERL-Programmen
147.vortex	Datenbank	2700	Aufbau und Manipulationen von drei in Beziehung stehenden Datenbanken

Tabelle 2-2 Anwendungsklassen, Kurzbeschreibungen und Referenzzeiten auf SPARCstation 10 Model 40 für CINT95

Die Ergebnisse der einzelnen Benchmarks werden, je nach Art der Anwendung, neben der Leistungsfähigkeit der CPU zusätzlich durch das Speichersystem und/oder das Ein- und Ausgabesystem des zu testenden UNIX-Computersystems bestimmt.
Die SPEC empfiehlt den Anwendern, alle Einzelmessungen zu betrachten bzw. durchzuführen, oder aber solche Einzelbenchmarks auszuwählen, die von der Anwendungsart den realen Applikationen des Anwenders ähneln. In diesem Zusammenhang muß kritisch angemerkt werden, daß seitens der SPEC nur wenige Angaben über die genauen Anforderungen der einzelnen Tests bereitgestellt werden. Für genaue Untersuchungen bleibt es dem Anwender überlassen, die, wie in [Lang92] bei Benchmarks gefordert, offengelegten Quellen zu analysieren, um präzise Angaben zu den Ressourcenforderungen und Testprofilen der einzelnen Programme zu erhalten. [Weic_El_96] enthält Angaben zu den statischen Speicheranforderungen der Benchmarks. Speziell die Integer-Benchmarks arbeiten jedoch auch mit großen dynamischen Datenmodellen, so daß die angegebenen Werte nur bedingte Aussagekraft besitzen. In [SPECwww] findet man seit kurzem Ergebnisse von Analysen der Einzelbenchmarks. Versuche mit den im Herbst 1995 neu veröffentlichten CPU-Benchmarks der 95er Generation haben gezeigt, daß derzeit übliche Arbeitsplatzrechner die Ressourcenforderungen der Benchmarks häufig nicht erfüllen können [Siev_iX_96], [Mund95]. Aus dieser Tatsache erwächst eine weitere Motivation für die Untersuchungen in dieser Arbeit, nämlich eben solche Ressourcenengpässe und deren Auwirkungen auf beliebige Anwendungen möglichst genau zu lokalisieren.

2.1.2 Meßverfahren der SPEC-CPU-Benchmarks CFP95 und CINT95

Grundsätzlich unterscheidet man für beide SPEC-CPU-Suiten 2 Bewertungsverfahren, die Geschwindigkeitsmessung und die Durchsatzmessung.
Bei der Geschwindigkeitsmessung wird die Abarbeitungszeit für jeden Einzelbenchmark der CFP- bzw. CINT-Suite auf dem zu testenden Rechner gemessen. Diese Abarbeitungszeit wird in Relation zur Abarbeitungszeit auf der sogenannten SPEC-Referenzmaschine (SPARCstation 10/40 ohne Cache) gebracht. Dieser Wert bildet das *SPECratio* für den entsprechenden Einzelbenchmark, das in jeder offiziellen Veröffentlichung von Testergebnissen angegeben werden muß.

$$SPECratio_{Benchmark\ i} = \frac{Referenzzeit_{Benchmark\ i}\ [\mathrm{s}]}{Laufzeit_{Benchmark\ i}\ auf\ Testmaschine\ [\mathrm{s}]}$$

Aus den SPECratio aller Benchmarks einer Suite wird der bekannte Gesamtwert SPECfp95 beziehungsweise SPECint95 *als geometrisches Mittel* berechnet.

$$SPECfp95 = \sqrt[10]{\prod_{\forall Benchmark\ i \in CFP95} SPECratio_{\ Benchmark\ i}} \qquad \text{bzw.}$$

$$SPECint95 = \sqrt[8]{\prod_{\forall Benchmark\ i \in CINT95} SPECratio_{\ Benchmark\ i}}$$

Die Messung der Abarbeitungszeiten und die daraus resultierenden Relativgeschwindigkeiten bezüglich der Referenzmaschine reichen jedoch nicht aus, um die Prozessorleistungsfähigkeit von Multiprozessorsystemen zu beschreiben. Dies liegt darin begründet, daß es sich bei den Benchmarks um sequentielle Prozesse handelt, die jeweils nur auf einer CPU laufen. Eine einfache Hochrechnung auf die Prozessorenzahl berücksichtigt nicht den Aufwand für die Koordinierung und eventuelle Ressourcenengpässe und ist damit nicht praktikabel. Aus diesem Grund wurde von SPEC ein weiteres Bewertungsverfahren für die Prozessorleistungsfähigkeit definiert, die sogenannte „Homogeneous Capacity Method"- Durchsatzmeßmethode. Dabei wird bestimmt, wieviel gleichartige Aufträge (Programme) in einer bestimmten Zeit ausgeführt werden können. Dazu werden mindestens so viele „Kopien" (Prozesse) eines Benchmarks gleichzeitig gestartet, wie das Rechnersystem Prozessoren besitzt. Es wird die Zeit vom Start des ersten bis zum Ende des letzten Benchmarkprozesses gemessen. Aus der gemessenen Zeit berechnet man die Leistungskennzahl für die Durchsatzleistung für den entsprechenden Benchmark, die als *SPECrate* bezeichnet wird. Die Berechnung erfolgt als Anzahl der benchmarkspezifisch normalisierten Aufträge , die das System an einem Tag ausführen kann.

$$SPECrate_{\ Benchmark\ i} = \#\ Kopien * \frac{Referenzzeit_{Benchmark i}\ [\mathrm{s}]}{längste\ Referenzzeit_{Suite}\ [\mathrm{s}]} * \frac{86400\ [\mathrm{s}]}{Laufzeit_{Benchmark\ i}\ mit\ \#\ Kopien\ auf\ Testmaschine\ [\mathrm{s}]}$$

Die längste Referenzzeit der CFP95-Suite beträgt 9600 Sekunden (145.fpppp), die der CINT95-Suite 4600 Sekunden (099.go). Der Wert von 86400 Sekunden resultiert aus der Anzahl von Sekunden eines

Tages. Analog zu den Geschwindigkeitsmessungen wird die Kennzahl für die gesamte Suite als *geometrisches Mittel* der einzelnen SPECrate-Werte gebildet, wobei bei einer Veröffentlichung alle Einzelwerte angegeben werden müssen.

$$SPECfp\_rate95 = \sqrt[10]{\prod_{\forall Benchmark\ i \in CFP95} SPECrate_{\text{Benchmark i}}} \qquad \text{bzw.}$$

$$SPECint\_rate95 = \sqrt[8]{\prod_{\forall Benchmark\ i \in CINT95} SPECrate_{\text{Benchmark i}}}$$

Bei Einprozessorsystemen ergeben sich die Durchsatzwerte direkt aus den Werten der Geschwindigkeitsmessungen, deshalb ist keine Veröffentlichung vorgeschrieben. Die Run- und Report-Rules der SPEC erlauben grundsätzlich auch Durchsatzmessungen mit einer größeren Anzahl von Benchmarkkopien gegenüber der Anzahl der Prozessoren. Bedingt durch Architektur und Konfiguration des Testsystems können sich daraus bessere Durchsatzwerte ergeben, wobei dieser Effekt primär durch die Befehlscaches eines Systems hervorgerufen wird.
Noch während der Gültigkeit der vorherigen Benchmarkgeneration (CFP92, CINT92) wurde eine weitere Differenzierung der nach eben beschriebenen Verfahren ermittelten Leistungskennzahlen eingeführt, die sogenannten *Baseline-Messungen*. Eine Vielzahl von Veröffentlichungen beschäftigte sich vorher mit dem Einfluß von Compileroptimierungen auf die Benchmarkresultate, z.B. [Weic_iX_94], [ChSuWo94].
Die mit der 95er Benchmarkgeneration obligatorisch gewordenen Baseline-Messungen erlauben den Einsatz von maximal 4 Compilerschaltern bei der Übersetzung der Benchmarks, wobei für die gesamte Suite dieselben Flags zu verwenden ist. Zu diesen Flags dürfen keine „Sammelflags“ und keine „unsicheren“ Flags gehören. Vor der Einführung dieser Baseline-Messungen war es unter den Herstellern verbreitet, jeden Benchmark mit einer Vielzahl von speziellen Schaltern zu optimieren, die jedoch in der Praxis selten eingesetzt werden. Oftmals bewirkten die Flags bei einzelnen Programmen rein zufällig eine Beschleunigung, während andere Anwendungen langsamer werden oder die Flags gar zum Abbruch führen („unsichere“ Flags). Da tage- oder wochenlange Optimierungsversuche nicht das Ziel des typischen Rechneranwenders sind, wurden für eine bessere Überschaubarkeit der Ergebnisse und für die Erhaltung der herstellerübergreifenden Vergleichbarkeit die Baseline-Messungen zur Pflicht gemacht. Vollständig optimierte Messungen können daneben weiterhin durchgeführt werden. SPEC selbst empfiehlt hauptsächlich die Betrachtung der Baseline-Werte.
Damit gibt es bei einer vollständigen Leistungsmessung mit den SPEC-CPU-Benchmarks insgesamt 8 Kennzahlen für die CPU-Leistungsfähigkeit eines Rechnersystems:

Baseline:	Optimierte Werte:
SPECfp_base95	SPECfp95
SPECfp_rate_base95	SPECfp_rate95
SPECint_base95	SPECint95
SPECint_rate_base95	SPECint_rate95

2.1.3 Erfahrungen aus bisherigen Anwendungen an der TU

In [Mund95] wurden die Möglichkeiten der Anwendung der SPEC-Benchmark-Suiten CFP95 und CINT95 zur Gewinnung von Aussagen zu den bereits genannten Anwendungszielen im Universitätsrechenzentrum untersucht, wobei für die praktischen Messungen eine Vorversion der 95er Benchmarkgeneration (CFP95.29 bzw. CINT95.32) zur Verfügung stand. Es wurde herausgearbeitet, daß eine Arbeit mit Anwendungsmethoden analog zu denen der SPEC-Mitglieder für die Untersuchungen am Campus-Netz wenig geeignet ist.
Um den Anspruch der Vergleichbarkeit der Meßergebnisse zu wahren, werden die Leistungsmessungen von den Herstellerfirmen unter stark idealisierten Meßbedingungen durchgeführt. Die zu untersuchenden Rechner sind dabei von der übrigen Rechnerwelt isoliert. Die Benchmarks laufen als einzige Anwendung auf dem zu testenden System, das heißt, obwohl Rechner mit UNIX-Systemen (Multi User System, Multi Processing System) getestet werden, werden diese für Leistungsuntersuchungen quasi in einen Einzelbetriebsmodus versetzt (Single User State). Bei den Durchsatzmessungen ist eine genau auf die Ressourcen des Systems (Prozessorenzahl) abgestimmte Anzahl von völlig identischen Anwendungen aktiv. Aus der Isolierung der Rechner folgt zwangsläufig, daß Benchmarkprogramme und zur Abarbeitung benötigte Datenbasen auf den lokalen Sekundärspeichern des Testsystems zur Verfügung stehen, so daß Ein- und Ausgabezeiten weitgehend minimiert werden. In Zeiten starker Vernetzung, Client-Server-Architekturen und verteilter Filesysteme ist diese Isolierung als etwas realitätsfern zu bewerten. Sie garantiert aber gleichzeitig eine Minimierung und Fixierung der Einflußfaktoren auf die Meßergebnisse. Die Ausstattung der Testrechner ist bei Herstellertests grundsätzlich so gewählt, daß während der Testläufe keine Ressourcenengpässe auftreten (große Hauptspeicher, große Caches, oder auch viele Harddisks an einer großen Zahl von Controllern beim SFS-Benchmark). Nicht immer entsprechen die getesteten Konfigurationen denen, die typischerweise verkauft und bei Anwendern eingesetzt werden. Die Preis- und Kostenkriterien finden bei den SPEC-Benchmarks keinerlei Berücksichtigung.

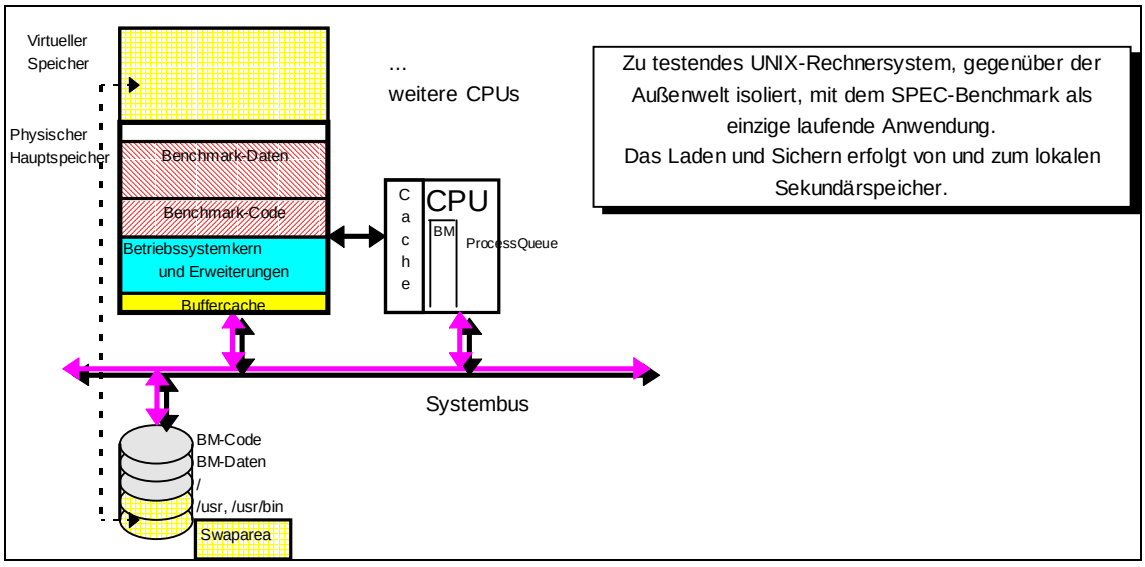


Abbildung 2-1 Modellbild der SPEC-Meßbedingungen

Die Wahl solcher Meßbedingungen für SPEC-CPU-Messungen ist als durchaus legitim zu bewerten. SPEC begründet die Bedingungen speziell damit, daß es sich bei **C**INT95 und **C**FP95 um sogenannte „***C***omponent-Benchmarks“ handelt, die spezielle Teilsysteme eines Rechensystems untersuchen (Prozessor, Speicherhierarchie inklusive Caches). Zum Test der Leistung beim Multi-Using oder der Leistungsfähigkeit bei Dateiein- und -ausgabe stehen andere Benchmarks (SDM bzw. SFS) zur Verfügung. Allerdings sind bei der praktischen Anwendung alle Komponenten in ihrem Zusammenwirken untrennbar verbunden, so daß Werte zu einzelnen Teilsystemen nicht unbedingt die Leistungsfähigkeit des Gesamtsystems widerspiegeln. In diesem Zusammenhang ist die Leistungsbeschreibung in Verkaufsprospekten oder in Hardwaretests von Zeitschriften, die häufig einseitig durch SPEC-CPU-Werte erfolgt, kritisch zu sehen. Durch die Angabe von SPEC-CPU-Leistungswerten bei kompletten Rechnersystemen entsteht häufig der Eindruck, es wird das Gesamtsystem bewertet. SPEC selbst fordert immer wieder auf, die Werte differenziert zu betrachten, was bereits damit beginnt, neben den SPEC*95-Werten auch die Ratio-Werte der Einzelbenchmarks zu verwenden. Für die Zukunft ist bei SPEC die Entwicklung heterogener Benchmarks geplant, die dem Zusammenwirken der Teilsysteme unter differenzierten Bedingungen stärker Rechnung tragen.

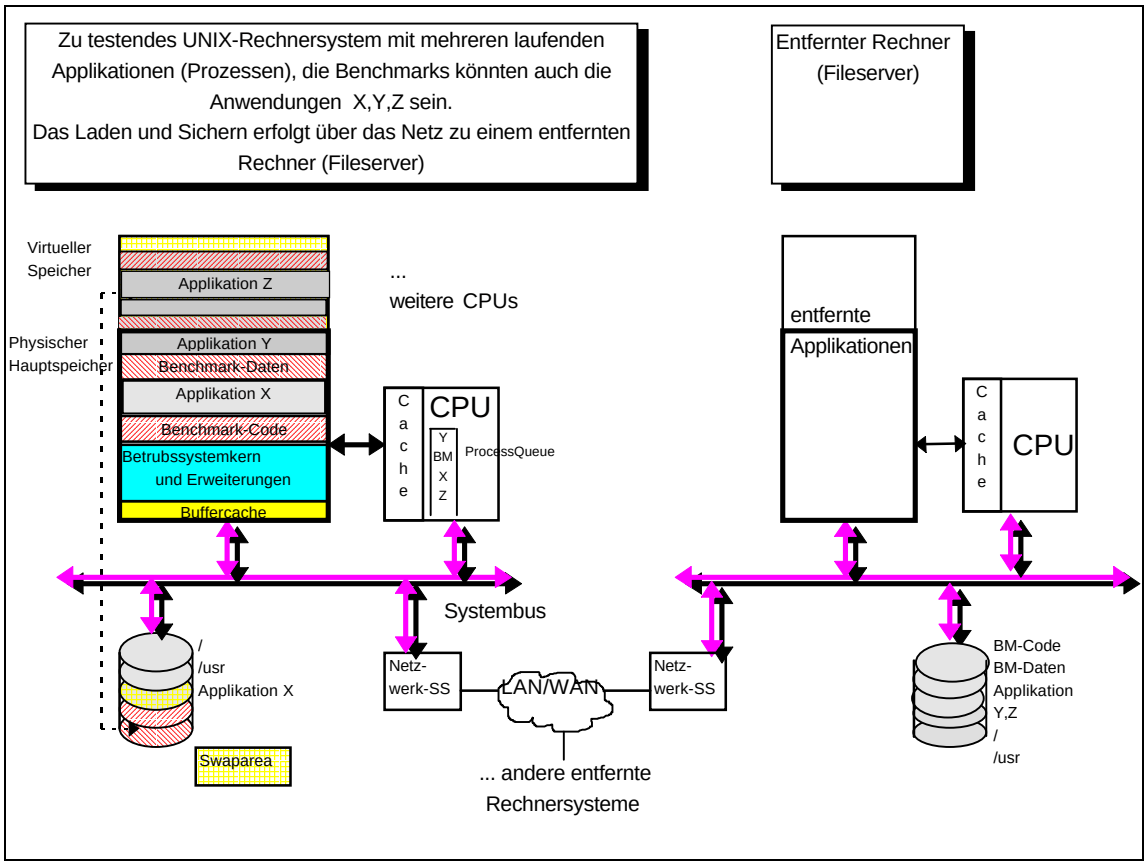


Abbildung 2-2 Modell eines praktischen Meßszenarios

Die Messungen an Rechnern des Universitätsrechenzentrums wurden unter stark von den typischen SPEC-Verhältnissen abweichenden Bedingungen durchgeführt. Die getesteten Compute-Server bzw. Workstations (Referenzmaschinen) waren während der Messungen voll im Universitätsnetz integriert, neben den Benchmarks waren unterschiedlich viele andere Applikationen auf den Rechnern aktiv. Ausführbare Benchmarks und deren benötigte Ein- und Ausgabedateien befanden sich nicht auf lokalem Sekundärspeicher, sondern auf einem entfernten Fileserver. Sie mußten zu Beginn und während der

Testläufe über das Netz (Ethernet, FDDI) geladen oder gespeichert werden. Neben dem reinen Transportaspekt (Auslastung der zwischenliegenden Netzstrecke, Transportverzögerung) spielen auch die Lasten auf anderen Rechnern und die konkurrierenden Ressourcenforderungen entfernt laufender Anwendungen auf gemeinsame Ressourcen, wie Übertragungskanäle und Fileserver, eine wichtige Rolle. Besonders drastisch wirkt es sich aus, wenn bereits der Testrechner die hohen Speicherforderungen der Benchmarks nicht erfüllen kann und dadurch beträchtliche Zeit für das Ein- und Auslagern von Speicherseiten durch die virtuelle Speicherverwaltung benötigt wird. Hierbei kommt neben dem Verwaltungsaufwand im Betriebssystem auch die Leistungsfähigkeit des lokalen Sekundärspeichersystems zum tragen.
Als Folge dieser Meßbedingungen sind auf diese Weise gewonnene Benchmarkresultate nicht mit denen vergleichbar, die von SPEC bzw. den Mitgliedsfirmen gemessen werden. Je nach Lastbedingungen werden dabei Abweichungen um Faktor 1,... bis Faktor 15 in den Laufzeiten der Benchmarks ermittelt. Eine Anpassung der Meßbedingungen an die Verhältnisse wie bei Herstellermessungen ist bei den URZ-Rechnern nicht möglich, da hierfür eine tagelange Isolierung der Rechner, sowie ein erheblicher Umbau in der Anwendung des globalen Universitäts-Filesystems (/uni/global) erforderlich wäre. Außerdem erscheint eine solche Maßnahme als nicht sinnvoll, da im Normalfall für aktuelle Maschinen der großen Hersteller die Leistungswerte aus Herstellermessungen vorliegen, beziehungsweise für ältere Maschinen die Meßwerte der Benchmarkgeneration SPEC92. Eigene Messungen unter nachgestellten Bedingungen müssen zu den gleichen Ergebnissen führen, wobei auch solche nachgestellten Messungen eine nicht unerhebliche Streuung der Resultate aufweisen können [Siev_iX_96]. Für „exotische“ Maschinen oder Rechner mit gegenüber der Standardkonfiguration modifizierter Ausstattung ließ sich bei der in [Mund95] untersuchten Vorabversion der 95er SPEC-CPU-Benchmarks eine Näherung für die CPU-Leistungswerte ermitteln. Die Zeitmessung erfolgt hier mit dem UNIX-Kommando „*/bin/time*“, das neben der Abarbeitungszeit für einen Prozeß auch die tatsächlich zugeteilte Prozessorzeit für die Abarbeitung dieses Prozesses, differenziert nach Prozessorzeit im User- und im Systemstatus, liefert. Die Summe dieser tatsächlichen Prozessorzeiten, in Verbindung mit der vorausgesetzten SPEC-Bedingung, daß die Benchmarks möglichst wenig I/O-Operationen enthalten, ist als Näherung für die Laufzeit eines Benchmarks auf einer unbelasteten Maschine geeignet. Diese inoffizielle Auswertungsmöglichkeit ist in der im Herbst 1995 veröffentlichten, offiziellen Version SPEC95.65 der SPEC-CPU-Benchmarks nicht mehr vorhanden. Hier wurden Steuerung, Zeitmessung und Auswertung der Resultate, sowie Prüfung der Korrektheit auf eine völlig neue, auf *GNU Make 3.74* und *Perl 5.001* basierende Ablaufsteuerung umgestellt. Diese hat den Vorteil, daß mit Perl auch die vollständige Auswertung der Benchmarkläufe erfolgen kann, um zu den SPEC-Report-Rules konforme Ergebnislisten verschiedener Formate (ASCII, HTML, Postscript) zu erzeugen. Dafür wurde die Zeitmessung mit *time* und *Bourne-Shell* aufgegeben. Die Zeitmessung erfolgt innerhalb der Perl-Ablaufsteuerung durch Messung von absoluter Start- und Endzeit eines Benchmarks mit der Perl-Funktion *time*.
Eine zusätzliche Messung der tatsächlich zugeteilten Prozessorzeiten läßt sich jedoch mit geringem Aufwand realisieren. Die notwendigen Schritte bei der Änderung der Ablaufsteuerung sind in der Anlage A dieser Arbeit dokumentiert. Damit ist es möglich, die Einzelwerte und die Summe der zugeteilten Prozessorzeit für den Benchmarkprozeß (USER-Mode) und von diesem ausgelöste Systemaktivitäten (SYS-Mode) zu bestimmen. Diese Prozessorzeit bildet eine Näherung für die Benchmarklaufzeit auf einer unbelasteten Maschine unter SPEC-ähnlichen Bedingungen. Bei den praktischen Untersuchungen in dieser Arbeit wurden neben der Ablaufzeit der Benchmarks auch die Prozessorzeiten ermittelt, da keine vollständig vergleichbaren SPEC-Resultate für die modifizierten

Universitätsrechner verfügbar sind. Diese Prozessorzeiten werden in der Auswertung der Messungen zur Normierung der lastbedingten Verzögerung der Benchmarks auf den unterschiedlich schnellen Rechnern verwendet und dort als maschinenspezifische Referenzzeit bezeichnet.
Generell kann man mit der modifizierten Zeitmessung Näherungen für die SPEC-CPU-Leistungskennzahlen für „exotische“ Rechner oder modifizierte Ausstattungen bestimmen, ohne daß eine vollständige Rechnerisolation zur Messung erforderlich ist.
Für die primären Ziele der Anwendung der Benchmarks im URZ,

- Bewertung der Leistungsfähigkeit verschiedener Rechner bei unterschiedlichen und schwankenden Lastverhältnissen,
- Aufdeckung von Ursachen für Leistungsdefizite unter diesen Bedingungen,

ist die CPU-Suite von SPEC in der vorliegenden Version allein jedoch noch nicht geeignet. Die unter Last gemessenen, schlechteren Leistungswerte werden nicht weiter begründet. Erschwerend wirkt sich zudem aus, daß seitens der SPEC nur wenig Informationen zu den einzelnen Benchmarks hinsichtlich Ressourcenforderungen und Ablaufverhalten veröffentlicht wurden [SPECwww], [SPEC95_1/2/3/4], [SPEC_DES.x]. Die Benchmarks liegen zwar als Quelltexte vor, eine Analyse von mehreren hundert KByte Quelltexten bei einigen Benchmarks ist jedoch mit einem hohen, dem Anwender nicht zumutbaren Aufwand verbunden.

Ohne Kenntnis der zugrunde liegenden Arbeitslast ist nach Kapitel 1 ist eine Leistungsbewertung jedoch nicht möglich [Lang92]. Ein Teil der Last während eines Tests unter Praxisbedingungen ist durch den Benchmark selbst definiert und damit bekannt, nach den SPEC-Aussagen wirkt diese Last auf Prozessor(en) und Speicherhierarchie des Testsystems. Unter den Bedingungen an der Universität werden die Testergebnisse daneben möglicherweise durch die Lastverhältnisse auf den benutzten Kommunikationskanälen und den externen Speichermedien, sowie die Prozeßumgebung auf dem Testrechner und die durch andere Prozesse auf Prozessor(en), Speicher- und Ein-/Ausgabesystem hervorgerufenen Lasten bestimmt. Folglich ist ein Lastmonitor erforderlich, der es erlaubt, die Last auf den relevanten Systemkomponenten möglichst genau zu messen und zu bewerten. Mit diesen Werten soll es möglich sein, die unterschiedlichen Testergebnisse bei unterschiedlichen Lastverhältnissen zu relativieren und zu begründen. Daneben soll festgestellt werden, ob tatsächlich alle der erwähnten Teilsysteme Einfluß auf die Benchmarkresultate haben, oder ob hinsichtlich einiger Teilsysteme tatsächlich eine Lastunabhängigkeit besteht.
Darum werden im Kapitel 3 charakteristische Lastmaße für die eventuell für die Benchmarkverzögerung in Frage kommenden Teile eines komplexen Rechnersystems

- Prozessor(en),
- Speichersystem,
- Ein-/Ausgabesystem (Sekundärspeichersystem, Netzwerksystem),

herausgearbeitet.
In den praktischen Messungen (Kapitel 4) und den zugehörigen Auswertungen (Kapitel 5) wird das Verhalten verschiedener Rechner unter Einbeziehung dieser Lastmaße untersucht. Ziel ist einerseits die Gewinnung von Informationen, welche Einflußfaktoren die Benchmarkergebnisse wie stark beeinflussen, oder ob sie ohne Einfluß sind. Auf der anderen Seite steht die Zielsetzung der Gewinnung von Aussagen zum Verhalten der Rechner verschiedener Hersteller unter verschiedenen Lastbedingungen. Damit sollen letztendlich Modifikationen von Systemen oder Neukaufentscheidungen begründet werden.

2.2 Leistungsbewertung mit TPC-Benchmarks

Neben den SPEC-Kennzahlen findet man bei Leistungsangaben zu Rechnersystemen häufig Werte der TPC-Benchmarks.

Im 1988 gegründeten *Transaction Processing Council* (TPC) finden sich unter den über 40 Mitgliedern ebenfalls fast alle „großen“ Hardware- und Datenbankhersteller. Ziel ist die Bereitstellung von Standard-Benchmarks für *Online Transaction Processing* (OLTP). Die Benchmarks versuchen, komplexe Anwendungen zu simulieren, die neben der Leistung von Hard- und Softwareteilen auch das Benutzerverhalten berücksichtigen. Dabei wird die Tatsache ausgenutzt, daß moderne, relationale Datenbanksysteme besondere Anforderungen an ein Rechnersystem stellen. Sie fordern viele Systemteile, wie Prozessoren, Hauptspeicher, Festplatten, Netzwerk und Betriebssystem, bis an ihre Grenzen [KöRoSc_ iX_95].

Ausgangspunkt der Entwicklung war der sogenannte DebitCredit-Benchmark, der Vorgänge bei Kontenbuchungen einer Bank simuliert. Daraus wurden die Benchmarks TPC-A und TPC-B entwickelt. Es werden Transaktionen mit 6 SQL-Statements (3 update, 1 insert, 1 select, 1 commit) über 4 Tabellen mit 100 bis 100.000 Sätzen simuliert. Nutzung von Hauptspeicherplatz und Festplattenanzahl sind dem Anwender überlassen, Betriebssystemmanipulationen sind hingegen nicht erlaubt. Durch diese Freiheitsgrade werden Systeme häufig so konfiguriert, daß das I/O-System keine Rolle spielt und damit die Ergebnisse für einen Vergleich der CPU-Leistungsfähigkeit von Rechnersystemen geeignet sind [Sinix93]. TPC-A und TPC-B unterscheiden sich dahingehend, daß bei TPC-B nur der lokale Anteil der Verarbeitung der Datenbankoperationen betrachtet wird, während TPC-A ein LAN/WAN-Treibersystem mit Terminalverwaltung und Verarbeitung von Ein- und Ausgabenachrichten einbezieht.

Als Leistungsmaß wird der Durchsatz in Transaktionen pro Sekunde (TPS) ermittelt, mit der Zusatzbedingung, daß 90 % der Transaktionen eine Antwortzeit unter 2 Sekunden einhalten. Wegen der beliebigen Hardwarekonfiguration wird der endgültige Vergleichswert dann als Quotient von Hardwarekosten durch TPS-Wert bestimmt. Die guten Leistungswerte beliebiger, teurer Hardware werden also über den Preis relativiert, worin ein wesentlicher Unterschied zum Bewertungsansatz bei den SPEC-Benchmarks besteht.

TPC-A und TPC-B unterlagen bald der Kritik, daß echte Anwendungen viel komplexer sind und erheblich mehr Tabellen und Verknüpfungen benutzen. Darum wurde 1992 der TPC-C eingeführt, der derzeit in Version 3 den aktuellen TPC-Benchmark darstellt. Seit Dezember 1995 dürfen keine TPC-A und TPC-B Ergebnisse mehr veröffentlicht werden.

TPC-C geht ebenfalls von einer Datenbank-Modellanwendung aus. Das System simuliert ein Auftragsabwicklungssystem für ein Großhandelsunternehmen. Dafür werden jetzt bis zu neun Tabellen verknüpft. Der Operationsmix setzt sich aus 49% großen Read-Write-Operationen, 43% kurzen Read-Write-Operationen sowie 8% Read-Only-Operationen zusammen.

Die Leistungsermittlung erfolgt bei TPC-C in Transaktionen pro Minute (TPM). Für eine realitätsnahe Gestaltung wurden die Transaktionen der simulierten Benutzer mit zusätzlichen Eigenschaften versehen. So werden unterschiedliche Zeiten für die Dateneingabe und für „Denkpausen“ simuliert. Auch Fehler des Anwenders werden berücksichtigt, für 1 % der Transaktionen wird vom DBMS ein Rollback ausgelöst.

Wegen der völlig freigestellten Hardwarekonfiguration erfolgt wiederum eine Relativierung der Durchsatzleistung TPM über den Preis für das Rechnersystem, so daß sich ein plattformübergreifend

vergleichbarer Wert von $/TPM ergibt. In die Kosten in $/TPM werden dabei auch die für fünf Jahre zu erwartenden Wartungskosten einbezogen.
TPC-C erlaubt mit realitätsnahen Testbedingungen eine gut vergleichbare Bewertung von Hard- und Software. Aufgrund der Komplexität ist TPC-C, im Gegensatz zu den SPEC-CPU-Benchmarks, als Systembenchmark einzustufen, bei dem logischerweise eine zusätzliche Abhängigkeit vom Datenbanksystem besteht, daß zur Durchführung der Leistungsmessungen auf einem Rechnersystem benutzt wird. TPC-C kann auch zum direkten Test der Leistungsfähigkeit verschiedener Datenbanksysteme eingesetzt werden [Rahm_iX_93], wenn die Tests unter gleichen Hardwarebedingungen stattfinden.
Die vollständigen Testregeln sind auf rund 60 Seiten in der „TPC-Benchmark C Standard Spezification“ beschrieben. Eine Veröffentlichung von Ergebnissen muß in einen „Full Disclosure Report“ erfolgen, der alle relevanten Informationen enthält, um die angegebenen Leistungswerte zu reproduzieren. Dieser Report wird von einem Gremium bestätigt.
Weitere Informationen findet man in [TPCwww], auch zu weiteren TPC-Benchmarks für OLTP-Tests für Mainframes, Client-Server-Belastungsmixe und Tests für Entscheidungssysteme, die aber zum Teil noch in der Einführung sind.

Quellen: [Sinix93], [KöRoSc_ iX_95], [Rahm_ iX_93]

2.3 Leistungsbewertung mit SSBA-Benchmarks

Die Suite Synthetique des Benchmarks de l'AFUU wurde von der AFUU (Assosiation Francaise des Utilisateurs d'Unix), der französischen UNIX-Benutzergruppe, zusammengestellt. Diese Suite hat den Vorteil, daß sie Public Domain erhältlich ist, die Verbreitung beschränkt sich jedoch hauptsächlich auf den französischen Raum.
Die deutsche UNIX-Zeitschrift „iX" verwendet für Hardwaretests eine abgewandelte Form der SSBA-Suite, wobei für die „iX-SSBA" eine spezielle Auswahl realisiert wurde. Zusätzlich wurden einige Erweiterungen und Anpassungen vorgenommen, die speziell auf die Bewertung von Multiprozessorsystemen ausgerichtet sind [Hüls_ iX_95].
Inhalt der Suite ist eine umfangreiche Sammlung von Benchmarks verschiedener Testfelder. Neben „klassischen" CPU-Benchmarks (*Whetstone, Doduc, Linpack, Dhrystone)*, Anwendungsbenchmarks (*Quicksort*), sind Benchmarks für E/A- und Filesysteme (*Saxer, Bsd, Bonnie*) enthalten, die auch Hauptspeicher- und Cache-Ausbau in den Tests berücksichtigen, indem entsprechend große Dateien zum Transfer gewählt werden. Enthalten sind auch UNIX-Workload-Benchmarks, wie *Byte* oder *MUSBUS*, der in modifizierter Form auch in der SPEC-SDM-Suite als Kenbus enthalten ist. Diese Tests messen, wie sich ein System bei hohen Prozeßanzahlen verhält.
Die Suite mißt die Leistung über das Zeitverhalten der Einzelbenchmarks, wobei einige Tests (Workload-Benchmarks) auch versuchen, über eine Skalierung der Einzelbenchmarks die Grenzen eines Testsystems auszuloten.
Die Durchführung der Testläufe ist voll automatisiert und damit unproblematisch, ein vollständiges Ergebnisprotokoll umfaßt etwa eine Seite. Auch hier gelten bestimmte Regeln für die Beschreibung von Testsystemen und -bedingungen, deren Umsetzung in die Ablaufumgebung integriert ist.
Es wird kein Mittelwert über die Ergebnisse der Einzelbenchmarks gebildet. Aufgrund der unterschiedlichen Anwendungen und Qualitäten der einzelnen Teilbenchmarks wäre dies auch kaum sinnvoll. Allerdings ist darin wohl auch einer der Gründe für die geringere Verbreitung und Akzeptanz unter den reinen Anwendern und Käufern von Rechnersystemen zu sehen, da sich ein Vergleich einer Vielzahl spezieller Werte eben aufwendiger gestaltet, als der Vergleich von 2 oder 4 SPEC-Kennzahlen.
Gute Anwendungsmöglichkeiten ergeben sich für den direkten praktischen Vergleich ähnlicher Systeme oder für modifizierte Konfigurationen von Systemen, sowie auch für die schon erwähnten Workload-Untersuchungen, da der zeitliche Aufwand für einzelne Tests relativ gering ist.

Quellen: [Weic91], [Hüls_ iX_95]

3 Entwurf einer Lasterfassung für das SPEC-Benchmarksystem

3.1 Werkzeugauswahl

Ausgehend von den Betrachtungen der vorherigen Kapitel soll die Last bezüglich relevanter Teilsysteme eines Rechnersystems erfaßt werden, um den Einfluß verschiedener Lasten auf die Abarbeitung der Benchmarks und damit auf das generelle Verhalten eines Rechnersystems zu bestimmen.
Basierend auf der Analyse der Meßbedingungen an der TU Chemnitz-Zwickau und dem Vergleich mit den Bedingungen, unter denen Leistungsmessungen bei SPEC-Mitgliedsfirmen durchgeführt werden, wurde herausgearbeitet, daß *mögliche Einflüsse* auf die Benchmarkabarbeitung aus

- Prozessorlast
- Hauptspeicherbelastung
- Disk-I/O-Belastung
- Netz-I/O-Belastung

resultieren können.
Vor der Definition von Lastmaßen für einzelne Teilsysteme ist es erforderlich, eine Menge von geeigneten Werkzeugen auszuwählen, mit denen die charakteristischen Lasten erfaßt werden können. Ausgehend von den Anforderungen an einen Monitor erfolgt die Auswahl geeigneter Werkzeuge unter verschiedenen Gesichtspunkten.
Zunächst ist eine möglichst einfache und aufwandsarme Realisierung des Lastmonitors anzustreben, um die Interferenz des Monitors gering zu halten. Dies erlaubt später eine bessere Anwendung des Lastmonitors zu allgemeinen Zwecken, um beispielsweise den Einfluß von Last auf beliebige, eigene Anwendungen bzw. die von diesen Anwendungen selbst erzeugten Lasten zu bestimmen.
Wichtigstes Auswahlkriterium bei der Werkzeugauswahl ist jedoch die Tatsache, daß die Last auf verschiedenen UNIX-Systemen erfaßt werden soll, die ermittelten Werte aber für diese verschiedenen Systeme vergleichbar sein müssen, um auch das Lastverhalten der Rechnersysteme unterschiedlicher Hersteller vergleichen zu können. Dies erfordert den Einsatz gleicher Werkzeuge bei allen Testsystemen bzw. solche Meßtools, die zumindest vergleichbare Werte liefern. Zwangsläufig muß man sich auf den Einsatz von Standardwerkzeugen des UNIX-Betriebssystems beschränken.
Drittes Ziel bei der Werkzeugauswahl ist eine zweckmäßige Abstraktion der Maße für die Lasten auf den einzelnen Teilsystemen. Es soll mit einem oder einigen wenigen Werten die Last auf einer Systemkomponente beschrieben werden, damit auch mit dieser Zusatzbewertung die Leistungskennzahlen überschaubar und vergleichbar bleiben.
Ausgehend von diesen Anforderungen wurden auf Basis von Literatur zur UNIX-Systemverwaltung [Louk91], [PeReLo93], [Rich91], [Wils95] die in der folgenden Tabelle aufgeführten Meßtools ausgewählt, mit denen die Lasterfassung für die einzelnen Teilsysteme realisiert werden soll. Die angegebenen Betriebssysteme sind eine Auswahl von UNIX-Systemen verschiedener Rechnerhersteller, unter denen eine Vielzahl der Rechner im Rechnernetz der TU Chemnitz-Zwickau betrieben wird. Die

genaue Beschreibung der betrachteten Testrechner erfolgt im Kapitel 4 dieser Arbeit. Vorher werden die Möglichkeiten der Wertesammlung für die einzelnen Teillasten auf Rechnern mit den UNIX-Betriebssystemen

- HP-UX 9.01 (Hewlett-Packard)
- IRIX 5.2 (Silicon Graphics)
- Digital UNIX 3.2 (Digital Equipment)
- SunOS 5.3 (Solaris2.3) (Sun Microsystems)

untersucht.

Abhängig davon, auf welchem „klassischen" UNIX-System das betrachtete Testsystem basiert, stehen grundsätzlich zwei verschiedene „Werkzeugkästen" zur Erfassung von CPU-, Speicher- und Disk-I/O-Last zur Verfügung. Bei auf BSD-UNIX (Berkeley) basierenden Systemen sind dies vmstat und iostat. Bei Systemen mit den Wurzeln in System V (AT&T) kann die Erfassung mit dem integrierten Monitoring-Tool sar (System Activity Reporter) erfolgen. Viele heutige UNIX-Systeme stellen „Mischformen" der beiden klassischen Linien dar, die damit auch die Funktionalität der Werkzeuge in sich vereinigen (SunOS5.3). Zusätzlich wird für die Erfassung der CPU-Last das Werkzeug uptime verwendet. Für die Erfassung der Netzlast werden auf Basis oben genannter Quellen die auf allen Testsystemen verfügbaren Werkzeuge netstat und nfsstat ausgewählt.

Zu bewertende Teilsysteme eines Rechnersystems	Betriebssystem und Systemfamilie			
	----------BSD-Style----------			
			-------System V-Style --------	
	HP-UX 9.01	DigitalUNIX 3.2	SunOS 5.3 (Solaris 2.3)	IRIX 5.2
Prozessor	uptime	uptime	uptime	uptime
	vmstat	vmstat	vmstat	
	iostat	iostat	iostat	
			sar -u	sar-u
Hauptspeicher	vmstat	vmstat	vmstat	
			sar -p -w -g	sar -w -p
Disk-I/O-System	iostat	iostat	iostat	
			sar -d	sar -d
Netzwerk-I/O-System	netstat	netstat	netstat	netstat
	nfsstat	nfsstat	nfsstat	nfsstat

Tabelle 3-1 Testsysteme und jeweils verfügbare Meßwerkzeuge

Im nächsten Abschnitt werden die Lastmaße für die zu betrachtenden Elemente eines Rechnersystems auf Basis der Werte, die von den aufgeführten Werkzeugen geliefert werden, definiert. Dies geschieht zunächst allgemein auf Basis von Betrachtungen zu den generellen, charakteristischen Vorgängen in den einzelnen Teilsystemen. Es wird betrachtet, welche Werte von den einzelnen Werkzeugen geliefert werden und welche dieser Werte signifikante Aussagen zu den Lastverhältnissen liefern.

Auf Basis der Erfassungsmöglichkeiten, die die einzelnen Werkzeuge bieten, werden dann schließlich die endgültigen Lastmaße für die mittleren Lasten bezüglich der Teilsysteme während eines Benchmarklaufes festgelegt. Dabei wird auch auf Besonderheiten oder Abweichungen bei den Werkzeugen auf den einzelnen Testsystemen eingegangen. Neben den oben genannten Quellen wurden dazu auch die UNIX-Manuals der einzelnen Systeme herangezogen.

3.2 Auswahl und Analyse von Lastfaktoren für Teilsysteme

3.2.1 CPU-Last

3.2.1.1 Beschreibungsmöglichkeiten für die CPU-Last

Die Prozessorbelastung (CPU-Last) eines Rechensystems läßt sich in verschiedenen Ebenen des Rechensystems unterschiedlich beschreiben. Interessant sind dabei besonders die Betrachtung in der Hardwareschicht und die Betrachtung der Betriebssystemschicht. Die anliegende Arbeitslast stellt sich dabei aus Sicht von Prozessor und aus Sicht des Betriebssystems sehr unterschiedlich dar.
Die theoretische Leistungsfähigkeit eines Prozessors wird dadurch bestimmt, wieviel Aufträge (Prozessorbefehle) er in einer definierten Zeiteinheit ausführen kann. Dies wird durch technische Parameter, wie Taktung und daraus resultierender Zykluszeit, sowie durch die Arten und den Aufbau der Anweisungen (Befehlssatz) charakterisiert.
Aus Prozessorsicht läßt sich eine Arbeitslast durch die Auslastung des Prozessors beschreiben. Stehen so viele Aufträge für den Prozessor zur Verfügung (transformiert durch die Schichten des Systems aus Anweisungen aus höheren Systemschichten), wie dieser maximal ausführen kann, so ist er vollständig ausgelastet. Erreicht die Auftragszahl aus Betriebssystem und höheren Anwendungen nicht die Maximalzahl, so werden von einem speziellen Systemprozeß (idle-Process) zusätzliche „leere“ Aufträge für den Prozessor generiert. In diesem Fall ist der Prozessor nicht (durch „nichtleere“ Aufträge) ausgelastet. Der die Arbeitslast beschreibende *CPU-Auslastungsgrad* ergibt sich in diesem Falle als Quotient aus der Anzahl „nichtleerer“ Aufträge durch Anzahl maximal abarbeitbarer Aufträge. In Analogie läßt sich die Betrachtung auf Basis der Ausführungszeiten für mögliche Aufträge und für idle-Aufträge durchführen. Hier ergibt sich der *CPU-Auslastungsgrad* für einen bestimmten Zeitraum als Verhältnis aus benötigter Zeit für Abarbeitung von „echten“ Aufträgen und der Gesamtlänge des Zeitraumes.
Aus Sicht des Betriebssystems stellt sich die Arbeitslast für den Prozessor anders dar. Das Betriebssystem hat die Aufgabe, eigene Aufträge und die Aufträge aus Anwendungen zu koordinieren und an die nächstniedrigere Schicht zur Ausführung in der Hardwareschicht weiterzureichen. In UNIX als Multiuser- und Multiprogrammsystem werden die Aufträge aus dem System und verschiedenen Anwendungen als Prozesse verwaltet. Bedingt durch die Anzahl von n Prozessoren im Rechensystem können zu einem fixen Zeitpunkt maximal n Prozesse abgearbeitet werden (*running processes*), indem die in Prozessorbefehle transformierten Anweisungen eines Prozesses p auf einem der Prozessoren ausgeführt werden. Stehen mehr Prozesse zu Abarbeitung an (*runable processes*), so werden diese in eine Warteschlange (*run queue*) eingereiht und zu einem späteren Zeitpunkt abgearbeitet. Aus Betriebssystemsicht stellt sich die aktuelle Arbeitslast damit als die Anzahl der Prozesse dar, die in der Run-Queue auf die Zuteilung eines Prozessors warten.
Bei Betrachtung von Zeitintervallen ist die Länge der Run-Queue starken Schwankungen unterworfen. In Zeitabschnitten, in denen die Anzahl neuer Aufträge unter der Anzahl abarbeitbarer Aufträge liegt, reduziert sich die Warteschlangenlänge. Gehen mehr Aufträge ein, als abgearbeitet werden können, so

wächst die Warteschlangenlänge[6]. Die Arbeitslast für ein Zeitintervall kann aus Betriebssystemsicht daher als *mittlere Länge der Run-Queue* in diesem Zeitraum beschrieben werden.
Warteschlangenlänge und *CPU-Auslastungsgrad* sind korreliert. Wenn keine Prozesse in der Warteschlange warten, dann generiert das Betriebssystem die bereits beschriebenen idle-Aufträge. Damit sinkt der CPU-Auslastungsgrad auf einen Wert unter 100%, wogegen er bei 100% bleibt, solange Prozesse in der Prozeßwarteschlange auf ihre Abarbeitung warten.

3.2.1.2 Erfassung und Verarbeitung der CPU-Last-Beschreibungsdaten

Die aus Prozessorsicht interessante *CPU-Auslastung* läßt sich bei BSD-UNIX-basierten Systemen mit den Monitoring-Werkzeugen *vmstat* und *iostat* ermitteln. Dabei werden für das zurückliegende Meßintervall folgende Werte geliefert:

- *vmstat:* CPU: us - Prozentsatz der CPU-Zeit für Nutzerprozesse
  sy - Prozentsatz der CPU-Zeit für Systemprozesse
  id - Prozentsatz für idle-Zeit
- *iostat:* CPU: us - Prozentsatz der CPU-Zeit für Nutzerprozesse
  ni - Prozentsatz der CPU-Zeit für Nutzerprozesse mit niedriger Priorität (*niced*)
  sy - Prozentsatz der CPU-Zeit für Systemprozesse
  id - Prozentsatz für idle-Zeit

Bei SystemV-basierten Systemen erhält man Informationen zu CPU-Auslastung eines Rechensystems durch Benutzung von *sar -u* . Dabei werden folgende Werte für jedes einzelne Meßintervall eines Zeitraumes, sowie die Durchschnittswerte für den Gesamtzeitraum ermittelt:

- *sar -u:* %usr - Prozentsatz der CPU-Zeit für Nutzerprozesse
  %sys - Prozentsatz der CPU-Zeit für Systemprozesse
  %wio - Prozentsatz der CPU-Zeit, in der CPU auf Beendigung von I/O-Operationen wartete
  %idle - Prozentsatz für idle-Zeit

Für die Betrachtung der CPU-Auslastung ist hier nur die Tatsache interessant, wie die CPU ausgelastet ist, nicht jedoch wodurch. Damit läßt sich für alle betrachteten Systeme die tatsächliche CPU-Auslastung leicht bestimmen als

CPU-Auslastung = 100% - (Prozentsatz für idle-Zeit) [%]

Für Mehrprozessorsysteme liefern die verwendeten Werkzeuge bereits einen Mittelwert für alle CPUs des getesteten Rechensystems. Die Auslastung wird von den Werkzeugen immer für eine definiertes zurückliegendes Intervall ermittelt, das beim Aufruf dem Meßwerkzeuges angegeben werden muß.

[6] Gehen permanent mehr Aufträge ein, als abgearbeitet werden können, so wächst die Länge der Run-Queue gegen unendlich, das Rechensystem ist überlastet (*overloaded system*).

Aus Sicht des Betriebssystems wurde die *Länge der Run-Queue* als charakteristische Lastgröße für die CPU-Last definiert. Für alle betrachteten UNIX-Systeme steht zur Gewinnung das Tool *uptime* zur Verfügung. *uptime* ohne Argumente liefert für das lokale System zum Zeitpunkt des Aufrufes :

- *uptime:*
    - die aktuelle Zeit
    - die Zeit, die das System aktiv ist
    - die Anzahl der angemeldeten Nutzer
    - die ∅-Anzahl der Prozesse in der run-queue über die letzte(n)
        - 1 Minute
        - 5 Minuten
        - 15 Minuten

      Eine Ausnahme bildet DigitalUnix , hier werden ∅-Werte geliefert für die letzten
        - 5 Sekunden
        - 30 Sekunden
        - 60 Sekunden.

Da alle Systeme einen Durchschnittswert für die letzte Minute liefern, bietet es sich an, während eines Benchmarklaufes mit *uptime* ein Sampling im Intervall von einer Minute durchzuführen und aus allen Samples im Gesamtzeitraum die *mittlere Warteschlangenlänge* zu berechnen. Die betrachteten Testsysteme verwalten bei mehreren Prozessoren für alle Prozessoren eine gemeinsame Warteschlange.
Weitere Möglichkeiten zur Ermittlung der Länge der Run-Queue bestehen mit *vmstat* und *sar -q*. Ersteres liefert die aktuelle Anzahl der ausführbaren Prozesse und damit keine sichere Aussage über die Last während eines zurückliegenden Intervalls. Eine Lösung könnte in der Wahl einer sehr hohen Samplerate für die Abfrage liegen, was jedoch aufgrund des damit produzierten Overheads kritisch zu betrachten ist. Der Lasterfassungsprozeß selbst würde eine erhebliche Last erzeugen. *sar -q* liefert mit dem Wert „runq-sz“ die durchschnittliche Länge der Warteschlange bereiter Prozesse für das letzte Meßintervall (analog zu *uptime*), steht jedoch nicht auf allen Systemen zur Verfügung.

3.2.1.3 Lastmaß für CPU-Last

Mit den so gewonnenen Daten ist es möglich, zwei Maße für die CPU-Last während eines bestimmten Zeitraumes, hier bestimmt durch die Abarbeitungszeit für einen Benchmark, zu definieren.

Aus Prozessorsicht wird als CPU-Lastmaß die *mittlere CPU-Auslastung* über *n* Meßintervalle *I* definiert:

$$\varnothing CPU\_Auslastung = \frac{\sum_{I=1}^{n} (100 - \%idle_I)}{n} \quad [\,\%\,]$$

Aus Betriebssystemsicht wird als CPU-Lastmaß die *mittlere Länge der Warteschlange* der ausführungsbereiten Prozesse (Run-Queue) über *n* Meßintervalle *I* während der Benchmarkabarbeitung definiert:

$$\varnothing Länge\_Run\_Queue = \frac{\sum_{I=1}^{n} Länge\_Run\_Queue_I}{n}$$

[Prozeßanzahl]

3.2.2 Hauptspeicherlast

3.2.2.1 Beschreibungsmöglichkeiten

Die Leistungsfähigkeit der Systemkomponente Hauptspeicher (HS) des Rechensystems wird zunächst durch 2 physikalische Kenngrößen beschrieben. Dies sind:

- Hauptspeichergröße, angegeben in MByte
- Bandbreite, angegeben in Bit/s

Die Hauptspeichergröße limitiert die Datenmenge, auf die die CPU während ihrer Aktivität zugreifen kann, ohne daß die Abarbeitung eines Prozesses stark verzögert wird. Die Bandbreite gibt die Anzahl der Bits an, die maximal in einer Sekunde durch die CPU vom Hauptspeicher gelesen werden können. Diese Bandbreite ist das typische Leistungsmaß, mit dem Rechnerhersteller die Leistungsfähigkeit des Speichersystems und des Übertragungssystems zwischen Prozessor und Speicher ihres Produktes beschreiben.

Moderne Rechnersysteme bauen auf einer mehrstufigen Speicherhierarchie mit mehreren kleineren, dafür jedoch schnelleren Zwischenspeichern (*Caches*) auf, die von der CPU benutzte Daten (und mit hoher Wahrscheinlichkeit darauf folgende Daten - *read ahead strategy*) puffern, um diese bei erneuter Referenzierung noch schneller bereitzustellen. Ausgabedaten der CPU werden gepuffert, um bei erneuter Referenzierung schnell zur Verfügung zu stehen, oder um zu einem späterem Zeitpunkt in den Hauptspeicher geschrieben zu werden *(write behind strategy)*, damit die CPU ihre Arbeit sofort fortsetzen kann. Diese Caches erhöhen jedoch nicht die tatsächlich nutzbare Menge physikalischen Speichers, sie halten Kopien von Hauptspeicherdaten für einen besonders schnellen Zugriff bereit. Dennoch ist ihr Einfluß auf die Leistungsfähigkeit des Speicher- und damit auch auf die des Gesamtsystems nicht zu unterschätzen. Der erreichte Leistungsgewinn läßt sich jedoch nicht allgemein beschreiben, da die Effektivität der Zwischenspeicher stark von der Menge und Strukturierung der referenzierten Daten abhängt, wobei dies sowohl für verarbeitete Daten als auch für Programmcode gilt.

Moderne Betriebssysteme benutzen das ***Konzept des virtuellen Speichers***, um den vorhandenen physischen Hauptspeicher zu verwalten und optimal für die Anforderungen aller Prozesse bereitzustellen. Dabei steht ein Adreßraum zur Verfügung, der ein Vielfaches des tatsächlichen physikalischen Hauptspeichers adressiert. Dieser ***virtuelle Speicher*** wird in Elemente einer bestimmten Größe gegliedert, die als Speicherseiten bezeichnet werden. Typische Seitengrößen liegen zwischen 4 KByte und 16 KByte. Übersteigt die Summe der Speicheranforderungen durch einzelne Prozesse die Menge des verfügbaren physischen Hauptspeichers, so werden einzelne Seiten auf einen externen Speicher (Harddisk) ausgelagert und die so frei gewordenen Speicherbereiche der speicherfordernden Anwendung zugeteilt. Dies ist möglich, da Anwendungen fast nie den insgesamt benötigten Speicher gleichzeitig benutzen. Eine Faustregel besagt, daß 80% der Ausführungszeit einer Anwendung in nur 20% des Programmcodes verbracht werden [Louk91]. Bei den von Prozessen verwendeten Daten ist ebenfalls stets eine Lokalität in bestimmten Abschnitten zu verzeichnen.

Die zusätzliche Zeit für das Auslagern von Seiten bewirkt, bedingt durch die wesentlich langsameren externen Speicher, daß die Gesamtabarbeitungszeit für die einzelnen Prozesse erheblich ansteigt, die sich in der Abarbeitungszeit widerspiegelnde Gesamtleistung des Rechensystems damit abfällt. Die Leistungsfähigkeit des Speichersystems ist damit „binär“, entweder ist der physische Hauptspeicher ausreichend und die Leistung ist konstant gut, oder der physische Hauptspeicher ist nicht ausreichend

und die Leistung fällt durch die Aus- und Einlagerungszeit stark ab. Wie stark die Leistung abfällt, hängt direkt davon ab, wie häufig solche Auslagerungen erforderlich sind. Man kann die Speicherlast deshalb so definieren, daß sie die Häufigkeit der erforderlichen Auslagerungen angibt:

Speicherlast = Auslagerungsrate (Page-Out-Rate) = # ausgelagerte Seiten / s

Für die Herausarbeitung der genauen Parameter zur Speicherlastbeschreibung bedürfen einige Teilrealisierungen der Verwaltung des Systems von virtuellem Speicher noch einer näheren Betrachtung. Die virtuelle Speicherverwaltung von UNIX basiert auf 2 Grundmechanismen, dem *Paging* und dem *Swapping*. Beim Swapping werden komplette Prozesse auf die Harddisk ausgelagert, um freien Hauptspeicher zu gewinnen. Bei der nächsten Aktivierung eines solchen Prozesses muß das Betriebssystem das Speicherabbild der Prozeßumgebung wieder vom externen Speicher in den Hauptspeicher laden. Beim Paging hingegen werden nur einzelne Seiten aus dem Hauptspeicher auf die Disk ausgelagert, die wichtigsten (gerade benutzten) Teile von Prozessen verbleiben im Hauptspeicher. Für diese Auswahl „merkt“ sich der Paging-Algorithmus Informationen, wann Seiten referenziert wurden und lagert im allgemeinen solche Seiten aus, die am längsten nicht mehr benutzt wurden. Die modernen UNIX-Systeme realisieren Paging und Swapping, wobei das Paging bevorzugt eingesetzt wird, da es eine detailliertere Kontrolle über den Speicher erlaubt. Swapping tritt nur in wenigen Situationen auf:

- Prozesse, die schon länger als 20 Sekunden inaktiv (*sleeping*) sind, können jederzeit ausgelagert werden, damit sie nicht unnötig Hauptspeicher blockieren.
- In Situationen extremer Speicherknappheit wird Swapping eingesetzt, um die Arbeitsfähigkeit des Gesamtsystems zu erhalten (*desperation swapping).*

Desperation Swapping kostet einen großen Teil der Systemleistung, da auch ein erheblicher Overhead für Harddisk-Operationen erforderlich ist. Daher versucht das Betriebsystem, Swapping wenn möglich zu vermeiden und, wenn es unbedingt erforderlich ist, solche Prozesse zu wählen, die häufig inaktiv sind. Solche potentiellen „Opfer“ des Swappings sind hauptsächlich interaktive Prozesse, die mit der Tastatur korrespondieren und dadurch oftmals längere Phasen der Inaktivität aufweisen (Editoren, Shells etc.). Solche hin und wieder auftretenden Swap-Out-Operationen haben dann einen relativ geringen Einfluß auf die Leistungsfähigkeit des Speichersystems, da in der Folge durch den frei werdenden Speicher zunächst besonders zügig weitergearbeitet werden kann.

Für die Ermittlung der Paging- und Swapping-Aktivitäten stehen folgende Werkzeuge zur Verfügung:

1. BSD-basierte Systeme: *vmstat*
2. System V -basierte Systeme: *sar*

Sie liefern grundsätzlich Informationen zu:

- Page-ins , gibt die Anzahl der Seitentransfers von der Disk in den HS an
- Page-outs, gibt die Anzahl der Seitentransfers vom HS auf die Disk an
- Swap-ins, Anzahl Prozesse, die von der Disk in den Speicher geladen wurden
- Swap-outs, Anzahl Prozesse, die vom HS auf die Disk ausgelagert wurden

Dabei ist noch ein wichtiger Fakt zu beachten. Alle modernen UNIX-Systeme unterstützen das sogenannte Anforderungs-Paging (*demand paging*). Dies bedeutet unter anderem, daß das

Betriebssystem den Paging-Mechanismus auch für den Start neuer Prozesse benutzt. Auch hierbei wird die Tatsache ausgenutzt, daß ein Prozeß nicht vollständig im HS sein muß, um ausgeführt werden zu können. Zum Start eines Prozesses werden nur die Speichertabellen im System so gesetzt, als wenn die Seiten ausgelagert worden wären. Beim Start des Prozesses (Sprung zu Startadresse) tritt ein Seitenfehler (*page fault*) auf, und die benötigte Seite des ausführbaren Programmes wird von der Disk geladen. Auf diese Art und Weise wird ein Programm nach Bedarf („as needed“) geladen. Ein vollständiges Laden eines Prozesses ist hingegen gleichzusetzen mit einen Swap-In.
Page-Ins und Swap-Ins sind damit normale Vorgänge, die beim Betrieb eines UNIX-Rechnersystems häufig auftreten, sie haben nicht immer etwas mit einer eventuellen Speicherknappheit bzw. der Speicherlast zu tun.
Um die Speicherlast zu bewerten, werden nachfolgend nur die *Page-Out- und Swap-Out-Aktivitäten* betrachtet. Implizit ist mit einer Auslagerung immer eine nachfolgende Einlagerung verbunden, man könnte damit die gemessenen Auslagerungsraten zusätzlich mit dem Faktor 2 normieren.

3.2.2.2 Erfassung und Verarbeitung der Speicherlast-Beschreibungsdaten

Für die Ermittlung der Speicherlast bieten sich systemabhängig zwei Werkzeuge an:

- *vmstat* (HP-UX, Digital Unix, Solaris2.3), dieses Werkzeug stammt aus BSD-UNIX
- *sar* (IRIX, Solaris2.3) , ein komfortables Werkzeug, daß auf SystemV zurückgeht

VMSTAT liefert folgende Informationen für ein anzugebendes Meßintervall:

- Prozeßanzahlen (runable, blocked for I/O, runable&swapped out)
  (Unterschiede bei Digital Unix)
- Memory:
  avm - Anzahl der verfügbaren Seiten an virtuellem Speicher (DEC: act)
  free - Anzahl der freien Seiten im physischen Hauptspeicher
- Page:
  si - Anzahl der von Disk geladenen Prozesse („swap-ins“)
  so - Anzahl der ausgelagerten Prozesse („swap-outs“)
  ...
  pi - Anzahl der pro Sekunde geladenen KByte (Seiten) („page-ins“)
  po - Anzahl der pro Sekunde ausgelagerten KByte (Seiten) („page-outs“)
  ...
  Unter Digital UNIX unterscheiden sich die Reportdaten etwas. Es werden Informationen über die Verwaltung verschiedener Seiten-Listen geliefert (freie Seiten, feste Seiten, die nicht für Page-Outs zugelassen sind, Gesamtzahl virt. Seiten), die eine Momentaufnahme darstellen. Dazu kommen weitere Informationen zu verschiedenen Fällen bei der Seitenadressierung, die jedoch alle keine Auswirkung auf die Last haben. Zum Paging werden folgende Angaben geliefert, die auf dieser Plattform Absolutwerte für ein Intervall darstellen:
  pin - Anzahl der geladenen Seiten (absolut für letztes Intervall)
  pout - Anzahl der ausgelagerten Seiten (absolut für letztes Intervall)
  Zum Swapping werden unter Digital Unix keinerlei Informationen bereitgestellt.
- Faults:
  Interrupt-Informationen
  in - Geräte-Interrupts pro Sekunde

sy - Systemrufe pro Sekunde
cs - Prozeßumschaltungen pro Sekunde

- CPU: CPU-Auslastungs-Informationen
  us - Prozentsatz der CPU-Zeit für Nutzerprozesse
  sy - Prozentsatz der CPU-Zeit für Systemprozesse
  id - Prozentsatz für idle-Zeit

- Disk: SunOS (Solaris) liefert bei vmstat unter diesem Punkt die Anzahl der Diskoperationen für die Disks des Rechnersystems, diese Zahlen werden nicht näher betrachtet. Disk-Aktivitäten sind Betrachtungsgegenstand des nächsten Abschnitts.

SAR bietet mehrere Schalter, die Informationen zur Speicherverwaltung für festgelegte Meßintervalle liefern:

- *sar -r* : freemem - Anzahl der verfügbaren physischen Speicherseiten
  freeswap - Anzahl der freien Disk-Blöcke für Paging/Swapping

- *sar -w* : Swapping-Aktivitäten
  swpin/s - Anzahl Swap-In-*Operationen* pro Sekunde
  swpot/s - Anzahl der Swap-Out-*Operationen* pro Sekunde
  bswin/a - Anzahl der 512-Byte-*Blöcke*, die pro Sekunde geladen wurden
  bswot/s - Anzahl der 515-Byte-*Blöcke*, die pro Sekunde ausgelagert wurden

- *sar -p*: Paging-Aktivitäten
  vflt/s - Anzahl erfolgloser Seitenadressierungen pro Sekunde
  pflt/s - Schutzverletzungen
  weitere verschiedene Untermengen von vflt/s („Copy-on-Write“ u.a.),
  die jedoch nichts zur Last aussagen
  ...
  rclm/s - Anzahl der *Speicherseiten* pro Sekunde, die zwangsweise durch
  Page-Out in die Liste der freien Speicherseiten eingetragen werden
  (entspricht Anzahl ausgelagerter Seiten pro Sekunde)

- *sar -g*: Paging-Aktivitäten (nur Solaris2.3)
  pgout/s - Anzahl *Page-Outs* pro Sekunde (d.h. Anzahl der Requests)
  ppgout/s - Anzahl der ausgelagerten *Seiten* pro Sekunde (entspr. rclm/s)
  pgfree/s - Anzahl der Seiten, die pro Sekunde der Liste der freien Seiten
  hinzugefügt wurde

Alle hier beschriebenen Werte liefern wichtige Hinweise für den Systemadministrator zur Konfigurierung und Rekonfigurierung des Systems, bzw. zu notwendigen Erweiterungen bei permanenten Engpässen. Zu Beginn dieses Abschnittes wurde bereits das Konzept des virtuellen Speichers mit speziellen Eigenschaften, wie *demand paging* diskutiert und dabei herausgearbeitet, daß zur Beschreibung der Speicherlast am besten die Auslagerungsrate, also die Anzahl der ausgelagerten

Seiten pro Sekunde, geeignet ist. Dieser Wert kann noch über die Seitengröße normalisiert werden, so daß sich als systemübergreifendes Lastmaß folgendes definieren läßt:

Speicherlast = Menge ausgelagerten Hauptspeicherinhalts / Sekunde

[Page-Out-Rate in KByte/s]

Für die Gewinnung der Lastinformationen bietet sich ein zeitgesteuertes Sampling an. Dabei werden nach definierten Intervallen die relevanten Daten für das abgelaufene Intervall mit *vmstat* bzw. *sar* protokolliert. Die so erhaltenen Durchschnittswerte für die einzelnen Intervalle für ***po + so*** (vmstat) beziehungsweise ***rclm/s + bswot/s*** (sar) werden dann noch einmal über den gesamten zu bewertenden Zeitraum gemittelt und ergeben damit die mittlere Speicherlast für den Beobachtungszeitraum (*∅Anzahl ausgelagerter Seiten / s).*
Für Digital Unix (DEC) ist diese Erfassung etwas anders aufzubauen, da hier nur der Absolutwert für die Anzahl der ausgelagerten Seiten ***pout*** im Intervall von vmstat geliefert wird. Diese Absolutwerte werden über den Meßzeitraum (Benchmarklaufzeit) hinweg aufsummiert und nach Ablauf der Messung durch die Länge des Meßzeitraumes (in Sekunden) dividiert, woraus sich ebenfalls die ausgelagerte Datenmenge pro Sekunde ergibt *(∅ Anzahl ausgelagerter Seiten / s*). Swapping-Informationen stehen bei diesem System nicht zur Verfügung.

vmstat erlaubt neben Intervallmessungen auch die Ausgabe von Statistikinformationen zu verschiedenen Ereignissen seit dem Bootvorgang. Dazu gehören

- swap ins, swap outs, pages swapped in, *pages swapped out*,
- total address trans. faults taken,
- page ins, page outs, pages paged in, *pages paged out*,
- total reclaims, reclaims from free list,

(originale Feldbezeichner von *vmstat*)

sowie weitere Statistikdaten zu Systemrufen und CPU-Auslastung. Durch Ermittlung dieser Werte für die Zeitpunkte vor und nach dem Meßzeitraum (Benchmarklaufzeit) und die Differenzenbildung läßt sich ebenfalls die durchschnittliche Anzahl dieser Ereignisse und Operationen pro Sekunde im Meßzeitraum ermitteln, wozu unter anderem auch die Mengen an ausgelagertem Speicher bei Paging bzw. Swapping gehören. Meßtechnisch ist diese Methode sogar zu bevorzugen, da damit der durch die Lastermittlung selbst erzeugte Overhead wesentlich kleiner ist als bei Messungen mit relativ kleinen Sampling-Intervallen. Die Sampling-Methode bietet dagegen den Vorteil, daß das vorliegende Protokoll detaillierte Rückschlüsse auf Lastspitzen (*peaks*) und besondere Situationen während der Messung zuläßt, die es ermöglichen, Abnormitäten oder größere Abweichungen im Vergleich zu anderen Messungen zu erklären.
Auch bei dieser Reportform verwendet Digital Unix ein anderes Format und liefert teilweise andere Daten. Die interessierenden Informationen, also die ausgelagerten Seiten und die Seitengröße des Systems, werden bereitgestellt (*pages paged out*, pagesize).

3.2.2.3 Lastmaß für die Hauptspeicherlast

Das Lastmaß für die Hauptspeicherlast für einen Beobachtungszeitraum (Benchmarklaufzeit) wird auf Basis der oben beschriebenen Daten als die Datenmenge definiert, die im Mittel pro Sekunde ausgelagert werden mußte, und im folgenden mit *Page-Out-Rate* bezeichnet:

$$Page\_Out\_Rate = \frac{Anzahl\_ausgelagerter\_Speicherseiten * Seitengröße[KByte]}{Zeit[s]}$$

[KByte/s]

3.2.3 Ein- und Ausgabelast für die externen Speicher (Disk-I/O-Last)

3.2.3.1 Beschreibungsmöglichkeiten für die Disk-I/O-Last

Die Bewertung der I/O-Performance bezüglich der externen, permanenten Speicher ist der wichtigste Punkt im Kontext der Leistungsfähigkeit des Ein- und Ausgabesystems. Disk-I/O-Operationen beeinflussen andere Komponenten des Gesamtsystems maßgeblich, so zum Beispiel auch die Leistungsfähigkeit des virtuellen Speichersystems.
Zu den als externe Speichermedien klassifizierten Geräte gehören neben den Festplattenspeichern (Harddisk) auch Diskettenlaufwerke (Floppy Disk), Magnetbandlaufwerke (Tape) sowie die nur lesbaren CD-ROM-Laufwerke.
Die Betrachtungen in dieser Arbeit sollen sich auf die typischen permanenten Speicher, die Harddisks beschränken, wobei sich die gemachten Aussagen auf die anderen Medien leicht übertragen lassen.
Die Leistungsfähigkeit des Disk-I/O-Systems wird von mehreren Einzelkomponenten bestimmt, von denen die wichtigste die Harddisk selbst ist. Deren Leistung wird durch 3 typische Herstellerangaben beschrieben:

- Suchzeit (*seek time*) - gibt an, wieviel Zeit benötigt wird, um den Lesekopf von einer Position auf eine neue Leseposition zu bringen. Dies hängt maßgeblich von der Entfernung der zu lesenden Spuren ab, weshalb Hersteller oft einen minimalen, einen maximalen und einen durchschnittlichen Wert angeben.
  Für die Planung der meisten Systeme ist diese Angabe die bedeutendste. Eine Vielzahl von Prozessen greift auf eine Vielzahl auf der Disk verstreuter Files zu, wobei häufig nur wenige Datenblöcke gelesen werden. Daher wird mehr Zeit damit verbracht, die Köpfe zu positionieren, als damit, tatsächlich Daten zu lesen. Dabei ergeben sich Verhältnisse von bis zu 10 :1 für Positionierzeit zur tatsächlichen Transferzeit [Louk91].

- Rotationsgeschwindigkeit (*rotational speed*) - gibt die Anzahl der Disk-Umdrehungen pro Minute an. Bei jedem Suchvorgang ist im Durchschnitt eine halbe Umdrehung erforderlich, bis auf der gefundenen Spur der richtige Sektor am Lese-/Schreibkopf erscheint.

- Datentransferrate (*(raw) transfer rate*) - Datenmenge, die in einer Sekunde gelesen/geschrieben werden kann. Diese Angabe ist besonders für solche Systeme wichtig, bei denen wenige Nutzerprozesse große Datenmengen schreiben und damit anteilig weniger Positionierzeit anfällt.

Es ist naheliegend, die Menge der über das Disk-I/O-Subsystem transferierten Daten in einer bestimmten Zeit als Maß für die Belastung des externen Speichersystems zu definieren. Die Werkzeuge *iostat* und *sar* liefern dabei Werte über Anzahl von Transfers pro Sekunde und die Menge der Transferdaten pro Sekunde für jede einzelne Disk.
Die nachfolgende Modellgrafik soll zeigen, daß es nicht immer genügt, die Werte für die einzelnen Laufwerke zu addieren. Es sind einige Konfigurationsdetails beim laufenden System zu betrachten, wenn man die Last in Relation zur theoretischen Maximallast betrachten will.

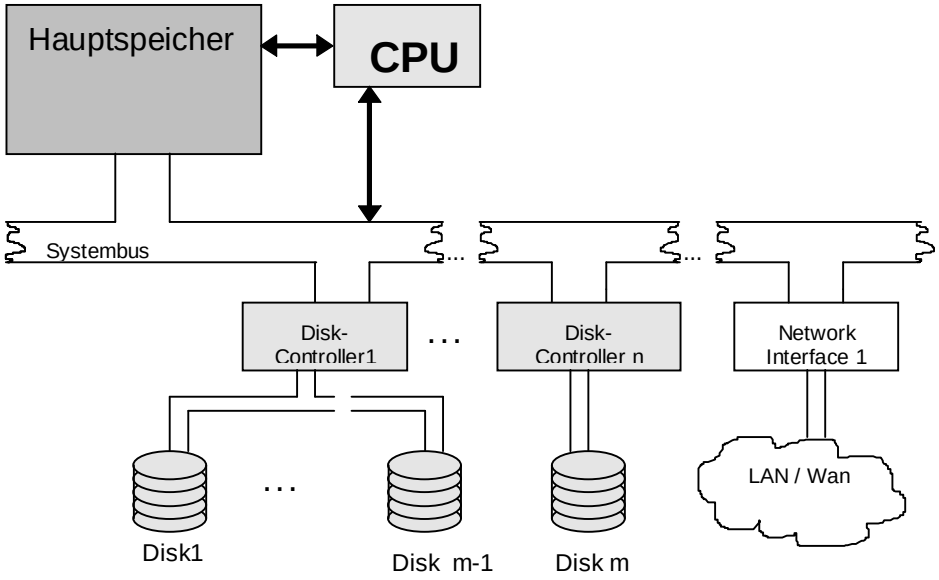


Abbildung 3-1 Modell mit Disk-I/O-Subsystem eines Rechnersystems

Die Menge der maximal transportierbaren Daten wird durch jene Teilkomponente bestimmt, die die geringste Transferrate zuläßt. Diese stellt dann den „Flaschenhals“ auf dem Weg zum Speichermedium dar. Gleiches gilt für Datentransfers von und zum Netzwerk-Interface des Systems, das in der Abbildung bereits integriert ist und mit dem sich der nächste Abschnitt beschäftigt. Diese Tatsache läßt sich auch funktional beschreiben.

$$max\_transfer\_rate_{Disk\text{-}I/O\text{-}Subsystem} = MIN (max\_transfer\_rate_{Systembus}, max\_transfer\_rate_{Disk\text{-}Controller}, max\_transfer\_rate_{Disk})$$

Um die Gesamttransferrate zu erhöhen, ist es möglich, mehrere Komponenten einer Klasse parallel zu verwenden, um den ankommenden Datenstrom aufzuteilen, so wie dies in der Abbildung mit mehreren Disk-Controllern und wiederum mehreren Disks dargestellt ist. Für die Ermittlung der aktuellen Transferrate ist dieser Fakt irrelevant, dabei können tatsächlich die Datendurchsatzraten der Platten verknüpft werden.

3.2.3.2 Erfassung und Beschreibung der Daten zur Disk-I/O-Last

Zur Ermittlung der Belastung des Disk-I/O-Teilsystems stehen, abhängig vom Testsystem, zwei Werkzeuge zur Verfügung:

- *iostat* (HP-UX, Digital UNIX, Solaris2.3), dieses Werkzeug stammt aus BSD-UNIX
- *sar* (IRIX, Solaris2.3) , ein komfortables Werkzeug, das auf SystemV zurückgeht

IOSTAT liefert auf den entsprechenden Systemen folgende Informationen für ein beim Aufruf zu spezifizierendes Meßintervall:

- Terminal (tty) : Terminal-Ein- und Ausgaben
  tin,
  out - Anzahl der über Terminal-Buffer ein- und ausgegeben Zeichen als absolute Anzahl für das letzte Intervall (bei Digital-UNIX Angabe in Zeichen pro Sekunde).
- CPU: CPU-Auslastungs-Informationen
  us - Prozentsatz der CPU-Zeit für Nutzerprozesse
  ni - Prozentsatz der CPU-Zeit für Prozesse mit niedriger Priorität (*niced*)
  sy - Prozentsatz der CPU-Zeit für Systemprozesse
  id - Prozentsatz für idle-Zeit
  Diese Angaben werden in ähnlicher Form auch von vmstat geliefert. Die Auswertung wurde schon im Abschnitt zur CPU-Last diskutiert.
- Disks: Informationen zu Datentransfers für jede Disk
  bps - ∅KByte/s, die von und zur Harddisk transportiert wurden
  tps - ∅Anzahl der Transfers/s
  msps - ∅Anzahl Millisekunden/ Transfer
  Der letzte Wert wird bei den einzelnen Systemen sehr unzuverlässig oder gar nicht ermittelt.

Beim iostat-Werkzeug von SunOS werden auf Wunsch (-D, -x) weitere Informationen zu Disk-Operationen bereitgestellt, dazu gehören die Anzahl der Read-Operationen, der Write-Operationen und die prozentuale Disk-Auslastung, die Warteschlangenlänge für Diskoperationen und die durchschnittliche Wartezeit. Da die Transferrate als charakteristisches Merkmal der Disklast bestimmt wurde, wird auf die Auswertung des erweiterten Reports verzichtet, da ein plattformübergreifender Vergleich durch fehlende Informationen zu den anderen Systemen nicht möglich ist .

SAR liefert Informationen über die Disk-I/O-Belastung bei Verwendung des Schalters *-d* :

- *sar -d* : Informationen zu Blockdevice-Operationen (Disks, Tapes,...)
  dsk*n* - Disk-Nummer
  %busy - Zeitanteil, in der die Disk mit Anforderungen beschäftigt war
  avque - ∅Anzahl der Prozesse, die in dieser Zeit warten mußten
  r+w/s - ∅Anzahl Lese-Transfers+Schreib-Transfers pro Sekunde
  blks/s - ∅Anzahl transportierter 512-Byte-Blöcke pro Sekunde
  avseek- ∅Suchzeit in Millisekunden (Transferzeit)

Die Transferrate für jede Disk ergibt sich damit als Produkt von

*Disk-I/O-Rate = blks/s * Blocksize [KByte]* [KByte/s]

Es ist somit für alle Systeme der Wert für die *mittlere Transferrate* $_{Disk\ i}$ [KByte/s] für ein vorher definiertes Intervall ermittelbar. Aus den Werten der einzelnen Disks wird für jedes Intervall die Transferrate [KByte/s] über alle Diskdrives summiert. Die so für n Intervalle ermittelten Transferraten

werden über den gesamten Beobachtungszeitraum (Benchmarklaufzeit) arithmetisch gemittelt. Diese mittlere Transferrate [KByte/s] stellt die Last im Disk-I/O-System über die Benchmarklaufzeit dar.
Auf die gleiche Art und Weise lassen sich zusätzlich die Werte für die mittlere Anzahl von Transfers pro Sekunde ermitteln, woraus sich dann letztendlich auch die durchschnittlichen Datenmengen pro Transfer berechnen lassen. Diese Daten können zur Interpretation der Disk-I/O-Last herangezogen werden, fließen jedoch in deren Lastmaß nicht ein. Der anfangs beschriebene Einfluß der Transferzahlen und Transfergrößen wird durch die Verwendung der sogenannten *Buffer Caches* erheblich abgeschwächt, so daß auf deren direkte Einbeziehung in das Lastmaß verzichtet werden kann.

3.2.3.3 Lastmaß für die Disk-I/O-Last

Auf Basis der ermittelbaren Daten wird die Summe der mittleren Disktransferraten über alle Disks als Maß für die Last auf dem Disk-I/O-Subsystem für einen endlichen Beobachtungszeitraum, der durch die Benchmarklaufzeit mit n Meßintervallen bestimmt ist, definiert.

$$Disk\_I/O\_Rate = \frac{\sum_{Intervall=1}^{n} \sum_{Disk=1}^{m} Transferrate_{Disk|Intervall}}{n}$$

[KByte/s]

3.2.4 Netzlast

3.2.4.1 Beschreibungsmöglichkeiten für die Netzlast

Die Netzlast ist eine Teilkomponente im allgemeinen Kontext der Ein- und Ausgabelast. In den letzten Jahren basieren immer mehr Anwendungen auf Kommunikation zwischen mehreren verschiedenen Rechnern und der Benutzung entfernter oder verteilter Filesysteme. Die Netzwerkleistungsfähigkeit eines Systems wird damit immer mehr zu einem der bestimmenden Faktoren für die Bewertung der Leistungsfähigkeit eines Rechnersystems.
Die SPEC-CPU-Benchmarks beziehen die Netzwerkleistungsfähigkeit nicht in die Bewertung eines Rechnersystems ein, dafür wurden und werden spezielle Bewertungsverfahren entwickelt. Bei SPEC-Messungen zur CPU-Leistung in den Labors der Hersteller werden die Meßbedingungen so idealisiert, daß tatsächlich kein Netzwerkeinfluß vorhanden ist. In einer Rechnerumgebung, die wie hier im Campus-Netz der TU Chemnitz-Zwickau auf der Nutzung eines globalen, gemeinsamen Filesystems beruht, läßt sich der Einfluß des Netzwerksystems nicht beseitigen, wenn man die Leistungsfähigkeit für die Bedingungen ermitteln möchte, unter denen reale Applikationen ablaufen müssen.
Die Ermittlung der Netzwerkleistungsfähigkeit beziehungsweise die eingebettete Bewertung der Netzwerklast gestaltet sich dabei ziemlich kompliziert, da viele Faktoren ein Rolle spielen, die nicht nur von Zuständen im lokalen, zu testenden Rechnersystem abhängen.
Die Kommunikation zwischen Rechnersystemen erfolgt über Protokollstacks, die allgemein durch das OSI-Schichtenmodell beschrieben werden.

7	Application Layer	Anwendung
6	Presentation Layer	Datendarstellung
5	Session Layer	Kommunikationssteuerung
4	Transport Layer	Transport
3	Network Layer	Vermittlung
2	Link Layer	Sicherung
1	Physical Layer	Bitübertragung

Abbildung 3-2 OSI-Schichtenmodell für Netzwerkkommunikation

Die oberste Schicht stellt die Anwendung dar, die das Rechnernetz benutzt, die unterste Schicht beschreibt das physische Medium, über das der Datenaustausch zwischen Rechnern letztendlich abgewickelt wird. Die dazwischenliegenden Schichten haben verschiedene Transformations- und Sicherungsaufgaben bei der Realisierung der Kommunikation über das Netzwerk. Für verschiedene Anwendungsgebiete stehen viele unterschiedliche Protokollstacks bereit, die jeweils auf spezielle Erfordernisse dieser Anwendungen zugeschnitten sind. Ein Beispiel zeigt die folgende Abbildung:

5-7	Anwendung	SMTP	FTP	TELNET	NFS	TFTP
4	Transport	TCP			UDP	
3	Vermittlung	IP und ICMP				
1,2	Subnetz	Ethernet	FDDI	X.25	SLIP	....

Abbildung 3-3 Internet Protokollstack

Die transportierten Daten durchlaufen diese verschiedenen Protokollschichten auf dem lokalen System, den beteiligten entfernten Systemen und teilweise auf den dazwischenliegenden Netzknoten (Repeater, Bridges, Router), die der Strukturierung des Kommunikationsnetzwerks dienen. Dabei lassen sich für alle beteiligten Systeme Informationen zu den einzelnen Schichten für die verschiedenen Stacks gewinnen. Diese Vielzahl von Informationen ist jedoch nicht geeignet, die Netzwerklast allgemein zu beschreiben.

Für die angestrebte Netzlastbewertung auf dem lokalen Rechner muß innerhalb des Kommunikationssystems abstrahiert werden. Das Lastmaß muß auf die Last bezüglich des lokalen Systems reduziert werden. Die Einbeziehung der entfernten Systeme würde wiederum alle Lastkomponenten eines Systems umfassen (CPU, Speichersystem, I/O-System), da diese die Bewältigung der Netzwerklast mitbestimmen. Diese Erfassung stellt jedoch ein Problem exponentieller Komplexität dar. Die „globale“ Last auf dem Kommunikationsweg sollte jedoch zumindest im lokalen Lastmaß reflektiert werden. Dies wird durch die Sicherungs- und die Kommunikationsschichten gewährleistet. Sollten durch Komponenten des Kommunikationssystems aufgrund hoher Last Daten verworfen (*droping*) oder gar zerstört werden (*data corruption, collisions*), so wird bei Erkennung des Fehlers oder nach Ablauf einer bestimmten Zeit (*timeout*) eine erneute Anforderung generiert, wodurch sich auch die lokal repräsentierte Netzlast erhöht.

Am besten geeignet zur Repräsentation der Netzlast im lokalen System ist die Menge der über die Netzwerk-Interfaces des Rechners transferierten Daten. Diese Netzwerk-Interfaces stellen die Verknüpfung von Schicht 1 (Physische Übertragung) und Schicht 2 (zugehörige Sicherungsschicht für diese Übertragungsart) dar. Jede Form der Rechnerkommunikation muß über diese Schnittstellen ablaufen, so daß hier das Datenaufkommen erfaßt werden kann. Dazu steht auf allen betrachteten Systemen das Werkzeug *netstat -i* zur Verfügung. Es ermittelt an der Schnittstelle zwischen Vermittlungsschicht (3) und Sicherungsschicht (2), wieviel Datenpakete über die Netzwerk-Interfaces des Rechners ein- und ausgehen.

Bei den Benchmarktests in dieser Arbeit wird über NFS auf entfernt gespeicherten Programmcode und auf ebenfalls auf dem entfernten Fileserver befindliche Anwendungsdaten zugegriffen. Daher bietet es sich an, auch die Informationen zu betrachten, die für das NFS-Protokoll über das Werkzeug *nfsstat* bereitgestellt werden. NFS ist ein Protokoll der Anwendungsschicht (7), welches zur Kommunikationssteuerung (5) auf dem RPC-Protokoll basiert (Abbildung 3-4). In der Praxis werden oft bis 95% aller RPC-Operationen durch NFS-Zugriffsoperationen ausgelöst. Daher bezieht *nfsstat* die RPC-Ebene in die Auswertungen ein [Louk91].

7	NFS MOUNT NIS REX
6	XDR
5	RPC
4	UDP, TCP
3	IP
2	....
1	....

Abbildung 3-4 NFS-Protokollstack mit wichtigen Hilfsdiensten

NFS ist eine Netzanwendung, basierend auf dem Client-Server-Modell. Die zu untersuchenden Rechnersysteme arbeiten als NFS-Clienten. Daher sind für diese Arbeit nur die Client-Daten von *nfsstat* von Interesse.
Die NFS-Zugriffszahlen von *nfsstat* repräsentieren wie erwähnt nur einen Teil der gesamten Netzlast, daher eignen sie sich nicht für die Definition eines allgemeinen Lastmaßes für die Netzlast. An der TU Chemnitz-Zwickau spielt der Faktor NFS-Performance jedoch eine wichtige Rolle, da durch die /uni/global-Organisationen viele Anwendungen nur über das gemeinsame Filesystem arbeiten. Daher wird für diese Teilmenge der Gesamtnetzlast zusätzlich ein eigenes Lastmaß gebildet.
Durch die zunehmende AFS[7] - Integration wird der Einfluß der speziellen NFS-Last perspektivisch jedoch abnehmen. Die Betrachtung der NFS-Last ist dann durch die Betrachtung der entsprechenden Größen des AFS zu ersetzen. Dabei sind durchaus unterschiedliche Verhalten zu erwarten, speziell dadurch, daß in AFS integrierte Rechner ein lokales Caching vornehmen und damit möglicherweise das Datenaufkommen im Netz reduzieren können.

3.2.4.2 Auswahl und Erfassung der Beschreibungsdaten für die Netzlast

Für die Sammlung der Daten zur Netzlast des lokalen Systems wird das Systemwerkzeug *netstat* verwendet. Abhängig von den gewählten Optionen stellt es eine Vielzahl von auf das Netzwerk bezogenen Daten bereit, die Informationen zu verschiedenen Schichten des Kommunikationsstacks liefern. Standardmäßig wird eine Momentaufnahme für Aktivitäten aller aktiven Sockets für jedes Protokoll geliefert. Mit der Angabe von Optionen lassen sich unter anderem folgende Daten aus den entsprechenden Verwaltungsstrukturen ermitteln:

- Statistiken über Benutzung des für die Netzwerkbenutzung verwendeten Hauptspeichers (*stream buffer caches*)
- Statistiken für verschiedene Adreßfamilien (inet, unix)
- Routingtabellen

[7] Andrew File System

- Per-Protokoll-Statistiken

Im letzten Abschnitt wurde bereits herausgearbeitet, daß die Menge des Datenverkehrs über die Netzwerkschnittstelle(n) gut geeignet ist, die Last im Netzwerk mit Bezug zum lokalen System zu bewerten. Für jede konfigurierte Schnittstelle erhält man mit

- *netstat -i* :

name -	Interface-Name
mtu -	maximum transfer unit, die maximale Größe eines Paketes, das von dieser Schnittstelle verschickt wird
net -	Netzwerk, an welches diese Schnittstelle angeschlossen ist
address -	Internet-Adresse der Schnittstelle (symbolischer Name), -n liefert die numerische IP-Adresse
ipkts -	Anzahl der empfangenen Datenpakete seit dem Systemstart
ierrs -	Anzahl der fehlerhaft empfangenen Datenpakete
opkts -	Anzahl der abgeschickten Datenpakete seit dem Systemstart
oerrs -	Anzahl der fehlerhaft verschickten Pakete
colls -	Anzahl der Paketkollisionen beim Abschicken von Daten über diese Schnittstelle (bei Ethernet-Schnittstellen)

Mit dem Schalter -I und dem Interface-Namen kann diese Statistik für einzelne Netzwerkschnittstellen angefordert werden. Für die Abfrage dieser Statistikdaten bietet *netstat* zwei verschiedene Modi an. Normalerweise werden bei den Schnittstelleninformationen die Absolutwerte seit dem Systemstart ausgegeben. Wird zusätzlich ein Intervall (n Sekunden) spezifiziert, so wird zunächst wieder eine Gesamtinformation seit dem Bootvorgang und anschließend aller n Sekunden eine Statistik für das letzte Intervall geliefert. Systemabhängig wird nach einer bestimmten Anzahl von Intervallen wieder ein Gesamtreport ausgegeben. Die intervallgesteuerte Abfrage ist nur in Verbindung mit -I (spezielles Interface) möglich und liefert die Daten nur für dieses (oder das Primärinterface) und eine Summe über alle Interfaces. Beim Fall von Systemen mit mehreren Schnittstellen zum Netzwerk, die noch dazu auf verschiedenen Technologien basieren können (Ethernet, FDDI, ATM, SLIP, PPP ...), ist damit ein Mehrfachaufruf von *netstat* erforderlich, wodurch ein nicht mehr akzeptabler Overhead für die Lastmessung entsteht, da diese Last direkten Einfluß auf CPU-, Speicher- und I/O-Performance nehmen kann. Daher wird für die Ermittlung der Netzlast die Variante bevorzugt, daß bei Start und Ende einer Messung (Benchmarklauf) die Absolutwerte ermittelt werden und durch Differenzenbildung die Absolutbelastung für den Meßzeitraum berechnet wird. Durch Relativierung über die Laufzeit ergibt sich der mittlere Durchsatz über den Meßzeitraum. Als Belastungsmaß werden die eingegangenen und abgeschickten Pakete gewählt, so daß sich als Einheit für diesen Durchsatz Pakete pro Sekunde ergibt.

$$\textit{Netz-I/O-Rate} = \frac{\textit{(\#In-Packets + \#Out-Packets) * packetsize(mtu) [Byte]}}{\textit{Zeit [s]}} \quad \text{[Byte/s]}$$

Um eventuelle Besonderheiten bei einzelnen Messungen besser bewerten zu können, werden auf die gleiche Art und Weise auch mittlere Raten für fehlerhafte Ein- und Ausgabepakete und der Anteil der Paketkollisionen (bei Ethernet-Schnittstellen) gebildet. In der Literatur werden verschiedene Angaben zu

akzeptablen Fehleranteilen an der Paketanzahl gemacht, [Louk91] bezeichnet 0,025% fehlerhafte Ausgabepakete und bis zu 10% Kollisionen als „normale“ Werte. Die Fehlerhäufigkeiten beschreiben nicht die Netzlast als solche, sie stellen jedoch gute Indikatoren für besondere Lastsituationen dar. Deshalb können sie bei der Interpretation besonderer Effekte hilfreich sein.
Es muß beachtet werden, daß die Ermittlung der Durchsatzrate in Paketen nur ein „unscharfes“ Bild der Netzlast liefert, da Pakete unterschiedliche Größen aufweisen. Allgemein sind Pakete zur Datenanforderung wesentlich kleiner als die Pakete mit gelieferten oder verschickten Daten und treten einzeln auf, während Datenpakete oft die Größe des *mtu*[8]-Wertes haben und in größeren Mengen verschickt oder empfangen werden. Leider bietet nur Digital UNIX die Möglichkeit, die Menge der Daten pro Schrittstelle in Byte zu ermitteln, woraus sich dann die durchschnittliche Paketgröße berechnen läßt. Für den Vergleich der unterschiedlichen Systeme besteht für die Erfassung der Datenmenge nur die Möglichkeit der Verknüpfung von Paketanzahl und der maximalen Paketgröße (mtu), um eine (theoretische) Schnittstellendurchsatzrate in KByte/s zu ermitteln. In Relation zur theoretischen, maximalen Durchsatzrate der jeweiligen Schnittstelle läßt sich damit auch die prozentuale Auslastung der Schnittstelle und in der Summe die Auslastung für das Gesamtsystem ermitteln, wobei auch diese Werte durch die Mißachtung der tatsächlichen Paketgrößen nur approximierte Näherungen sind.
Im vorigen Abschnitt wurde das Datenaufkommen über das RPC/NFS-Protokoll als wichtige Teillast bei der Bewertung der Netzlast herausgearbeitet. Zur Datensammlung dient das Werkzeug *nfsstat*. Es liefert statistische Informationen zum NFS-Protokoll (Schicht 7) und zum RPC-Protokoll (Schicht 5). Da die zu untersuchenden Maschinen als NFS-Clienten arbeiten, sind nur die Statistiken für die Client-Funktionalität relevant (nfsstat -c). Daneben ist es möglich, getrennte Statistiken für NFS Version 2 und Version 3 zu generieren, was hier nicht von Interesse ist. Im Einzelnen stehen folgende Daten zur Verfügung :

- *nfsstat -c* :

rpc: Informationen der *Remote Procedure Call* -Schnittstelle

calls -	Gesamtzahl der Funktionsaufrufe
badcalls -	Anzahl fehlgeschlagener Aufrufe
retrans -	Anzahl wiederholter Aufrufe (wegen timeout)
badxid -	Empfangene Antworten ohne entsprechende Anforderungen (out of date transaction ID)
timeout -	Anzahl der in der konfigurierten Wartezeit nicht beantworteten Rufe (erzeugt Retransmission)
wait	- Anzahl RPCs, die mangels Kapazität verzögert wurden

nfs: Informationen zum *Network File System* -Protokoll

calls -	Gesamtanzahl der NFS-Funktionsrufe
badcalls -	Anzahl fehlgeschlagener Funktionsrufe
nclget -	Anzahl Versuche, eine Aufrufstruktur zu benutzen
nclsleep -	Wartezeit für Kernel bis zur Bereitstellung einer Aufrufstruktur

[8] maximum transfer unit

null, getattr, setattr, root, lookup, readlink, read, wrcache, write, create, remove, rename, link, symlink, mkdir, rmdir, readdir, fsstat

--> Anzahl und prozentualer Anteil spezieller NFS-Rufe an der Gesamtzahl der NFS-Calls, die Verteilung ist applikationsabhängig

In der Praxis beträgt der Anteil der NFS-Calls an den Funktionsrufen des RPC-Protokolls 95% bis 100%. Als NFS-Last-Näherung könnte damit auch die Anzahl der RPC-Rufe verwendet werden. Diese Erweiterung würde sich deshalb anbieten, weil durch zusätzliche Betrachtung von erforderlichen Neusendungen bzw. der registrierten Timeouts der RPC-Ebene (ähnlich zu den Fehlerraten bei netstat) gute Indikatoren für ungewöhnliche Verhältnisse während der Lastmessung gegeben sind.
Für die Erfassung der Werte bieten sich wieder 2 Modi an, die Differenzen von Absolutwerten und die Intervallmessungen (nfsstat -i Sekunden). Wegen der erheblichen produzierten Datenmenge und des damit verbundenen Verarbeitungs-Overheads, sowie des nur ergänzenden Charakters dieser NFS-Last, wird die Anfangs- und Ende-Messung mit Differenzenbildung als ausreichend betrachtet. Die Anzahl der NFS-Rufen bzw. RPCs wird in Relation zur Benchmarklaufzeit betrachtet und liefert die *NFS-Call-Rate* in calls/Sekunde.

$$NFS\text{-}Call\text{-}Rate = \frac{\#\ NFS\text{-}Calls\ [calls]}{Zeit\ [s]} \quad [calls/s]$$

Der prozentuale Anteil von RPC-Retransmissions, -Timeouts, -Badcalls wird als Interpretationshilfe zur manuellen Auswertung außergewöhnlicher Ergebnisse gespeichert, fließt jedoch nicht direkt in das Lastmaß ein.

3.2.4.3 Definition von Lastmaßen für die Netzlast

Auf Basis dieser ermittelbaren Daten werden unter Beachtung der Erfassungsmöglichkeiten bei den verwendeten Werkzeugen folgende Maße für die Repräsentation der Netzlast während eines Benchmarklaufes definiert:

$$Netz\_I/O\_Rate = \frac{\sum_{SS=1}^{n} (((ipkts_{SS_{END}} + opkts_{SS_{END}}) - (ipkts_{SS_{START}} + opkts_{SS_{START}})) * mtu_{SS})}{Benchmarklaufzeit}$$

[Byte/s oder KByte/s]

SS	*Netzwerkinterface (insgesamt n im System)*
$ipkts_{SS}$	*Systemzählerstand für über Schnittstelle SS eingegangene Pakete*
$opkts_{SS}$	*Systemzählerstand für über Schnittstelle SS abgeschickte Pakete*
mtu ss	*maximale Paketgröße der Schnittstelle SS*
Start,End	*Wert des entsprechenden Zählers bei Start bzw. Ende eines Benchmarks*

Als wichtige Teillast der Netzlast wird die NFS-Last auf Basis der Daten von *nfsstat* als *NFS-Call-Rate* definiert:

$$NFS\_Call\_Rate = \frac{calls_{NFSEND} - calls_{NFSSTART}}{Benchmarklaufzeit}$$

[Calls/s]

3.3 Realisierung der Lasterfassung

In der vorangegangenen Definition von Lastmaßen zur Beschreibung der Systemlast wurde bereits auf die möglichen Betriebsmodi der einzelnen Meßwerkzeuge hingewiesen und diskutiert, welche Erfassungsmöglichkeit sich jeweils in Abwägung von Aussagekraft der Werte und dem dafür erforderlichen Aufwand anbietet. Diese Aussagen werden hier zusammengefaßt und die Umsetzung in einem Prototyp für einen Lastmonitor vorgestellt.
Alle gewählten Werkzeuge außer *uptime* bieten die Möglichkeit ein Meßintervall zu spezifizieren, so daß es möglich ist, in bestimmten Abständen eine Statistik für einen zurückliegenden, zeitlich genau definierte Zeitabschnitt zu erstellen. Eine solches *intervallgesteuertes Monitoring* wird auch als *Sampling* (Kapitel 1) bezeichnet, aller n Zeiteinheiten wird ein Sample protokolliert. Problematisch gestaltet sich die Tatsache, daß im voraus nicht fest steht, wie lange die Daten erfaßt werden müssen, da die Laufzeit eines Benchmarks eben durch die zu erfassende Last bestimmt wird. Es muß ein genügend langer Protokollierungszeitraum gewählt werden, aus dem dann der zu bewertende Zeitraum für den eigentlichen Benchmark im nachhinein herausgelöst wird.
Besonders komfortabel ist die Benutzung des ursprünglich aus System V stammenden *System Aktivity Reporters (sar).* Die Samples werden jeweils nach durch die Intervallänge beschriebenen Zeiteinheiten in ein speicherplatzsparendes Binärfile geschrieben. Nach dem Benchmarklauf kann man *sar* erneut benutzen, um aus diesem Binärfile spezielle Statistiken für den durch Start- und Endzeit des Benchmarks bestimmten Zeitraum zu erstellen. *sar* generiert dann einen Report bestehend aus den Samples innerhalb des Zeitraumes und ermittelt selbständig Durchschnittswerte für diesen Abschnitt. Bei den anderen Werkzeugen müssen diese Auswertungen manuell erfolgen.
Das Werkzeug *Uptime*, das zur Ermittlung der Länge der Prozeßwarteschlange verwendet wird, bietet keinen Intervallmechanismus. Hier muß dieser durch eine eigene Zeitsteuerung (sleep) realisiert werden.
Daneben bieten die meisten Werkzeuge die Möglichkeit einer summierten Statistik oder die Ausgabe von Zählerständen von Ereigniszählern seit dem Systemstart. Durch Differenzenbildung von End- und Startwert zu einem Meßzeitraum, der hier von der Benchmarklaufzeit bestimmt wird, kann die Ereignisanzahl im Meßzeitraum ermittelt werden. Dies bietet den Vorteil, daß nur wenige Werte auszuwerten sind und diese präzise für Start- und Endzeitpunkt ermittelt werden können.
Zur Bewertung der Last während eines Benchmarklaufes müssen die für die Lastmaße ermittelten Werte zusammengefaßt werden. Dazu werden die für Intervalle vorliegenden Werte über den gesamten Meßzeitraum hinweg arithmetisch gemittelt. Somit wird die mittlere Last während eines Benchmarklaufes betrachtet. Bei der Auswertung der Absolutwerte seit dem Systemstart werden die ermittelten Zahlen für bestimmte Ereignisse ebenfalls gemittelt, und zwar über die Länge des Benchmarklaufes. Für Werkzeuge mit beiden Möglichkeiten ergeben sich daraus normalerweise dieselben mittleren Lasten.
Natürlich ist es grundsätzlich ein Unterschied, ob eine bestimmte mittlere Last aus einer kontinuierlichen Belastung resultiert, die vom Rechnersystem gleichmäßig verarbeitet wird, oder ob sich dieser Mittelwert aus Phasen ohne Belastung und Phasen besonders hoher Belastung (*Peaks*) ergibt, wobei diese hohen Belastungen evtl. die Fähigkeiten des Systems überschreiten. In der technischen Realisierung werden alle betrachteten Lasten jedoch über Warteschlangen beziehungsweise Puffer abgearbeitet. Dadurch werden Lastschwankungen ausgeglichen, so daß sich für längere Beobachtungszeiträume relative Gleichverteilungen einstellen. Für die eigentliche, die Last bewältigende Systemeinheit wird die kontinuierliche Belastung durch die vorgelagerten Queues und Puffer quasi

simuliert. Daher ist die Anwendung des arithmetischen Mittels zur Beschreibung einer Last über einen längeren Zeitraum hinweg durchaus geeignet und legitim.
In diesem Zusammenhang ist die Protokollierung von Lasten, die auf Anzahlen von Ereignissen basieren, durch Differenzenbildung von Zählerständen an Meßende und Meßbeginn auch ausreichend. Die ersten Versuche in dieser Arbeit haben daneben aufgezeigt, daß die Genauigkeit der Messungen einzelner Werkzeuge gewissen Schwankungen unterworfen ist. Dies betrifft speziell die Werkzeuge *vmstat* und *iostat* im Samplingbetrieb und ebenso die Ergebnisse von *sar*. Speziell niedrige Werte bei Page-Out-Raten und Disktransferraten führen durch Rundung zu Fehlern und bei Rundung auf Null zu dem Eindruck, daß keinerlei Last vorhanden war. Treten viele Samples mit solchen Fehlern auf, so wird logischerweise auch der aus den Samples berechnete Mittelwert für die Gesamtlaufzeit verfälscht. Betrachtet man statt dessen die absoluten Werte der Ereigniszähler bei Benchmarkstart und -ende und berechnet daraus die mittleren Raten für die Ereignisse, dann werden die Rundungsfehler umgangen, die präzise mittlere Last kann berechnet werden. Bei höheren Lasten ist nicht mehr relevant, da die Rundungsabweichungen in Relation zu den Lastwerten sehr klein sind.
Zu Interpretationszwecken bietet es sich bei Werkzeugen mit nicht zu großen Datenmengen auf jeden Fall an, die Lastwerte über Sampling zu ermitteln und zu archivieren, um Abweichungen und spezielle Effekte erklären und begründen zu können. Diese Methode wird für die Erfassung von CPU-, Speicher- und Disk-I/O-Last mit *uptime*, *vmstat* und *iostat* bzw. *sar* gewählt. Die Erfassung der prozentualen CPU-Auslastung und der mittleren Warteschlangenlänge kann nur im Samplingverfahren erfolgen, da diese Maße nicht durch Anzahlen von Ereignissen, sondern nur durch Zeiträume mit bestimmenden Zuständen charakterisiert sind.
Wegen des größeren Overheads bei der Erfassung der Netzlast mit *netstat* und *nfsstat* wird dort die Methode der Differenzen von Absolutzahlen verwendet. Als Indikatoren für besondere Bedingungen während des Testlaufes stehen hier Fehlerraten für die einzelnen Netzwerkprotokolle zur Verfügung.

Damit stehen die Erfassungsmodi für die einzelnen Lastmaße fest. Die Aufstellung im nächsten Abschnitt faßt für die zu bewertenden Teilsysteme noch einmal die gewählten Lastmaße und die gewählten Modi zur Erfassung der Werte zusammen. Außerdem wird auf andere interessante Größen verwiesen, deren Erfassung als Zusatzinformation und Fehlerindikator sich anbietet, die jedoch nicht in eines der Lastmaße einfließen.
Auf Basis dieser Aufstellung wird danach die praktische Realisierung der Lasterfassung für die Messungen in dieser Arbeit dargestellt.

3.3.1 Zusammenfassung der Lastmaße und Erfassungsmodi

- *CPU-Last*

CPU-Auslastung:

Werkzeuge: vmstat (BSD-Style)
sar -u (SystemV-Style)
Verfahren: Sampling im Intervall 1 Minute
Ergebnis: n %idle-Werte für n Meßintervalle
Auswertung:

$$\varnothing CPU\_Auslastung = \frac{\sum_{Intervall=1}^{n}(100 - \%idle_{Intervall})}{n}$$

[%]

Zusatz: Differenzierung in prozentualen Anteil für Nutzer- und Systemprozesse möglich

Run-Queue-Länge:

Werkzeuge: uptime
Verfahren: Sampling im Intervall 1 Minute
Ergebnis: n Werte für Run-Queue-Länge für n Meßintervalle
Auswertung:

$$\varnothing Run\_Queue\_Länge = \frac{\sum_{Intervall=1}^{n} Run - Queue - Länge_{Intervall}}{n}$$

[Prozeßanzahl]

- *Speicherlast*

Page-Out-Rate:

Werkzeuge: vmstat -s (BSD-Style-Systeme)
Verfahren: Abfrage Systemzähler bei Start- und Ende des Benchmarks und Bildung der Differenz
Ergebnis: Absolutanzahl der durch Paging und Swapping ausgelagerten Seiten während der Benchmarklaufzeit
Auswertung:

$$\varnothing Page\_Out\_Rate = \frac{((PPageO_{ENDE} - PPageO_{START}) + (PSwapO_{ENDE} - PSwapO_{START})) * PageSize}{Benchmarklaufzeit}$$

[KByte/s]

PPageO - Pages Paged Out (Anzahl beim Paging ausgelagerter Speicherseiten)
PSwapO - Pages Swapped Out (Anzahl beim Swapping ausgelagerter Seiten)

Zusatz: -Anzahl der Page-Outs und Swap-Outs steht auch zur Verfügung
-Bei der Erfassung der CPU-Auslastung mit vmstat -S werden auch Paging- und Swapping-Informationen zu einzelnen Intervallen protokolliert, diese können für erweiterte Interpretationen betrachtet werden

bzw.

Werkzeuge: sar -w -p (-g) (SystemV-Style-Systeme)
Verfahren: Sampling im Intervall von einer Minute
Ergebnis: n Werte für Auslagerungsraten für n Meßintervalle
Auswertung: sar ermittelt selbst einen Durchschnittswert über n Sample-Intervalle für einen durch Start- und Endzeit spezifizierten Zeitraum

$$\varnothing Page\_Out\_Rate = \frac{\sum_{Intervall=1}^{n} PagesPagedOut\_per\_sec_{Intervall} * PageSize}{n}$$

[Kbyte/s]

- *Disk-I/O-Last*

Disk-I/O-Rate:

Werkzeuge:	iostat	(BSD-Style-Systeme)
	sar -d	(SystemV-Style-Systeme)
Verfahren:	Sampling im Intervall von einer Minute	
Ergebnis:	n Werte für Disktransferraten für jede Harddisk für n Intervalle	
Auswertung:		

$$Disk\_I/O\_Rate = \frac{\sum_{Intervall=1}^{n} \sum_{Disk=1}^{m} Transferrate_{DiskIIntervall}}{n}$$

[KByte/s]

Zusatz: -Anzahl der Transfers von und zu Disk(s) steht auch zur Verfügung

- *Netz-I/O-Last*

Netz-I/O-Rate:

Werkzeuge: netstat -i für jedes Netzwerkinterface
Verfahren: Abfrage von Protokollzähler bei Start und Ende des Benchmarks
Ergebnis: Anzahl ein- und ausgegangener Pakete für jede der n Schnittstellen (SS)
Auswertung:

$$Netz\_I/O\_Rate = \frac{\sum_{SS=1}^{n} (((ipkts_{SS_{END}} + opkts_{SS_{END}}) - (ipkts_{SS_{START}} + opkts_{SS_{START}})) * mtu_{SS})}{Benchmarklaufzeit}$$

[KByte/s]

Zusatz: Kollisionsraten (Ethernet) und Fehlerraten für Ein- und Ausgabepakete ermittelbar

NFS-Call-Rate:

Werkzeuge: nfsstat -c
Verfahren: Abfrage von NFS-Protokollzähler bei Start und Ende des Benchmarks
Ergebnis: absolute Anzahl der NFS-Calls während des Benchmarklaufes
Auswertung:

$$NFS\_Call\_Rate = \frac{calls_{NFS_{END}} - calls_{NFS_{START}}}{Benchmarklaufzeit}$$

[calls/s]

Zusatz: Retransmissionsraten als Fehlerindikator auf RPC-Ebene ermittelbar

3.3.2 Praktische Realisierung im Lastmonitor

Die originale Ablaufumgebung der SPEC-CPU-Benchmarks bietet an verschiedenen Stellen eines Meßlaufes die Möglichkeit, an sogenannten Monitor-Punkten („monitoring hooks“) Programme in den Ablauf einzubinden. Zwei solcher Punkte befinden sich unmittelbar vor dem Start des eigentlichen Benchmarks und der Startzeit-Messung bzw. unmittelbar nach dem Ende des Benchmarks und der Messung der Endzeit.
Der erste Punkt wird benutzt, um den entworfenen Lastmonitor zu starten. Wegen der unbestimmten Laufzeit eines Benchmarks wird zunächst eine hohe Anzahl von Samples konfiguriert, damit nicht der Lastmonitor vor dem Benchmark endet. Die für den Sample-Mode konfigurierten Tools protokollieren ab diesem Zeitpunkt im Intervall von einer Minute die aktuellen Lasten jeweils in eine Datei. Außerdem werden mit *vmstat, netstat* und *nfsstat* die Zählerstände der beschriebenen Ereigniszähler abgespeichert.
Im zweiten Monitor-Punkt wird der Lastmonitor terminiert und mit erneutem Aufruf von *vmstat, netstat* und *nfsstat* der Stand der entsprechenden Ereigniszähler bei Benchmarkende gespeichert. Ergebnis eines solchen Testlaufes ist ein Satz von Dateien, die die Benchmarklaufzeiten, die Samplingprotokolle von *uptime, vmstat, iostat* bzw. *sar*, sowie die mit *vmstat, netstat, nfsstat* ermittelten Anfangs- und Endstände der Ereigniszähler enthalten. Unabhängig vom eigentlichen Benchmarklauf erfolgt später die Auswertung entsprechend der zuvor definierten Lastmaße (Perl-Skript get_run_data.sh). Dabei müssen die Spezifika auf den einzelnen Systemen beachtet werden, z.B. unterschiedliche Formate oder Einheiten bei den gelieferten Werten.

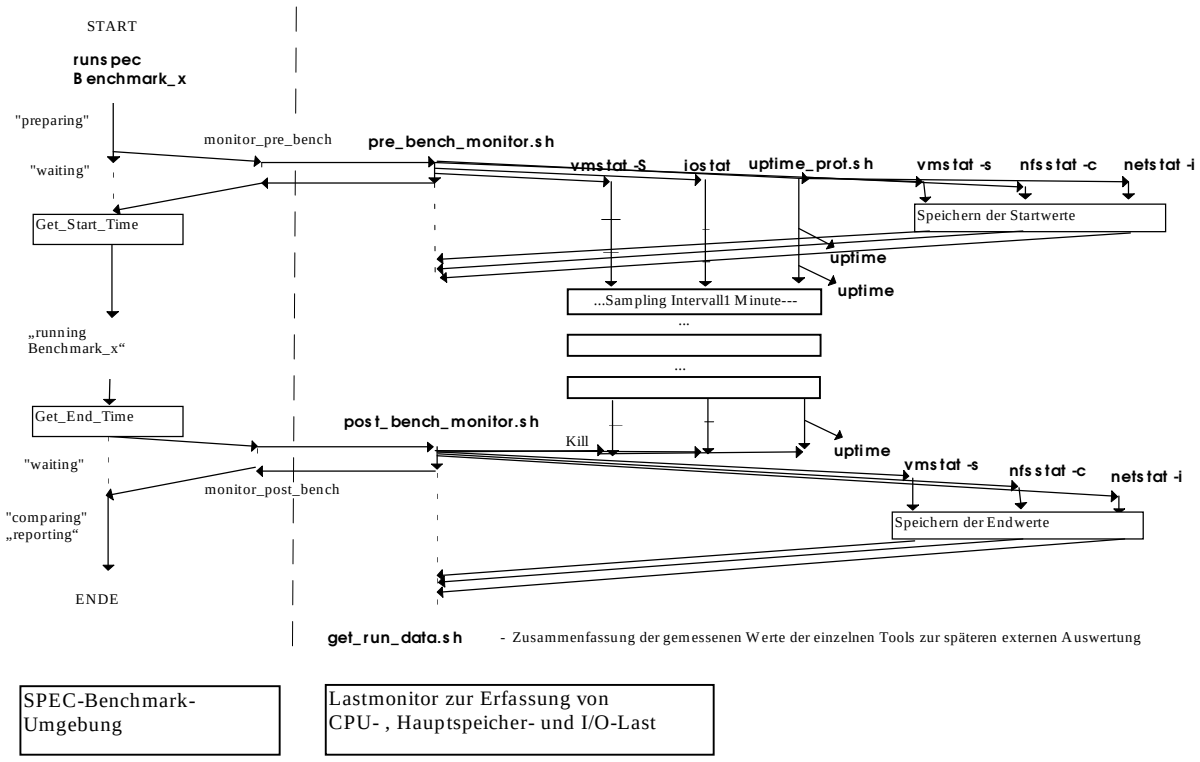


Abbildung 3-5 Ablaufschema einer SPEC-CPU-Leistungsmessung in Verbindung mit einem Lastmonitor

Die Abbildung zeigt den Ablauf auf einem System aus der BSD-Familie. Das Ablaufschema für ein SystemV-basiertes System stellt sich ähnlich dar, hierbei entfallen alle Aufrufe von *vmstat* und *iostat*, dafür ist ein Sampling mit *sar* zu starten. Bei SunOS 5.3 können alle Werkzeuge eingesetzt werden.

Die Einbindung von solchen Nutzerzusätzen erfolgt in einem Konfigurationsfile („config-id“.cfg) im Verzeichnis $SPEC/config. In diesem Konfigurationsfile werden auch die Optimierungen für Baseline- und volloptimierte Messungen, sowie aus Portabilitätsgründen erforderliche Compilerschalter spezifiziert. Der Start der Leistungsmessung muß dann unter Angabe des Konfigurationsfiles mit dem Schalter

`$ runspec -c config_id ...` erfolgen.

Die Angaben

```
...
monitor_pre_bench= pre_bench_monitor.sh
monitor_post_bench= post_bench_monitor.sh
...
```

im allgemeinen Teil des Konfigurationsfiles bewirken hier die Ausführung der entsprechenden Shell-Scripte unmittelbar vor und nach dem Benchmarklauf. Diese Shell-Scripte starten und beenden die einzelnen Meßwerkzeuge zur Lasterfassung, so wie dies in der Abbildung dargestellt ist. Eventuell benötigte Umgebungsvariablen der SPEC-Ablaufumgebung (Arbeitsverzeichnisse, Benchmark-Name, Nutzernummern, ..) werden von dieser exportiert und stehen in den Monitor-Shells zur Verfügung ([SPECdoc]-howto.txt).
In der Anlage B dieser Arbeit ist der Ablauf einer Lastmessung innerhalb eines SPEC-Benchmark-Laufes an einem Beispiel dargestellt, zu diesem Beispiel gehören auch die zum Starten und zum Beenden des Monitorings konstruierten Shell-Skripte.

4 Durchführung der praktischen Messungen

In den praktischen Untersuchungen dieser Arbeit werden mit den SPEC-CPU-Benchmarks Leistungsmessungen für verschiedene Rechner des URZ der TU Chemnitz-Zwickau vorgenommen. Dabei werden die Vorgänge in den Systemen während der Benchmarkläufe mit einem im vorigen Kapitel entwickelten Lastmonitor erfaßt.
Ziel ist die Gewinnung von Informationen, welche Teilsysteme und Teillasten eines Rechnersystems einen wesentlichen Einfluß auf die Abarbeitung der Benchmarks ausüben. Daneben soll das Lastverhalten der Rechner und Betriebssysteme der einzelnen Hersteller hinsichtlich des generellen Verhaltens der Plattform und hinsichtlich konkreter Engpässe bei den Testsystemen betrachtet werden.
Solche Untersuchungen erfordern eine Vielzahl von Messungen, um mit einem großen Stichprobenumfang möglichst das gesamte Lastspektrum eines Rechensystems zu erfassen, das im normalen Betrieb auftritt. Eine Durchführung von Messungen mit den vollständigen SPEC-CPU-Suiten würde den zeitlichen Rahmen dieser Arbeit sprengen oder nur wenige Messungen mit den einzelnen Benchmarks erlauben. Deshalb werden im nächsten Abschnitt für die praktischen Messungen jeweils 2 Benchmarks aus der CFP95- bzw. der CINT95-Suite ausgewählt. Mit diesen sollen auf jedem der 4 ausgewählten Testsysteme etwa 30...50 Testläufe zu verschiedenen Zeiten durchgeführt werden.
Daran schließt sich eine Beschreibung der Konfigurationen der einzelnen Testsysteme an, auf denen die Messungen durchgeführt werden.
Damit ist der Untersuchungsraum für die praktischen Messungen dieser Arbeit definiert.

4.1 Auswahl von Benchmarks für praktische Messungen

4.1.1 Benchmark 101.tomcatv (CFP95)

Dieser Benchmark stammt aus dem Anwendungsbereich der Flüssigkeitsdynamik und der geometrischen Translation. Dabei werden Netze um allgemeine geometrische Bereiche generiert. Der Benchmark ist in Fortran implementiert und arbeitet mit Fließkommazahlen doppelter Genauigkeit.
Kern des Benchmarks sind Manipulationen an einer Vielzahl von Matrizen und Feldern, auf die in sehr unterschiedlichen Reihenfolgen zugegriffen wird. Der Hauptspeicherbedarf für die Daten liegt relativ hoch bei ca. 15 MByte, die wegen des gestreuten Zugriffes auch fast ständig vollständig im HS benötigt werden. Durch die gestreuten Zugriffe ist keine zu starke Cache-Freundlichkeit anzunehmen. Der Programmcode benötigt nur etwa 100 KByte Hauptspeicher und weist eine relativ hohe Lokalität innerhalb einzelner Schleifen auf, so daß der Code als Cache-freundlich bewertet werden kann.
Im Gegensatz zu einigen anderen CFP-Benchmarks enthält dieser Benchmark auch einen erhöhten Anteil von Ein- und Ausgaben, die sich, wie häufig in numerischen Anwendungen, auf Programmstart und Programmende konzentrieren. Das Benchmarkprogramm selbst muß einmal geladen werden. Innerhalb des Tests werden zu Beginn des Programms ein 99 Byte großes Initialisierungsfile und ein 10

MByte (!) großes File mit dem Datenmodell des Test gelesen, am Ende wird ein etwa 21 KByte großes Ergebnisfile geschrieben.

4.1.2 Benchmark 102.swim (CFP95)

Anwendungsfeld der zugrunde liegenden numerischen Anwendung ist die Wettervorhersage. Es wird ein Gleichungssystem für ein „Wasser-Verflachungs-Modell“ durch endliche Differenzenapproximation gelöst. Dieser Benchmark ist ebenfalls in Fortran implementiert, das Programm benutzt allerdings nur Fließkommazahlen mit einfacher Genauigkeit. Der Speicherbedarf liegt ebenfalls bei etwa 15 MByte Hauptspeicher, wobei der Zugriff bei diesem Benchmark eher in sequentiellen Folgen erfolgt. Innerhalb des bis zu 110 KByte großen Programmcodes herrscht eine ziemlich hohe Lokalität innerhalb einiger Unterprogramme, so daß dieser Benchmark als insgesamt speicherfreundlicher als 101.tomcatv einzuschätzen ist.
Das Programm wird während des Benchmarks zweimal mit unterschiedlichen Iterationstiefen gestartet.
Die Ein- und Ausgaben des Programms beschränken sich dabei auf das Lesen eines 51 Byte großen Initialisierungsfiles und das Schreiben eines 475 Byte großen Protokollfiles. Dieses I/O-Profil ist nicht unbedingt typisch für numerische Anwendungen, dafür besteht aber nahezu Unabhängigkeit vom Ein- und Ausgabesystem.

4.1.3 Benchmark 124.m88ksim (CINT95)

Dieser Benchmark hat die Simulation des Motorola-88100-Prozessors bei der Durchführung eines Speichertests und dem Lauf des bekannten Dhrystone-Benchmarks [Weic84] zum Gegenstand. Die Realisierung erfolgte in C. Der Benchmark weist hohe Speicherforderungen auf und ist prozessorintensiv, ohne aber die FPU mit spezieller Fließkommaarithmetik zu belasten. Die Speicherforderungen des Benchmarks erfolgen dynamisch. In [Weic_ El_96] wird nur der statische Speicherbedarf mit 242 KByte für Code und 139 KByte für Daten angegeben. Um den tatsächlichen Speicherbedarf zu ermitteln, wurden während mehrerer Benchmarkläufe die tatsächlich allokierten Speichermengen des Benchmarkprozesses überwacht. Dies ergab einen maximalen Speicherbedarf von ca. 25 MByte, wobei die Größe des tatsächlich speicherresistenten Anteils zeitweilig kleiner war, ohne daß eine sichtbare Beeinträchtigung der Benchmarkabarbeitung erkennbar war, was wiederum für eine gewisse Lokalität innerhalb der Verteilung der Speicherzugriffe spricht.
Der Benchmarkcode muß einmal geladen werden. Die Ein- und Ausgabeintensität ist gering , es wird zu Beginn des Tests ein 213 Byte großes Initialisierungsfile gelesen und während des Ablaufes wird auf zwei ca. 70 KByte große Datenfiles zugegriffen. Die Ergebnisprotokollierung erfolgt am Ende in ein File von ca. 12 KByte Größe. Damit kann eine relative Unabhängigkeit von direkten Einflüssen des I/O-Systems abgeschätzt werden.

4.1.4 Benchmark 126.gcc (CINT95)

Dieser Benchmark stammt aus einem Anwendungsgebiet, das auch ein Hauptanwendungsfeld im Rechenbetrieb des URZ darstellt. Mit dem GNU-C-Compiler wird eine Reihe von Programmen übersetzt. Aus bereits mit einem Präprozessor vorübersetzen Quellcode wird optimierter SPARC-Maschinencode erzeugt. Dabei wird eine Reihe aufwendiger Optimierungen durchgeführt, dazu gehören unter anderem Speicherausrichtung, Sprungoptimierung und Schleifenauflösung (*loop unrolling*).
Dieser Benchmark weist eine hohe Ein- und Ausgabeintensität auf. Insgesamt werden während eines Testlaufes 56 einzelne Übersetzungen gestartet. Neben dem Laden des Programmcodes von ca. 2 MByte erfordert dies jeweils das Lesen eines Quellcode-Files mit Größen von 21 bis 306 KByte und das Speichern der erzeugten Programmfiles, die Größen zwischen 29 und 308 KByte aufweisen. Insgesamt werden über den gesamten Benchmarklauf hinweg etwa 6,4 MByte Daten gelesen und 8,4 MByte Ausgabedaten geschrieben.
Derartige Übersetzungsprozesse weisen generell einen hohen dynamischen Hauptspeicherbedarf auf. In der Literatur findet sich wieder nur ein Verweis auf die statische Größe von 210 KByte für Daten. Die hier ebenfalls durchgeführte experimentelle Ermittlung des dynamischen Speicherbedarfs durch Überwachung der Prozesse und des allokierten Speichers ergab sehr unterschiedliche Forderungen zwischen 1,6 MByte und 28 MByte für die einzelnen Übersetzungsprozesse, wobei im Mittel 9 MByte gemessen wurden.
Das von den anderen Benchmarks abweichende Programmprofil mit der besonderen Ein- und Ausgabeintensität und -verteilung, sowie die schwankenden Speicherforderungen, lassen auch von den anderen Tests abweichende Einflüsse erwarten. Besonders interessant sind diese unter dem Aspekt, daß dieser Benchmark sicher derjenige in der Auswahl ist, der typischen Anwendungen im Praxisbetrieb am ähnlichsten ist.

4.2 Beschreibung der betrachteten Testsysteme

Die nachfolgend beschriebenen Rechnersysteme werden für die praktischen Untersuchungen dieser Arbeit verwendet. Die angegebenen symbolischen IP-Maschinennamen bzw. die Kurzformen werden in der Folge auch zur Bezeichnung der einzelnen Systeme verwendet. Die angegebenen Daten entsprechen in etwa denen, die nach den SPEC-Reporting-Rules bei einer offiziellen Veröffentlichung von Ergebnissen einer SPEC-CPU-Leistungsmessung aufgeführt sein müssen.

Anwendungsfeld	Compute-Server	Compute-Server	Workstation (Referenzmaschine)	Workstation (Referenzmaschine)
Name	samson.hrz.tu-chemnitz.de	tantalus-f.hrz.tu-chemnitz.de	silly.hrz.tu-chemnitz.de	decency.hrz.tu-chemnitz.de
Hardware				
Modell	SPARCstation 20 Modell 61	HP 9000 S. 700 Modell 735/125	RW 420-XS (IRIS Indigo)	DEC 3000-300 AXP
Hersteller	Sun Microsystems	Hewlett-Packard	Silicon Graphics	Digital Equipment
CPU	SuperSPARC 60 MHz	PA 7150 PA-RISC_1.1 125 MHz	Mips R4000 2.2 100 MHz	Alpha 21064 125 MHz
Anzahl CPUs	2	1	1	1
FPU	Integriert	Integriert	Integriert	Integriert
Primärcache	20KByte(I) + 16KByte(D) on chip per CPU	256 KByte(I) + 256 KByte(D) on chip	8KByte(I) + 8KByte(D) on chip	8KByte(I) + 8KByte(D) on chip
Sekundär-cache	1 MByte (D+I) off chip per CPU	ohne	1 MByte (D+I) off chip	256 KByte (D+I) off chip
Hauptspeicher	192 MByte	208 MByte	48 MByte	32 MByte
Virtueller Sp.	424 MByte	350 MByte	166 MByte	? MByte
Harddisks	2 x 2GByte	1 GByte + 400 MByte	2 GByte + 540 MByte	1,9 GByte
Netzwerk-SS	FDDI 134.109.2.6 (samson) Ethernet 134.109.132.6 (samson-e)	FDDI 134.109.2.3 (tantalus-f) Ethernet 134.109.132.3 (tantalus-e)	Ethernet 134.109.200.34 (silly)	Ethernet 134.109.200.68 (decency)
Software				
Betriebssystem	SunOS 5.3 (Solaris 2.3)	HP-UX 9.01	IRIX 5.2 IP 20	Digital UNIX Release 214
Compiler	Sun Fortran Sun C 3.0.1	HP-UX Fort. 9.01 HP-UX C 9.01	SGI FortranComp. SGI C Compiler	DEC Fortran 77 DEC C Compiler
Filesystem	NFS	NFS	NFS	NFS

Tabelle 4-3 Konfigurationsbeschreibung für die Testsysteme

4.3 Prozessorzeiten als Referenz für die Auswertung

Die folgende Tabelle enthält eine Aufstellung der Prozessorzeiten, die für die Abarbeitung der Benchmarks auf den Testsystemen im Normalfall benötigt werden, auch diese sind in den einzelnen Messungen unterschiedlich starken Schwankungen unterworfen. In [Mund95] wurde ermittelt, daß die Betrachtung der zugeteilten Prozessorzeit für den Benchmarkprozeß (USER-Mode) und die von ihm ausgelösten Systemaktivitäten (SYS-Mode) geeignet ist, einen Näherungswert für ein SPECratio für den jeweiligen Benchmark zu liefern, falls es nicht möglich ist, eine Leistungsmessung unter den SPEC-üblichen Meßbedingungen durchzuführen. Die Tabelle enthält neben den gemessenen CPU-Zeiten auch die daraus berechneten Näherungen für die SPECratio, sowie beim Rechner Tantalus die Vergleichswerte für eine SPEC-Veröffentlichung eines derartigen Systems in [SPEC95_3], wobei dort in der Softwarekonfiguration als Betriebssystem die neuere Generation HP-UX 10.00 benutzt wurde.

		Mach_Ref_Time als Summe d. Prozessorzeit in USER- und SYS-Mode Approximiertes SPECratio SPECratio aus offizieller SPEC-Veröffentlichung			
Modell	SPEC-Reference	SPARCstation 20 Modell 61	HP 9000 S. 700 Modell 735/125	RW 420-XS (IRIS Indigo)	DEC 3000-300 AXP
CINT95 Baseline-Optimierung		-fast -xO4 124: -DSYSV 126: -Dalloca=__builtin_alloca	EXTRA_CFLAGS = -Aa -D_HPUX_SOURCE EXTRA_LDFLAGS= -Wl,-aarchive OPTIMIZE = -O	OPTIMIZE = -O -mips2 -non_shared -Olimit 2000 -jmpopt 126: OPTIMIZE = -cckr -O2 -DSGI -lm src.alt/alloca.c	OPTIMIZE = -migrate -O5 -std EXTRA_LDFLAGS= -non_shared -om 124: EXTRA_CFLAGS= -DLEHOST
124.m88ksim	1900	1230	780	1810	1700
		1.54	2.44	1.05	1.12
		-	3.41	-	-
126.gcc	1700	900	570	1200	1600
		1.89	2.98	1.42	1.06
		-	4.62	-	-
CFP95 Baseline-Optimierung		-fast -xO4	-O	OPTIMIZE = -O -mips2 -non_shared -Olimit 2000 -jmpopt	-fast -O5 -non_shared -om
101.tomcatv	3700	1410	750	2400	1550
		2.62	4.93	1.54	2.39
		-	5.05	-	-
102.swim	8600	2450	880	2510	2650
		3.51	9.77	3.43	3.25
		-	10.6	-	-

Tabelle 4-4 Gemessene Prozessorzeiten, approximierte SPECratio sowie Vergleichswerte offizieller SPEC-Messungen

Die angegebenen Prozessorzeiten werden im Kapitel 5 für die Normierung der Ergebnisse der Testläufe unter Lastbedingungen für die verschiedenen Testsysteme benutzt und dort als maschinenspezifische Referenzzeiten (Mach_Ref_Time) bezeichnet. Anlage A dokumentiert die an der SPEC-Ablaufsteuerung erforderlichen Änderungen, damit die spezifischen Prozessorzeiten für die Benchmarkläufe gemessen werden können.

4.4 Untersuchungsraum für die praktischen Messungen

Nach der Auswahl der vier Testsysteme an der TU Chemnitz, der vier Benchmarkvertreter, sowie der Definition geeigneter Lastmaße, die die Systemlast während einer Messung beschreiben sollen, steht der Untersuchungsraum für die praktischen Messungen fest, die mit den SPEC-CPU-Suiten und dem vorgestellten Lastmonitor durchgeführt werden. Dieser Untersuchungsraum ist in der Abbildung zusammengefaßt.

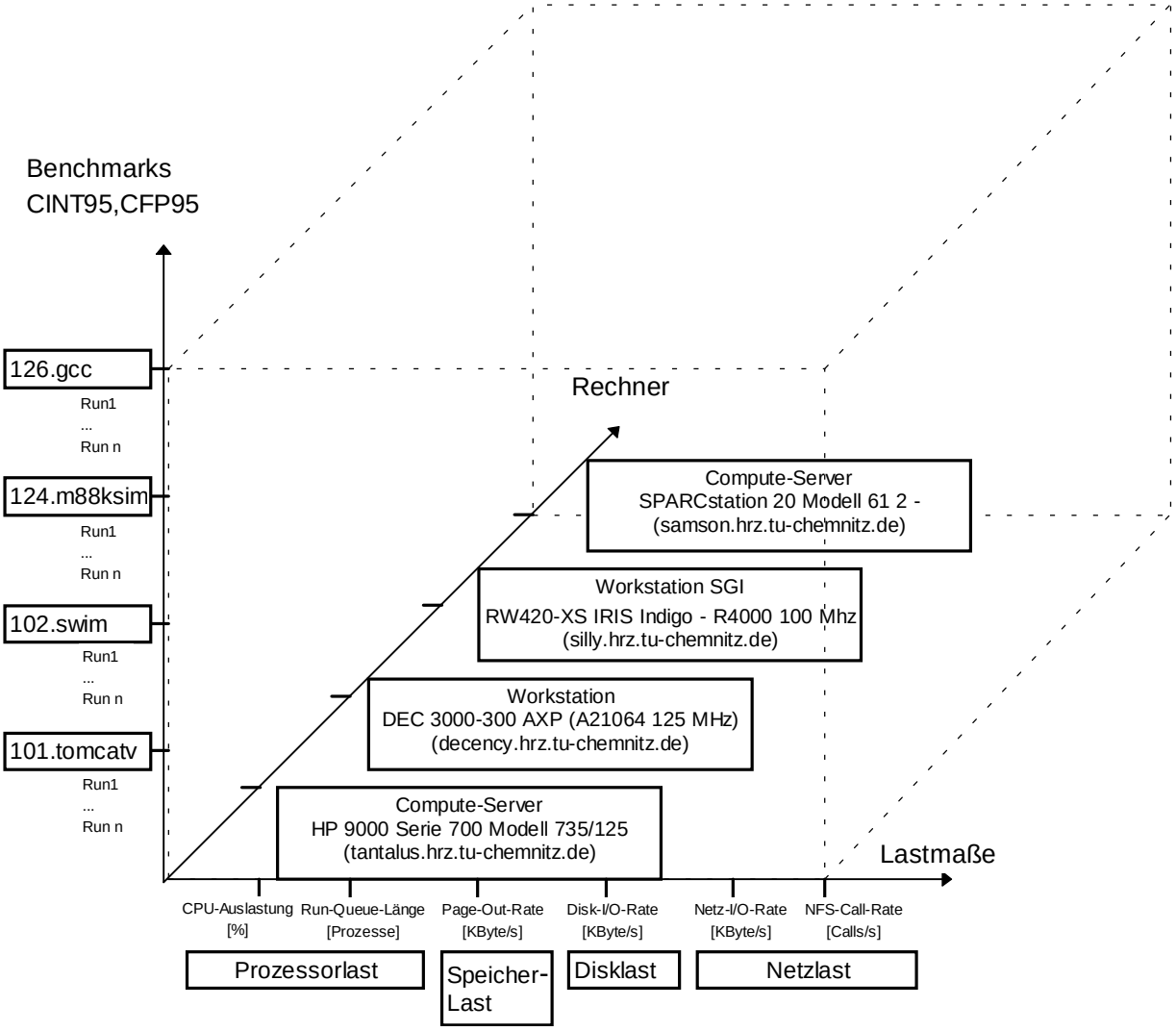


Abbildung 4-1 Untersuchungsraum für praktische Leistungs- und Lastmessungen

Für jeden Testrechner wird für die einzelnen Benchmarks ein Umfang von n=30...50 Testläufen angesetzt, um ein möglichst breites und typisches Lastspektrum zu erfassen. Insgesamt ergibt sich daraus ein Aufwand von ca. 800 Benchmarkläufen, für die jeweils die Benchmarklaufzeit und die Meßreihen zu den 6 Lastmaßen protokolliert und ausgewertet werden müssen.

5 Bewertung der Meßreihen zur Last-Leistungs-Messung

5.1 Vorgehen bei der Auswertung

Die Auswertung der Messungen soll unter zwei den Anwendungszielen der Benchmarks angepaßten Aspekten erfolgen.
Um die Wertigkeit der Leistungsaussagen mit SPEC-Benchmarks zu bestimmen, sollen zum einen die Einflußgrößen ermittelt werden, die bei Messungen unter den beschriebenen Bedingungen an der Universität eine signifikante Änderung der Benchmarkresultate hervorrufen. Daneben sollen die exemplarisch ausgewählten Testrechner hinsichtlich Ihres Verhaltens unter differenzierten Lastbedingungen bewertet werden, was den ursprünglichen Anwendungszielen des URZ für die SPEC-Benchmarks zu Erweiterungs- und Tuningzwecken entspricht.
In den vorausgegangen Kapiteln wurden 4 Elemente eines Rechnersystems mit insgesamt 6 verschiedenen Maßen zur repräsentativen Beschreibung der anliegenden Last herausgearbeitet, die eine maßgebliche Beeinflussung des Laufzeitverhaltens der Benchmarks hervorrufen können. Ziel ist die Beschreibung des Verhaltens des Gesamtsystems als Funktion dieser Lasten.
Das einzelne Benchmark-Programm stellt eine Testapplikation für eine bestimmte Anwendungsklasse dar. Die gesamte Systemlast setzt sich aus einer definierten, vom Benchmark selbst erzeugten Last und der Last aus anderen auf dem Rechnersystem laufenden Applikationen zusammen. Der verwendete Lastmonitor erfaßt die Gesamtlast im System, ohne eine Trennung in Benchmark- und Umfeldlast vorzunehmen. Eine Lasterfassung ohne jegliche Applikationen über einen definierten Zeitraum hinweg würde jene Last liefern, die vom Erfassungssystem selbst erzeugt wird (Interferenz). Eine Messung mit SPEC-ähnlichen Bedingungen, also dem Benchmark als einzige aktive Anwendung, würde es ermöglichen, die von diesem generierte Systembelastung zu bestimmen. Auf derartige Untersuchungen wurde in dieser Arbeit verzichtet, da eine Isolierung nicht vertretbare Eingriffe in den Rechenbetrieb des URZ erfordern würde.
In der Auswertung der Meßreihen wird ermittelt, wie stark der Benchmark-Prozeß als Beispielapplikation durch die anliegende Systemlast gebremst wird, die Abarbeitung sich also um einen lastbedingten Faktor x verzögert.
Kompromisse werden dabei dahingehend eingegangen, daß für die einzelnen Lastmaße die mittleren Lasten über die gesamte Abarbeitungszeit des Benchmarks hinweg betrachtet werden, obwohl die Lasten tatsächlich mehr oder weniger starken Schwankungen unterworfen sind. Durch die Wahl eines großen Stichprobenumfangs bei der Anzahl der Messungen kann dies aber in Kauf genommen werden, da bei einer hohen Anzahl von Messungen davon auszugehen ist, daß insgesamt die typische Lastverteilung erfaßt wird.
Die Beschreibung des funktionalen Zusammenhangs zwischen Lastfaktoren und Benchmarkverzögerung gestaltet sich dadurch schwierig, daß die Lasten auf den einzelnen Systemteilen teilweise korreliert sind. Außerdem führt der unterschiedlich starke Einfluß einzelner Teillasten dazu, daß einige Abhängigkeiten durch besonders starke Faktoren verdeckt werden können. Eine direkte Beschreibung der

Abhängigkeiten durch eine Funktion von bis zu 6 Lastmaßen stellt sich damit als nahezu unmöglich dar. Darum wird in dieser Arbeit eine graphische Darstellung des funktionalen Zusammenhangs zwischen Lastmaßen und Abarbeitungszeit gewählt, die mit Hintergrundinformationen zum betrachteten Rechnersystem und zum Einzelbenchmark eine Interpretation der Abhängigkeiten erlaubt.

5.1.1 Ansatz bei der Bewertung der Meßergebnisse

Es wird davon ausgegangen, daß ein Benchmark BM auf einem Rechner X eine Abarbeitungszeit von t Sekunden benötigt, wenn diese Applikation die einzige aktive auf dem entsprechenden System ist. Diese maschinenspezifische Referenzzeit (Mach_Ref_Time) für einen Benchmark BM auf X ist die Zeit, die bei einer SPEC-Herstellermessung ermittelt wird und dem entsprechenden Datenblatt des Herstellers oder einem SPEC-Newsletter entnommen werden kann. Wenn keine adäquate SPEC-Veröffentlichung existiert und keine solche isolierte Messung durchführbar ist, dann ist es statt dessen möglich, die einem Benchmark während eines Testlaufes tatsächlich zugeteilte Prozessorzeit (für Nutzer- und ausgelöste Systemprozesse) zu messen und diesen Wert als maschinenspezifische Referenzzeit zu betrachten [Mund95]. Dieser Weg wurde in dieser Arbeit gegangen, da für die verwendeten Testsysteme keine (vergleichbaren) SPEC-Messungen vorliegen. Die gemessenen Prozessorzeiten für die verwendeten Benchmarks auf den einzelnen Systemen finden sich in Tabelle 4-4 im Abschnitt 4.3 dieser Arbeit.
Vernachlässigt man die Einflüsse der Peripherie eines Rechnersystems, dann wird die Belastung allein durch die Anzahl der aktiven Anwendungen bzw. die Anzahl der bereiten, auf die Bearbeitung wartenden Prozesse beschrieben. Diese Last wird im Betriebssystem durch die Länge der Prozeßwarteschlange (Run-Queue) repräsentiert. Nimmt man an, daß alle Prozesse jeweils die CPU voll auslasten und die gleiche Laufzeit aufweisen, dann bestimmt sich die theoretische Laufzeit (Erwartungswert) für den Benchmark als Produkt von maschinenspezifischer Referenzzeit (Mach_Ref_Time) und der mittleren Länge der Prozeßwarteschlange (Run-Queue-Länge). Der Aufwand für die Koordinierung mehrerer Anwendungen im Betriebssystem wird bei diesem Erwartungswert wie alle anderen äußeren Lasteinflüsse vernachlässigt.
Bei der Betrachtung der Meßergebnisse soll nun ermittelt werden, wie stark die tatsächliche Abarbeitungszeit der gewählten Beispielapplikationen (Benchmarks) bei n Messungen unter zufälligen Lastbedingungen vom jeweiligen Erwartungswert abweicht. Von Interesse ist dabei speziell, wie stark die Abweichung ist und welchem mathematischen Zusammenhang die Laufzeit und damit die Abweichung folgt.
Die einzelnen abweichenden Laufzeitergebnisse werden in Verbindung mit den Werten für die einzelnen Lastmaße für die zu bewertenden Teilsysteme betrachtet, um die Abweichung vom Erwartungswert zu begründen und die Abhängigkeiten von den Lasten bezüglich einzelner Systemkomponenten zu bestimmen.

5.1.2 Darstellung der Meßergebnisse

Zur Darstellung der Meßergebnisse ist für eine Vergleichbarkeit des Lastverhaltens der unterschiedlich leistungsfähigen Testsysteme eine normierte Darstellung erforderlich. Für jeden Benchmark werden für die einzelnen Rechner 2 normierte Diagramme zur Ergebnisdarstellung verwendet.

Um unterschiedlich schnelle Rechner hinsichtlich des Lastverhaltens vergleichbar zu machen, wird dabei nicht die absolute Laufzeit des Benchmarks betrachtet, sondern der Quotient aus tatsächlicher Abarbeitungszeit und der maschinenspezifischen Referenzzeit (Mach_Ref_Time) für den Benchmark. Dieser Wert wird nachfolgend als ***Verzögerungsfaktor*** bezeichnet und ist für unterschiedliche Rechnerklassen vergleichbar.

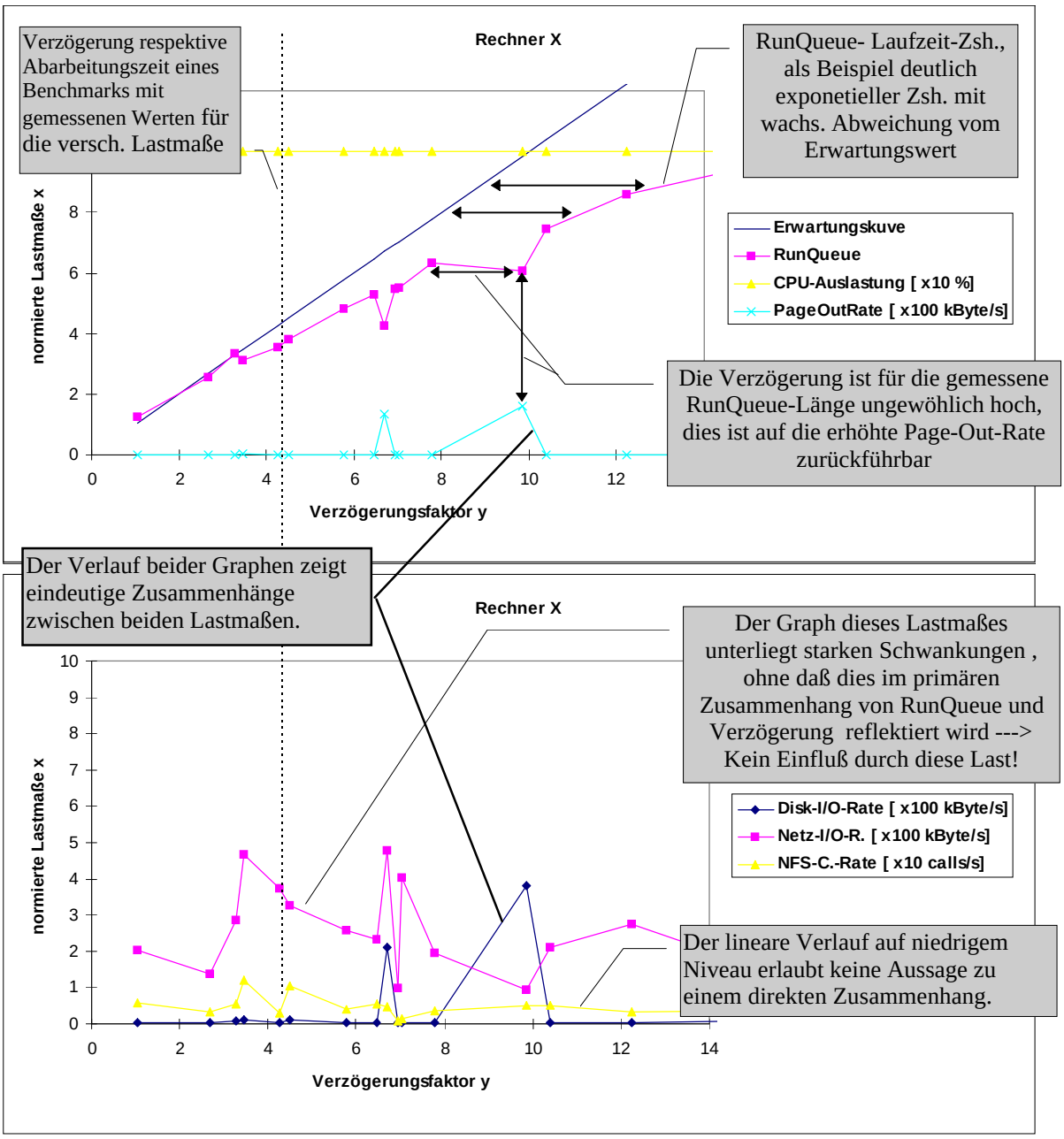


Abbildung 5-1 Auswertungsschema für Meßreihe mit Benchmark BM auf Rechner X

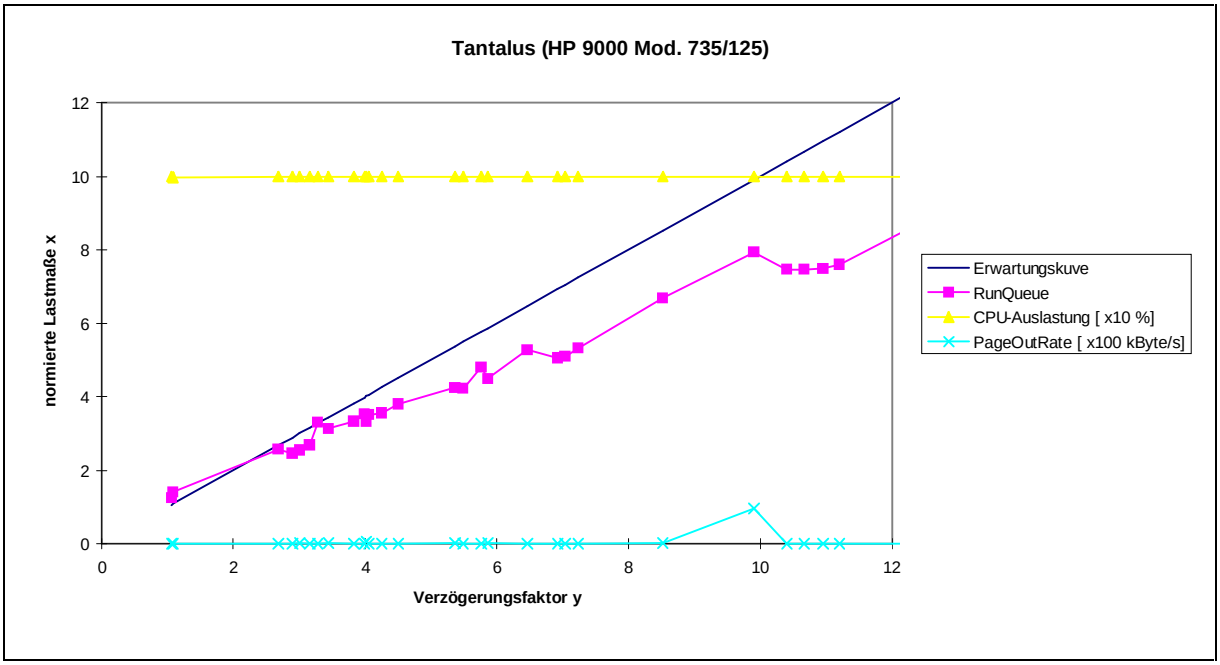
Dieser Verzögerungsfaktor wird an der waagerechten Achse der Diagramme dargestellt. An den senkrechten Achsen der Diagramme werden die Werte für die einzelnen Lastmaße, die zum Testlauf mit dem entsprechenden Verzögerungsfaktor gehören, ***normiert*** dargestellt. Der tatsächliche Lastwert bestimmt sich dabei als Produkt aus dem normierten Wert an der Diagrammachse und dem in der Legende angegebenen Multiplikator mit der entsprechenden Einheit für das Lastmaß.

Die Normierung der Werte für die einzelnen Lastmaße erfolgte auf Basis der höchsten gemessenen absoluten Werte für das jeweilige Lastmaß über alle Benchmarks auf allen Testsystemen.
Diagramm 1 enthält dabei neben den Lastmaßen „Run-Queue-Länge“, „CPU-Auslastung“ und „Page-Out-Rate“ eine Kurve, die den Erwartungswert für den primären Zusammenhang von Laufzeit und Warteschlangenlänge darstellt. Diagramm 2 enthält die übrigen Lastmaße „Disk-I/O-Rate“, „Netz-I/O-Rate“ und „NFS-Call-Rate“. Auf einer Senkrechten durch beide Diagramme finden sich damit immer die Werte für die einzelnen Lastmaße für einen Testlauf mit der Laufzeit t beziehungsweise dem aus der Laufzeit resultierenden Verzögerungsfaktor y. Die Abbildung 5-1 illustriert das Schema für die Auswertung und Interpretation einer fiktiven Meßreihe mit möglichen Zusammenhängen.
Bei dieser Auswertungsmethode wird klar, daß jeweils nur die Haupteinflußgröße bestimmt werden kann, indem Beziehungen zwischen den Graphen bestimmt werden. Für eine Anwendung zu Tuningzwecken ist dies jedoch ein geeigneter Weg zur Bestimmung von Schwachstellen und leistungsverbessernden Erweiterungen.
Für eine bessere Überschaubarkeit der Diagramme werden für die Darstellung nicht alle Meßergebnisse verwendet. Für jeden Testrechner gibt es charakteristische Lastbereiche mit mittlerer Prozeßlast, in denen eine starke Häufung von ähnlichen Meßergebnissen auftritt. Hier werden wenige exemplarische Resultate in die Darstellung aufgenommen.
Im nächsten Abschnitt 5.2 werden die Ergebnisse der Meßreihen nach dem eben beschriebenen Schema dargestellt, dies erfolgt jeweils in der Reihenfolge 101.tomcatv, 102.swim, 124.m88ksim, 126.gcc
In den nachfolgenden Abschnitten erfolgt eine Interpretation der gemessenen Ergebnisse aus verschiedenen Betrachtungsrichtungen, die aus den am Anfang der Arbeit genannten Zielen resultieren. Abschnitt 5.3 betrachtet dabei die Einflüsse auf die einzelnen ausgewählten SPEC-Benchmarks. Im Abschnitt 5.4 erfolgt die Betrachtung der generellen Einflüsse der Teilsysteme eines Rechnersystems, die im Abschnitt 2.1 (unter Beachtung der im URZ gegebenen Bedingungen) als leistungsbeeinflussende Komponenten herausgearbeitet wurden. Abschließend wird im Abschnitt 5.5 das konkrete Lastverhalten der betrachteten Testsysteme unter Einbeziehung von Konfigurations- und Architekturmerkmalen diskutiert.

5.2 Ergebnisse der Meßreihen

5.2.1 Compute-Server HP 9000 Serie 700 Modell 735/125 (tantalus.hrz)

- *Benchmark 101.tomcatv*



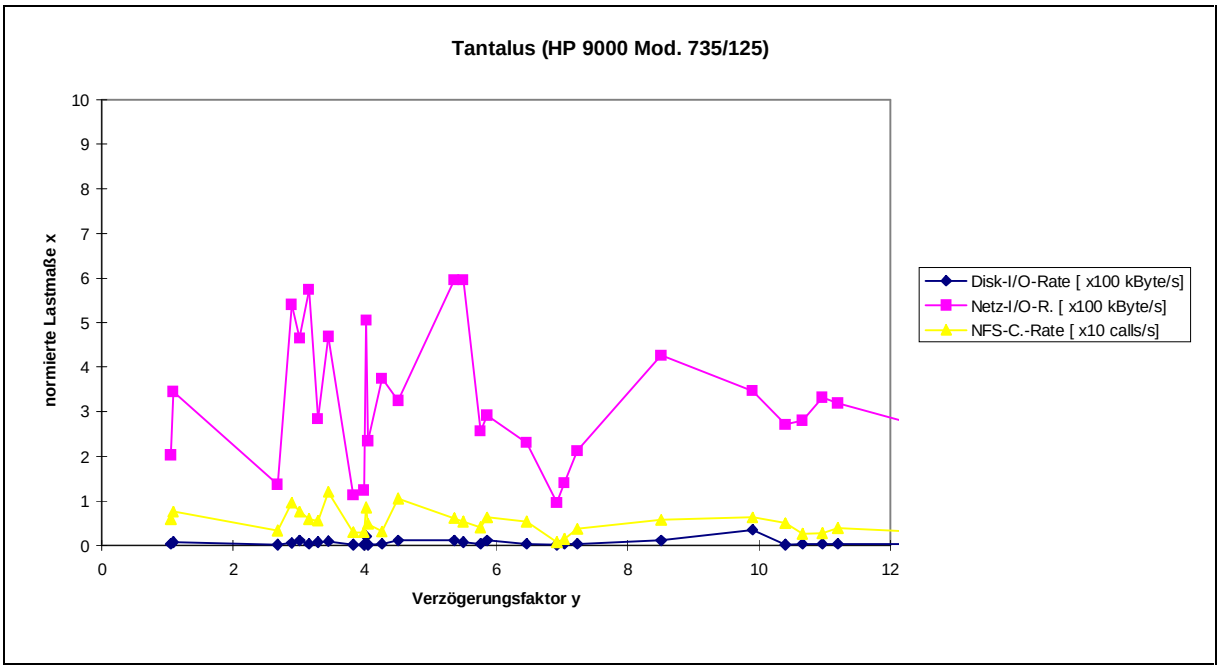


Diagramm 5-1 Benchmark 101.tomcatv (CFP95)

Es ist ein deutlicher Zusammenhang zwischen Benchmarkverzögerung und der Run-Queue-Länge zu erkennen. Allerdings wächst mit steigender Prozeßlast die Verzögerung schneller, als dies bei den gemessenen Warteschlangenlängen zu erwarten ist. Zwischen Prozeßlast und Laufzeitverlängerung läßt sich sogar ein exponentieller Zusammenhang vermuten. Die Prozessorauslastung liegt bei allen Messungen nahe 100%, eine Verzögerung durch Peripherieteile liegt folglich nicht vor.
Speicherauslagerungen finden in geringen Raten von 0 bis 10 KByte/s statt, so daß eine Beeinflussung hier nicht feststellbar ist. Auch bei einer einzigen Messung mit ca. 100 KByte/s Auslagerungsrate ist kein Zusammenhang zur Benchmarkverzögerung sichtbar.
Korrespondierend zur Page-Out-Rate treten nur niedrige Transfermengen zur lokalen Harddisk auf, aus denen sich ebenfalls kein Einfluß ableiten läßt.
Die gemessenen Netz-I/O-Raten streuen stark zwischen 100 und 600 KByte/s, ohne daß sich beim Vergleich einzelner Spitzenwerte eine globale Verbindung zur Abweichung der Benchmarkverzögerung vom Erwartungswert bei der entsprechenden Prozeßlast erkennen läßt. Möglich ist aber, daß ein Teil der Abweichung vom Erwartungswert auf das vergleichsweise hohe Niveau der Netzlast auf diesem Rechner zurückzuführen ist. Die NFS-Call-Rate zeigt eine mit der NETZ-I/O-Rate korrespondierende Werteverteilung, auch hier ist kein Einfluß der stark streuenden Call-Raten auf den Zusammenhang von Prozeßlast und Laufzeit erkennbar.

- *Benchmark 102.swim*

Bei allen Testläufen mit 102.swim auf diesem Rechner wurde der Prozessor vollständig ausgelastet. Es läßt sich auch hier ein exponentieller Zusammenhang zwischen der Länge der Prozeßwarteschlange und der Verzögerung bei der Abarbeitung des Benchmarks erkennen. Aus den geringen Speicherauslagerungsraten und Disktransferraten lassen sich keine Beeinflussungen der Benchmarklaufzeiten ableiten.
Die Netz-I/O-Raten streuen stark in einem Wertebereich von 100 bis 700 KByte/s. Die bereits beim Benchmark 101.tomcatv gemachte Vermutung, daß sich ein Teil der Abweichung der Benchmarkverzögerung vom Erwartungswert auf die hohe Grundlast von ca. 300 KByte/s auf dem Netz und die damit verbundenen Wartezeiten für Ein- und Ausgaben zurückführen läßt, findet hier Bestätigung, da bei den wenigen Messungen mit geringen Netzraten von ca. 100 KByte/s der Verzögerungsfaktor sich gegenüber den anderen Messungen geringfügig verringert. Ein Einfluß einzelner hoher Netzraten oder der stark streuenden NFS-Call-Raten auf die Abarbeitungszeit des Benchmarks läßt sich daraus aber nicht ableiten.

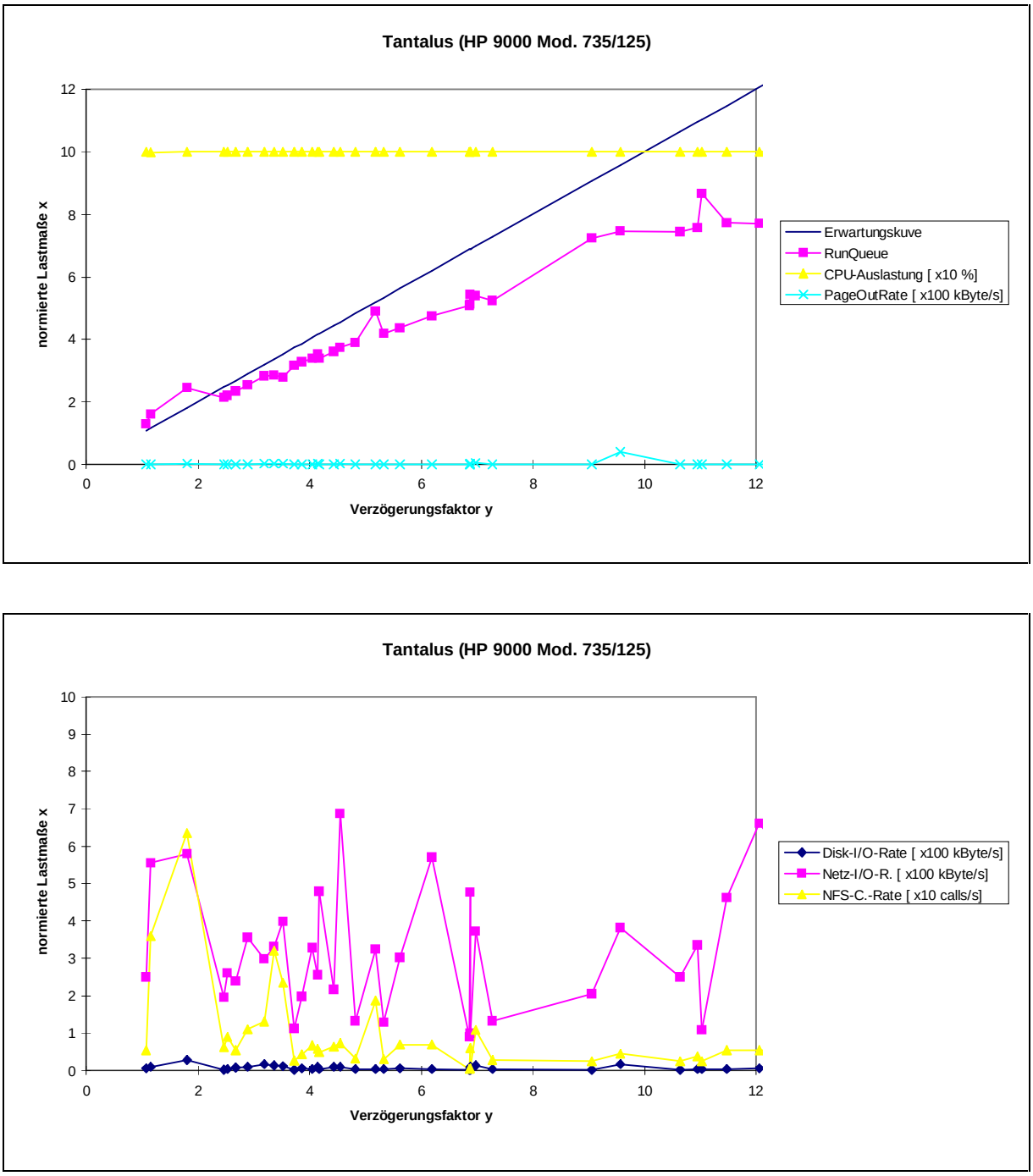
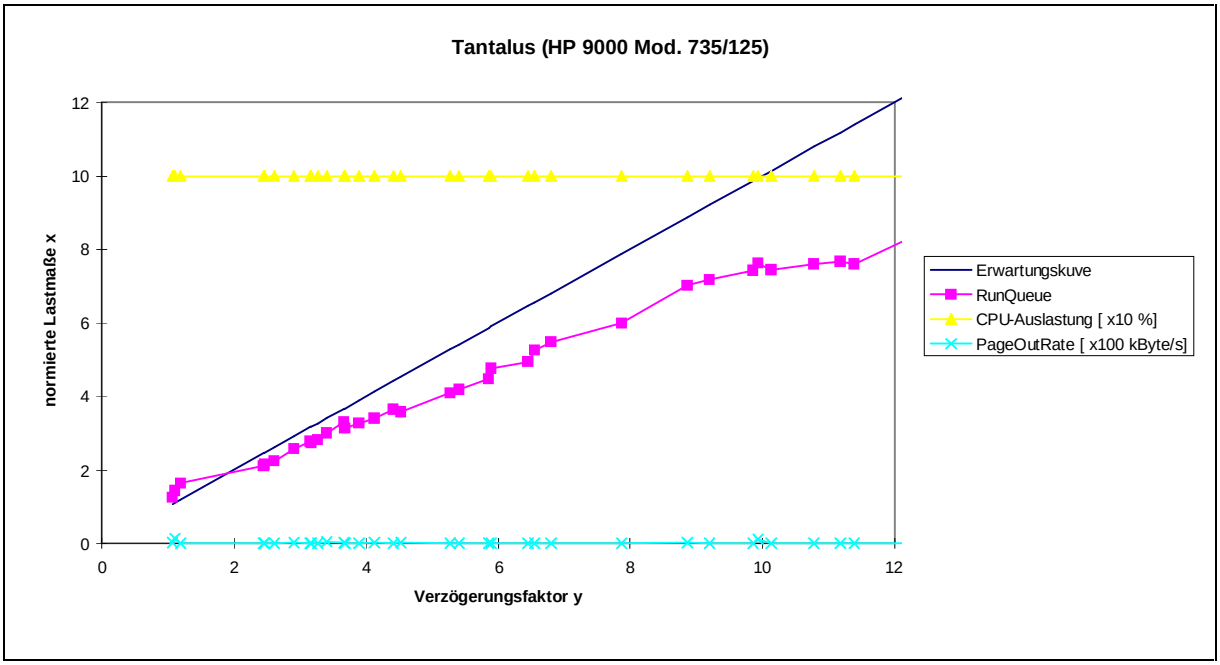


Diagramm 5-2 Benchmark 102.swim (CFP95)

- *Benchmark 124.m88ksim*

Die Testreihe mit dem Benchmark 124.m88ksim zeigt einen exponentiellen Zusammenhang zwischen der Prozeßlast, repräsentiert durch die Länge der Run-Queue, und der für die Abarbeitung des Benchmarks benötigten Zeit. Aus den geringen Speicherauslagerungsraten von 0 ... 15 KByte/s läßt sich kein Einfluß auf diesen Zusammenhang ableiten. Ebensowenig ist bei den im direkten Vergleich immerhin um Faktor 10 schwankenden Netz-I/O-Raten und NFS-Call-Raten ein Einfluß auf den Zusammenhang von Prozeßlast und Benchmarkverzögerung erkennbar. Die Netz-I/O-Rate liegt auch hier stets auf relativ hohem Niveau von durchschnittlich 200 ... 300 KByte/s.



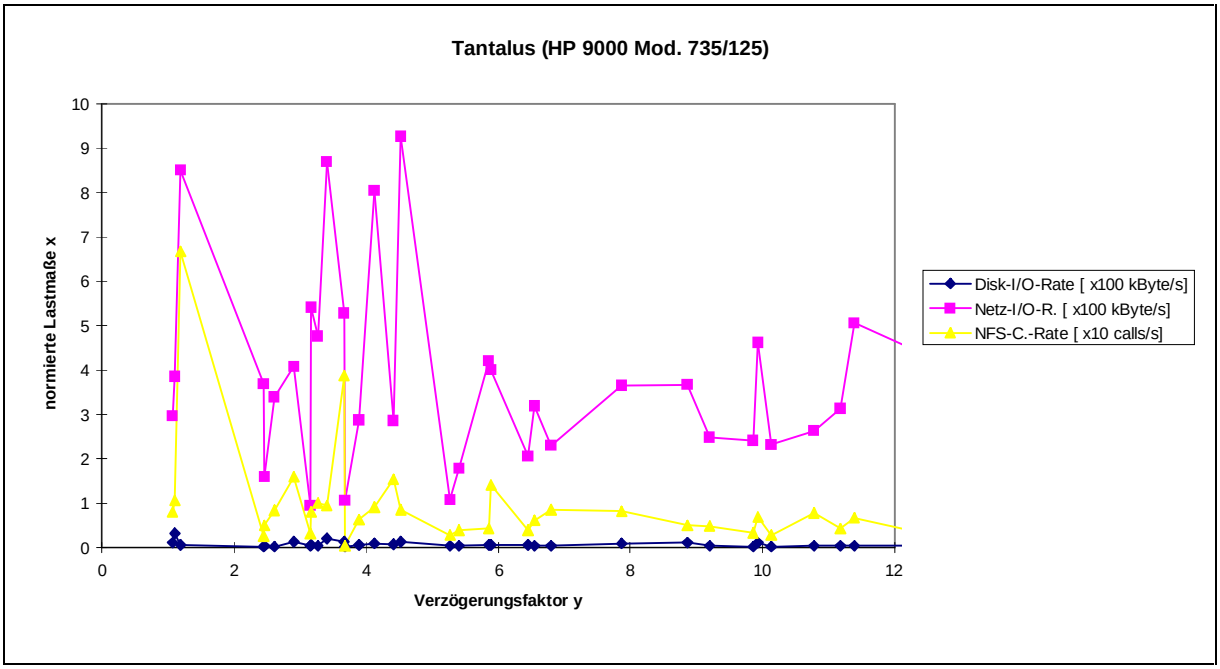
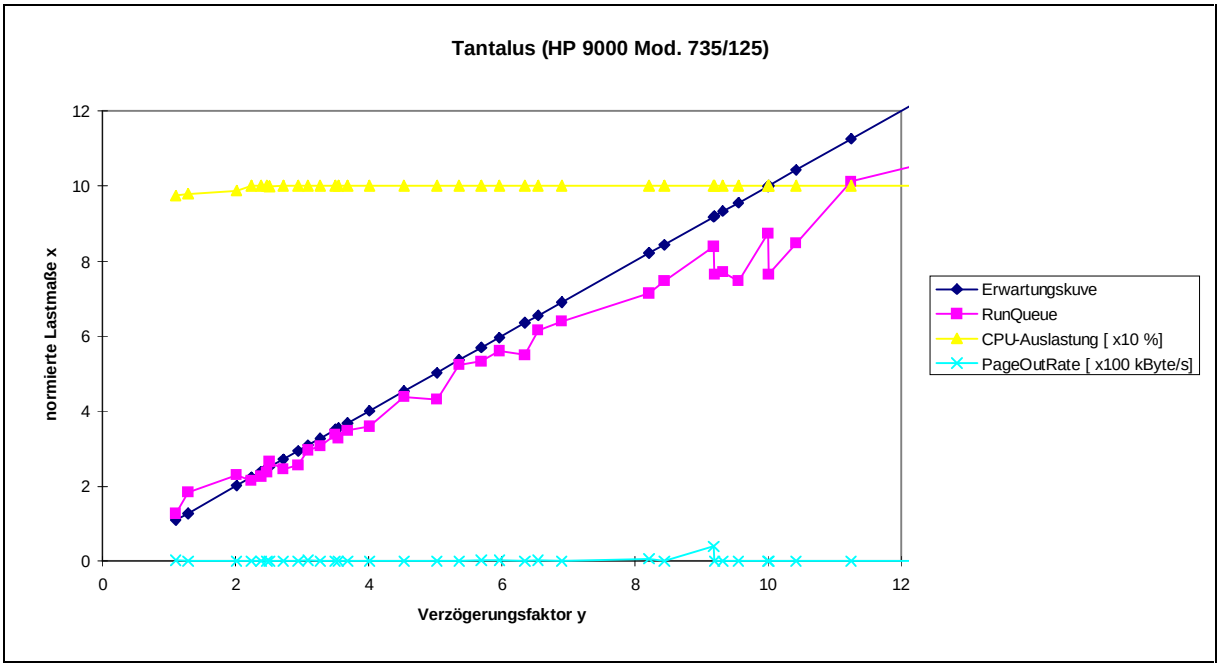


Diagramm 5-3 Benchmark 124.m88ksim (CINT95)

- *Benchmark 126.gcc*

Auch bei diesem Benchmark besteht der generelle Zusammenhang zwischen der Anzahl aktiver Prozesse und der Verzögerung bei der Benchmarkabarbeitung. Im Gegensatz zu den anderen Benchmarks ist hier jedoch ein nahezu linearer Zusammenhang zwischen der Länge der Run-Queue und der resultierenden Benchmarklaufzeit bzw. -verzögerung ablesbar, die Differenzen zwischen dem theoretischen Erwartungswert und der tatsächlichen Verzögerung fallen speziell im Bereich der höheren Prozeßlast geringer aus, als bei den vorherigen Tests. Bezüglich der verschiedenen Einflußkomponenten lassen sich

dennoch ähnliche Aussagen wie zuvor treffen. Auf Grund der geringen Speicherauslagerungs- und Disk-I/O-Raten sind Einflüsse der virtuellen Speicherverwaltung und des Disk-Subsystems nicht bestimmbar oder nicht vorhanden. Die Netzlast schwankt zwischen I/O-Raten von 120 bis 2300 KByte/s, ohne daß eine signifikante Abweichung vom linearen Zusammenhang von Run-Queue-Länge und Verzögerung feststellbar ist, gleiches gilt für die NFS-Call-Rate, die zwischen 2 und 48 Calls/s streut und im Mittel unter 10 Calls/s liegt. Damit ist ein direkter Einfluß der Netzlast auf die Verzögerung der Benchmarkabarbeitung nicht gegeben.



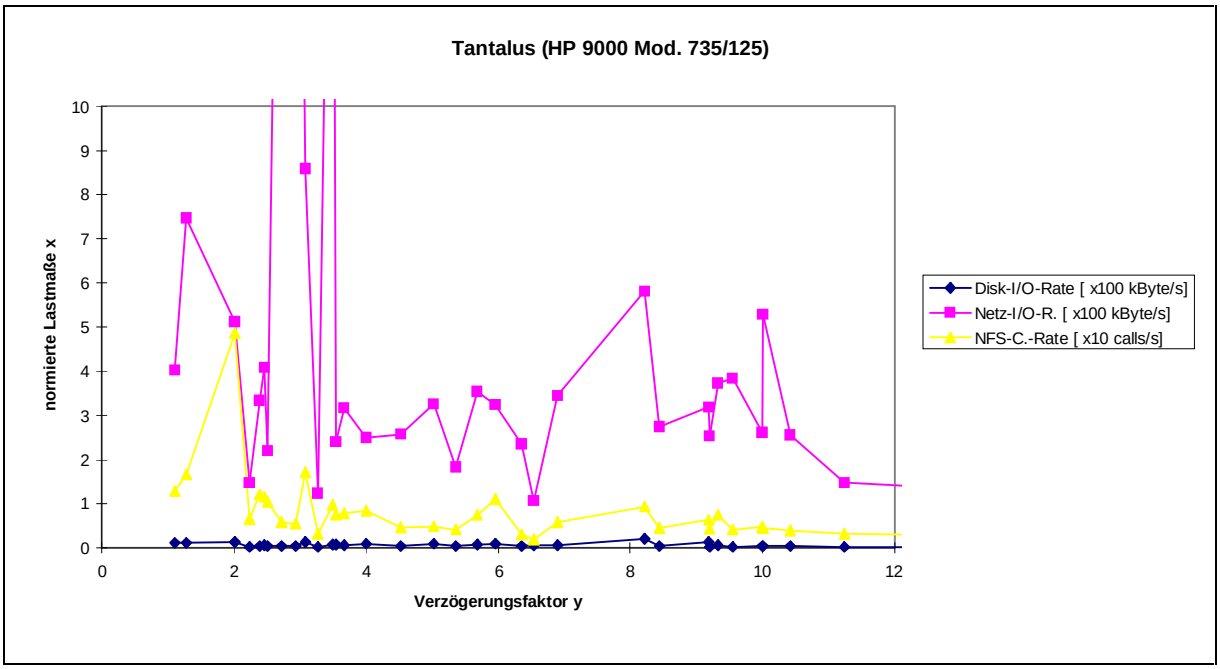
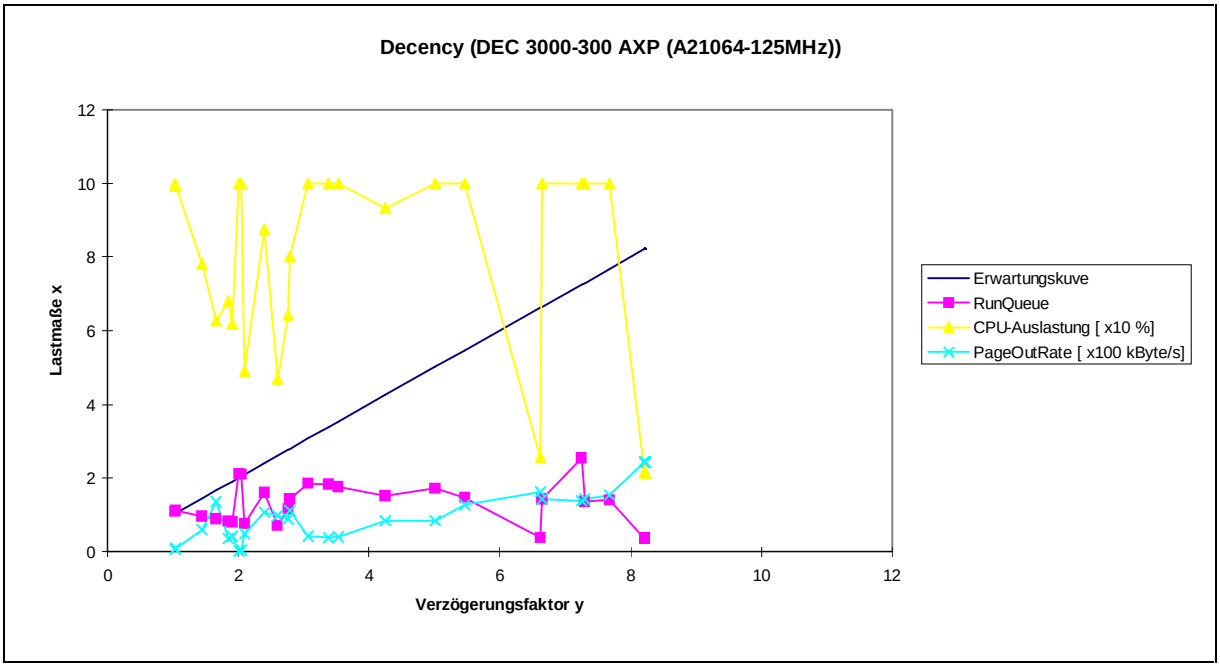


Diagramm 5-4 Benchmark 126.gcc (CINT95)

5.2.2 Workstation DEC 3000-300 AXP (decency.hrz)

- *Benchmark 101.tomcatv*



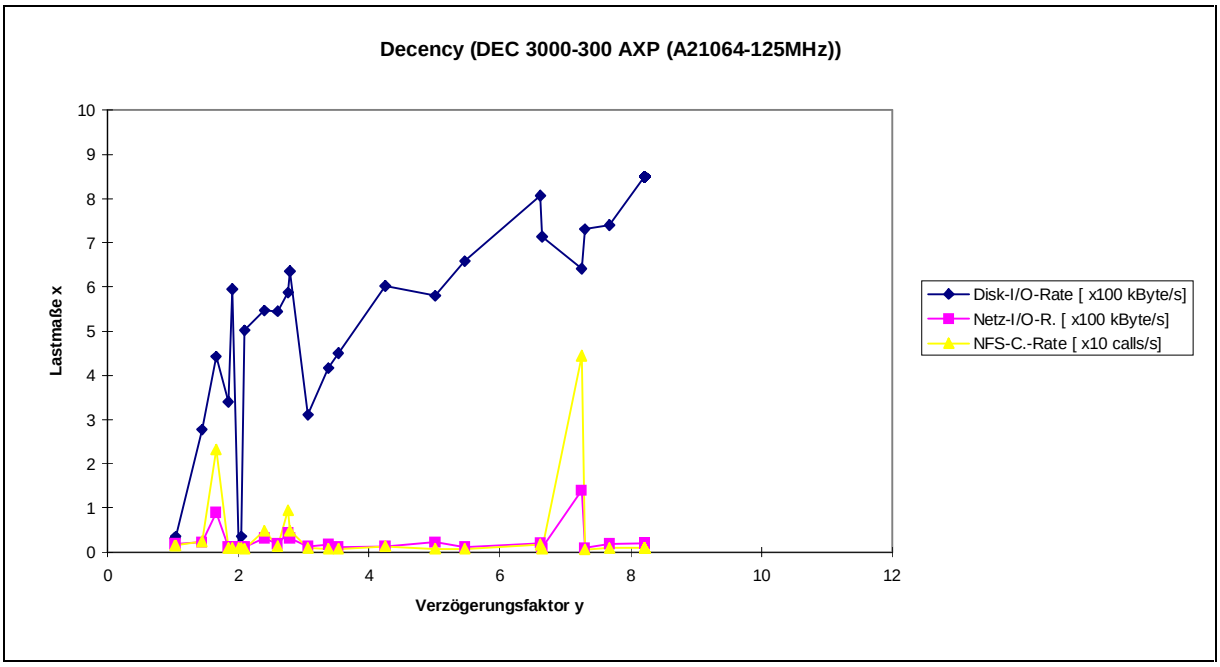


Diagramm 5-5 Benchmark 101.tomcatv (CFP95)

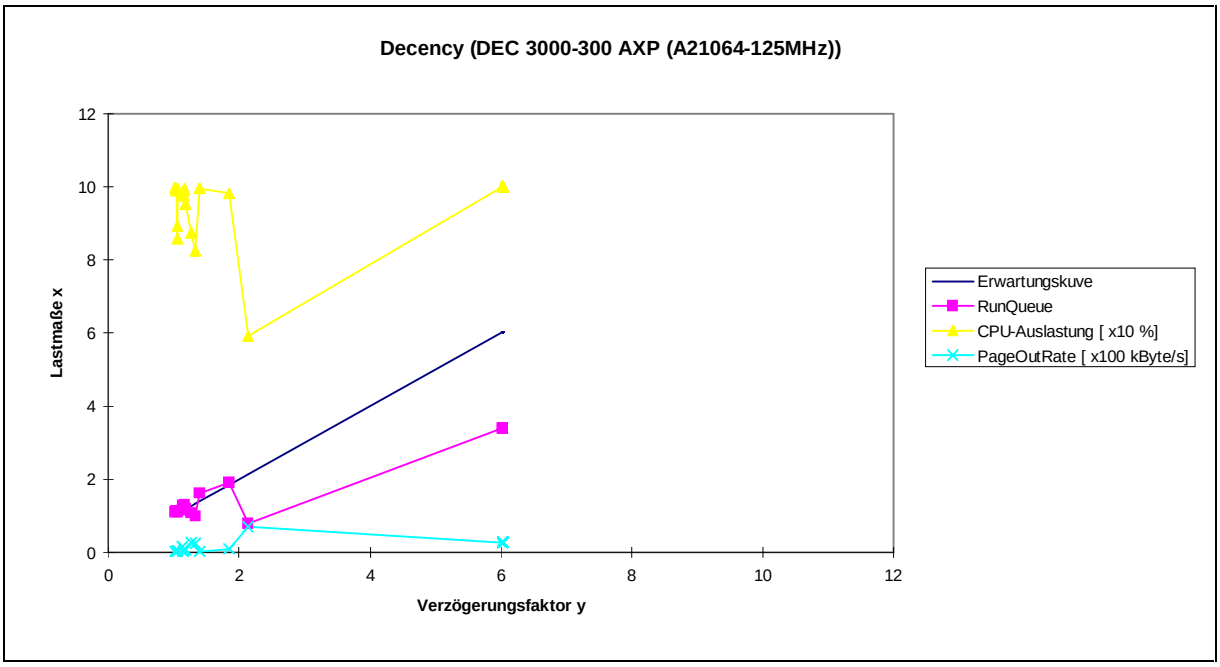
Bedingt durch die Hardwarekonfiguration für den Betrieb als „kleine“ Workstation, speziell durch die geringe Speicherausstattung mit nur 32 Megabyte, zeigt sich bei 101.tomcatv auf diesem System ein sehr unausgeglichenes Lastverhalten. Nur bei wenigen Messungen im unteren Prozeßlastbereich, das heißt bei Run-Queue-Längen zwischen 1 und 2, liegt die Benchmarkverzögerung im Bereich des

Erwartungswertes für die gemessene Warteschlangenlänge. Bei diesen Messungen wird die CPU auch vollständig ausgelastet. Sobald die Prozeßlast steigt oder selbst bei geringer Prozeßlast ein anderer Prozeß eine gewisse Menge Hauptspeicher benötigt, ist ein sprunghafter Anstieg der Speicherauslagerungen zu verzeichnen. Der Zusammenhang zwischen der Run-Queue-Länge und der Benchmarkverzögerung wird durch die Speichereinflüsse vollständig überlagert. Bedingt durch die Wartezeiten auf Speicheraus- und -einlagerungen treten hohe Idle-Anteile für die Prozessorauslastung auf, die effektive CPU-Auslastung sinkt auf Werte zwischen 90% und 20% (!). Dementsprechend liegen die gemessenen Warteschlangenlängen sogar unter Eins. Das überlastete Speichersystem sorgt damit dafür, daß der schnelle Alpha-Prozessor seine Leistungsfähigkeit (approximiertes SPECratio für 101.tomcatv bei 2.4) nicht ausspielen kann.
Man kann hierbei sogar feststellen, daß die Benchmarkverzögerung beinahe linear mit der Menge ausgelagerten Hauptspeichers steigt. Mit dem Anstieg der Page-Out-Operationen (und gleichzeitiger Page-In-Operationen) geht ein erheblicher Anstieg der Disk-I/O-Rate einher, der durch das Laden eventuell benötigter Systemprogramme von der Harddisk noch zusätzlich verstärkt wird. Die dabei erreichten durchschnittlichen Transferraten von bis zu 850 KByte/s, gemessen über Zeiträume von mehreren Stunden hinweg, stellen die lokale Harddisk als zweiten limitierenden Faktor neben dem Hauptspeicher in diesem System heraus.
Eine Beeinflussung durch die vergleichsweise geringen Netz-I/O-Raten und NFS-Call-Raten liegt hingegen nicht vor, bzw. kann sie durch die Überlagerungen durch die Aktivitäten der Speicherverwaltung nicht erkannt werden.

- *Benchmark 102.swim*

Der Benchmark 102.swim weist wie 101.tomcatv einen Speicherbedarf von ca. 14 MByte für die benötigten Daten auf, auf die jedoch offensichtlich ausgewogener zugegriffen wird, was auch schon in der Programmanalyse abgeschätzt wurde. Die erforderlichen Hauptspeicherauslagerungen liegen zwischen 2 und 70 KByte/s. Grundsätzlich folgt das Laufzeitverhalten dem erwarteten primären Zusammenhang von Run-Queue-Länge und resultierender Verzögerung der Benchmarkabarbeitung. Je nach Menge der erforderlichen Speicherauslagerungen weicht die Verzögerung vom linearen Zusammenhang zur Run-Queue ab, wobei diese Abweichung im unteren Lastbereich in etwa mit einer Erhöhung des Verzögerungsfaktors um 1 pro 100 KByte/s Auslagerungsrate beschrieben werden kann. Mit den Auslagerungen ist im Bereich geringer Prozeßlast wiederum ein Absinken der CPU-Auslastung auf 90% bis 60% verbunden.
Bei der am stärksten verzögerten Messung liegt der Verzögerungsfaktor etwa um Wert 3 über dem Run-Queue-Erwartungswert. Bei dieser Messung ist eine Netz-I/O-Rate von 255 KByte/s und eine NFS-Call-Rate von 85 Calls/s, gegenüber den sonstigen Werten von bis zu 100 KByte/s und bis zu 10 Calls/s, zu verzeichnen, wodurch die starke Abweichung begründet werden kann. Speziell die NFS-Call-Rate von 85 Calls/s stellt einen hohen Wert dar, bei dessen Bewältigung die Leistungsgrenze eines NFS-Servers durchaus erreicht werden kann, wenn dieser von mehreren Client-Systemen frequentiert wird. Die hohe Call-Rate muß von einem anderen als dem Benchmarkprozeß ausgelöst worden sein, da dieser Lastwert bei fast allen anderen Messungen, speziell bei denen ohne Zusatzbelastung, nur im Bereich von 1 bis 3 Calls/s liegt. Dennoch werden die „Benchmark-eigenen“ Calls bei der hohen NFS-Last möglicherweise so verzögert, daß daraus auch die starke Verzögerung des Benchmarks resultiert. Aus

den übrigen, relativ niedrigen Meßwerten für Netz-I/O-Raten und NFS-Call-Raten lassen sich keine Aussagen zum Einfluß auf den Benchmark ableiten.



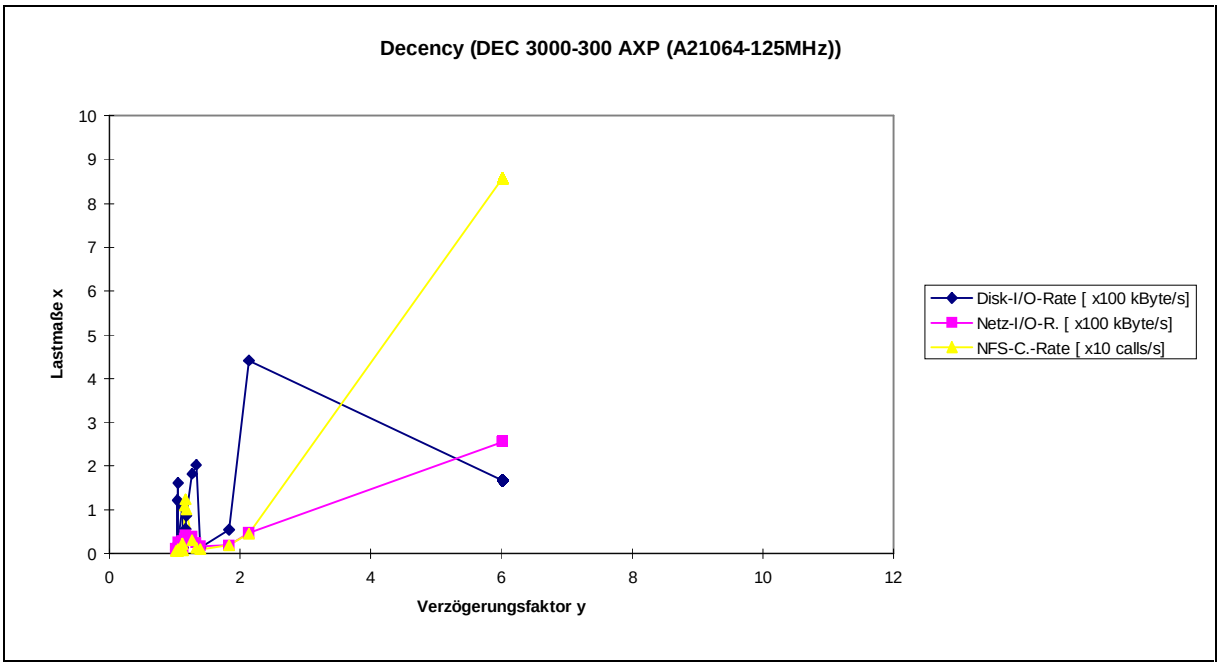
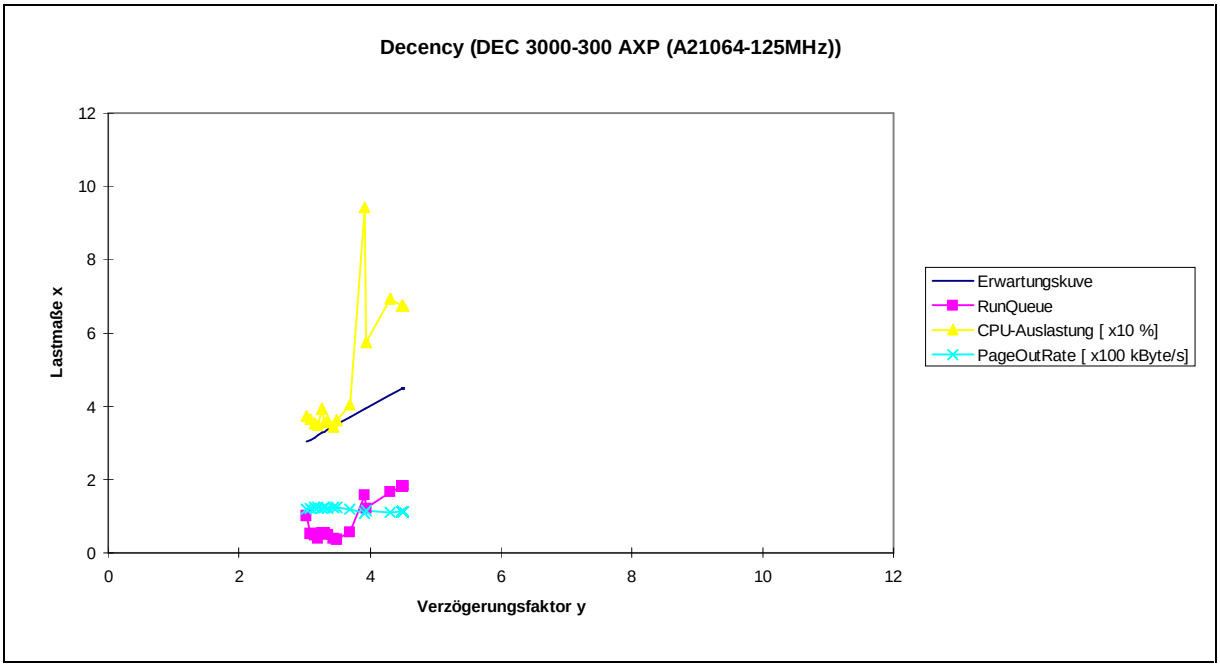


Diagramm 5-6 Benchmark 102.swim (CFP95)

- *Benchmark 124.m88ksim*



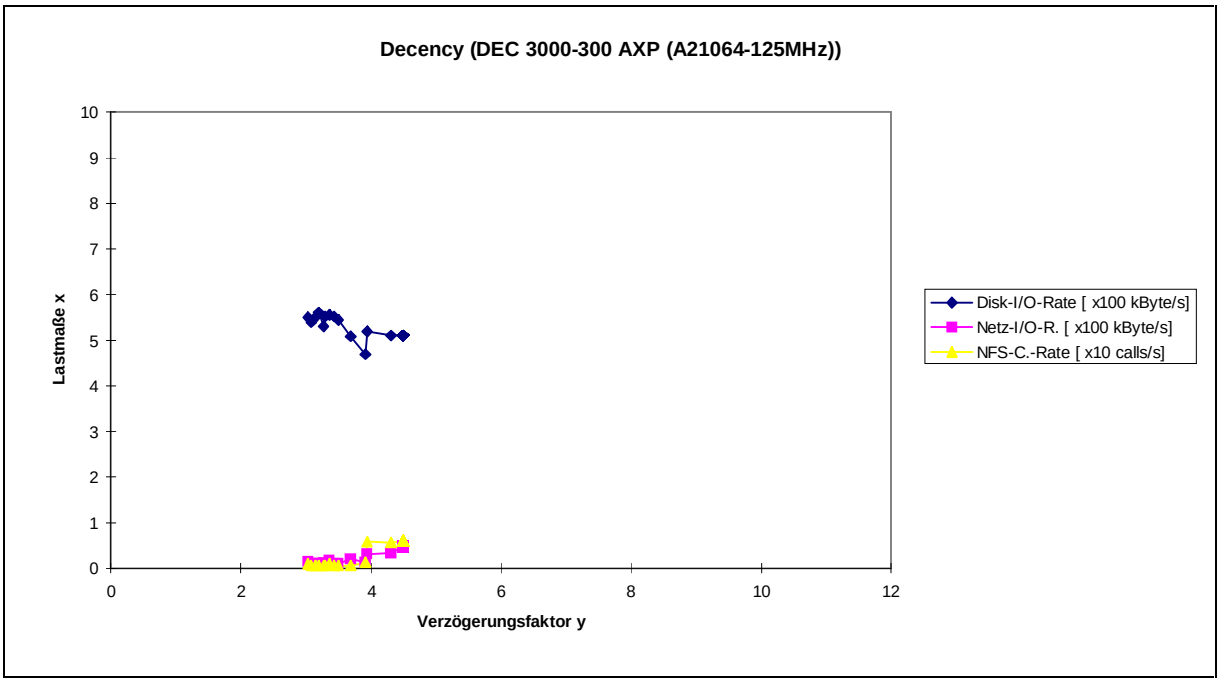


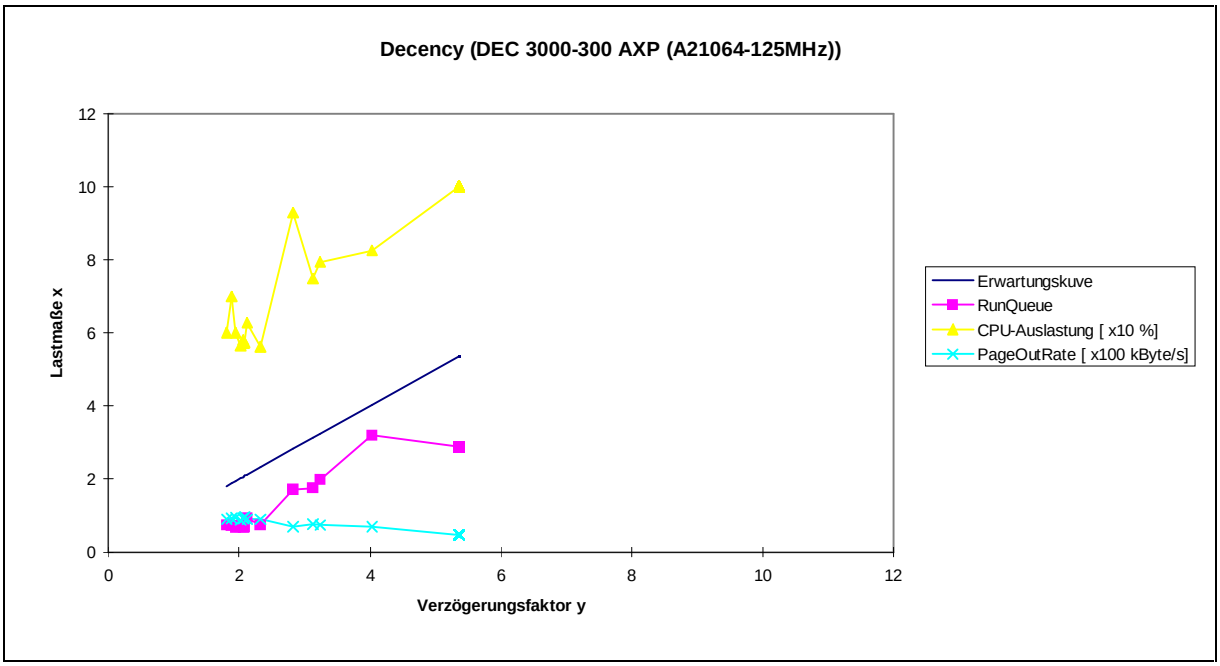
Diagramm 5-7 Benchmark 124.m88ksim (CINT95)

Beim Benchmark 124.m88ksim liegen die Ergebnisse aller Messungen in einem relativ schmalen Lastbereich. Zwischen der Länge der Prozeßwarteschlange und der Benchmarkverzögerung ist dabei ein linearer Zusammenhang ablesbarbar, allerdings liegt der Verzögerungsfaktor etwa um den Wert 3 über dem theoretischen Erwartungswert, der der mittleren Warteschlangenlänge entspricht. Diese relativ konstante Abweichung korrespondiert mit der für alle Messungen nahezu gleichen Page-Out-Rate von 110 bis 120 KByte/s. Eine zufällig immer gleiche Auslagerungsrate bei Messungen über mehrere Wochen hinweg und verschiedenen Prozeßbelastungen ist dabei äußerst unwahrscheinlich. Die Disk-

I/O-Rate ist ebenfalls relativ konstant und liegt im Bereich von ca. 550 KByte/s mit jeweils etwa 47 Transfers/s. Alle Resultate mit gering nach oben oder unten abweichenden Disk-I/O-Raten sind mit einer geringen Erhöhung bzw. Verringerung der Benchmarkverzögerung bei Betrachtung der aktuellen Warteschlangenlänge verbunden. Damit ist bei 124.m88ksim auf diesem System die lokale Harddisk als limitierende Einflußgröße bei der Benchmarkabarbeitung identifiziert.
Mit zunehmender Prozeßanzahl steigt die Prozessorauslastung von Werten um 30% bis gegen 100%, da die zusätzlichen, möglicherweise weniger speicherintensiven Prozesse die CPU auslasten können, während der Benchmarkprozeß auf Speicherein- und -auslagerungen wartet.
Aus den konstant geringen Netz-I/O- und NFS-Call-Raten läßt sich keine Beeinflussung der Benchmarkabarbeitung ableiten.

- *Benchmark 126.gcc*

Auch bei 126.gcc ist ein deutlicher linearer Zusammenhang zwischen Run-Queue-Länge und Benchmarkverzögerung sichtbar. Bei der mit steigender Last leicht fallenden Speicherauslagerungsrate von ca. 90 KByte/s im niedrigen Lastbereich liegt der Verzögerungsfaktor etwa um Wert 1 über dem aus der Warteschlangenlänge resultierenden Erwartungswert.
Die korrespondierende Disk-I/O-Rate sinkt bei steigenden Prozeßlasten vom bereits als kritisch bewerteten Wert von 550 KByte/s auf etwa den halben Wert, wobei daraus umgekehrt wieder ein Anstieg der Prozessorauslastung von anfangs nur 60% bis zur vollständigen CPU-Auslastung resultiert.
Netz-I/O-Rate und NFS-Call-Rate liegen allgemein auf geringen Niveau um 100 KByte/s bzw. um 10 Calls/s, so daß keine Aussage zum Einfluß möglich ist. Beim längsten Benchmarklauf ist allerdings wieder eine besonders hohe Call-Rate von 50 Calls/s gemessen worden, was als mögliche Ursache für die stärkere Verzögerung in diesem Einzelfall in Frage kommt, jedoch keine Aussage zu einem generellen Zusammenhang zuläßt.



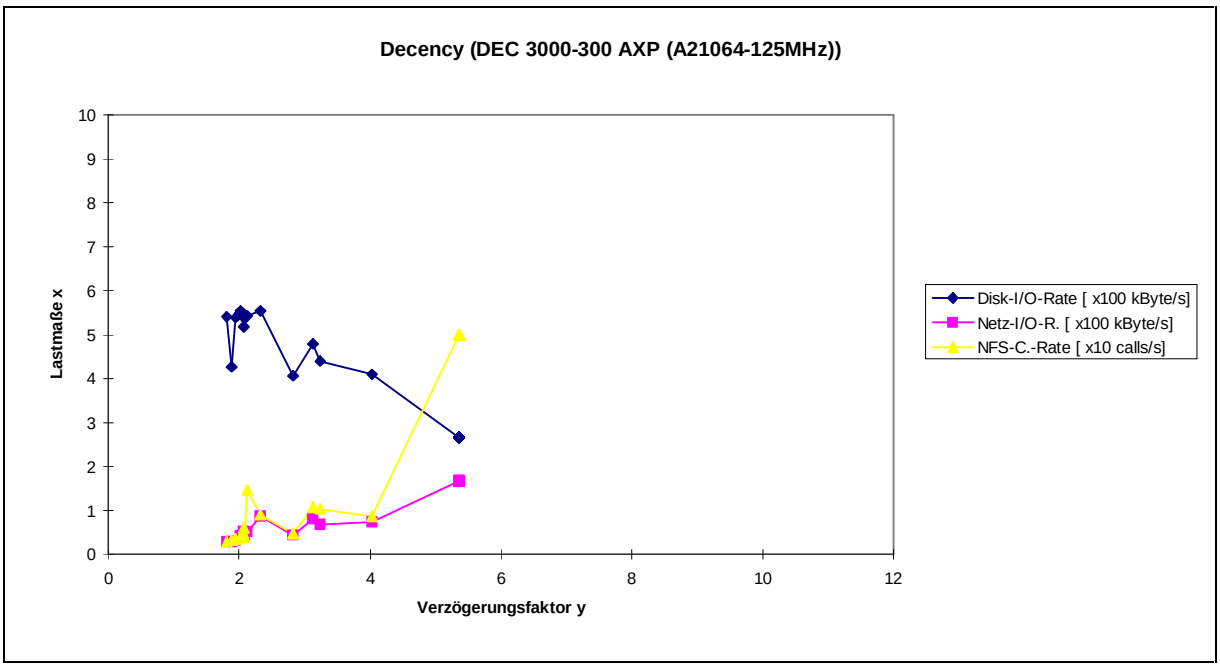
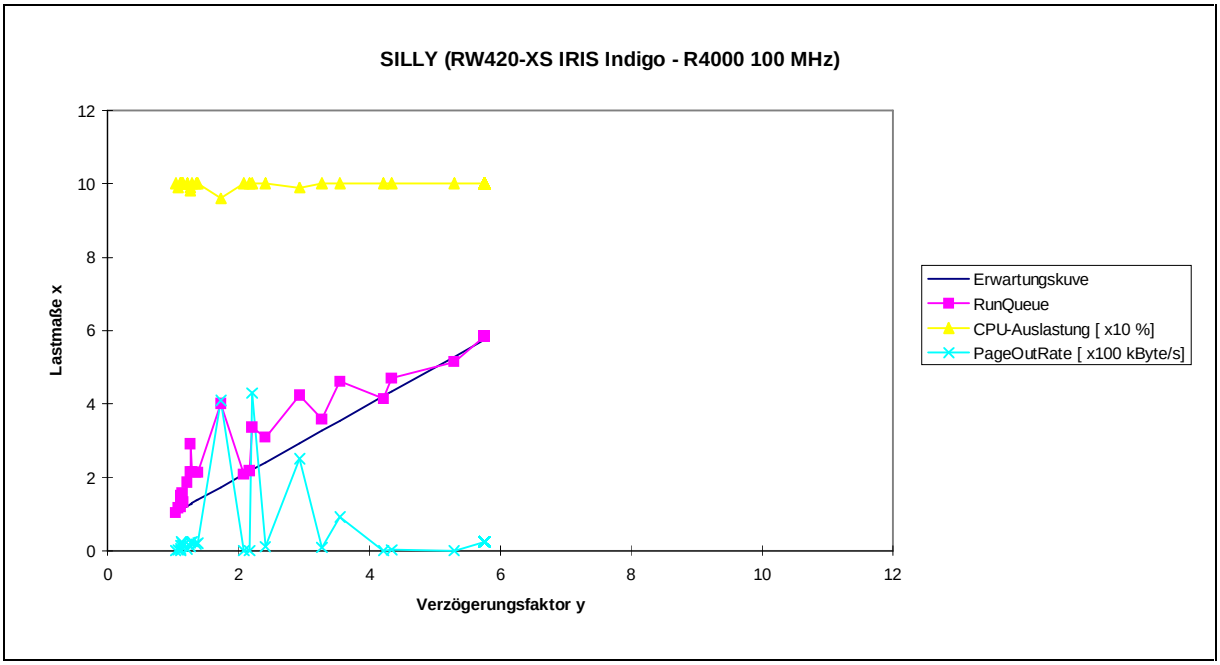


Diagramm 5-8 126.gcc (CINT95)

5.2.3 Workstation SGI RW420-XS IRIS Indigo - R4000 (silly.hrz)

- *Benchmark 101.tomcatv*



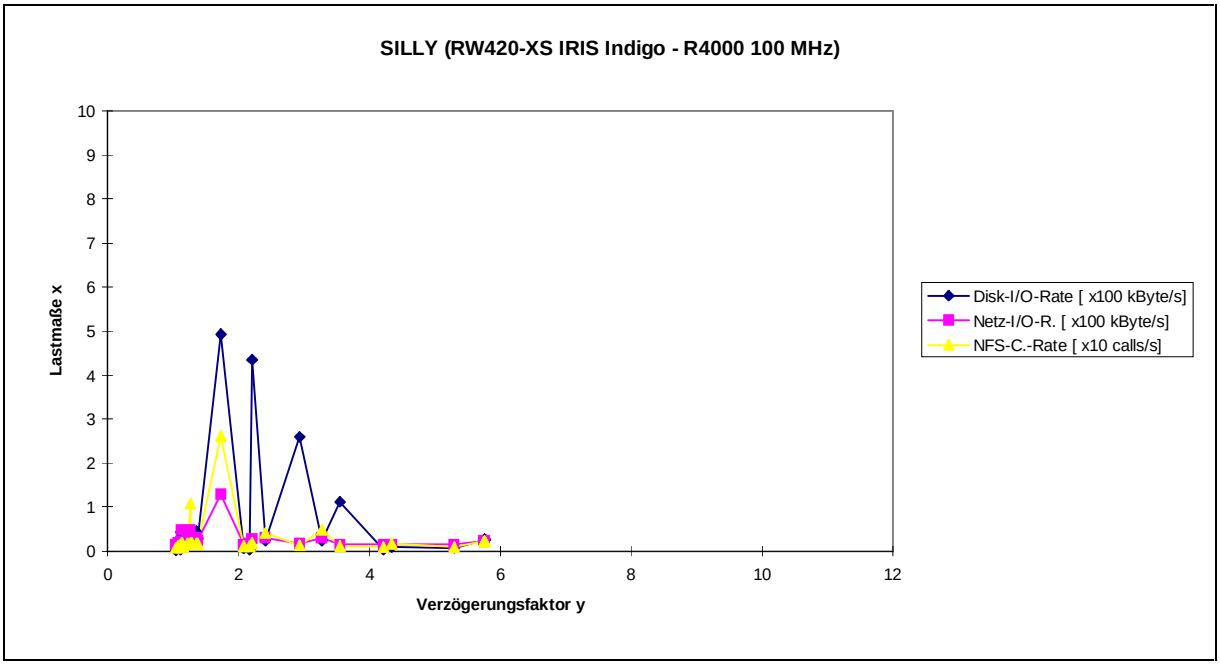


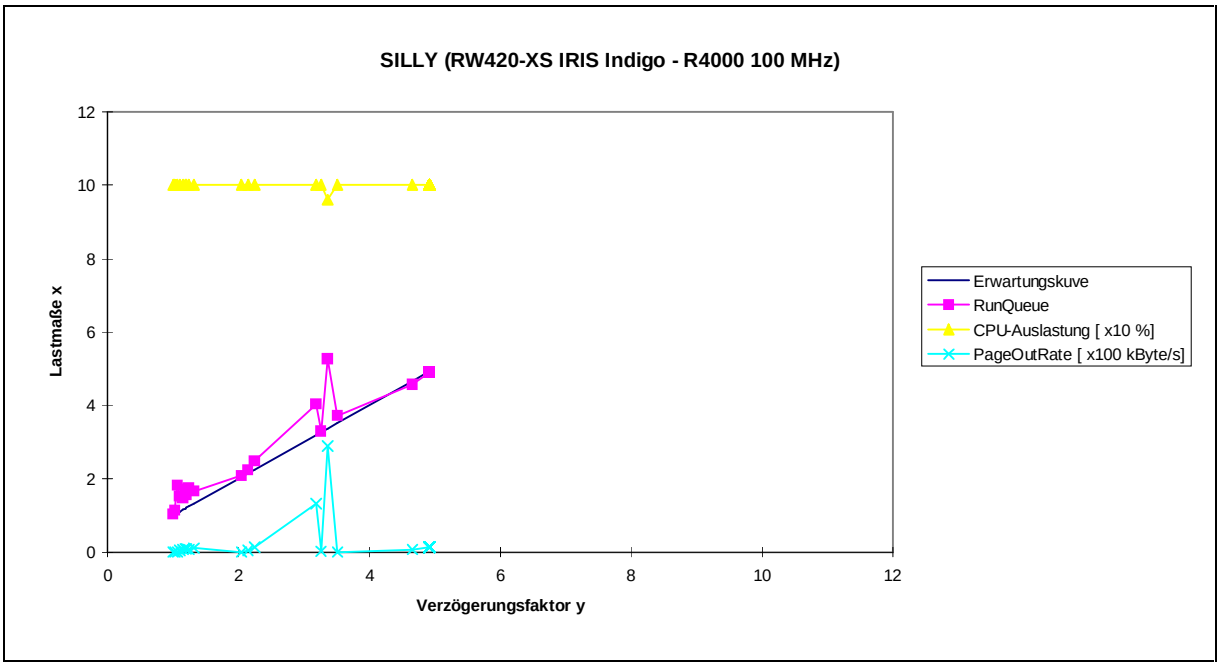
Diagramm 5-9 Benchmark 101.tomcatv (CFP95)

Dieses Rechnersystem wird an TU Chemnitz-Zwickau ebenfalls als Workstation eingesetzt. Mit der Hauptspeicherausstattung von 48 MByte zeigt sich jedoch ein wesentlich homogeneres Lastverhalten, als bei der zuvor betrachteten DEC-Workstation.

Zwischen der Länge der Prozeßwarteschlange und der Benchmarkverzögerung besteht beim Benchmark 101.tomcatv generell der erwartete, lineare Zusammenhang. Bis in die höchsten gemessenen Lastbereiche entspricht die Verzögerung dabei dem Erwartungswert, der sich durch die gemessene Warteschlangenlänge bestimmt. Die Prozessorauslastung liegt bei fast allen Messungen nahe 100%, was zeigt, daß beim Betrieb dieses Systems keine akute Beeinträchtigung des Systemverhaltens durch eine periphere Einflußgröße zu verzeichnen ist.
Ein interessantes Verhalten zeigt sich bei den Messungen, bei denen Operationen der virtuellen Speicherverwaltung registriert wurden. Dabei werden immerhin Speicherauslagerungsraten von bis zu 400 KByte/s verzeichnet. Die Darstellung zeigt hier, daß bei solchen Messungen der Verzögerungsfaktor um 1..2 niedriger liegt, als dies bei der gemessenen Warteschlangenlänge zu erwarten wäre. Eine Beschleunigung der Abarbeitung des Benchmarks bei der Benutzung von Auslagerungsspeicher gegenüber der Benutzung physischen Hauptspeichers ist jedoch als ausgeschlossen zu betrachten. Man kann nur spekulieren, daß dieser Effekt auf eine abweichende Realisierung der Prozeß- und Speicherverwaltung innerhalb des IRIX-Systems zurückzuführen ist. Man kann vermuten, daß für die Verwaltung des virtuellen Speichers erforderliche Systemprozesse jeweils in die real kürzere Prozeßwarteschlange einfließen und diese „künstlich“ verlängern. Diese Vermutung bestätigt sich, wenn man zusätzlich die Verteilung der Prozessorzeit für System- und Nutzerprozesse betrachtet. Während bei Messungen ohne Speicherauslagerungen nahezu 100% der Prozessorzeit für Nutzerprozesse zugeteilt werden, werden bei Messungen mit höheren Speicherauslagerungsraten bis zu 20% der Prozessorzeit für Systemprozesse benötigt. Damit läßt sich auch hier ablesen, daß die Benchmarkabarbeitung von der virtuellen Speicherverwaltung beeinflußt wird, allerdings ist eine Quantifizierung des Einflusses hier nicht möglich.
Die Verläufe der Kurven von Page-Out-Rate und Disk-I/O-Rate ähneln sich stark. Dies läßt auf geringe lokale Diskzugriffsintensität durch Anwendungen und Betriebssystem schließen, so daß bei den gemessenen Raten kein direkter Einfluß durch das lokale Sekundärspeichersystem feststellbar ist.
Auch zwischen den relativ niedrigen Netz-I/O-Raten und NFS-Call-Raten und der Benchmarkverzögerung läßt sich kein Einfluß erkennen, dies gilt auch für jene Messung mit der erhöhten NFS-Call-Rate von immerhin 25 Calls/s.

- *Benchmark 102.swim*

Über den gesamten Lastbereich, der bei den Messungen erfaßt wurde, läßt sich deutlich der lineare Zusammenhang zwischen der Länge der Prozeßwarteschlange und der resultierenden Verzögerung des Benchmarklaufes erkennen, wobei die gemessenen Verzögerungen auch hier im Bereich der Erwartungswerte liegen.
Für die Bewertung des Einflusses der Speicherbelastung gelten die gleichen Aussagen wie bei 101.tomcatv, es ist eine Beeinflussung zu verzeichnen, die aufgrund der „verfälschten“ Warteschlangenlänge jedoch nicht quantifiziert werden kann.
Aus allen anderen betrachteten Systemkomponenten und den darauf gemessenen Lasten lassen sich wegen des geringen Lastniveaus keine Einflüsse ableiten oder liegen keine Einflüsse vor.



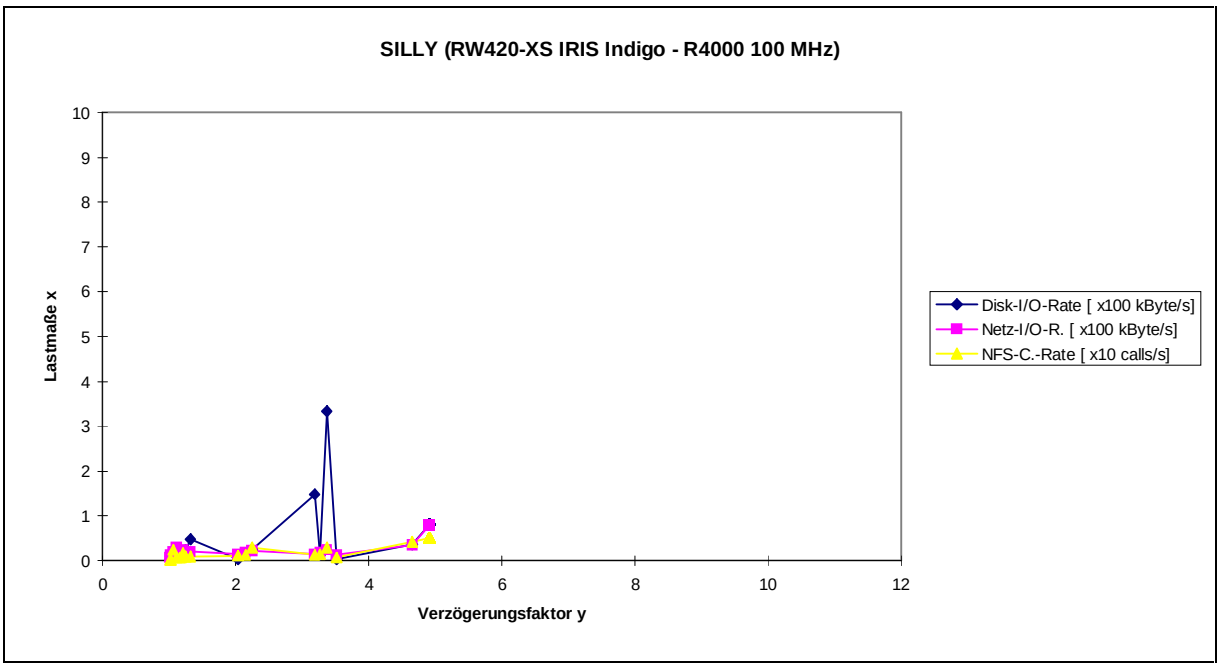
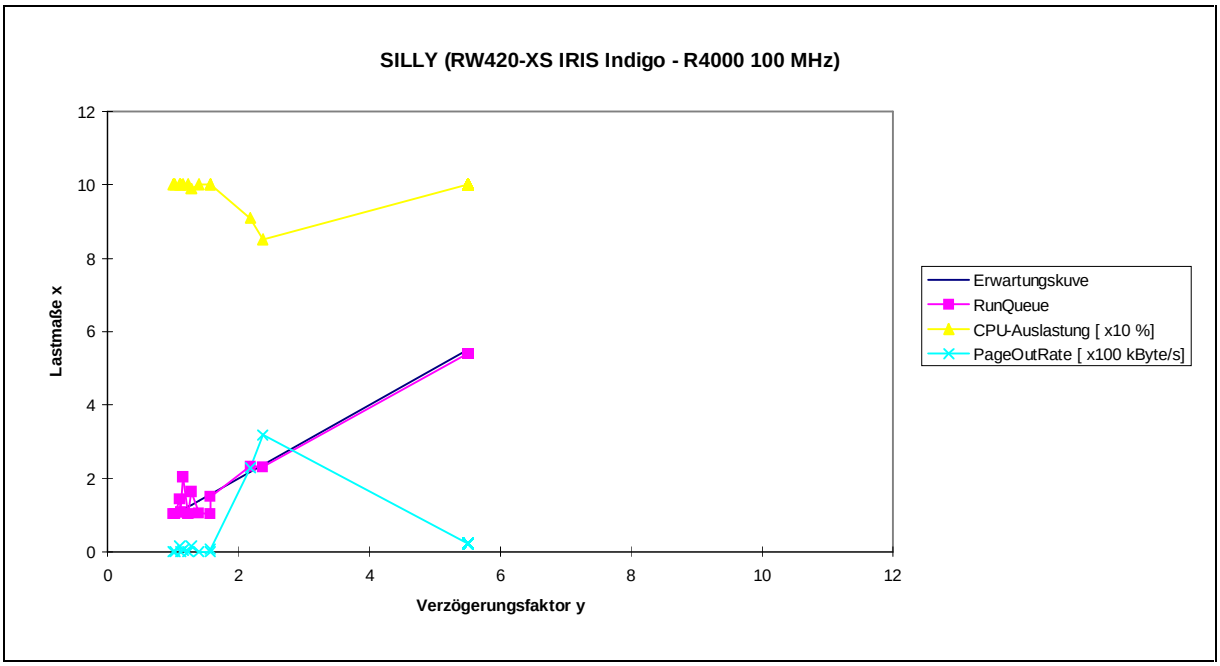


Diagramm 5-10 Benchmark 102.swim (CFP95)

- *Benchmark 124.m88ksim*

Trotz eines großen Stichprobenumfangs wurde bei den Messungen zu diesem Benchmark, bis auf wenige Ausnahmen, nur ein schmaler Bereich des möglichen Lastumfanges abgedeckt. Für den normalen Workstationbetrieb ist dieser Lastbereich jedoch als typisch zu bewerten. Aus den geringen Lastmengen für die einzelnen Lastmaße können hier keine repräsentativen Bewertungen abgeleitet werden. Wie bei den beiden bereits betrachteten Fließkomma-Benchmarks ist jedoch auch hier ein linearer Zusammenhang zwischen Warteschlangenlänge und Benchmarkverzögerung erkennbar, bei dem auch nur geringe Abweichungen vom Erwartungswert verzeichnet werden. Gewisse Streuungen der

Verzögerung um die Erwartungswerte treten im Bereich geringer Prozeßbelastung zwischen 1 und 2 Prozessen auf.
Eine einzige Messung mit einer erhöhten Page-Out-Rate zeigt keinerlei Einfluß auf die Benchmarkverzögerung, die genau im Erwartungsbereich liegt.



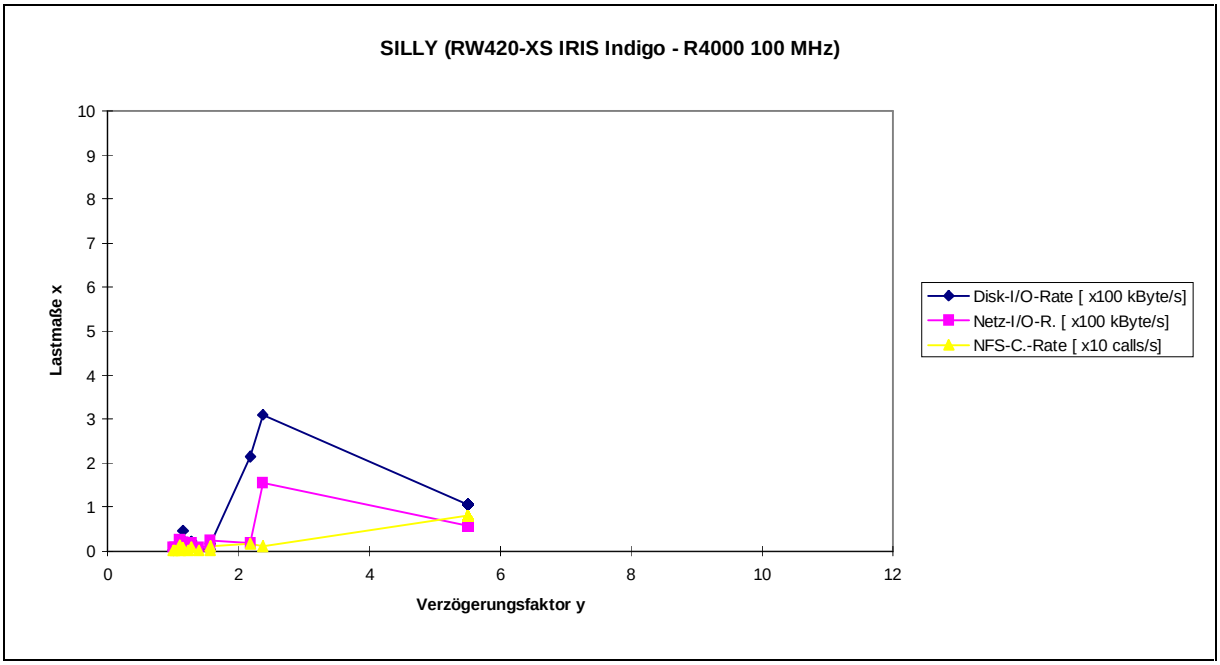


Diagramm 5-11 Benchmark 124.m88ksim (CINT95)

- *Benchmark 126.gcc*

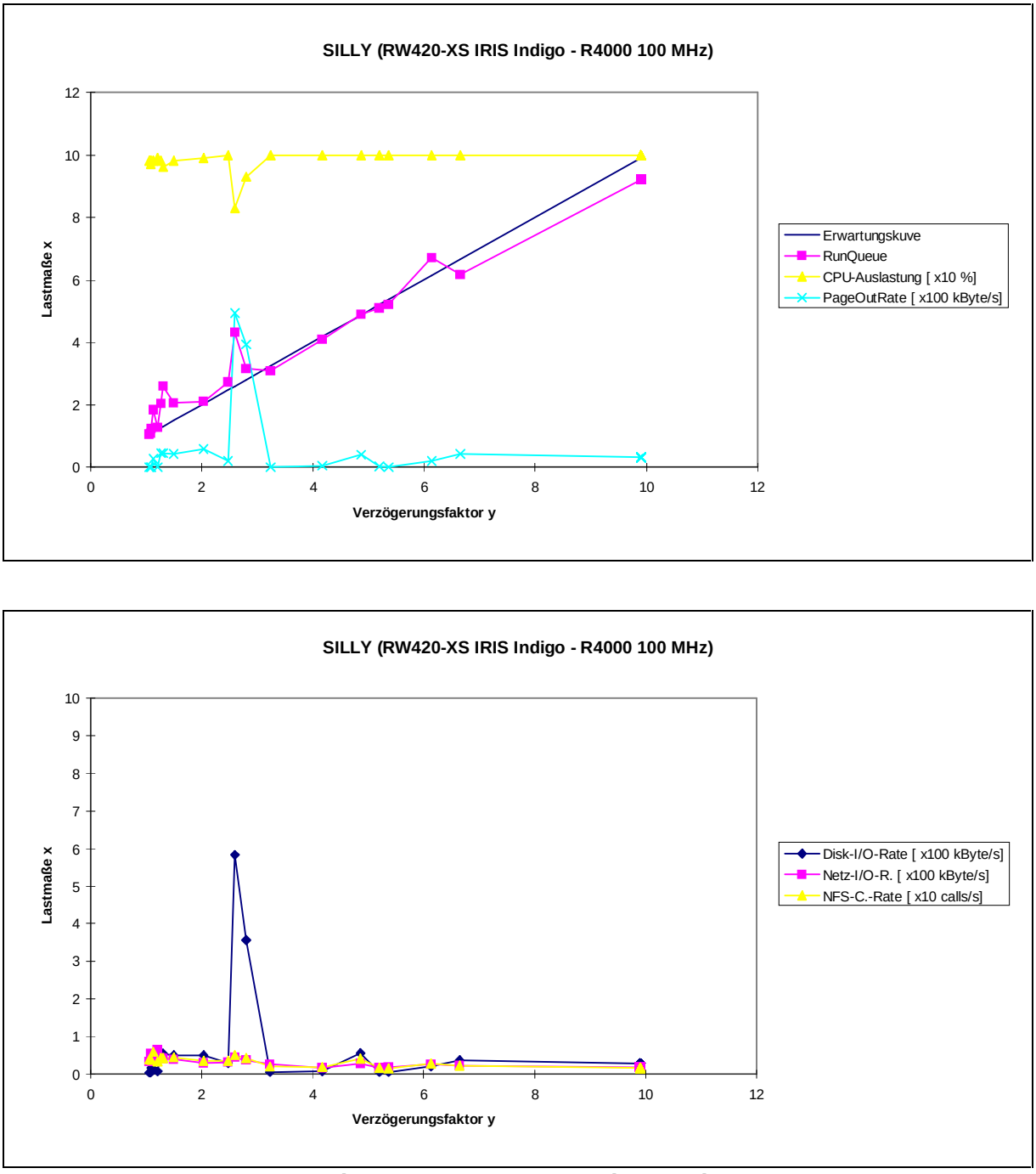


Diagramm 5-12 126.gcc (CINT95)

Bis in den Bereich der hohen Prozeßlasten mit 10 aktiven Prozessen besteht der lineare Zusammenhang zwischen der Länge der Run-Queue und der Verzögerung bei der Benchmarkabarbeitung. Dabei entspricht die gemessene Verzögerung nahezu immer dem Erwartungswert.
Die zwei Messungen mit hohen Auslagerungsraten von ca. 400 bzw. 500 KByte/s zeigen wiederum den Effekt der verfälschten Warteschlangenlänge, hier sinkt die CPU-Auslastung auf nur noch 80% (davon entfallen noch 15% der CPU-Zeit auf Systemprozesse), was deutliches Indiz für Wartezeiten auf Ein-

und Ausgaben ist. Diese müßten jedoch bei einer realistischen Warteschlangenlänge von 4 Prozessen mit der Abarbeitung der verbleibenden Prozesse ausgefüllt werden, so daß normalerweise keine Leerlaufzeiten für die CPU auftreten dürften. Interessant sind insbesondere die mit den Auslagerungen verbundenen hohen Disk-I/O-Raten von bis zu 600 KByte/s, die als Dauerbelastung unter Umständen schon die Leistungsgrenze der Harddisks darstellen und dann auch Ursache für zusätzliche Verzögerungen beim Speicherzugriff und damit für die auftretenden Leerlaufzeiten darstellen. Bei den anderen Messungen werden durchschnittliche Auslagerungsraten von 20..50 KByte/s registriert, die keine Abweichungen der linear steigenden Benchmarkverzögerung vom Erwartungswert bewirken.
Die konstant niedrigen Netz-I/O-Raten werden vom Rechnersystem problemlos bewältigt, eine dadurch bedingte Verzögerung ist bei keiner Messung feststellbar. Die NFS-Call-Rate liegt, bedingt durch das Anwendungsprofil des Benchmarks, mit 4 bis 6 Calls/s deutlich höher als bei den anderen Benchmarks mit im Mittel nur 1 bis 2 Calls/s, wobei diese erhöhte Last jedoch keine relevante Benchmarkverzögerung hervorruft.

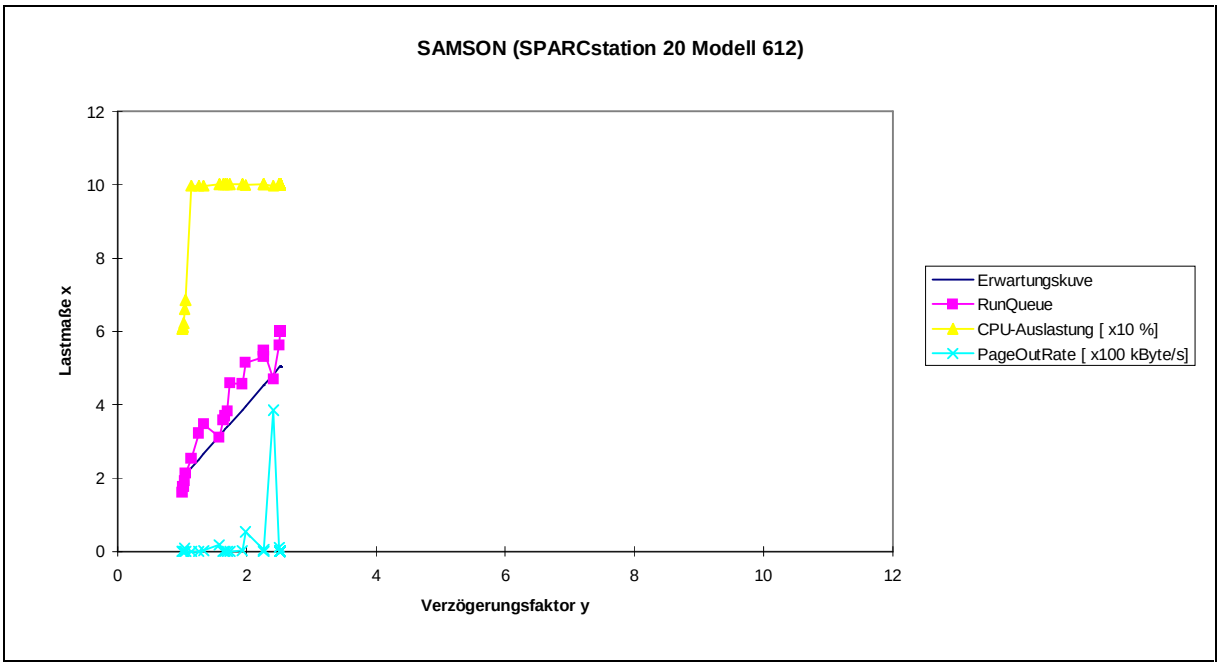
5.2.4 Compute-Server SPARCstation 20 Modell 61 (2 x SuperSPARC-CPU) (samson.hrz)

Zur Betrachtung der Meßreihen auf diesem Multiprozessorsystem ist eine Erweiterung des gewählten Bewertungsansatzes für Multiprozessorsysteme mit n Prozessoren erforderlich.
Die zur Abarbeitung bereiten Prozesse befinden sich auch hier in einer Warteschlange, aus der sie vom Betriebssystem einem oder mehreren freien Prozessoren zugeteilt werden (symmetrisches Multiprocessing).
Ausgehend von der maschinenspezifischen Referenzzeit (Mach_Ref_Time) für eine beliebige Applikation oder einen Benchmark sind zwei Betrachtungsweisen für die Festlegung des Erwartungswertes für die Abarbeitungsverzögerung denkbar. Wenn die Anwendung auf n Prozessoren vollständig parallelisierbar wäre, dann würde sich die zu erwartende Verzögerung generell als (RunQueue*Mach_Ref_Time)/n bestimmen, für Warteschlangenlängen kleiner als n also faktisch eine Beschleunigung für den Anwendungsprozeß bedeuten. Dies setzt aber eine Verteilbarkeit der Applikation voraus, die nur in den wenigsten praktischen Anwendungen gegeben ist. In den Dokumentationsfiles zu den einzelnen SPEC-Benchmarks [SPEC_DES_x] finden sich mehrfach Aussagen, daß keine Parallelisierbarkeit gegeben bzw. beim Entwurf berücksichtigt worden ist. Auch die verwendeten Compiler erzeugen keinen speziellen parallelen Code.
Daher muß der Erwartungswert für die Verzögerung anders definiert werden. Für Warteschlangenlängen RunQueue < n ergibt sich ein theoretischer Verzögerungsfaktor 1, da jeder Prozeß immer einen freien Prozessor „findet“, eine Beschleunigung durch Verteilung vom Entwurf der Applikationen her aber ausgeschlossen ist. Es wird damit nur ein Teil der möglichen Prozessorkapazität ausgenutzt. Für Warteschlangenlängen RunQueue > n ergibt sich tatsächlich ein Erwartungswert für die Verzögerung von (RunQueue*Mach_Ref_Time)/n, da die verschiedenen Prozesse dann auf die n Prozessoren verteilt werden. Von diesem Fall wird auch bei der folgenden Auswertung der Meßreihen für den Compute-Server „Samson“ ausgegangen.

- *Benchmark 101.tomcatv*

Das Laufzeitverhalten des Benchmarks entspricht den im vorigen Abschnitt beschriebenen Erwartungen. Bei Run-Queue-Längen unter 2 wird stets in etwa die Referenzzeit für diese Maschine von ca. 1410 Sekunden oder leicht darüber erreicht, was dem Verzögerungsfaktor 1 entspricht. Dabei steigt die Prozessorauslastung mit steigender Warteschlangenlänge von anfangs nur 60% bis zur vollständigen Auslastung, die sich ab der Run-Queue-Länge 2 einstellt. Mit weiter steigender Prozeßanzahl stellt sich der erwartete lineare Zusammenhang zwischen Verzögerung und Prozeßanzahl ein. Dabei liegt der Verzögerungsfaktor sogar etwa um den Wert 1 niedriger als der Erwartungswert, der sich aus der halben Warteschlangenlänge ergibt.
Die Speicherauslagerungsrate schwankt zwischen 0 und 385 KByte/s, ohne daß ein allgemeingültiger Zusammenhang zur Prozeßlast besteht. Bei der Speicherbestückung mit 192 MByte sind Speicherauslagerungen nur von der Summe der Speicherforderungen aller aktiven Prozesse abhängig, die jedoch völlig zufällig bestimmt sind. Allerdings sind bei fast allen Messungen Speicherauslagerungen zu verzeichnen, auch wenn diese teilweise bei mittleren Raten von nur 0.1 KByte/s liegen und durch sporadisches Paging oder Swapping verursacht werden. Bei den wenigen

Messungen mit höheren Page-Out-Raten über 10 KByte/s läßt sich abschätzen, daß sich der normale Verzögerungsfaktor um 0.25 bis 0.5 für pro 100 KByte/s erhöht. Damit liegen auch alle Messungen mit Speicherauslagerungen zumindest noch im Bereich des ursprünglichen Erwartungswertes.



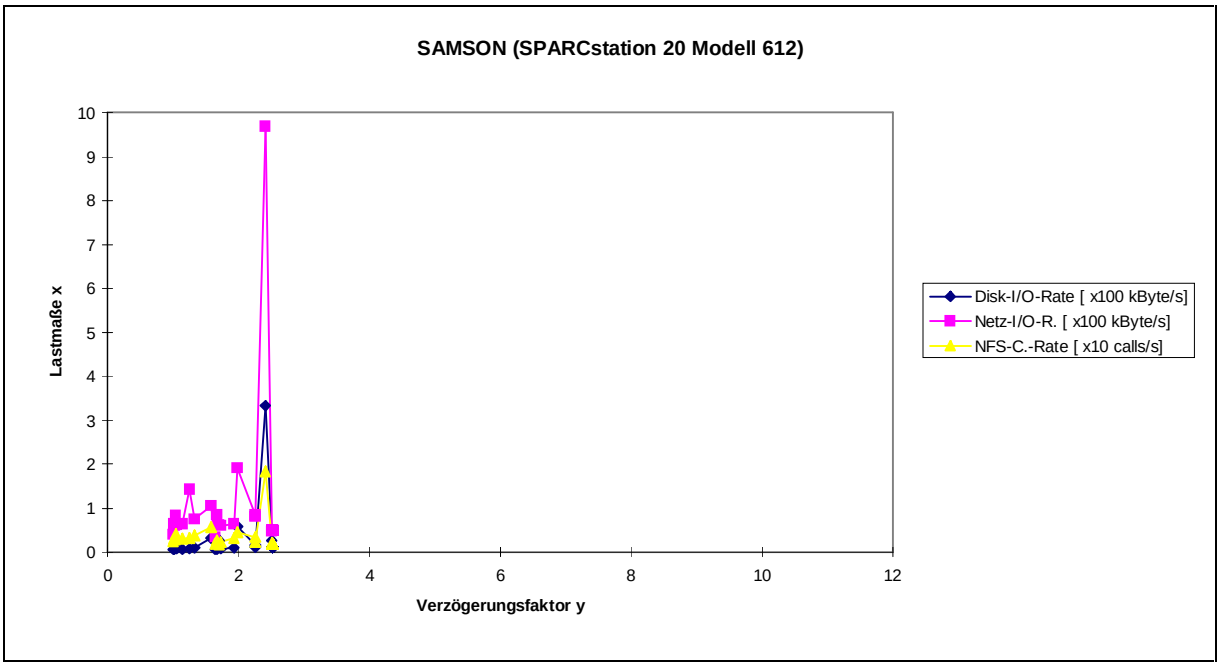
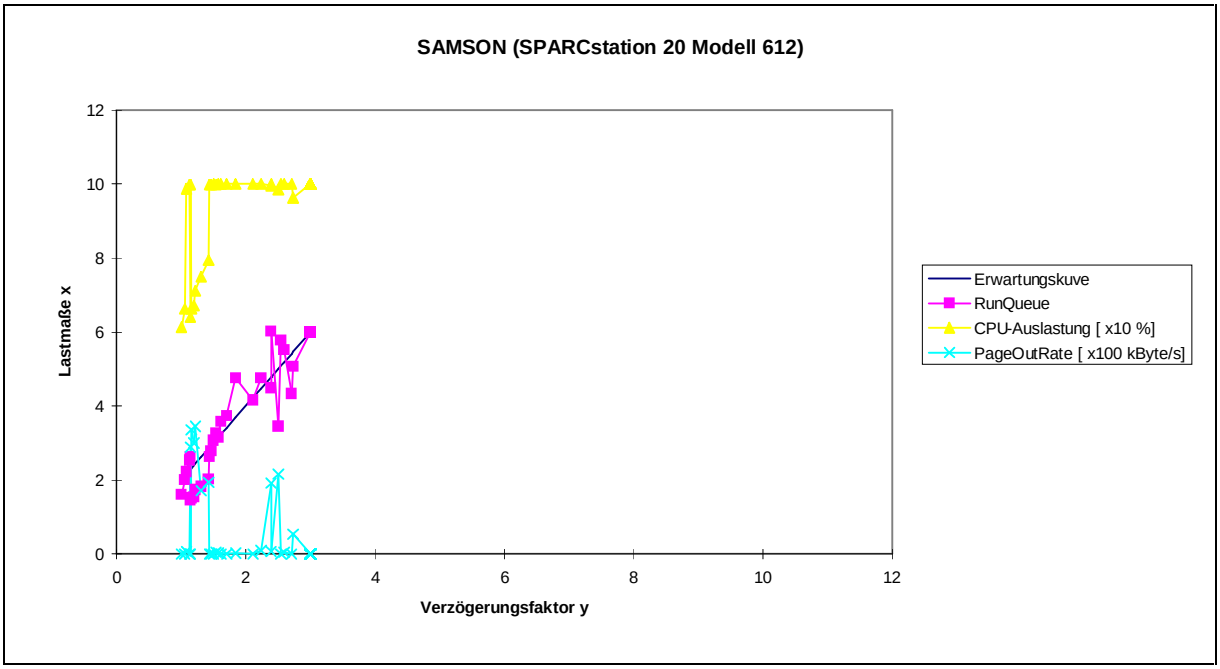


Diagramm 5-13 Benchmark 101.tomcatv (CFP95)

Die Netz-I/O-Rate schwankt zwischen 30 und knapp 1000 (!) KByte/s und liegt im Mittel um 100 KByte/s. Die NFS-Call-Rate, die im Mittel bei 3 Calls/s liegt, streut dabei zwischen 2 und 18 Calls/s. Aussagen zu direkten Einflüssen auf die Benchmarkverzögerung sind dabei nicht zu treffen, besonders da in der Messung mit den besonders hohen Werten auch eine erhöhte Page-Out-Aktivität zu verzeichnen ist, die bereits als bremsendes Element identifiziert wurde und damit eventuelle Netzeinflüsse verdeckt.

- *Benchmark 102.swim*



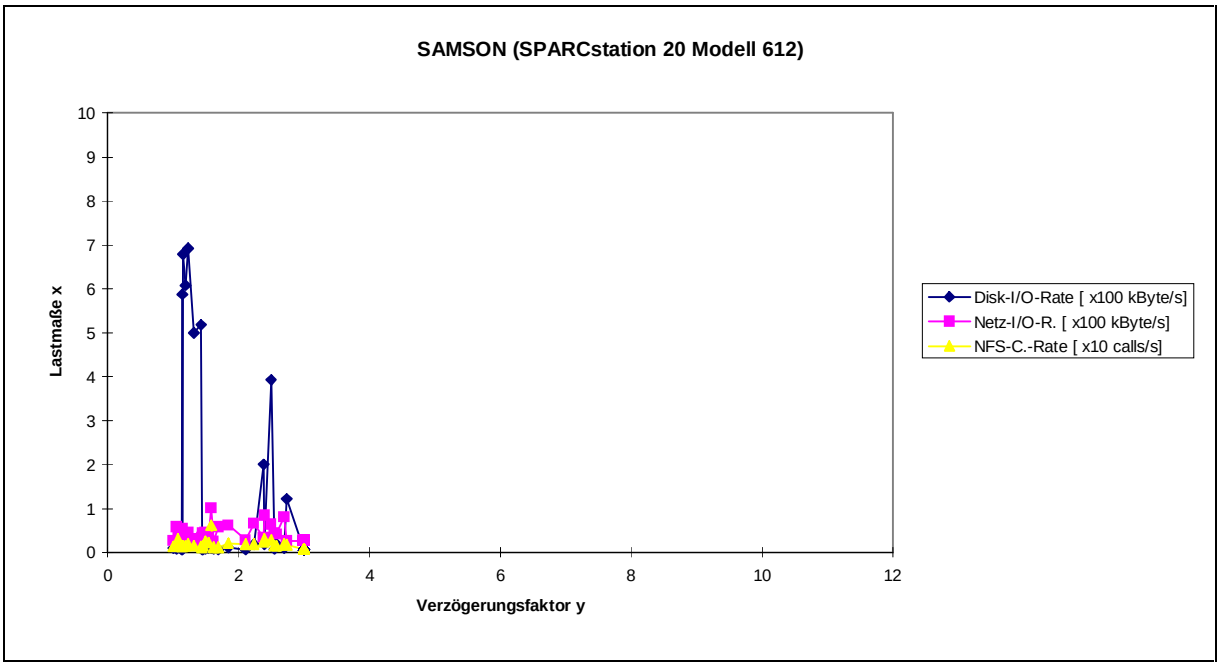


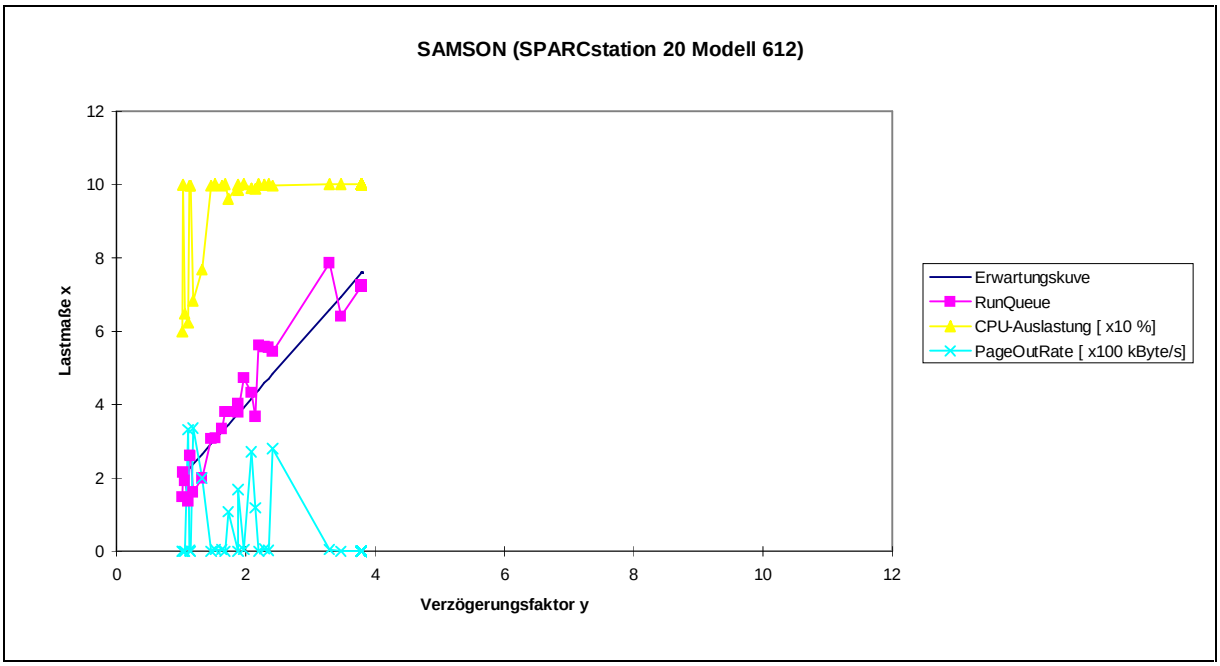
Diagramm 5-14 Benchmark 102.swim (CFP95)

Auch dieser Benchmark verhält sich grundsätzlich erwartungskonform hinsichtlich der Verzögerung bei steigender Prozeßlast. Bei Run-Queue-Längen unter 2 Prozessen wird für die Abarbeitung etwa die maschinenspezifische Referenzzeit von 2450 Sekunden benötigt. Mit steigender Warteschlangenlänge folgt die Verzögerung dann generell dem Erwartungswert von Run-Queue-Länge/2, wobei bei 102.swim häufigere Abweichungen vom linearen Zusammenhang zu verzeichnen sind als bei 101.tomcatv.

Schon bei niedriger Prozeßlast treten bei einigen Testläufen hohe Page-Out-Raten zwischen 200 und 350 KByte/s auf, bei denen auch bei RunQueue < 2 Verzögerungsfaktoren über 1 gemessen werden. Mit diesen Page-Out-Raten sind geringere CPU-Auslastungen von nur 80% auch bei mehr als zwei Prozessen verbunden. Die Disk-I/O-Raten von 500 bis 700 KByte/s als dauernde Belastung erreichen dabei bereits die Belastungsgrenze des Disksystems, da hier auch Transferzahlen von ca. 100 Transfers/s ermittelt wurden.
Im Bereich höherer Prozeßzahlen sind starke Streuungen um den Erwartungswert für die Verzögerung zu verzeichnen, wobei der generelle, lineare Verlauf des Zusammenhangs von Run-Queue und Verzögerung trotzdem erhalten bleibt. Die stärkeren Verzögerungen sind erwartungsgemäß mit hohen Speicherauslagerungs- und Disktransferraten verbunden. Die Resultate mit Verzögerungsfaktoren, die unter dem Erwartungswert liegen, können hingegen mit den betrachteten Last- und Einflußgrößen nicht begründet werden. Als Erklärung kommt damit eigentlich nur die Beschleunigung durch Cache-Speicher in Frage.
Zwischen Netz-I/O-Rate bzw. NFS-Call-Rate und den gemessenen Verzögerungen sind keine Kausalitäten erkennbar. Die Schwankungen der Verzögerungen sind unabhängig von denen der Netzlasten, die, verglichen mit anderen Messungen und Maschinen, auf einem normalen, niedrigen Niveau liegen.

- *Benchmark 124.m88ksim*

Bei diesem Benchmark der CINT-Suite zeigt sich grundsätzlich ein Laufzeitverhalten mit Verzögerungen im Bereich der Erwartungswerte.
Schon bei niedrigen Prozeßlasten werden Speicherauslagerungsraten bis zu 350 KByte/s gemessen, mit denen wiederum Disk-I/O-Raten von bis zu 700 KByte/s (bei bis zu 90 Transfers/s) zur Swap-Disk verbunden sind. Trotz daher zu erwartender Wartezeiten treten die teilweisen „Leerlaufzeiten“ der CPUs nur bei Messungen mit Warteschlangenlängen < 2 auf. Bei Warteschlangenlängen > 3 Prozesse ist selbst bei Page-Out-Raten von 300 KByte/s keine deutliche Abweichung vom linearen Verlauf der Erwartungskurve erkennbar. Insgesamt zeigen die Verzögerungen wieder deutliche Streuungen in positiver und negativer Richtung, ohne daß die Abweichungen auf einzelne Lastspitzen bei einzelnen Komponenten zurückführbar sind. Tendenziell kann man Verzögerungsfaktoren im Bereich Erwartungswert -1 abschätzen, die sich bei höheren Speicherauslagerungsraten und zunehmenden Warteschlangenlängen bis zum Erwartungswert oder geringfügig darüber erhöhen.
Die Netz-I/O-Rate liegt bei allen Messungen auf niedrigem Niveau zwischen 20 und 100 KByte/s, was wiederum keine Aussage zu einem direkten Einfluß auf die Verzögerung zuläßt. Die bei einer Messung registrierte hohe Netzlast von 600 KByte/s zeigt keine Auswirkung auf die gemessene Verzögerung, so daß die Prognose möglich ist, daß eine Abhängigkeit von der Netzlast nicht besteht, was auch mit dem Anwendungsprofil des Benchmarks begründbar ist. Analoge Aussagen lassen sich aus den niedrigen NFS-Call-Raten von 1..5 Calls/s ableiten, eine signifikante Verzögerung läßt sich bei der beobachteten relativen Gleichverteilung nicht erkennen oder begründen.



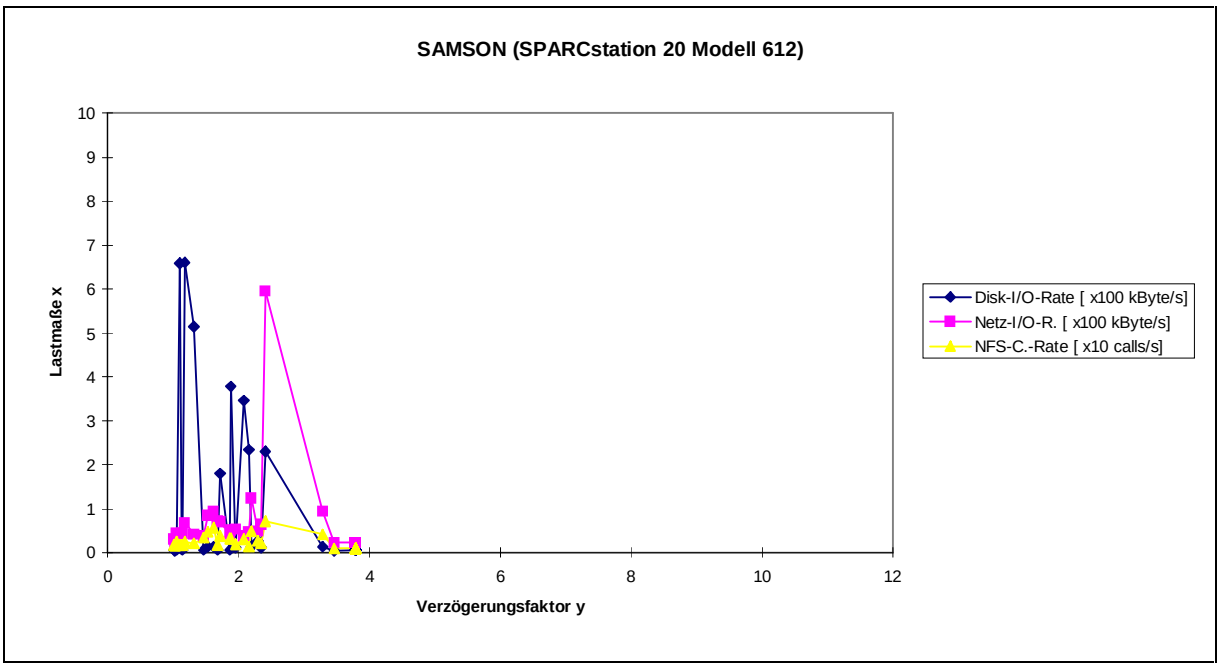
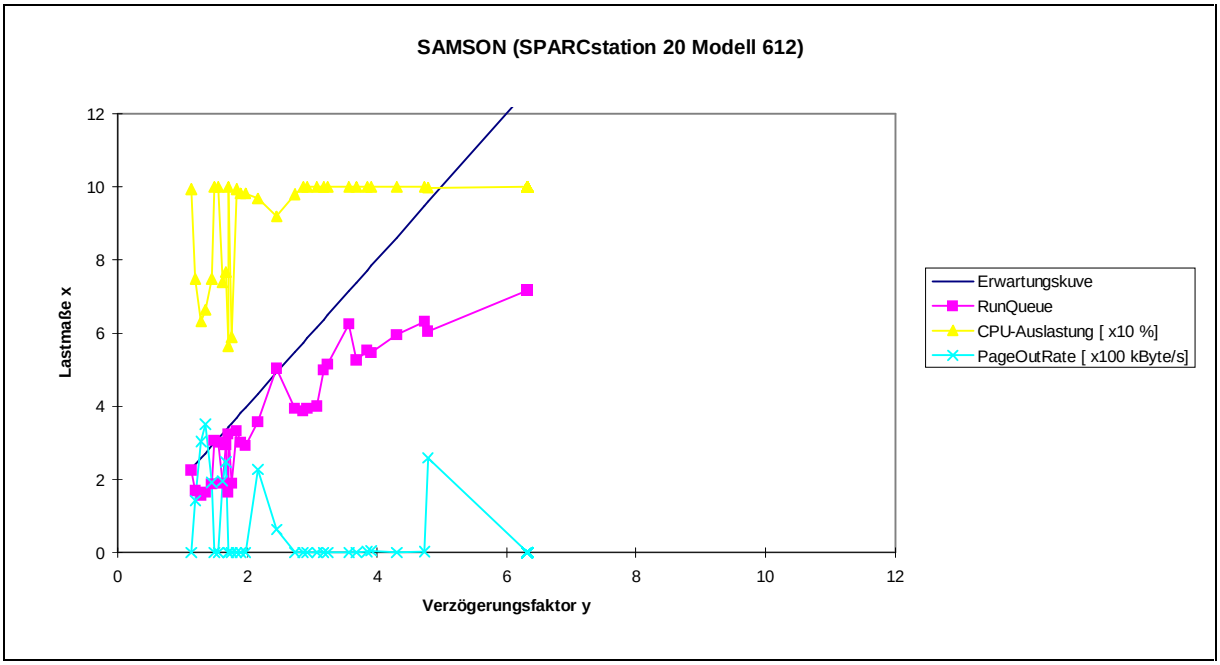


Diagramm 5-15 Benchmark 124.m88ksim

- *Benchmark 126.gcc*



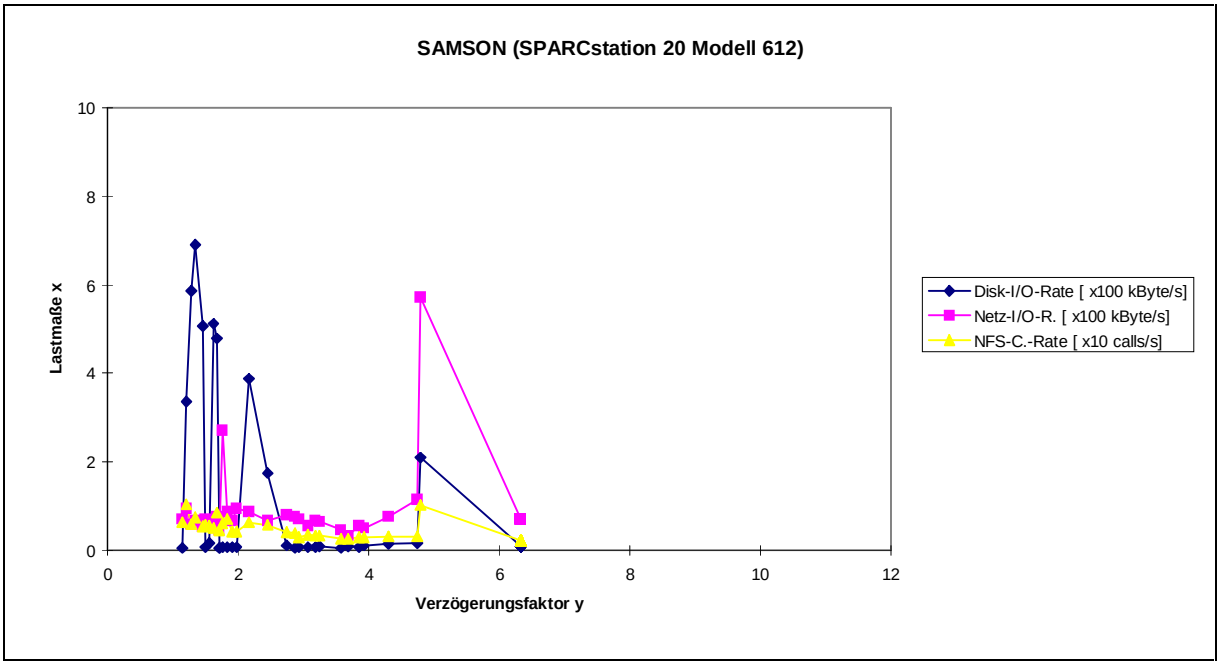


Diagramm 5-16 Benchmark 126.gcc (CINT95)

Bei diesem Benchmark zeigt sich auf dem Rechner Samson ein deutlich von den anderen Tests abweichendes Verhalten. Die registrierten Verzögerungen liegen deutlich über den aus den gemessenen Längen der Run-Queue zu erwartenden Werten. Aus der graphischen Darstellung kann man sogar eine exponentielle Abhängigkeit der Verzögerung von der Warteschlangenlänge ableiten. Bei Längen zwischen 2 und 5 Prozessen liegen die gemessenen Benchmarklaufzeiten noch im Bereich der Erwartungswerte, bei weiter steigender Prozeßbelastung ist ein überproportionaler Anstieg der Laufzeitverzögerung zu verzeichnen. Im hohen Prozeßlastbereich ist dieser Verlauf sogar unabhängig

von auftretenden Speicherauslagerungen. Im unteren Lastbereich mit 2..5 Prozessen ist hingegen das bisher typische Verhalten zu registrieren, daß sich für 100 KByte/s der Verzögerungsfaktor im Mittel um 0.25 bis 0.5 erhöht.
Eine zusätzliche Betrachtung der Verteilung der CPU-Zeiten bei Messungen mit vollständiger CPU-Auslastung zeigte einen mittleren Anteil von 25% für Systemprozesse mit steigender Tendenz bei wachsender Prozeßanzahl. Bei den anderen Benchmarks wurden gewöhnlich Systemzeitanteile von maximal 10% beobachtet. Die Bearbeitung der häufig höher priorisierten Systemprozesse muß als Ursache für den überproportionalen Anstieg der Benchmarkverzögerung betrachtet werden. Der erhöhte Systemzeitanteil läßt sich auf die erhöhten Ein- und Ausgabeaktivitäten zurückführen, die ihre Ursachen im Anwendungsprofil des Benchmarks haben.
Netz-I/O-Raten und NFS-Call-Raten sind über alle Messungen hinweg wieder ziemlich gleich verteilt, allerdings liegen die mittleren Werte bei diesem Benchmark etwa doppelt so hoch, wie bei den anderen Tests auf dieser Maschine, was auf das Ein- und Ausgabeprofil des Benchmarks zurückführbar ist.

5.3 Zusammenfassung der Einflußfaktoren auf ausgewählte SPEC-CPU-Benchmarks

In den folgenden Aufstellungen wird für die ausgewählten Benchmarks zusammengefaßt, welche Abhängigkeiten der Benchmarks von den möglichen Lastfaktoren in den durchgeführten Meßreihen tatsächlich festgestellt werden konnten und wie diese quantifiziert werden konnten.

5.3.1 Lasteinflüsse bei 101.tomcatv

Kategorie:	CPU-Benchmark für Fließkomma-Arithmetik (CFP95) (doppelte Genauigkeit)
Gegenstand:	Flüssigkeitsdynamik und geometrische Translation (Netzgenerierung um allgemeine geometrische Bereiche)
Charakteristik:	Komplexe Matrix-Operationen mit stark differenzierten Zugriffsfolgen und mehreren Iterationsstufen
Ressourcenbedarf:	- ca. 15 MByte Hauptspeicher für Daten (statisch) - 100 KByte Hauptspeicher für Code
Ein-/Ausgabeprofil:	- einmaliges Laden des Benchmarkprogramms - Lesen eines 99 Byte großen Initialisierungsfiles und eines ca. 10 MByte großen MODELL-Files bei Benchmarkstart - Schreiben eines ca. 21 KByte großen Resultatfiles bei Benchmarkende
Gemessene Laufzeiten:	790 bis 15500 Sekunden

Lastabhängigkeiten:

- CPU-Last (Run-Queue-Länge):

Es besteht ein genereller Zusammenhang zwischen Prozeßlast und Benchmarkverzögerung, der sich systemabhängig linear oder gering exponentiell darstellt.

- Speicherlast (Page-Out-Rate):

Mit steigender Auslagerungsrate geht eine besonders starke Benchmarkverzögerung einher. Im Extremfall geht der Zusammenhang zur Prozeßlast verloren, und es zeigt sich eine nahezu lineare Abhängigkeit von Benchmarkverzögerung und Speicherlast (Diagramm 5-5). Im Normalfall erhöht sich der prozeßlastbedingte Verzögerungsfaktor systemabhängig um Werte zwischen 0,5 bis 2 für Auslagerungsraten von je 100 KByte/s

- Disk-I/O-Last (Disk-I/O-Rate):

Eine generelle, direkte Abhängigkeit der Verzögerung von der Disk-I/O-Last besteht nicht, eine indirekte Abhängigkeit ergibt sich aus der Korrelation zur Speicherlast.

- Netzlast (Netz-I/O-Rate, NFS-Call-Rate):

Die Benchmarkverzögerung steht trotz einer erheblichen Menge an Eingabedaten in keinem Zusammenhang zu Schwankungen bei Netz-I/O- und NFS-Call-Raten.

5.3.2 Lasteinflüsse bei 102.swim

Kategorie:	CPU-Benchmark für Fließkomma- Arithmetik (CFP95) (einfache Genauigkeit)
Gegenstand:	Wettervorhersage (Lösung von „Wasser-Verflachungs-Gleichungen“ unter Anwendung endlicher Differenzenapproximation)
Charakteristik:	Matrix-Operationen in mehreren Iterationsstufen mit zwei verschiedenen Iterationstiefen
Ressourcenbedarf:	- 15 MByte Hauptspeicher für Daten (statisch) - 110 KByte Hauptspeicher für Code
Ein-/Ausgabeprofil:	- zweimaliger Programmstart dabei jeweils: Laden eines 51 Byte großen Initialisierungsfiles Schreiben eines 475 Byte großen Ergebnisfiles
Gemessene Laufzeiten:	950 bis 18900 Sekunden

Lastabhängigkeiten:

- CPU-Last (Run-Queue-Länge):

Es besteht ein genereller Zusammenhang zwischen Prozeßlast und Benchmarkverzögerung, der sich systemabhängig linear oder gering exponentiell darstellt.

- Speicherlast (Page-Out-Rate):

Mit steigender Auslagerungsrate geht eine sichtbare Benchmarkverzögerung einher. Typischerweise erhöht sich der prozeßlastbedingte Verzögerungsfaktor systemabhängig um bis zu Wert 1 für Auslagerungsraten von 100 KByte/s.

- Disk-I/O-Last (Disk-I/O-Rate):

Eine generelle, direkte Abhängigkeit der Verzögerung von der Disk-I/O-Last besteht nicht, eine indirekte Abhängigkeit ergibt sich aus der Korrelation zur Speicherlast.

- Netzlast (Netz-I/O-Rate, NFS-Call-Rate):

Durch die geringe Ein- und Ausgabeintensität des Benchmarks ist eine Abhängigkeit unter den gewöhnlich herrschenden Bedingungen auszuschließen. Bei den Messungen konnten keine Zusammenhänge ermittelt werden.

5.3.3 Lasteinflüsse bei 124.m88ksim

Kategorie:	CPU-Benchmark ohne Test der Fließkomma-Arithmetik (CINT95)
Gegenstand:	Simulation des Motorola-88100-Prozessors bei einem Speichertest und dem Dhrystone-Benchmark
Charakteristik:	Speicher- und prozessorintensive Simulation ohne I/O-Einfluß
Ressourcenbedarf:	- 25 MByte Hauptspeicher für Daten (dynamisch) - 250 KByte Hauptspeicher für Code
Ein-/Ausgabeprofil:	- einmaliges Laden des Benchmarks - Laden eines 213 Byte großen Initialisierungsfiles bei Start - Lesen von 2 jeweils ca. 70 KByte-Files während des Benchmarks - Schreiben eines 12427 Bytes großen Ergebnisfiles
Gemessene Laufzeiten:	840 bis 12200 Sekunden

Lastabhängigkeiten:

- CPU-Last (Run-Queue-Länge):

Es besteht ein genereller Zusammenhang zwischen Prozeßlast und Benchmarkverzögerung, der sich systemabhängig linear oder bei höherer Prozeßlast gering exponentiell verhält.

- Speicherlast (Page-Out-Rate):

Mit steigender Auslagerungsrate geht eine drastische Benchmarkverzögerung für den sehr speicherintensiven Benchmark einher. Dabei erhöht sich der prozeßlastbedingte Verzögerungsfaktor um bis zu Wert 2 für Auslagerungsraten von 100 KByte/s (Diagramm 5-7 Decency)

- Disk-I/O-Last (Disk-I/O-Rate):

Eine generelle, direkte Abhängigkeit der Verzögerung von der Disk-I/O-Last besteht nicht.
Bei hohen Speicherauslagerungsraten zur lokalen Harddisk kann diese allerdings zum Flaschenhals eines Rechnersystems werden (Diagramm 5-7).

- Netzlast (Netz-I/O-Rate, NFS-Call-Rate):

Bei der geringen Ein- und Ausgabeintensität des Benchmarks ist eine Abhängigkeit unter gewöhnlichen Bedingungen auszuschließen. Auch starke Netzlastschwankungen wirken sich nicht auf die üblichen Verzögerungen aus.

5.3.4 Lasteinflüsse bei 126.gcc

Kategorie:	CPU-Benchmark ohne Test der Fließkomma-Arithmetik (CINT95)
Gegenstand:	Übersetzung von mit Präprozessor vorbereiteten Quellen in optimierten SPARC-Assembler-Code mit GNU-C-Compiler
Charakteristik:	56 Übersetzungen mit unterschiedlich großen, dynamischen Hauptspeicher-Datenmodellen und regelmäßiger E/A-Aktivität
Ressourcenbedarf:	- zwischen 1,6 MByte und 28 MByte Hauptspeicher für Daten (dynamisch) bei den einzelnen Übersetzungsprozessen, im Mittel 9 MByte - 2 MByte Hauptspeicher für Code
Ein-/Ausgabeprofil:	- 56-maliges Laden des Benchmarkprogramms, dabei jeweils: Lesen eines Source-Files zwischen 21 und 306 KByte Schreiben des Ergebnisfiles zwischen 29 und 380 KByte - Summe: 6,4 MByte Eingabedaten 8,4 MByte Ausgabedaten - Verteilung der Ein- und Ausgaben über die gesamte Laufzeit
Gemessene Laufzeiten:	630 bis 11900 Sekunden

Lastabhängigkeiten:

- CPU-Last (Run-Queue-Länge):

Es besteht ein grundsätzlicher Zusammenhang zwischen Prozeßlast und Benchmarkverzögerung, der sich systemabhängig linear oder bei höherer Prozeßlast gering exponentiell verhält. Die von den anderen Benchmarks abweichende E/A-Struktur bewirkt auf den einzelnen Testsystemen unterschiedliche Ergebnisse. Auf Rechner Tantalus liegt die Verzögerung niedriger und damit näher am Erwartungswert als bei den anderen Benchmarks. Auf den Rechnern Silly und Samson ist das Gegenteil der Fall, die Verzögerung ist stärker als bei den anderen Benchmarks, bei Samson ist ein exponentieller Anstieg feststellbar.

- Speicherlast (Page-Out-Rate):

Mit steigender Auslagerungsrate geht eine leichte Benchmarkverzögerung einher. Der Speicherbedarf des Benchmarks schwankt dynamisch, so daß nur bestimmte Phasen stärker von einer Paging-bedingten Verzögerung betroffen sind. Dabei erhöht sich der prozeßlastbedingte Verzögerungsfaktor auf allen Systemen nur um maximal Wert 1.

- Disk-I/O-Last (Disk-I/O-Rate):

Eine direkte Abhängigkeit der Verzögerung von der Disk-I/O-Last kann nicht festgestellt werden.

- Netzlast (Netz-I/O-Rate, NFS-Call-Rate):

Die Ein- und Ausgabeintensität des Benchmarks ist höher als bei den anderen Benchmarks. Im Verhältnis zur Gesamtzeit ist der Anteil dennoch so gering, daß unter den normalen Bedingungen auf den einzelnen Testsystemen keine Abhängigkeit der Verzögerung von der Netzlast feststellbar ist. Die gemessenen mittleren Netzlasten liegen auf den einzelnen Rechnern in etwa auf demselben Niveau wie bei den anderen Benchmarks. Auch aus einzelnen minimalen und maximalen Extremwerten lassen sich keine Zusammenhänge ableiten.

5.4 Diskussion der Einflüsse der Teilsysteme von komplexen Rechnersystemen

Im Kapitel 2 wurde auf die Meßverfahren und Meßbedingungen bei Leistungsuntersuchungen der SPEC-Mitgliedsfirmen und auf abweichende Bedingungen bei realen Applikationen im Praxiseinsatz, speziell auf die Nutzungsbedingungen im Rechenzentrum der TU Chemnitz, eingegangen. Aus diesen Bedingungen wurden Einflußelemente eines komplexen Rechnersystems abgeleitet, die möglicherweise eine signifikante Rolle bei der Abarbeitung eines Benchmarks spielen können. Im Kapitel 3 wurden für diese Einflußkomponenten Lastmaße erarbeitet, durch deren Bewertung über eine Reihe von Tests hinweg der tatsächliche Einfluß von Teilsystemen auf die Benchmarkergebnisse und damit auf die Leistungsfähigkeit von Rechner festgestellt werden sollte.
Dieser Abschnitt faßt die in Abschnitt 5.2 detailliert dargestellten Ergebnisse der Testreihen im Hinblick auf die Systemlast-Komponenten CPU-Last, Speicherlast, Disk-I/O-Last und Netz-I/O-Last zusammen. Die Betrachtungen erfolgen allgemein zu den jeweiligen Einflußgrößen, wobei auf Besonderheiten bei einzelnen Anwendungsgebieten bzw. den die Anwendungsgebiete repräsentierenden Benchmarks hinsichtlich ihrer Profile bei Auffälligkeiten besonders eingegangen wird. Daneben müssen natürlich auch die unterschiedlichen Konfigurationsbedingungen der Testsysteme (s. Abschnitt 4.2) berücksichtigt werden.
Neben den im Kapitel 3 definierten Lastmaßen werden in Einzelfällen weitere Meßwerte zu einzelnen Systemaktivitäten betrachtet, die von den verwendeten Systemtools geliefert werden, in die Lastmaße jedoch nicht explizit einfließen. Bei der Betrachtung solcher Werte, wie der Verteilung von Prozessorzeit für Nutzer- und Systemprozesse oder der Fehlerraten bei den Netzwerkprotokollen, wird auf diese jeweils speziell hingewiesen.

Einfluß der CPU-Last

Da die Benchmarks von SPEC als Tests zur Bewertung der Prozessorleistungsfähigkeit entworfen und definiert wurden, ist zu erwarten, daß von der CPU-Last die maßgebliche Beeinflussung der Benchmarkergebnisse bei Messungen unter Lastbedingungen ausgeht.
Die praktischen Untersuchungen haben ergeben, daß bei allen vier betrachten Benchmarks ein direkter Zusammenhang zwischen der Anzahl der laufenden Prozesse (Run-Queue-Länge) und der zeitlichen Verzögerung gegenüber einer Messung ohne fremde Systemlast besteht. Die Laufzeit für eine solche Messung ohne Fremdlast wurde auf Basis der tatsächlich benötigten Prozessorzeit (im User- und Systemmode) für einen Benchmarklauf approximiert (Abschnitt 4.3), da vergleichbare Ergebnisse von SPEC für die im URZ vorliegenden Konfigurationen und Ausstattungen der Testrechner nicht verfügbar sind..
Dabei zeigt sich in den jeweils ersten Diagrammen diese Abhängigkeit als nahezu linearer Zusammenhang von Run-Queue-Länge und Benchmarkverzögerung. Mit zunehmender Prozeßbelastung ist ein zusätzlicher Anstieg der Verzögerung der Benchmarkabarbeitung feststellbar, der sich aus dem zusätzlichen Aufwand für die Prozeßkoordinierung im Betriebssystem ergibt. Dies wird bei zusätzlicher Betrachtung der Verteilung der CPU-Zeit für Nutzer- und Systemprozesse deutlich, die von vmstat,

iostat oder sar geliefert wird. Es ist ein Trend ablesbar, daß sich mit steigender Prozeßlast (Warteschlangenlänge) der prozentuale Anteil der Systemzeit an der CPU-Zeit erhöht. Resultierend aus der höheren Priorisierung einiger Systemprozesse ergibt sich daraus eine erhöhte Verzögerung der Nutzerprozesse, zu denen die Benchmarks gehören. Die Stärke der Verzögerung ist weniger von den einzelnen Benchmarks, als vielmehr vom benutzten Rechnersystem abhängig. So zeigt sich beim Rechner Tantalus (Diagramme 5-1 bis 5-4) mit steigender Run-Queue-Länge ein exponentieller Anstieg der Verzögerung, während bei Samson und Silly (5-9 bis 5-16) nur ein etwas steilerer Anstieg des linearen Zusammenhanges von Warteschlangenlänge und Benchmarkverzögerung gegenüber dem theoretischen Erwartungswert zu verzeichnen ist. Beim Rechner Decency ist eine sichere Aussage zu dieser Problematik nicht möglich, da die Messungen mit höherer Prozeßlast so stark von den Lasten der Einzelkomponenten des Systems beeinflußt werden, daß eine isolierte Betrachtung der CPU-Last nahezu unmöglich ist.

Betrachtet man die einzelnen Benchmarks separat für einzelne Testsysteme, so läßt sich feststellen, daß die Benchmarks mit sehr geringen Ein- und Ausgabeanteilen (102.swim, 124.m88ksim) von dieser CPU-Last-bedingten Verzögerung stärker betroffen sind als 101.tomcatv und 126.gcc, die mehr Aktivitäten auf dem I/O-System hervorrufen. Dies ist damit begründbar, daß diese Benchmarks bereits eine gewisse innewohnende Verzögerung durch das Warten auf I/O-Operationen aufweisen, durch die die aus dem Betriebssystem resultierende Verzögerung verdeckt wird.

Ein Einfluß der CPU-Last hinsichtlich der Prozessorauslastung kann bei keinem Benchmark festgestellt werden. Generell sind die Benchmarks so prozessorintensiv, daß bei allen Benchmarks auf den getesteten Systemen eine hundertprozentige Auslastung der (oder einer) CPU erreicht wird. Die Betrachtung der CPU-Auslastung liefert dafür aber wichtige Hinweise zum Einfluß peripherer Systemkomponenten, da mit hohen Lasten bzgl. des Speicher- bzw. des I/O-Systems teilweise sinkende CPU-Auslastungen durch Wartezeiten verbunden sind. Besonders deutlich werden diese Wartezeiten bei den Messungen in den Diagrammen 5-5 bis 5-8 zum Rechner Decency, wo sich die Wartezeiten als Idle-Anteil an der Prozessorauslastung zeigen.

Besondere Beachtung verdient noch der Aspekt der Multiprozessorsysteme. Die Diagramme 5-13 bis 5-16 zu den Messungen auf dem Rechner Samson mit 2 Prozessoren zeigen, daß bei keinem Benchmark eine Beschleunigung durch Parallelisierung gegeben ist. Bei Warteschlangenlängen < 2 benötigen die einzelnen Benchmarks stets in etwa dieselbe Abarbeitungszeit auf dem Niveau der maschinenspezifischen Referenzzeit, unabhängig von der konkreten Warteschlangenlänge. Die CPUs sind dabei nur teilweise ausgelastet. Verallgemeinert heißt dies, daß bei Systemen mit n Prozessoren für Run-Queue-Längen kleiner n als Abarbeitungszeit für einen SPEC-CPU-Benchmark die Zeit benötigt wird, die auf einem solchen System auch mit nur einem Prozessor zur Abarbeitung des Benchmarks allein benötigt wird. Bei Run-Queue-Längen größer n liegt die CPU-Last-bedingte Abarbeitungsverzögerung dann bei Run-Queue-Länge / n, wobei hier noch der anfangs beschriebene vom System abhängige Zusatzanteil für die Prozeßkoordinierung im Betriebssystem hinzukommt. Bei gleichartigen Prozessen, so wie bei SPECrate-Messungen ist jedoch auch eine Beschleunigung durch Data- bzw. Instruction-Caches möglich.

Einfluß der Hauptspeicherlast

Die Hauptspeicherlast kann neben der CPU-Last (Prozeßlast) als die zweite primäre Einflußgröße auf die Meßergebnisse mit den SPEC-CPU-Benchmarks unter Lastbedingungen bezeichnet werden. Die Last ist dabei als solche bezüglich des virtuellen Speichersystems zu betrachten. Es ist dabei generell uninteressant, wie stark der physische Hauptspeicher ausgelastet ist, solange die Speicherforderungen aller Applikationen erfüllt werden können. Signifikante Verzögerungen eines Benchmarks ergeben sich dann, wenn die virtuelle Speicherverwaltung eines Systems Auslagerungen von Speicherseiten auf externen Speicher vornimmt und später auf diese ausgelagerten Speicherbereiche zugegriffen wird, da externe Speicherzugriffe um ein vielfaches länger dauern, als direkte Zugriffe zum physischen Hauptspeicher.
Es ist damit klar, daß der Einfluß der Hauptspeicherlast eng mit der Größe des Hauptspeichers verbunden ist, dessen Auslastung wiederum von der Anzahl der Prozesse und deren Speicheranforderungen abhängt. Dementsprechend sind auf dem Rechner Tantalus (208 MByte physischer Hauptspeicher) nur wenige Speicherauslagerungen beobachtbar, während auf dem Rechner Decency (32 MByte) die virtuelle Speicherverwaltung das leistungsbestimmende Teilsystem darstellt. Auf den beiden anderen Systemen sind differenzierte Page-Out-Aktivitäten registriert worden. Wie stark sich eine bestimmte Page-Out-Rate auf die Verzögerung eines Benchmarks konkret auswirkt, wird im Abschnitt 5.5 zum Lastverhalten der einzelnen Testsysteme diskutiert.
Die Floatingpoint-Benchmarks 101.tomcatv und 102.swim haben beide einen hohen Speicherbedarf von ca. 14 MByte für die von ihnen verwendeten Daten, hinzu kommen noch einige Hundert KByte für den Programmcode. Bei 101.tomcatv besteht eine stärkere Abhängigkeit von der gemessenen Speicherlast als bei 102.swim. Bei den Messungen auf den Rechnern Decency und Silly liegen die registrierten Page-Out-Raten bei 101.tomcatv (Diagramme 5-5, 5-9) deutlich höher als bei 102.swim (Diagramme 5-6, 5-10) und treten auch erheblich häufiger auf.
Die beiden Festkomma-Benchmarks weisen sehr hohe, dynamische Speicherbedürfnisse von bis zu 25 MByte auf. 124.m88ksim braucht diese Speichermenge ständig, während bei 126.gcc der tatsächliche Bedarf zwischen 1,9 und 28 MByte schwankt, wie Beobachtungen der aktuell verwendeten Speichermengen während der Testläufe zeigten.
Dementsprechend ist bei 124.m88ksim eine stärkere Verzögerung der Benchmarkabarbeitung durch Speicherauslagerungen beobachtbar als bei 126.gcc. Beim am stärksten speicherabhängigen Testsystem (Decency) ist dies am deutlichsten erkennbar. Bei allen Messungen mit 124.m88ksim (Diagramm 5-7) werden Auslagerungsraten von 110...120 KByte/s gemessen, die Laufzeit erhöht sich um etwa die zweifache Normallaufzeit. Messungen mit 126.gcc (Diagramm 5-8) liefern Page-Out-Raten von 70...90 KByte/s, die Benchmarkverzögerung erhöht sich dadurch etwa um die einfache Referenzzeit des Benchmarks auf dieser Maschine. Diese geringere Abhängigkeit von 126.gcc läßt sich einerseits mit den kurzen Laufzeiten der speicherintensiven Benchmarkteile begründen. Daneben weist 126.gcc durch die höhere Ein- und Ausgabeintensität eine natürliche I/O-Verzögerung auf, in der Verzögerungen durch Page-Out-Aktivitäten verborgen sein können.
Der Einfluß der Speicherlast ist durch die Nutzung virtuellen Speichers eng mit der im nächstem Abschnitt zu diskutierenden Disk-I/O-Last verbunden.

Einfluß der Disk-I/O-Last

Eine grundsätzliche Beeinflussung der Benchmarkergebnisse durch unterschiedliche Belastungen des lokalen Sekundärspeichersystems konnte bei keinem der verwendeten Benchmarks auf den verschiedenen Testrechnern ermittelt werden.
Schon bei Betrachtung der gegebenen Testbedingungen ist eine generelle Abhängigkeit der Benchmarks von der Last bezüglich des lokalen Disk-I/O-Subsystems auszuschließen. Die Benchmarkabarbeitung an sich erfordert, im Gegensatz zu den üblichen SPEC-Meßbedingungen, theoretisch keinerlei lokale Harddisk-Zugriffe, da alle zur Abarbeitung erforderlichen Quellen und Daten sich auf einem entfernten Fileserver befinden.
Ein indirekte Abhängigkeit von der Disklast besteht in diesem Umfeld möglicherweise dahingehend, daß das Betriebssystem eigene Programme von der lokalen Disk lädt oder Systemdaten dorthin speichert, und daß dadurch die Nutzerprozesse verzögert werden. Da aber die Benchmarks von ihrer Konzeption her mit einer minimalen Menge von einfachen Systemfunktionen arbeiten, die zum permanent im Hauptspeicher befindlichen Systemkern gehören, ist eine Abhängigkeit der Benchmarks von Festplattenzugriffen des Betriebssystems nahezu auszuschließen. Bei den meisten Messungen auf den verschiedenen Testrechnern wurden nur geringe mittlere Disk-I/O-Raten zwischen 1 und 10 KByte/s ermittelt, die mit Sicherheit von anderen Prozessen verursacht wurden.
Höhere Disk-I/O-Raten wurden grundsätzlich nur für solche Testläufe registriert, bei denen Paging- oder Swapping-Aktivitäten aufgetreten sind. Die Laufzeitbeeinflussung durch die virtuelle Speicherbelastung korreliert damit mit der Verzögerung der Zugriffe auf den Auslagerungsbereich der Harddisk gegenüber direkten Hauptspeicherzugriffen.
Zusammenfassend kann festgestellt werden, daß eine generelle, direkte Abhängigkeit zwischen Disk-I/O-Last und Benchmarkverzögerung bei keinem der 4 gewählten Benchmarks besteht. Eine indirekte Abhängigkeit ergibt sich bei hoher Speicherlast durch die Transformation von Hauptspeicherzugriffen in Diskzugriffe durch die virtuelle Speicherverwaltung. Auf Spezifika bei einzelnen Testrechnern wird in der Diskussion des Lastverhaltens der einzelnen Testsysteme eingegangen.

Einfluß der Netz-I/O-Last

Der Einfluß der Ein- und Ausgabelast bezüglich der Netzwerkanbindung ist einerseits vom Ein- und Ausgabeprofil des jeweiligen Benchmarks, andererseits von der ausstattungsbedingten, theoretischen Durchsatzleistung der Netzwerkschnittstelle(n) des Testsystems abhängig.
Die Benchmarks 102.swim und 124.m88ksim weisen eine so minimale Aktivität an Ein- und Ausgaben auf, daß man eine Beeinflussung durch die Netzlast von vornherein ausschließen kann. 101.tomcatv und 126.gcc erfordern mehr Netztransfers zur Bereitstellung benötigter Daten, wobei sie sich in der Intensität und Häufigkeit der Ein- und Ausgaben erheblich unterscheiden. Aus dieser Sicht war anzunehmen, daß, wenn überhaupt eine Abhängigkeit von der Netzlast besteht, diese am stärksten bei 126.gcc zum Tragen kommt. Tatsächlich werden auf allen Testsystemen die höchsten mittleren Netz-I/O-Raten bei 126.gcc und 101.tomcatv registriert. Eine zusätzliche Verzögerung der Benchmarkabarbeitung kann damit jedoch nicht in Verbindung gebracht werden.
Beide getesteten Compute-Server verfügen über jeweils eine FDDI- (100 MBit/s) und eine Ethernet-Netzwerkschnittstelle (10 MBit/s). Die beiden als Workstation genutzten Referenzmaschinen sind über eine 10 MBit/s-Ethernetschnittstelle in das Universitätsnetz eingebunden. Entsprechend der

Anwendungsfelder der verschiedenen Testrechner sind für die Netz-I/O-Rate und die NFS-Call-Rate erhebliche Unterschiede in den typischen Lastprofilen zu verzeichnen.
Auf dem Compute-Server Tantalus (Diagramme 5-1 bis 5-4) sind hohe Netz-I/O-Raten zwischen 100 und 900 KByte/s - bei 126.gcc in Extremfällen bis 2300 KByte/s - zu verzeichnen, wobei die durchschnittlichen Netz-I/O-Raten um 300 KByte/s liegen. Die gemessenen NFS-Call-Raten betragen dabei 3 bis 70 Calls/s, wobei die durchschnittliche Rate in etwa bei 10 Calls/s liegt. Da die gemessenen Netzlasten bei einzelnen Messungen unter verschiedenen Prozeßlastniveaus stark streuen, sich diese Streuungen jedoch nicht auf den Verlauf der Kurve des Zusammenhanges von Prozeßlast und Benchmarkverzögerung auswirken, kann festgestellt werden, daß von den Netzlasten keine allgemeine Verzögerung der Benchmarkabarbeitung ausgeht. Auf dem zweiten Compute-Server im Testfeld, dem Rechner Samson, zeigt sich ein ähnliches Streuungsverhalten. Die registrierten Lasten liegen allerdings auf einem niedrigeren Niveau (Diagramme 5-13 bis 5-16).
Die Begründung für die relative Unabhängigkeit von der Netzlast liefern die Meßreihen auf den als Workstation eingesetzten Rechnern Silly und Decency. Hier liegen sowohl Netz-I/O-Rate als auch NFS-Call-Rate auf konstant niedrigem Niveau (Diagramme 5-5 bis 5-12) mit 20 bis 50 KByte/s bzw. 1 bis 2 NFS-Calls/s. Damit zeigt sich, daß die von den Benchmarks selbst erzeugten Netzlasten gering sind und folglich nur eine minimale Abhängigkeit von „Benchmark-fremden“ Netzlasten bestehen kann. Die als Ausnahmefälle auftretenden hohen Lasten zeigen auch auf diesen Rechnern keine Auswirkungen auf die normale, durch die Prozeßlast bedingte Benchmarkverzögerung.
Bei der Auswertung der Messungen des DEC-Systems wurde bei 102.swim und 126.gcc auf zwei einzelne Messungen hingewiesen, bei denen bei besonders hohen NFS-Call-Raten eine stärkere Benchmarkverzögerung auftrat, als zu erwarten war, ohne daß andere Indizien die Verzögerung begründen konnten. Da ein derartiger Zusammenhang bei den anderen Testsystemen nicht ermittelt werden konnte, ist eine Abhängigkeit der Verzögerung von der NFS-Call-Rate aus nur 2 Einzelfällen nicht ableitbar. Man kann für diese Einzelfälle nur vermuten, daß die von den verschiedenen Prozessen frequentieren NFS-Server bei diesen Messungen besonders belastet waren und es dadurch auch zu einer Verzögerung bei der Erfüllung der wenigen NFS-Requests der Benchmarks kam. Diese Tatsache ist jedoch auf die „globale“ NFS-Last im Gesamtnetz zurückzuführen, und nicht auf das generelle Verhalten eines Testsystems oder eines Benchmarks. Solche Lastspitzen kommen im täglichen Rechenbetrieb als Ausnahmesituationen vor, daraus sollte aber nicht auf generelle Abhängigkeit geschlossen werden.
Ein weiteres Indiz für die Unabhängigkeit der untersuchten Benchmarks von der Netzlast liefert die Betrachtung der durch die Meßwerkzeuge zusätzlich bereitgestellten Fehlerraten zu den betrachteten Netzwerkschichten. Obwohl in Einzelfällen bis zu 40% Paketkollisionen im Ethernet und bis zu 10% Neuanforderungen (Retransmissions) des NFS-Protokolles registriert wurden, ergaben sich auch bei diesen Messungen keine zusätzlichen Verzögerungen für die Benchmarkabarbeitung, auch wenn in [Louk91] 10% Kollisionen und 5% Retransmissions als maximal akzeptable Fehlerraten für ein ausgewogenes Systemverhalten angegeben werden.
Abschließend kann man aus der Aussage zur Unabhängigkeit von der Netzlast eine weitere zu möglichen Abhängigkeiten unter den üblichen SPEC-Bedingungen ableiten. Unter diesen befinden sich Programmcode und Daten normalerweise auf der lokalen Harddisk des Testsystems. Damit reduziert sich die Zugriffszeit auf die Programme und Daten um die Netzübertragungszeiten auf die Zeiten der lokalen Festplattenzugriffe, die unter den Testbedingungen dieser Arbeit beim entfernten Fileserver erfolgen. Da hier bereits der Einfluß der Zugriffe über das Netzwerk als äußerst gering ermittelt wurde, ist der Disk-I/O-Einfluß unter originalen SPEC-Bedingungen ebenfalls als minimal einzuschätzen. Diese

Annahme deckt sich mit den pauschalen Aussagen in SPEC-Veröffentlichungen, daß der allgemeine Einfluß des Ein-/Ausgabesystems sehr gering ist [SPEC95_3], [SPEC_DES.x]. In Verbindung mit den Feststellungen zum nicht vorhandenen Einfluß des Disk-I/O-Systems im vorangegangenen Abschnitt, kann diese Aussage durch die durchgeführten Testreihen experimentell bestätigt werden.

Abschätzungen zum Einfluß anderer Elemente eines komplexen Rechnersystems

Bereits bei der Betrachtung der Meßbedingungen und der anschließenden Auswahl von Lastmaßen zur Ermittlung der Einflüsse wurde auf Elemente eines Rechnersystems hingewiesen, von denen mit Sicherheit ein Einfluß auf die Benchmarkergebnisse und damit die Systemleistung ausgeht, deren Einfluß jedoch mit der Messung einzelner Lasten nicht bestimmbar ist. Dazu gehören das Betriebssystem, die auf das gesamte Rechnersystem abgestimmten Compiler, sowie die Daten- und Befehlscaches als erste Stufe der komplexen Speicherhierarchie. Die Betrachtung der allgemeinen Tendenzen beim Lastverhalten der verschiedenen Testrechner erlaubt es jedoch, zumindest einige Vermutungen zum Einfluß dieser Systemkomponenten aufzustellen.
Der Einfluß der verschiedenen Betriebssysteme zeigt sich weniger am Verhalten der einzelnen Benchmarks, sondern vielmehr am Lastverhalten der jeweiligen Rechnersysteme. So liegt die Vermutung nahe, daß der exponentielle Anstieg der Benchmarkverzögerung bei steigender Prozeßlast beim HP-PA-RISC-Rechner (Tantalus) auf interne Spezifika des Betriebssystems zurückzuführen ist. Beim SGI-Mips-R4000-System (Silly) unter IRIX zeigen sich hingegen besondere Effekte bei der Betrachtung der Last hinsichtlich der virtuellen Speicherverwaltung. Hier lassen sich Ursachen in der systeminternen Prozeßverwaltung oder aber in den zum System gehörigen Meßtools vermuten.
Im Abschnitt 1.2.1 wurde darauf hingewiesen, daß der Einfluß der Cache-Speicher in der gesamten Speicherhierarchie mit einem Softwaremonitor nicht direkt meßbar ist. Betrachtet man die Meßreihen für die einzelnen Testsysteme, so lassen sich einzelne positive Abweichungen der Benchmarkverzögerungen erkennen, die von den ermittelten Werten für die betrachteten Teillasten unabhängig sind. Bei diesen kann man auf einen Cache-Einfluß spekulieren. Betrachtet man die Messungen auf dem Rechner Tantalus und die wenigen nicht von der virtuellen Speicherverwaltung betroffenen Messungen auf dem Rechner Decency, so läßt sich generell feststellen, daß die registrierten Verzögerungen in Höhe des Erwartungswertes oder noch etwas höher liegen. Diese beiden Rechner sind nur mit vergleichsweise kleinen Caches (Primärcaches) ausgestattet (Abschnitt 4.2). Die beiden anderen Testrechner verfügen neben den kleinen Primärcaches über einen weiteren Sekundärcache mit einer Größe von 1 MByte. Bei diesen Rechnern werden bei allen Benchmarks jeweils einige Testläufe registriert, bei denen die gemessene Verzögerung unter dem prozeßlastbedingten Erwartungswert lag. Beim SPARC-Multiprozessorsystem Samson mit 1 MByte Sekundärcache pro CPU lag der Verzögerungsfaktor teilweise bis zu Wert 1 niedriger als der Erwartungswert. Daß diese „Beschleunigung“ nicht bei allen Messungen ermittelt werden konnte, bestätigt nur die Vermutung, daß die Caches die Ursache bilden, da gerade Cache-Trefferraten stark vom jeweils völlig unterschiedlichen Gesamtzustand des Prozeßumfeldes abhängen.

5.5 Zusammenfassende Betrachtung des Lastverhaltens der Testrechner

Ausgangspunkt für die Auseinandersetzung mit den SPEC-Benchmarks an der TU Chemnitz-Zwickau war das Ziel, mit diesen Meßinstrumentarien Aussagen

- zur allgemeinen Leistungsfähigkeit einzelner Universitätsrechner,
- zur besonderen Eignung einzelner Rechnerplattformen für typische Applikationen,
- zum Einfluß bestimmter Applikationen auf das Gesamtverhalten einzelner Rechner,
- zur Erkennung von Engpässen und dementsprechend notwendigen Erweiterungen

zu gewinnen. Unter diesem Aspekt erfolgen hier abschließend Betrachtungen zum spezifischen Verhalten der bei den Messungen untersuchten Testsysteme. Zur Ausstattung der einzelnen Testrechner wird auf Abschnitt 4.2 verwiesen.

5.5.1 Compute-Server HP 9000 S.700 Modell 735 (tantalus.hrz.tu-chemnitz.de)

Dieser Rechner wird im Universitätsrechenzentrum der Technischen Universität Chemnitz als Compute-Server eingesetzt. Dementsprechend wird er von einer Vielzahl von Nutzern mit verschiedenen Applikationen belastet. Dabei sind neben vielen kurzzeitigen Nutzungen, die zu bestimmten Spitzenzeiten auftreten, stets eine Reihe von über einen längeren Zeitraum (mehrere Stunden oder Tage) laufenden Applikationen zu verzeichnen. Hierbei handelt es sich häufig um numerische Anwendungen von Studenten und Mitarbeitern verschiedener Fachbereiche, die die Hauptlast auf diesem Rechner darstellen. Entsprechend den hohen Ressourcenforderungen vieler wissenschaftlicher Anwendungen ist der Rechner mit einer aufwendigen Ausstattung versehen, wichtigste Elemente hierbei sind 208 MByte Hauptspeicher und ein 100MBit-FDDI-Netzwerkinterface. Der Prozessor HP-PA-RISC 7150 erreicht die höchsten SPECratio der betrachteten Rechner. Eine SPEC-Veröffentlichung der offiziellen Benchmarkresultate erfolgte für eine solche Maschine in [SPEC95_3], wegen der unterschiedlichen Versionen von Betriebssystem und Compiler sind die Ergebnisse jedoch nur bedingt vergleichbar.
Der Rechner zeigt unter verschiedenen Lastbedingungen bei allen Benchmarks ein ähnliches Verhalten. Mit steigender Prozeßlast ist eine leicht exponentiell steigende Verzögerung bei der Abarbeitung der Benchmarks als repräsentative Beispielapplikationen zu beobachten. Ein signifikanter Anstieg der Verzögerung tritt in etwa ab Warteschlangenlängen von 6 Prozessen auf, vorher sind nahezu lineare Anstiege knapp über den Erwartungswerten zu verzeichnen. Diese wachsende Verzögerung ist mit dem zusätzlichen Aufwand für die Verwaltung der Prozesse im Betriebssystem begründbar. Indiz hierfür ist der im Mittel wachsende Anteil der Prozessorzeit, die Systemprozessen zugeteilt wird.
Ein Beeinflussung der Benchmarkverzögerung durch die anderen betrachteten Systemkomponenten konnte nicht ermittelt werden. Aufgrund des großen Hauptspeichers treten auch bei hohen Prozeßlasten kaum Speicherauslagerungen durch Paging/Swapping auf, bei den wenigen Einzelfällen mit allerdings niedrigen Page-Out-Raten ist eine dadurch bedingte Verzögerung nicht erkennbar. Bedingt durch die Struktur des Filesystems der TU Chemnitz (`/uni/global`) sind die meisten Programme und Daten auf zentralen Fileservern gespeichert. Zusammen mit den wenigen Speicherauslagerungen werden daher

nur minimale Transferraten zu den lokalen Harddisks registriert. Ein eventuell bremsender Einfluß des lokalen Disk-Subsystems ist damit nicht nachweisbar.
Bei den hohen Nutzerzahlen des Rechnersystems sind um so mehr Netztransfers zu registrieren. Die approximierten Netz-I/O-Raten stellen sowohl hinsichtlich der Durchschnittsraten von 300 bis 400 KByte/s als auch hinsichtlich der Spitzenwerte von bis zu 3 MByte/s bei den gcc-Messungen die höchsten Werte aller Vergleichsrechner dar. Diese Netzlasten werden vom System mit FDDI- und Ethernet-Interface offensichtlich problemlos bewältigt. Ein Einfluß auf die gemessenen Verzögerungen ist weder bei Maximal- noch bei Minimallasten erkennbar.
Allenfalls ist es möglich, daß durch das hohe Grundniveau an Netztransfers eine gewisse Verzögerung bei allen Messungen auftritt. Dies könnte erklären, warum die Verzögerungen auch im niedrigen Prozeßlastbereich geringfügig, jedoch beständig, über den Erwartungswerten liegen. Da die NFS-Call-Raten ebenfalls starken Schwankungen unterliegen, ohne daß sich dies in den Verzögerungen repräsentiert, kann man zusammenfassend feststellen, daß eine Beeinflussung der erbrachten Rechenleistungen des Rechners durch die überlicherweise zu bewältigenden Netzlasten nicht feststellbar ist.
Insgesamt verhält sich der Rechner Tantalus unter den betrachteten Bedingungen völlig erwartungskonform, die Laufzeit bestimmter Applikationen ist unter normalen Bedingungen nur von der Prozeßlast abhängig. Die Einhaltung von Qualitätsansprüchen hinsichtlich der Laufzeit ist damit an eine maximal zulässige Anzahl aktiver Prozesse gebunden, diese Tatsache kann z.B. künftig bei einer dynamischen Lastverteilung mit dem Batch-System DQS[9] ausgenutzt werden.

5.5.2 Workstation DEC 3000-300 AXP (decency.hrz.tu-chemnitz.de)

Aus den Meßreihen zu diesem als Workstation eingesetzten Rechner muß abgeleitet werden, daß Systeme mit einer Speicherausstattung von 32 MByte zur Anwendung für rechenintensive Applikationen, wie sie die SPEC-CPU-Benchmarks repräsentieren, auch im reinen Workstationbetrieb ungeeignet sind.
Der 21064-Alpha-Prozessor liefert auf Basis der reinen Prozessorzeiten durchaus gute SPECratio-Werte für die betrachteten Benchmarks, wobei die approximierten SPECratio für die Fließkommatests deutlich besser sind (2,38 für 101.tomcatv, 3,24 für 102.swim) als die für die Integertests, die nur knapp über denen der SPEC-Referenzmaschine liegen (1,12 für 124.m88ksim, 1,06 für 126.gcc).
Bis auf wenige Fälle ohne fremde Prozeßlast bei den beiden Fließkomma-Benchmarks kommt die eigentliche Prozessorleistungsfähigkeit nicht zum tragen. Aufgrund des nahezu permanenten Hauptspeichermangels wurden ständige Paging-Aktivitäten registriert, die eine starke Verzögerung der Benchmarkabarbeitung hervorrufen. Das ständige Warten auf die Aus- und Einlagerung von Speicherseiten bewirkt ein Absinken der tatsächlichen CPU-Auslastung auf teilweise nur noch 20%. Bei steigender Prozeßlast steigt die CPU-Auslastung dadurch, daß weniger speicherintensive Prozesse die Paging-bedingten Wartezeiten der Benchmarkprozesse ausnutzen können.
Die Graphen zu den Meßreihen zeigen bei 101.tomcatv eine direkte Abhängigkeit der Benchmarkverzögerung von der Page-Out-Rate. Bei den anderen Benchmarks wurden für alle Messungen jeweils etwa gleiche Speicherauslagerungsraten registriert. Daraus resultieren entsprechend konstante Erhöhungen des Verzögerungsfaktors gegenüber dem Erwartungswert. Diese Erhöhungen

[9] Distributed Queuing System

liegen in etwa bei Wert 2 für 100 KByte/s bei 124.m88ksim und bei Wert 1 für 100 KByte/s bei 126.gcc.
Die Meßwerte für die mittleren Disk-I/O-Lasten korrespondieren in ihrem Kurvenverlauf direkt mit der Speicherauslagerungsrate. Die Page-Out-Rate wird damit in die Disk-I/O-Last transformiert, die ebenfalls für viele Messungen auf nahezu gleichem Niveau um 500 bis 600 KByte/s liegt, nur bei 101.tomcatv erreichen die Disk-I/O-Raten bis zu 850 KByte/s.
Die relativ konstanten Page-Out-Raten und Disk-I/O-Raten für die meisten Tests erlauben die Vermutung, daß das Disk-I/O-System das zweite leistungslimitierende Teilsystem bei diesem Rechner neben dem Hauptspeicher darstellt. Durch die geringe Speicherbestückung ausgelöste Paging-Aktivitäten bewirken konstant hohe Disk-I/O-Raten über Zeiträume von mehreren Stunden hinweg, die durch die Intensität der Zugriffe im Grenzbereich der Disk-I/O-Leistung liegen.
Durch die starke Beeinflussung der Benchmarklaufzeit durch die Speicherauslagerungsrate und die damit verbundene Disk-I/O-Rate, ist es auf diesem Testsystem nicht möglich, aus den allgemein geringen Netzlasten eine Abhängigkeit der Benchmarkverzögerung von den vorhandenen Netzlasten abzuleiten.
Eine deutliche Leistungsverbesserung ist für dieses System bei einer Aufrüstung des Hauptspeichers zu erwarten. Die als nächste zu betrachtende Workstation von Silicon Graphics verfügt über nur die anderthalbfache Hauptspeichergröße, zeigt aber auch bei hohen Prozeßzahlen ein deutlich ausgewogeneres Lastverhalten mit deutlich geringerer Paging-Aktivität. Der Einfluß der Realisierungen innerhalb des Betriebssystems liegt dabei außerhalb der Bewertungen. Mit einem Rückgang der Speicherauslagerung wäre eine adäquate Reduktion der Disk-I/O-Last verbunden.

5.5.3 Workstation SGI RW420-XS IRIS Indigo - R4000 (silly.hrz.tu-chemnitz.de)

Zur Benutzung als Workstation mit Applikationen, die den Anwendungsprofilen der verwendeten Benchmarks entsprechen, ist dieses System geeignet und auch ausreichend ausgestattet. In SPEC-Veröffentlichungen [SPEC_doc] werden 64 MByte Hauptspeicher für einen sinnvollen Einsatz der Benchmarks vorausgesetzt. Die Tests auf diesem Rechner haben gezeigt, daß zumindest für die 4 ausgewählten Benchmarks die 48 MByte dieser Rechnerkonfiguration für eine verzögerungsfreie Anwendung ausreichen. Diese Aussage läßt sich damit auf ähnliche numerische Applikationen übertragen. Für den für eine Workstation typischen Betrieb mit einem aktiven Nutzer ist ein homogenes Systemverhalten zu erwarten.
Die auf Basis der reinen Prozessorzeiten approximierten SPECratio liegen für die einzelnen Benchmarks zwischen 1 und 3,5 (Tabelle 4-2 im Abschnitt 4.3). Selbst bei einer Prozeßlast von zu 10 Prozessen in der Prozeßwarteschlange liegt die Benchmarkverzögerung noch auf Erwartungswertniveau. Ein durch die Prozeßlast bedingter Leistungsabfall wie beim HP-PA-RISC-System kann nicht festgestellt werden. Der Prozessor ist bis auf wenige Ausnahmen (bei hohen Speicherauslagerungsraten) vollständig ausgelastet.
Im niedrigen Prozeßlastbereich liegt die Verzögerung teilweise sogar niedriger als der aus der Warteschlangenlänge gebildete Erwartungswert. In vorangegangen Abschnitten wurde bereits vermutet, daß diese „Beschleunigung“ auf die nicht direkt bewertbaren Cache-Einflüsse zurückführbar sein könnte, da dieses System neben jeweils 8 KByte Daten- und Befehlscache über einen 1 MByte-Sekundärcache verfügt.

Messungen mit Aktivitäten der virtuellen Speicherverwaltung zeigen generell eine Abweichung des Zusammenhangs von Benchmarkverzögerung und Run-Queue-Länge vom Erwartungswert. Dabei ist jedoch ein interessanter Effekt zu registrieren. Normalerweise liefern Messungen mit Paging Verzögerungswerte, die über dem Erwartungswert liegen, da Harddisk-Zugriffe mehr Zeit beanspruchen als Hauptspeicherzugriffe. Bei diesem System wird das Gegenteil ermittelt, die Verzögerung liegt unter dem Erwartungswert für die gemessene Prozeßlast. Über die Ursache dieses Effektes kann nur spekuliert werden. Möglicherweise unterscheidet sich der Verwaltungsmechanismus für die zu bearbeitenden Prozesse in diesem IRIX-System von den anderen UNIX-Systemen im Testfeld dahingehend, daß auch Systemprozesse mit höherer Priorität in die normale Warteschlangenlänge einfließen, oder daß das verwendete Meßwerkzeug *uptime* über den rstat-Daemon solche Warteschlangenlängen ermittelt, die durch die Systemprozesse der Speicherverwaltung verfälscht wurden. Generell zeigt sich bei allen Benchmarks ein solcher Einfluß von Page-Out- und Swap-Out-Aktivitäten. Der Einfluß der Auslagerungsraten kann daher nicht quantifiziert werden. Er ist jedoch gegenüber dem DEC-System geringer einzuschätzen, da kaum Leerlaufzeiten des Prozessors resultieren und die interaktive Arbeit nur unwesentlich behindert wird. Mit zunehmender Prozeßlast ist bei speicherbeeinflußten Tests ein Abnehmen der Abweichung vom Erwartungswert zu registrieren, da die Wartezeit auf das Laden ausgelagerter Speicherseiten schon zum Teil in die Wartezeit für die Prozessorzuteilung einfließt.
Die gemessenen Lasten auf dem Disk-I/O-System korrespondieren wiederum mit den registrierten Aktivitäten der Hauptspeicherverwaltung. Andere Disk-I/O-Lasten konnten nur auf einem sehr geringen Niveau gemessen werden. Ein Verzögerung der Benchmarks durch diese ist auszuschließen.
Die gemessenen Netz-I/O-Raten und NFS-Call-Raten liegen allgemein auf dem niedrigsten Niveau aller bewerteten Systeme, dies bestätigt zunächst die SPEC-Aussagen zum geringen I/O-Anteil innerhalb der Benchmarks. Daraus resultierend zeigt sich bei den Messungen, bei denen überdurchschnittliche Netzlasten ermittelt wurden, daß von diesen keine zusätzliche Verzögerung der Benchmarkabarbeitung ausgeht.

5.5.4 Compute-Server SPARCstation 20 Modell 612 (samson.hrz.tu-chemnitz.de)

Dieses an der TU Chemnitz als Compute-Server betriebene Multiprozessorsystem zielt aus Konfigurationssicht eher auf eine gute Durchsatzleistung als auf eine besonders schnelle Abarbeitung von Applikationen im Einzelnutzerbetrieb. Dementsprechend liegen die auf Basis der Prozessorzeiten ermittelten SPECratio nur geringfügig besser als die der DEC-Workstation (Decency), jedoch deutlich unter denen des HP-PA-RISC-Systems (Tantalus).
Generell folgt die Benchmarkverzögerung bei steigender Prozeßlast dem Erwartungswert. Dieser bestimmt sich für dieses 2-Prozessor-System als Produkt der halben Warteschlangenlänge mit der maschinenspezifischen Referenzzeit. Bei Warteschlangenlängen kleiner 2 ist die maschinenspezifische Referenzzeit als Abarbeitungszeit zu erwarten, da die Benchmarks nicht im Hinblick auf Parallelisierbarkeit entworfen wurden, und die benutzten Compiler Sun C 3.0.1 bzw. Sun Fortran 3.0.1 keinen „parallelen Code“ erzeugen. Dementsprechend liegt die Prozessorauslastung bei Run-Queue-Längen < 2 unter 100%, während bei höheren Prozeßlasten eine permanente Prozessorauslastung gegeben ist.
Bei einer ganzen Reihe von Messungen liegt die tatsächliche Verzögerung niedriger als der Erwartungswert. Noch einmal sei für diese „Beschleunigung“ auf eventuelle Cache-Einflüsse verwiesen,

die mit den hier verwendeten Meßmethoden nicht nachweisbar sind. Jedoch verfügt auch dieses System neben 20 KByte Befehls- und 16 KByte Datencache als Primärcache pro CPU über jeweils weitere 1 MByte Sekundärcache (Befehle und Daten).
Mit einer Hauptspeicherbestückung von 192 MByte sollte, ähnlich wie beim HP-System, Unabhängigkeit von Speicherlasteinflüssen herrschen. Tatsächlich werden aber bei den Testreihen zu allen Benchmarks auch im unteren Prozeßlastbereich mittlere Page-Out-Raten von bis zu 400 KByte/s gemessen. Dies bewirkt auf diesem System Erhöhungen der Verzögerungen etwa um Wert 0.5 für 100 KByte/s Auslagerungsrate gegenüber den typischen Verzögerungen bzw. dem Erwartungswert. Im Gegensatz zum HP-Compute-Server werden bei diesem System bei fast allen Messungen Speicherauslagerungen registriert, auch wenn die mittleren Raten in vielen Fällen unter 1 KByte/s liegen. Diese geringen, aber regelmäßigen Seitenauslagerungen liefern die Erklärung, warum bei diesem System die interaktive Arbeit häufig merklich verzögert wird. Da typische Paging-Strategien die am längsten nicht benutzten Seiten auslagern, ist die Wahrscheinlichkeit hoch, daß davon zuerst solche Prozesse betroffen sind, die mit langsamen Eingabegeräten verbunden sind oder auf Eingaben warten. Dies sind eben interaktive Prozesse, wie Shells oder Programme mit Eingabeaufforderungen.
Die gemessenen Netzraten liegen im Vergleich zum HP-System im Mittel deutlich niedriger. Netz-I/O-Raten bis 100 KByte/s und NFS-Call-Raten unter 10 Calls/s werden von zwei Netzwerkschnittstellen problemlos bewältigt. Auch aus den wenigen, bei allen Testreihen vorkommenden, höheren Netzraten lassen sich keine Verbindungen zu einer daraus resultierenden Verzögerung herstellen.
Beim Benchmark 126.gcc ist auf diesem Rechner, abweichend von den anderen Benchmarks, der Effekt zu verzeichnen, daß die Verzögerung mit zunehmender Prozeßlast exponentiell wächst. Eine Erklärung ist mit den betrachteten Lastmaßen nicht möglich. Diese liefern ähnliche Werte wie für die anderen Benchmarks. Man kann über Ursachen im Betriebssystem spekulieren, wenn man andere Werte betrachtet, die von den Meßtools geliefert werden. So haben die zusätzlichen Betrachtungen aller Meßergebnisse ergeben, daß von der Prozessorzeit auf diesem System im Mittel 25 % auf Systemprozesse entfallen, während es bei den anderen Benchmarks nur maximal 10 % sind. Dabei ist tendenziell ein weiteres Ansteigen des Systemanteils bei wachsender Prozeßlast festzustellen. Der höhere Systemzeitanteil läßt sich mit der höheren Ein- und Ausgabeintensität des 126.gcc begründen. Wegen der häufig höheren Priorisierung von Systemprozessen kann man darauf die stärkere Verzögerung der Nutzerprozesse zurückführen.
In den Netz- oder Disk-I/O-Lasten ist jedoch zu den anderen Benchmarks keine Abweichung durch das besondere Ein- und Ausgabeverhalten erkennbar.

6 Fazit und Ausblick

In dieser Arbeit wurden mögliche Einflußfaktoren auf die Ergebnisse der SPEC-CPU-Benchmarks CFP95 und CINT95 in praktischen Messungen untersucht.
Durch die Betrachtung der unterschiedlichen Meßbedingungen bei Rechnerherstellern und an der TU-Chemnitz wurden Prozessor, Hauptspeichersystem, Disk-I/O-System und Netz-I/O-System als die Teilsysteme eines Rechnersystems bestimmt, die unter den Meßbedingungen an der TU Chemnitz-Zwickau möglicherweise einen Einfluß auf die Ergebnisse der Benchmarks unter verschiedenen Belastungszuständen ausüben.
Unter Berücksichtigung der charakteristischen Eigenschaften und Vorgänge in diesen Teilsystemen bei UNIX-Rechnern wurden Lastmaße erarbeitet, mit denen die jeweilige Belastung des Teilsystems über bestimmte Zeiträume hinweg bewertet werden kann.
Mit auf den gewählten Testrechnern der TU Chemnitz-Zwickau verfügbaren UNIX-Standardwerkzeugen wurde ein Lastmonitor entworfen, der es erlaubt, die mittleren Werte für diese Lastmaße während eines Benchmarklaufes zu bestimmen. Auf die Nutzung systemspezifischer Werkzeuge wurde bewußt verzichtet, um möglichst vergleichbare Werte für die Lastmaße für ein möglichst breites Rechnerspektrum bestimmen zu können.
Im experimentellen Teil der Arbeit wurden mit 4 ausgewählten Einzelbenchmarks der CINT95- und CFP95-Suiten Leistungsmessungen auf 4 verschiedenen UNIX-Rechnersystemen der TU Chemnitz-Zwickau durchgeführt, bei denen mit einem Prototyp des entworfenen Lastmonitors die Gesamtlast im Rechnersystem erfaßt wurde.
Die Auswertung der umfangreichen Meßreihen erfolgte einerseits im Hinblick auf Einflüsse auf die einzelnen Benchmarks als Repräsentanten bestimmter Applikationsklassen und auf generelle Abhängigkeiten von den betrachteten Teilsystemen. Auf der anderen Seite wurden Aussagen zum Verhalten der unterschiedlichen Rechnerplattformen unter verschiedenen Lastbedingungen und im Hinblick auf die Einsatzgebiete der einzelnen Rechner an der Universität gewonnen. Dabei konnten Unterschiede im Verhalten der Testsysteme unter verschiedenen Lastbedingungen ermittelt werden, sowie auch einzelne Ressourcenengpässe auf bestimmten Systemen.
Die Betrachtung der Lasten auf den ausgewählten Teilsystemen eines Rechners ergab, daß von der CPU-Belastung (Prozeßlast) und der Belastung des virtuellen Hauptspeichersystems ein signifikanter Einfluß auf die Leistungswerte der Benchmarks ausgeht. Eine Abhängigkeit von der Disk-I/O-Last konnte unter den Bedingungen an der Universität nur als indirekte Abhängigkeit in Verbindung mit Auslagerungsaktivitäten der virtuellen Speicherverwaltung festgestellt werden. Trotz stark differenzierter Ein- und Ausgabeprofile der verwendeten Einzelbenchmarks konnte ein Einfluß verschiedener Lasten auf dem Netzwerk-I/O-System unter normalen Bedingungen nicht ermittelt werden.
Zusammenfassend läßt sich feststellen, daß es auch mit den verwendeten einfachen UNIX-Werkzeugen in Verbindung mit den SPEC-CPU-Benchmarks möglich ist, Aussagen zum Leistungs- und Lastverhalten von UNIX-Rechnern zu gewinnen, die über Aussagen der SPEC-Messungen auf isolierten Systemen hinausgehen. Dies betrifft insbesondere Durchsatzleistungen in Bereichen höherer Systemlast, die bei den SPEC-Durchsatzmessungen (SPECrate) keine Berücksichtigung finden.

Perspektivisch sollte es bei Verwendung eines effektiven Werkzeuges (zum Beispiel Perl in Verbindung mit Tcl/Tk) zur Datenauswertung und -aufbereitung einfach möglich sein, aus dem Prototyp des Lastmonitors ein universelles Tool zur Erfassung eines breiten Spektrums der Rechnerlast zu entwickeln. Durch einen ständigen Einsatz eines solchen Lastmonitors eröffnen sich dann auch Möglichkeiten für weitere Anwendungsgebiete.
Mit dem gewählten Auswertungsverfahren könnte der Einfluß verschiedener Lasten auf die Abarbeitung beliebiger Anwendungen ermittelt werden. Dies könnte eingesetzt werden, um zu prüfen, wie sich der Einsatz bestimmter Softwareprodukte auf das gewöhnliche Lastverhalten des Einsatzrechners auswirkt und ob für den Einsatz eines solchen Produktes für die zu erwartenden Lastbedingungen eine Erweiterung der Hardwarekonfiguration erforderlich ist. Für eine dynamische Lastverteilung, zum Beispiel mit dem Batch-System DQS, bietet sich ein Einsatz zur Ermittlung von Grenzbereichen der maximal verarbeitbaren Last für einzelne Rechner an.

Anlage A -

Änderungen in der SPEC- Ablaufumgebung

Zentrales Element der SPEC-Ablaufsteuerung ist das Perl-Script „runspec“ im Verzeichnis $SPEC/bin. Hier wird der Ablauf eines kompletten Benchmarks vollständig gesteuert. Dazu wird eine Reihe von Funktionsbibliotheken importiert, die die einzelnen Teilfunktionen der Suite realisieren. Dazu gehört auch die Funktionsbibliothek „benchmark.pm“ im Verzeichnis $SPEC/bin. Sie enthält eine Sammlung aller Funktionen, die den direkten Ablauf einer Messung (Vorbereiten von Verzeichnissen, Zeitmessung, Ergebnisvergleich, Reporterstellung) steuern. Dort wird u.a. die Funktion „run_benchmark“ definiert. In dieser wurden für die Messungen zu dieser Arbeit einige Änderungen vorgenommen, die nachfolgend dokumentiert sind. Mit diesen Änderungen ist es möglich, auch die tatsächlich zugeteilten Prozessorzeiten für den Benchmarkprozeß und alle Kindprozesse, jeweils im USER-Mode und im SYSTEM-Mode zu messen. Außerdem wurde die Abarbeitungsfolge geringfügig modifiziert, um die gemessenen Zeiten sofort im „monitor_post_bench“-Abschnitt nachbearbeiten und auswerten zu können. Die gemachten Änderungen sind im folgenden Listing der Funktion „run_benchmark“ in *kursiver* Schrift dargestellt und in durch ### gekennzeichnete Kommentare eingeschlossen.

```
sub run_benchmark {
    my ($me, $size, $tune, $ext, $mach, $users, $fdocommand) = @_;
    my ($return_code) = 1;
    my ($i, @start_time, @stop_time, @diff_time, $name, $pid, $action, $rc);
    my (@exitval, @num_targets, @todo, %children, $dir, $run_num);
    my ($str, @temp, @temp2, @submit_commands, $dir, %config, $count, $submit);
    my $benchtop  = $me->path;
    my $benchname = $me->name;
    my $benchnum  = $me->num;
    my $benchmark = $me->benchmark;
    my $spectop = $'spec{'top'};
    my $count = 0;
    my $command = "";
    my $temp;
    my $wait_pid;
    my $noisy = ($'user{'teerunout'} =~ m/^y(es)?/i) &&
                        ($'user{'teeout'} =~ m/^y(es)?/i);
    ;

### Modification by Carsten Mund (cmu)
###  I need one file more for saving usertime and systime for benchmark process
### OLD LINE:        local (*TIMEFILE, *FILE);
    local (*TIMEFILE, *FILE, *TIMEFILE_cmu);
### End (cmu)

    %config = &'build_config($me->benchmark, $tune, $ext, $mach);
    for (sort &'key_match("submit*", \%config)) {
            push (@submit_commands, $config{$_});
    }
    $submit = 0;

    $i = 0;
    my (%repl) = (
                            'statement' => \$count,
                            'command'   => \$command,
                            'dir'       => \$dir,
                            'user'      => \$i,
                            'numusers'  => $users,
                            'benchmark' => $benchmark,
                            'benchtop'  => $benchtop,
                            'benchnum'  => $benchnum,
                            'benchname' => $benchname,
                            'spectop'   => $spectop,
                          );

# Build a list of what to do in each directory and clobber any output files
    for ($i = 0; $i < $users; $i++) {
            $dir = $me->{'work'}{'dirs'}[$i];
            chdir $dir;
            unlink $me->outputs($size);
            @temp = $me->invoke($size, $ext);
            @temp2 = ();
            $count = 0;
            for (@temp) {
                $command = $_;
                if (defined $fdocommand) {
                        $command = &'subst(\%repl, $fdocommand);
                }
                if (defined $config{'monitor_wrapper'}) {
                        $command = &'subst(\%repl, $config{'monitor_wrapper'});
                }
                if (@submit_commands && $'spec{'mode'} eq "rate") {
```

```
                $command = &'subst(\%repl, $submit_commands[$submit]);
            }
            push (@temp2, $command);
            $count++;
        }
        @{$todo[$i]} = @temp2;
        $num_targets[$i] = @temp2;
        &'log(20, "$benchmark user $i\n");
        for (@temp2) {
            &'log(21, "    $_\n");
        }
        $submit = 0 if ++$submit > $#submit_commands;
    }

# Open up the exitval file now so that if we fail somewhere we at least have
# a partial file in the directory
    $run_num = (&'find_biggest_file($me->work, $me->name."_".$'spec{'timefile'}))[3];
    $name = &'joinpaths($me->work,$me->name."_".$'spec{'timefile'}.".$run_num");
    if (open  (TIMEFILE, ">$name")) {
        print TIMEFILE $me->benchmark, "\n";
    } else {
        &'log(0, "Can't write to '$name': $!\n");
    }

### Modification by Carsten Mund (cmu)
### Need this File for saving the real processor times used by benchmark in usermode and
### in sysmode
### I will do this in the same way like SPEC is  saving the  runtime
$name_cmu = &'joinpaths($me->work,$me->name."_"."usertime".".$run_num");
    if (open  (TIMEFILE_cmu, ">$name_cmu")) {
        print TIMEFILE_cmu $me->benchmark, "\n";
     }
    else {
        &'log(0, "Can't write to '$name_cmu': $!\n");
    }
### End (cmu)

    chdir $'spec{'top'};
    if ($config{'monitor_pre_bench'}) {
        my $command = &'subst(\%repl, $config{'monitor_pre_bench'});

### Modification by Carsten Mund (cmu)
### I need the $run_num in the the pre- and post- monitors
### I put the number  as a shell export to the monitor command
### to prevent from conflicts I put my name to the name of the variable
        $cmu_adding = "carsten_num=$run_num ; export carsten_num ;";
        my $command=join (' ', $cmu_adding, $command);
###  I Can use „$carsten_num“ in the „monitor_pre_bench“-section now
### End (cmu)
        system $command;
    }
# Start off first job for each directory
# We fork off a job based on what was returned by the 'invoke command.
# If it's an array we pass it to exec as an array so we don't incur a shell
# If it's a scalar we pass it as a scalar, this will invoke the system's
# /bin/sh equivalent.

###  Modification by Carsten Mund (cmu)
### Get a vector of 4 elements containing CPU time in user mode and system mode
### for this process and all children of this process (values at start of  a benchmark)
    @start_time_cmu = times;
### End (cmu)

    @start_time = &'spectime;
    for ($i = 0; $i < $users; $i++) {
        chdir $me->{'work'}{'dirs'}[$i];
        $action = shift(@{$todo[$i]});
        $fork_time = time;
        if ($pid = fork) {
            $children{$pid} = $i;
            $usedpid{$pid} = 1;
            if ($noisy) {
                if (ref($action) eq "ARRAY") {
                    print join (" ", @$action), "\n";
                } elsif (ref($action) eq "CODE") {
                    print "perl code\n";
                } else {
                    print $action, "\n";
                }
            }
        } elsif (defined $pid) {
            if (ref($action) eq "ARRAY") {
                exec @{$action};
            } elsif (ref($action) eq "CODE") {
                &{$action};
            } else {
                exec $action;
            }
            exit 1;
        } else {
            kill TERM, keys %children;
            &'log (0, "Error Forking Children: $!\n");
            exit 1;
        }
        if ($'log_open) {
            if (ref($action) eq "ARRAY") {
                $str = join (" ", @$action);
            } elsif (ref($action) eq "CODE") {
                $str = "Perl code forked";
```

```
            } else {
                    $str = $action;
            }
            &'log (98, "Forked child $pid for user $i at $fork_time: $str\n");
        }
}

while (keys %children) {
    # 'Heartbeat' for running benchmarks.  This is NOT very portable
    # mainly due to the waitpid call.  (Need it otherwise there is a race
    # condition)
        if ($'user{'heartbeat'}) {
            $temp = $'user{'heartbeat'};
            local ($|) = 1;
            while (($wait_pid = waitpid(-1, $'WNOHANG)) <= 0) {
                    select(undef,undef,undef, $temp);
                    print ".";
                    print ("\n"), $count = 0 if $count++ > 78;
            }
            $rc  = $?;
        } else {
            $wait_pid = wait;
            $rc  = $?;
        }
        $wait_time = time;
        &'log (98, "Child $wait_pid terminated with return code $rc at $wait_time\n") if ($'log_open);
        if ($wait_pid == -1) {
            &'log (0, "No children, but I think I have these left: ",
                    join (", ", sort bynum keys %children), "\n");
            exit 1;
        } elsif (defined $children{$wait_pid}) {
            $dir = $children {$wait_pid};
            delete $children {$wait_pid};
            push (@{$exitval[$dir]}, $rc);
            chdir $me->{'work'}{'dirs'}[$dir];
            $action = shift(@{$todo[$dir]});
            next if !defined $action;
            $fork_time = time;
            if ($pid = fork) {
                    $children{$pid} = $dir;
                    $usedpid{$pid} = 1;
            } elsif (defined $pid) {
                    if (ref($action) eq "ARRAY") {
                        exec @{$action};
                    } elsif (ref($action) eq "CODE") {
                        &{$action};
                    } else {
                        exec $action;
                    }
                    exit 1;
            } else {
                    kill TERM, keys %children;
                    &'log (0, "Error Forking Children: $!\n");
                    exit 1;
            }
        } else {
            &'log(98, "Child I didn't spawn reported its death: $wait_pid\n");
            if ($usedpid{$wait_pid}) {
                    &'log(98, "Ok, maybe I spawned it... We'll let it pass\n");
            } else {
                    exit 1;
            }
        }
        if ($'log_open) {
            if (ref($action) eq "ARRAY") {
                    $str = join (" ", @$action);
            } elsif (ref($action) eq "CODE") {
                    $str = "Perl code forked";
            } else {
                    $str = $action;
            }
            &'log (98, "Forked child $pid for user $dir at $fork_time: $str\n");
        }
    }
    @stop_time = &'spectime;

### Modification by  Carsten Mund (cmu)
### Get a vector of 4 elements containing CPU time in user mode and system mode
### for this process and the children of this process
### values at END of  a benchmark
    @stop_time_cmu = times;
### End (cmu)

### Modification by Carsten Mund (cmu)
### The following line came some lines later in the original SPEC script
### But I need the time information in the „monitor_post_bench“-section
    @diff_time = &'sub_time(@stop_time, @start_time);
### end (cmu)

### Modification by  Carsten Mund (cmu)
### Write out  the run times of  a benchmark to my file „$benchmark_usertime. $run_num“
### I have to to this at this point because I want to use this file in the „monitor_post_bench“-section
### first the original start time,  end time and difference in seconds of SPEC measuring
    print TIMEFILE_cmu join(": ", "START", &'ctime($start_time[0]));
    print TIMEFILE_cmu join(": ", "ENDE", &'ctime($stop_time[0]));
    printf TIMEFILE_cmu "real: %.2f s \n", @diff_time;
### Compute and print out  the value of the cpu time for process and children in user mode
        printf  TIMEFILE_cmu  "user:  %.2f  s  \n",  (@stop_time_cmu[2]  -  @start_time_cmu[2])  +
(@stop_time_cmu[0] - @start_time_cmu[0]);
### Compute an print out the value of  the cpu time for process and children in system mode
```

```
        printf TIMEFILE_cmu "sys: %.2f s \n", (@stop_time_cmu[3] - @start_time_cmu[3]) +
(@stop_time_cmu[1] - @start_time_cmu[1]);
### Print some parameters of this run
    print TIMEFILE_cmu "$size, $tune, $ext, $users\n";
    close (TIMEFILE_cmu);
### End (cmu)

    chdir $'spec{'top'};
    if ($config{'monitor_post_bench'}) {
            my $command = &'subst(\%repl, $config{'monitor_post_bench'});

### Modification by  Carsten Mund (cmu)
### I need the $run_num in the post-monitor
### I put them as a shell export to the monitor command
### to prevent from conflicts I put my name to the name of the variable
            $cmu_adding = "carsten_num=$run_num ; export carsten_num ;";
            my $command=join (' ', $cmu_adding, $command);
###  I Can use „$carsten_num“ in the „monitor_post_bench“-section now
### end (cmu)
            system $command;
    }

### Modification by  Carsten Mund (cmu)
### I put the next line from this original place to the place directly before
### the the start of monitor_post_bench, because I need the time information there
###    @diff_time = &'sub_time(@stop_time, @start_time);
### End (cmu)

    &'log (97, "Benchmark Times:\n",
          "  Start:    ", join(", ", @start_time, &'ctime($start_time[0])),
          "  Stop:     ", join(", ", @stop_time, &'ctime($stop_time[0])),
          "  Elapsed:  ", join(", ", @diff_time, &'to_hms(@diff_time)),
               "\n");

# Write out a file in each directory which contains the exit values of
# each run
    for ($i = 0; $i < $users; $i++) {
            $name = &'joinpaths($me->{'work'}{'dirs'}[$i], $'spec{'exitvals'}.".$run_num");
            if (open  (FILE, ">$name")) {
               print FILE $me->benchmark, "\n";
               print FILE join(", ", @start_time, &'ctime($start_time[0]));
               print FILE join("\n", @{$exitval[$i]}), "\n";
               if ($num_targets[$i] != @{$exitval[$i]}) {
                        print FILE "ERROR: Missing Runs!\n";
                        &'log(97, "User $i appears to be missing some runs?\n");
               }
               close (FILE);
            } else {
               &'log(0, "Can't write to '$name': $!\n");
            }
    }

# Write out a file containing the run time of the benchmarks
    print TIMEFILE join(", ", @'bench_time, &'ctime($'bench_time[0]));
    print TIMEFILE join(", ", @start_time, &'ctime($start_time[0]));
    print TIMEFILE join(", ", @stop_time,  &'ctime($stop_time[0]));
    print TIMEFILE join(", ", @diff_time,  &'to_hms(@diff_time)), "\n";
    print TIMEFILE "$size, $tune, $ext, $users\n";
    close (TIMEFILE);

    sleep $config->{'wait_after_run'} if ($config->{'wait_after_run'} > 0);

    chdir $'pwd;
}
```

Mit diesen Änderungen ist es nach wie vor möglich, die SPEC-Umgebung so zu verwenden, wie dies in den Dokumentationsfiles [SPEC_doc] beschrieben ist. Die originale SPEC-Messung, -Auswertung und Reporterstellung wird von diesen Änderungen nicht berührt. Weitere Änderungen sind an der Ablaufumgebung grundsätzlich nicht erforderlich.
Start und Auswertung der Messungen mit den Lasterfassungstools sind vollständig von den in die Ablaufumgebung integrierten Monitoring-Points aus möglich bzw. erfolgen in einem nachgestellten externen Auswertungsschritt.

Anlage B -

Exemplarischer Ablauf einer Lasterfassung

Am Beispiel der Lasterfassung während des Benchmarklaufes auf der SPARCstation 20 Model 61 (Samson) unter Solaris 2.3 wird die generelle Realisierung dargestellt. Dieses System wurde für die Demonstration gewählt, da hier alle ausgewählten Meßwerkzeuge verfügbar sind.
Die Auslösung der Lasterfassung während eines Benchmarklaufes erfolgt über den SPEC-integrierten Mechanismus „monitor_pre_bench“, die Terminierung über den „monitor_post_bench“. Die Spezifikation dieser Aktionen unmittelbar vor und nach einem Benchmarklauf erfolgt im Konfigurationsfile („config-id“.cfg) für das Rechnersystem, in dem auch die zu nutzenden Optimierungen angegeben werden müssen ([SPEC_doc]- config.txt). Die Einbindung dieses Konfigurationsfiles wird mit dem Schalter *`-c config-id`* beim Benchmark-Aufruf realisiert.

```
#
# Sun configuration file for SPEC95
# 9 October 1995
VENDOR      = sun
action      = validate
tune        = base
ext         = sunos53
#output_format= asc,ps

monitor_pre_bench = pre_bench_monitor_sun.sh %benchname% %benchtop%/run

## monitor_wrapper =

monitor_post_bench = post_bench_monitor_sun.sh %benchname% %benchtop%/run

# These are listed as benchmark-tuning-extension-machine
default=default=default=default:
mark_runs    = 1
users        = 1

# cmu
OPTIMIZE   = -fast -xO4
....
# Anschließend Angabe der Optimierungen
```

Es werden jeweils Shell-Scripts gestartet, denen der Benchmark-Name und das Arbeitsverzeichnis als Parameter übergeben werden. Im Script *pre_bench_monitor_sun.sh* werden die Messungen mit den einzelnen Werkzeugen gestartet:

```
#!/bin/sh
### thinks that $1 is the aktual benchmark name
### thinks that $2 ist the path to the actual run-Directory

if [ $tool_path ]
then echo ":" ${tool_path} ":"
else tool_path="/a/zuse-f/home/urz/fs22/c/cmun/DIP/Measure_tools"
fi

### hard coded parameters für maximal 300000 sekunden Laufzeit
samplerate=60
samples=500

### Abfarge des Hostnamens zur Filenamen-Generierung
host=`hostname -s 2>/dev/null`
if [ $host ]
then echo $host;
else host=`hostname`
fi

mydate=`date +"%d-%m-%y"`
starttime=`date +"%H:%M:%S"`

### Schreibt Startdatum und Startzeit in eindeutige temporäre Files, wird zur Auswertung gebraucht
echo "$mydate" >___STARTDATE.${host}.$1.${carsten_num}
echo "$starttime" >___STARTTIME.${host}.$1.${carsten_num}

### Jetzt werden die einzelnen Meßwerkzeuge gestartet, dies geschieht für eine  schnelle Ausführung
### im Hintergrund, vor Rückkehr zur Benchmarkshell wird gewartet
### $carsten _num ist eine neue Shell-Umgebungsvariable, die die Ablaufnummer identifiziert und von der
### Benchmarkshell für den Monitor exportiert wird (s. Änderungen Anlage A)

# Anfangs-Zählerstand der Netzwerk.Protokoll-Zähler in temporäres File
netstat -i >_START.${host}.$1.${carsten_num}.net &
# Optional: $samplerate- Sekunden-Sampling für Daten des Netzwerk-Interfaces
#netstat -I fa0 $samplerate >_START.${host}.$1.${carsten_num}.fa0 &
#an0id=$!
#netstat -I le0 $samplerate >_START.${host}.$1.${carsten_num}.le0 &
#an1id=$!

# Anfangszählerstand für Ereigniszähler der virtuellen Speicherverwaltung in temporäres File
vmstat -s >_START.${host}.$1.${carsten_num}.vm &
# $samplerate -Sekunden-Sampling mit vmstat, schreibt in Sample-File
vmstat -S $samplerate $samples >${host}.$1.${mydate}.${starttime}.vm_live_prot &
vm_live_prot_id=$!

# $samplerate-Sekunden-Sampling für uptime wird in diesem Script organisiert, da bei uptime kein Sample-Mode vorhanden
ist
# Schreibt in Sample-File
$tool_path/uptime_prot.sh ${host}.$1 ${mydate} ${starttime} $samples $samplerate &
up_id=$!
```

```
#Anfangszählerstand  für Ereigniszähler des NFS-Protokolls in temporäres File schreiben
nfsstat -c >_START.${host}.$1.${carsten_num}.nfs &

# $samplerate-Sekunden -Sampling für Disk-I/O-Aktivitäten  mit iostat in ein Sample-File
iostat $samplerate $samples >${host}.$1.${mydate}.${starttime}.io_live_prot &
io_live_prot_id=$!

# $samplerate-Sekunden -Sampling für alle Systemaktivitäten  mit  SAR in ein  Binär-Sample-File
sar -o ${host}.$1.${mydate}.${starttime}.sar $samplerate $samples >/dev/null &
sar_id=$!
sadc_id=`$tool_path/get_sadc_child_pid $sar_id`
### ermittelt Prozeßnunmmer des eigentlichen Sampling-Prozesses sadc

### Generierung eines Shell-Scriptes zur späteren  Terminierung aller Sampling-Prozesse
### erforderlich, da die erforderliche Laufzeit für das Sampling im voraus nicht bekannt ist
echo "#!/bin/sh" >killshell.${host}.$1.${carsten_num}.sh
echo "kill -9 $lan0id" >>killshell.${host}.$1.${carsten_num}.sh
echo "kill -9 $lan1id" >>killshell.${host}.$1.${carsten_num}.sh
echo "kill -9 $vm_live_prot_id" >>killshell.${host}.$1.${carsten_num}.sh
echo "kill -9 $io_live_prot_id" >>killshell.${host}.$1.${carsten_num}.sh
echo "kill -9 $up_id" >>killshell.${host}.$1.${carsten_num}.sh
echo "kill -9 $sadc_id" >>killshell.${host}.$1.${carsten_num}.sh
chmod 0755 killshell.${host}.$1.${carsten_num}.sh

# ENDE pre_bench_monitor_sun.sh
```

Danach geht die Ablaufsteuerung zurück an das runspec-Script, das den Benchmark startet. Nach dem Benchmarklauf wird das bei „monitor_post_bench“ spezifizierte Shell-Script *post_bench_monitor_sun.sh* gestartet:

```
#!/bin/sh
### thinks that $1 is the actual benchmark name
### thinks that $2 ist the path to the actual run-Directory

if [ $MY_SPEC ]
then echo ":" $MY_SPEC ":"
else
MY_SPEC=`pwd`
export MY_SPEC
fi

prot_path=$MY_SPEC/PROTO

if [  $tool_path ]
then echo ":" $tool_path ":"
else
tool_path="/a/zuse-f/home/urz/fs22/c/cmun/DIP/Measure_tools"
export tool_path
fi

### get data for File-Naming
host=`hostname -s 2>/dev/null`
if [ $host ]
then echo $host;
else host=`hostname`
fi

### Liest Startzeit und Startdatum aus eindeutigen temporären Files für Protokoll-Namen-Konstruktion
### wird benötigt , um die „eigenen“ temporären Files zu identifizieren
starttime=`cat ___STARTTIME.${host}.$1.$carsten_num`
startdate=`cat ___STARTDATE.${host}.$1.$carsten_num`

###  Konstruiert Protokollnamen für diesen Benchmarklauf zur Datensicherung
general_proto="${prot_path}/${host}.$1.${startdate}.${starttime}"
echo "ALLG. PROTOKOLL-IDENT = $general_proto"

### Startet  erneut  die Abfragen der Zählerstande für  Ereignisse der  Netzwerkschnittstellen,
### der virtuellen Speicherverwaltung und des NFS-Protokolls und schreibt diese auch in temporäre Files
netstat -i >_ENDE.$host.$1.${carsten_num}.net &
vmstat -s >_ENDE.$host.$1.${carsten_num}.vm &
nfsstat -c >_ENDE.$host.$1.${carsten_num}.nfs &
### Mit dem im  Pre_Bench_Monitor konstruierten Shellscript werden die Sampling-Prozesse terminiert
killshell.${host}.$1.${carsten_num}.sh
wait

### Sicherung der Sampling-Files  ins Protokollverzeichnis für spätere Verarbeitung
cp ${host}.$1.${startdate}.${starttime}.io_live_prot ${general_proto}.io_live_prot
cp ${host}.$1.${startdate}.${starttime}.vm_live_prot ${general_proto}.vm_live_prot
cp ${host}.$1.${startdate}.${starttime}.up_prot ${general_proto}.up_live_prot
cp ${host}.$1.${startdate}.${starttime}.sar ${general_proto}.sar
### Löschen der lokalen Sampling-Files
rm ${host}.$1.${startdate}.${starttime}.io_live_prot
rm ${host}.$1.${startdate}.${starttime}.vm_live_prot
rm ${host}.$1.${startdate}.${starttime}.sar
rm ${host}.$1.${startdate}.${starttime}.up_prot

### Sichern des Files „$Benchmark_usertime.$run_num“ aus dem Arbeitsverzeichnis des Einzelbenchmarks
### Das File enthält die tatsächlichen Prozessorzeiten, dafür waren die in Anlage A beschriebenen Änderungen
filelist=`ls $2/$1_usertime.${carsten_num}`
echo "gefundene Zeitfiles: $filelist"
for f in $filelist
do
        f [ -s $f ]
        then
                mv $f  ${general_proto}.usertime
        fi
done

###  Von den temporären Files mit den Start- und Endzählerständen können eventuell benötigte gesichert werden,
### wenn man ergänzende Daten betrachten möchte
#cat _START.$host.$1.${carsten_num}.nfs _ENDE.$host.$1.${carsten_num}.nfs  >${general_proto}.nfsdata
#cat _START.$host.$1.${carsten_num}.net _ENDE.$host.$1.${carsten_num}.net  >${general_proto}.netdata
#cat _START.$host.$1.${carsten_num}.vm _ENDE.$host.$1.${carsten_num}.vm >${general_proto}.vmdata
#cat _START.$host.$1.${carsten_num}.fa0 >${general_proto}.fa0
#cat _START.$host.$1.${carsten_num}.le0 >${general_proto}.le0

### Jetzt werden mit in Perl realisierten  Skripten die temporären Files mit den Zählerständen bei Benchmarkstart
### und -ende ausgewertet, dabei werden die Besonderheiten in den Formaten der einzelnen Testsysteme
### berücksichtigt, es werden die für die Lastmaße interessanten Werte aus den komplexen Reports selektiert
### als Parameter erwarten die Scripts jeweils die Namen  der Files mit Start- und Endwerten, sowie evtl. ein
```

```
### Rechner-id
${tool_path}/extract.nfs   _START.$host.$1.${carsten_num}.nfs   _ENDE.$host.$1.${carsten_num}.nfs  >${general_proto}.nfs
${tool_path}/extract.net   _START.$host.$1.${carsten_num}.net  _ENDE.$host.$1.${carsten_num}.net samson >${general_proto}.net
${tool_path}/extract.vm  _START.$host.$1.${carsten_num}.vm  _ENDE.$host.$1.${carsten_num}.vm  samson >${general_proto}.vm

### now eventually sar-extracting (also possible to do this later time)

### Löschen des Kill-Scriptes
rm killshell.${host}.$1.${carsten_num}.sh

###  Löschen sämtlicher temporärer Files diese Benchmarklaufes
rm _START.$host.$1.${carsten_num}.*
rm _ENDE.$host.$1.${carsten_num}.*
rm ___START*.$host.$1.${carsten_num}

### End post_bench_monitor_sun.sh
```

Nach dem „monitor_post_bench“ geht die Steuerung erneut zurück an die SPEC-Ablaufsteuerung mit „runspec“. Hier erfolgt die Auswertung des eigentlichen Benchmarklaufes mit Verifizierung der korrekten Abarbeitung und Generierung von SPEC-REPORT-Rules-konformen Protokollen für eine Veröffentlichung.
Da nur einzelne Benchmarks getestet wurden, wurde auf die dadurch unvollständigen Reports verzichtet. Ob der Benchmarklauf die richtigen Ergebnisse berechnet hat, wird auch auf den Bildschirm ausgegeben.
Als Ergebnis eines Benchmarklaufes mit zugehöriger Lastmessung steht damit ein Satz von Files zur Verfügung. Dies sind zunächst die Sampling-Files von Messungen mit vmstat, iostat, uptime und sar. Nachfolgend werden jeweils kurze Ausschnitte dargestellt, die direkt in den Lastmaßen zu nutzenden Werte sind ***fett-kursiv*** dargestellt, ergänzend zu betrachtende Werte sind nur *kursiv* dargestellt.

vmstat-Sample-Protokoll: ($host.$benchmark.$startdate.$starttime.vm_live_prot)

```
....
procs     memory          page                disk          faults      cpu
 r b w   swap  free  si  so pi po     fr de sr s1 s3 -- --   in   sy   cs us sy id
 2 0 0 336672 11780   0   0  0 304    321 0 503 0 5  0  0  654  191  102 93  7  0
 2 0 0 336652 11164   0   0  0 446    471 0 429 0 8  0  0  688  197  101 93  7  0
 2 0 0 336644 11364   0   0  0 564    582 0 421 0 13 0  0  770  193  103 92  8  0
 1 0 0 380956 49740   0   0 18 191    196 0 167 0 12 0  0  424  192  108 92  7  0
 ...
```

iostat-Sample-Protokoll: ($host.$benchmark.$startdate.$starttime.io_live_prot)

```
...
tty          sd1           sd3           cpu
 tin tout Kps tps serv  Kps tps serv  us sy wt id
   0    7   2   0   25  309  45   73  65 10 15 11
   0    0   1   0    9  310  18   66  82 18  0  0
   0    0   2   0   17  802  59   52  73 13 13  0
   0    0   2   0   12  691  90   36  73 11 17  0
...
```

uptime-Sample-Protokoll: ($host.$benchmark.$startdate.$starttime.up_live_prot)

```
....
Tue Apr 23 03:06:34 MET DST 1996
  3:06am  up 5 day(s), 16:23,  0 user,  load average: 5.17, 4.23, 3.39
  3:07am  up 5 day(s), 16:24,  0 user,  load average: 4.70, 4.29, 3.48
  3:09am  up 5 day(s), 16:25,  0 user,  load average: 3.06, 3.88, 3.39
  3:10am  up 5 day(s), 16:26,  0 user,  load average: 3.07, 3.68, 3.35
.....
```

sar-Sample-Protokoll: ($host.$benchmark.$startdate.$starttime.sar)

Beim Sampling mit sar wird ein Binärfile erzeugt. Durch erneuten sar-Aufruf mit Angabe des Filenamens und einem der in 3.1 angegeben Schalter lassen sich daraus ähnliche Sample-Reports generieren.

Alle diese Sample-Protokolle können für Betrachtungen zur Verteilung der Werte für die einzelnen Lastmaße herangezogen werden. Zur Beschreibung der Last für die einzelnen Lastmaße während eines Benchmarklaufes wird jedoch jeweils das arithmetische Mittel über die Einzelwerte für Lastmaße gebildet. Dies geschieht in einem extern zu startenden Perl-Programm, daß damit die Auswertungsschicht des Lastmonitors bildet.
Das konstruierte Perl-Script *get_run_data2* erwartet die Übergabe einer Rechnerbezeichnung, auf deren Basis die Auswertung der unterschiedlichen Formate der Sample-Protokolle der Testrechner erfolgt.
Das Programm erstellt aus allen verfügbaren Daten einen Report ($host.$benchmark.$startdate.$starttime.full_rep) mit mittleren Lasten während eines Benchmarklaufes. Neben der Auswertung der Sample-Protokolle fließen in den Report die bereits beim Monitoring erzeugten Files ein, die die Ergebnisse der Auswertung der verschiedenen Start- und Endzählerstände enthalten ($host.$benchmark.$startdate.$starttime.vm, $host.$benchmark.$startdate.$starttime.net $host.$benchmark.$startdate.$starttime.nfs), sowie die Werte für die absolute Benchmarklaufzeit und die zugeteilten Prozessorzeiten ($host.$benchmark.$startdate.$usertime). Ein vollständiger Report für eine Messung auf dem SPARC-Rechner Samson ist nachfolgend dargestellt. Dieser enthält die Ergebnisse einer Messung mit BSD- und SystemV-Werkzeugen, für die anderen Rechner ist jeweils nur ein Teil der Daten verfügbar. Zur Erstellung der

Diagramme für den Rechner Samson wurden dabei die Werte der BSD-Werkzeuge (iostat, vmstat) verwendet.
Die Reports aller Messungen dieser Arbeit werden zu dieser Arbeit extern am Lehrstuhl Betriebssysteme der TU Chemnitz-Zwickau bereitgestellt. Die Erstellung der Auswertungsdiagramme (Abschnitt 5.2) erfolgte auf Basis dieser Reports, jedoch extern mit Microsoft Excel.
Für die Kennzeichnung der Werte zu den einzelnen Lastmaßen wird folgende Form gewählt:

fett-kursiv - Werte zur Bildung von Lastmaßen
kursiv - ergänzend betrachtete Werte
#Kommentar - Kommentar

```
================================================================================
101.tomcatv                                  # Benchmarkname
START: Tue Apr 23  3:06:40 MET 1996
ENDE: Tue Apr 23  4:05:45 MET 1996
real: 3545.00 s                              # Benchmarklaufzeiten
user: 1475.51 s                              # Prozessorzeit Nutzerprozeß
sys: 6.18 s
ref, base, sunos53, 1
--------------------------------------------------------------------------------
Tue Apr 23 03:06:34 MET DST 1996             # Run-Queue-Länge
Average_workload(1min-avgs from uptime 48 samples):         5.33333
--------------------------------------------------------------------------------
 procs     memory           page            disk          faults      cpu
 r b w   swap  free  si  so pi po fr de sr s1 s3 -- --   in   sy   cs us sy id
    vmdata            #abs              avg
         r             172       3.01754385
         b               6       0.10526315
         w               0                0
      swap        23669680       415257.543
      free         4554592       79905.1228
        si               0                0
        so               0                0
        pi            1872       32.8421052
        po            9032       158.456140  #Page-Out-Rate
        fr            9784       171.649122
        de               0                0
        sr           10097       177.140350
        s1              11       0.19298245
        s3             721       12.6491228
        --               0                0
        --               0                0
        in           24007       421.175438
        sy           12892       226.175438
        cs            7201       126.333333
        us            5254       92.1754385
        sy             389       6.82456140
        id              57                1  # %idle-Anteil CPU-Auslastung
(vmstat with 57 samples)
--------------------------------------------------------------------------------
      tty          sd1           sd3          cpu
 tin tout Kps tps serv  Kps tps serv  us sy wt id
    IOdata            #abs            avg
       tin               0              0
      tout               0              0
        Kps            115      1.91666666  # Disk-I/O-Rate Disk1
        tps             11      0.18333333
       serv            847      14.1166666
        Kps          11247      187.449999  # Disk-I/O-Rate  Disk2
        tps            726      12.0999999
       serv           4110            68.5
         us           5545      92.4166666
         sy            398      6.63333333
         wt             37      0.61666666
         id             21      0.34999999
(iostat with 60 samples)
--------------------------------------------------------------------------------
SunOS sisyphus 5.4 Generic_101945-36 sun4m    04/23/96

03:06:31    %usr    %sys    %wio   %idle    #CPU-Auslastung 100%-idle
Average       92       7       1       0
(sar found 60 samples in samson.tomcatv.23-04-96.03:06:12.sar )
--------------------------------------------------------------------------------
SunOS sisyphus 5.4 Generic_101945-36 sun4m    04/23/96

03:06:31 runq-sz %runocc swpq-sz %swpocc
Average      5.2     95                      #Run-Queue-Länge
(sar found 60 samples in samson.tomcatv.23-04-96.03:06:12.sar )
--------------------------------------------------------------------------------
SunOS sisyphus 5.4 Generic_101945-36 sun4m    04/23/96

03:06:31 swpin/s bswin/s swpot/s bswot/s pswch/s
Average     0.00     0.0    0.00     0.0     126 #Swap-Anteil  Page-Out-Rate  in Blöcken/s
(sar found 60 samples in samson.tomcatv.23-04-96.03:06:12.sar )
--------------------------------------------------------------------------------
SunOS sisyphus 5.4 Generic_101945-36 sun4m    04/23/96

03:06:31  pgout/s ppgout/s pgfree/s pgscan/s %ufs_ipf
Average      4.71    38.84    42.04   176.83     0.00    #Page-Out-Rate in Seiten/s
(sar found 60 samples in samson.tomcatv.23-04-96.03:06:12.sar )
--------------------------------------------------------------------------------
SunOS sisyphus 5.4 Generic_101945-36 sun4m    04/23/96

03:06:31   device        %busy   avque   r+w/s  blks/s  avwait  avserv
Average    sd1              0     0.1       0       4    28.3   126.4
           sd3             14     0.7      12     375     1.3    53.0
                                          #Disk-I/O-Rate in Blöcken/s
(sar found 181 samples in samson.tomcatv.23-04-96.03:06:12.sar )
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
samson.hrz.tu-chemnitz.de#ipkts: (4352) 21670
samson.hrz.tu-chemnitz.de#ierrs: (4352) 0
samson.hrz.tu-chemnitz.de#opkts: (4352) 8986        #Paketanzahlen und Größen für
samson.hrz.tu-chemnitz.de#oerrs: (4352) 0           # die verschiedenen Schnittstellen
samson.hrz.tu-chemnitz.de#coll: (4352) 0            # als Absolutwerte für Laufzeit
samson-e.hrz.tu-chemnitz.de#ipkts: (1500) 12668
samson-e.hrz.tu-chemnitz.de#ierrs: (1500) 0
samson-e.hrz.tu-chemnitz.de#opkts: (1500) 53
samson-e.hrz.tu-chemnitz.de#oerrs: (1500) 0
samson-e.hrz.tu-chemnitz.de#coll: (1500) 0
--------------------------------------------------------------------------------
calls                     old : 2727180
retransmissions of requests old : 16701
calls                     new : 2736892
```

```
retransmissions of requests new : 16706
#calls: 9712                                          # NFS-Calls absolut für
#retransmissions: 5                                   # Laufzeit
%percentage: 0.051482701812191104807
------------------------------------------------------------------------------
             0 swap ins
             0 swap outs
             0 pages swapped in                       #Absolutzahl ausgelagerter
             0 pages swapped out                      #Seiten während der Laufzeit
         25482 page ins                               #des Benchmarks
         17081 page outs
         28517 pages paged in
        140368 pages paged out
        636623 pages examined by the clock daemon
==============================================================================
```

Die verbleibenden Auswertungsschritte, wie Berechnung der mittleren Netz- und Speicherraten aus den Absolutwerten für die Ereignisse und der Benchmarklaufzeit, erfolgten im externen Auswertungsschritt in Microsoft Excel. Die Normierung der Lastmaße über die in der Definition angegebenen Faktoren (Seitengrößen, Blockgrößen, Paketgrößen) wurde auf Basis der Informationen aus den Manuals ebenfalls in der externen Auswertung realisiert. (Alle Excel-Tabellen dieses letzten Auswertungsschrittes werden ebenfalls extern auf Diskette bereitgestellt.)

Die dargestellte Realisierung einer Messung und Auswertung stellt nur einen Funktionsprototyp dar, der soweit ausgebaut wurde, daß die für die Benchmark-Untersuchungen dieser Arbeit erforderlichen Messungen möglichst effektiv durchgeführt und ausgewertet werden konnten.

Mit Perl und einem Werkzeug wie Tcl/Tk sollte es relativ einfach möglich sein, auf Basis dieses Entwurfes einen universellen Lastmonitor für die betrachteten Lasten zu implementieren. Problematischster Punkt bei der Verarbeitung sind dabei die unterschiedlichen Formate und Modi, in denen die verwendeten Werkzeuge auf den betrachteten Systemen die gesuchten Werte bereitstellen.

Literaturverzeichnis

[ChSuWo94] Chan, Y., Sudarsanam, A., Wolfe, A.: *The Effect of Compiler-Flag Tuning on SPEC Benchmark Performance*. Computer Architecture News 22,4, S.60-70, 1994.

[Curn76] Curnow, H. J., Wichmann B. A., *A Synthetic Benchmark*. The Computer Journal 19,1, S. 43-49, 1976.

[Hofb96] Hofbauer, J.*: Leistungsbewertung von Workstations mit SPEC-SFS-Benchmarks für den Einsatz als Fileserver*. Projektarbeit am Lehrstuhl Betriebssysteme der TU Chemnitz-Zwickau, 1996.

[Hüls_iX_95] Hülsenbusch, R.: *Feinabgleich - Anpassung der SSBA-Suite an Multiprozessorsysteme: iX-SSBA*, iX 2/1995, S. 38-39.

[Kalfa90] Kalfa, W.: *Betriebssysteme*, Akademie-Verlag Berlin, 1990.

[KöRoSc_iX_95]Körner, A., Rohe, K., Schöpf, P.: *Simulierte Anwender - TPC-Benchmarks: Zwischen Anspruch und Wirklichkeit*. iX 11/1995, S.158-162.

[Lang 92] Langendörfer, H.: *Leistungsanalyse von Rechensystemen - Messen, Modellieren, Simulation*, Carl Hanser Verlag München Wien, 1992.

[Louk91] Loukides, M.: *System Performance Tuning*. A Nutshell Handbook, O'Reilly & Associates, Inc., Sebastobol, 1991.

[Mund95] Mund, C.: *Bewertung der Compute-Leistung von Workstations mit SPEC-CPU-Benchmarks*. Projektarbeit am Lehrstuhl Betriebssysteme der TU Chemnitz-Zwickau, 1995.

[PeReLo93] Peek, J., O'Reilly, T., Loukides, M. u.a.: *UNIX Power Tools*. A Nutshell Handbook, O'Reilly & Associates, Inc., Sebastobol / Random House, Inc. , New York, 1993.

[PERFwww] *PDS:The Performance Database Server*.
(URL: http://performance.netlib.org/performance/html/spec.html), Juni 1996.

[Rahm_iX_93] Rahm, E.: *Schnelle Transaktionen - DBMS Leistungsvergleich mit TPC-Benchmarks.* iX 5/1993, S. 144-148.

[Rich91] Richter, H.: *UNIX Systemverwaltung*. Addison-Wesley, München, 1991.

[Siev_iX_96] Sieverding, H.: *FehlSPECulation - Erste Erfahrungen mit der SPEC95*. iX 2/1996, S.84-91.

[Sinix93] *SINIX Benchmarking - UNIX-Leistungsdaten richtig beurteilen.* Siemens Nixdorf Informationssysteme AG, 33102 Paderborn, Bestell-Nummer U 8999-J-Z141-2, 1993.

[SPEC95_i] *SPEC Newsletter*. Volume 7; Issue i; (Issue 1 März 1995, Issue 2 Juni 1995, Issue 3 September 1995, Issue 4 Dezember 1995),
SPEC, 10754 Ambassador Drive, Suite201, Manassas, VA 22110, USA; 1995.

[SPEC_DES.x] *SPEC Benchmark Description*. Benchmarkbeschreibungen für einzelne Benchmarks, Files „DESCR.x“ der Quellen der SPEC-Benchmarks, SPEC CPU Release spec95.65, 1995.

[SPECdoc] *SPEC CPU95 Documentation.*
Installationsverzeichnis $SPEC/doc der Benchmark-Suite spec95.65,
howto.txt - *WELCOME to SPEC95!* -Tips zur Installation und Testdurchführung,
config.txt - Konstruktion von Konfigurationsfiles,
runspec.txt - *Runspec - SPEC run and reporting tool.* -Beschreibung des Tools,
tool_build.txt - Spezielle Installationsprobleme,
runrules.txt - *SPEC Run and Reporting Rules for CPU95 Suites,*
SPEC Open Systems Steering Commitee, 1995.

[SPECwww] *SPEC World Wide Web Site.*
(URL: http://www.specbench.org), Juni 1996.

[TPCwww] *TPC WWW-Server.*
(URL: http://www.tpc.org), Juni 1996,
TPC, 777 N.First St., Suite 600, San Jose, CA 95112-6311.

[Weic84] Weicker, R.P.: *Dhrystone: A Synthetic Systems Programming Benchmark*. Communikation of the ACM 27,10, S. 1013-1030, Oktober 1984.

[Weic91] Weicker, R. P.: *Benchmarking: Status, Kritik und Aussichten*. In: Messung, Modellierung und Bewertung von Rechensystemen, 6.GI/ITG-Fachtagung, Hrsg. Lehmann, A. Lehmann, F., S. 259-277, Informatik-Fachberichte, Springer-Verlag Berlin, 1991.

[Weic_El_96] Weicker, R.P.: *Meßlatte für Workstations.* Elektronik 5/1996, S.114-130.

[Weic_iX_94] Weicker, R.P.: *Nicht nur für SPECialisten - Weiterentwicklungen der SPEC-Testsuite*. iX 2/1994, S.124-130.

[Wils95] Wilson, R. C.: *UNIX TEST TOOLS AND BENCHMARKS.* Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458, 1995.