

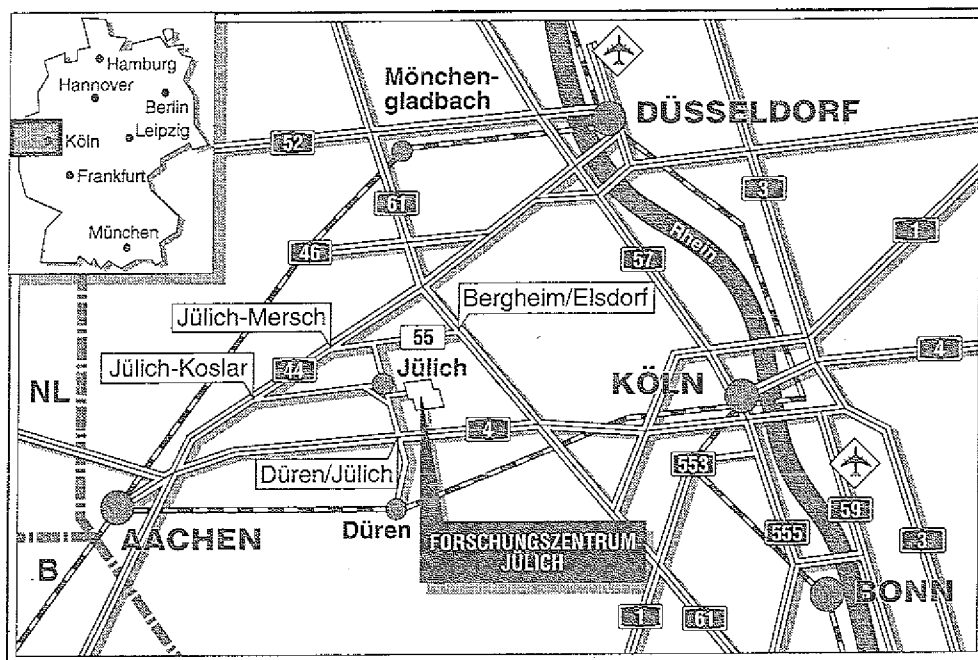
Zentralinstitut für Angewandte Mathematik

**Ein Benchmark zur Kapazitätsplanung
bei interaktiven Rechnersystemen**

Georg Berk

1871

1872



Berichte des Forschungszentrums Jülich ; 2597

ISSN 0366-0885

Zentralinstitut für Angewandte Mathematik Jül-2597

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek

Postfach 1913 · D-5170 Jülich · Bundesrepublik Deutschland

Telefon: 02461/61-6102 · Telefax: 02461/61-6103 · Telex: 833556-70 kfa d

1. The first part of the paper discusses the importance of the study of the history of the United States. It is argued that the study of the history of the United States is essential for a full understanding of the country and its people. The paper then goes on to discuss the various factors that have shaped the history of the United States, including the role of the federal government, the influence of the states, and the impact of the people.

2. The second part of the paper discusses the importance of the study of the history of the United States. It is argued that the study of the history of the United States is essential for a full understanding of the country and its people. The paper then goes on to discuss the various factors that have shaped the history of the United States, including the role of the federal government, the influence of the states, and the impact of the people.

3. The third part of the paper discusses the importance of the study of the history of the United States. It is argued that the study of the history of the United States is essential for a full understanding of the country and its people. The paper then goes on to discuss the various factors that have shaped the history of the United States, including the role of the federal government, the influence of the states, and the impact of the people.

4. The fourth part of the paper discusses the importance of the study of the history of the United States. It is argued that the study of the history of the United States is essential for a full understanding of the country and its people. The paper then goes on to discuss the various factors that have shaped the history of the United States, including the role of the federal government, the influence of the states, and the impact of the people.

5. The fifth part of the paper discusses the importance of the study of the history of the United States. It is argued that the study of the history of the United States is essential for a full understanding of the country and its people. The paper then goes on to discuss the various factors that have shaped the history of the United States, including the role of the federal government, the influence of the states, and the impact of the people.

Ein Benchmark zur Kapazitätsplanung bei interaktiven Rechnersystemen

Georg Berk

Inhaltsverzeichnis

1.0	Einleitung	1
2.0	Leistungsbewertung von Rechnersystemen	5
2.1	Interaktive Lasten und Batch-Lasten	5
2.2	Produktivität der Systembenutzer	6
2.3	Ziele der Leistungsbewertung	8
2.4	Methoden der Leistungsbewertung	9
2.5	Modell eines Transaktionssystems	11
2.6	Warteschlangenmodell nach Denning und Buzen	12
2.7	Das Kennlinienverfahren	19
2.7.1	Antwortzeiten bei der Anwendung des Kennlinienverfahrens	19
2.7.2	Systemauslastung	21
2.7.3	Einfluß von Kenngrößen	22
3.0	Aufbau des Meßsystems	25
3.1	Prozeßverwaltung	25
3.2	Asynchrone Prozesse	27
3.2.1	Möglichkeiten zur Interprozeß-Kommunikation	28
3.2.2	Eigenschaften einer PIPE	28
3.3	Zielsysteme	30
3.4	Betriebsmittelbelastungen	32
3.5	Meßmethode	35
3.6	Genereller Aufbau eines Tochterprozesses	36
3.7	Meßmethode für einen interaktiven Benchmark	38
3.8	Realisierung systeminvarianter Parameter	38
3.8.1	Skalierung der CPU-Belastung	39
3.8.2	Skalierung der E/A-Belastung	40
3.9	Erzeugung von Prozessen mit verschiedenen Klassen	40
3.10	Befehlsdatei zur Benchmark-Steuerung	41
3.11	Ausgabedatei	41
3.12	Übergabe der benötigten Daten und Parameter	42
3.13	Start der erzeugten Prozesse	43
3.14	Kommunikation der asynchronen Prozesse	43
4.0	Analyse der Arbeitslast	45
4.1	Ausgangsdaten	46
4.2	Speicherbedarf	49
4.3	Belastung der Ein-/Ausgabeeinheiten	50
4.4	Spezifizierung von Lastschränken	52
5.0	Anwendung des Meßsystems	53
5.1	Kalibrierung	53
5.2	Einfluß des Benchmarksystems auf die Meßergebnisse	56
6.0	Ergebnisse	59

6.1	Durchführung des Kennlinienverfahrens	59
6.1.1	Berechnung der Kenngrößen	59
6.1.2	Kapazitätsabschätzung	69
6.2	Bestimmung der Auslastung	70
6.3	Konstanz des Denkzeitdehnfaktors	74
6.4	Simulation interaktiver Benutzer im realen Betrieb der CRAY X-MP ...	77
6.5	Vergleich der untersuchten Systeme	78
6.6	Erfahrungen mit dem Programmsystem Interbench	80
7.0	Ausblick	83
8.0	Literaturverzeichnis	85
Anhang A.	Beispiel einer Befehlsdatei	89
Anhang B.	Parameter in den verwendeten Klassen	91
Anhang C.	Beispiel einer Ausgabedatei	93

1.0 Einleitung

Die Anforderungen an die Leistungsfähigkeit von Rechnersystemen sind in den vergangenen Jahren deutlich gestiegen. Ausgehend von der von-Neumann-Architektur bestanden die ersten Ansätze zur Leistungssteigerung in einer technologiebestimmten Geschwindigkeitssteigerung der Hardware. Diese Vorgehensweise stößt zunehmend an physikalische Grenzen, so daß vorrangig die Entwicklung innovativer Architekturen zur weiteren Leistungssteigerung betrieben wird.

Die Firma CRAY-Research entwickelt seit Anfang der siebziger Jahre Hochleistungsrechner mit Pipeline-Struktur. Diese Systeme werden für bestimmte Anwendungsfälle optimiert, die größte Leistungsfähigkeit wird damit bei der Bearbeitung großer Vektoren und allgemeiner mehrdimensionaler Felder erreicht.

Die International Business Machines Corp. (IBM) verfolgt ein anderes Konzept zur Entwicklung von Großrechnern. Diese Systeme werden nicht auf einen Anwendungsfall spezialisiert, sondern für hohe Leistungsfähigkeit bei der Abarbeitung allgemeiner Datenverarbeitungsaufgaben ausgelegt. Allerdings ermöglichen auch hier zusätzliche Spezialprozessoren (*Vector Feature*) die Vektorverarbeitung.

Die bisher erwähnten Systeme besitzen einige wenige, aber sehr leistungsfähige Prozessoren. In der jüngeren Zeit wird das Bestreben verstärkt, massiv parallele Systeme zu entwickeln. Hier werden Hunderte bis zu einigen Tausenden von Prozessoren zu einem System zusammengefaßt. Diese Prozessoren sind in der Regel keine teuren Spezialentwicklungen, sondern Standardangebote der Halbleiterhersteller. Somit bleibt jeder einzelne Prozessor preiswert.

Ein Problem dieser Architekturen besteht in der Notwendigkeit der Kommunikation der Prozessoren untereinander. Beispielsweise ist bei den Transputersystemen dieses Problem durch zwei unterschiedliche Baugruppen auf dem Prozessor gelöst. Die eine Baugruppe ist, wie bisher, für die eigentliche Rechenleistung verantwortlich, während die zweite Baugruppe ausschließlich die Kommunikation durchführt. Diese Aufgabenteilung gewährleistet eine (annähernd) gleichbleibende Rechenleistung unabhängig vom Umfang der Prozessorkommunikation.

Mit wachsender Prozessorzahl ist es nicht mehr sinnvoll, eine direkte Kommunikationsverbindung zwischen allen Prozessoren zu realisieren, da sowohl der Hardware- als auch der Softwareaufwand die vertretbaren Grenzen bei weitem überschreiten würde. Ein Weg der Realisierung von Verbindungsstrukturen zur Kommunikation zwischen den Prozessoren ist durch *Hypercube*-Architekturen gegeben.

Im Gegensatz zu diesen, primär erst in Forschungseinrichtungen und Universitäten genutzten Hochleistungsrechnern ist in den vergangenen Jahren in der breiten Bevölkerung die Akzeptanz von *Personal Computern* deutlich angestiegen. Diese Systeme haben sich für allgemeine Anwendungen durchgesetzt. Basierend auf dem von-Neumann-Prinzip besteht keine oder nur geringe Parallelität der Betriebsmittel. *Personal Computer* können in Gewerbebetrieben zur Verwaltung der Kundendateien, des Lagerbestandes und der Auftragslage eingesetzt werden. Die gleichen Systeme erleichtern auch die Geschäftskorrespondenz erheblich. Ebenso werden diese Systeme im häuslichen Bereich eingesetzt.

Ein wichtiges neues Konzept zur Steigerung der Leistungsfähigkeit ist durch RISC-Architekturen (*Reduced Instruction Set Computer*) gegeben. Diese Systeme be-

sitzen nur einen eingeschränkten Befehlssatz. Die vorhandenen Befehle können allerdings sehr schnell abgearbeitet werden. Somit werden auch Befehlssequenzen, welche in anderen Architekturen durch einen einzigen, leistungsfähigen Befehl ersetzt werden können, in vergleichbarer Geschwindigkeit abgearbeitet.

Im naturwissenschaftlich-technischen Bereich liegt ein Schwerpunkt der Nutzung bei sehr rechenintensiven Verfahren, da eine exakte, geschlossene Problemlösung häufig nicht gefunden werden kann. Beispielhaft sollen hier die Wettervorhersage oder die Probleme der Vielteilchenphysik angeführt werden.

Zu diesem Zweck werden zentrale Rechnersysteme mit extrem hoher Leistungsfähigkeit eingesetzt, die *Supercomputer*. Die enorme Rechenleistung kann durch Einsatz mehrerer sehr leistungsfähiger Recheneinheiten erzielt werden, welche das Problem bei entsprechender Auslegung parallel bearbeiten können [CRAY,c], dabei ist die Leistungsfähigkeit der Einzelprozessoren durch Einführung von Funktionseinheiten für spezielle Aufgaben, wie Vektoreinheiten oder Pipelines, gegenüber Universalrechnern deutlich gesteigert [Hoß]. Viele Algorithmen greifen auf diese Möglichkeiten zurück; so profitieren hiervon naturgemäß die Verfahren der linearen Algebra. Solche Systeme werden heute noch bevorzugt im Batch-Betrieb eingesetzt. Dies hat insbesondere die Vorteile des hohen Durchsatzes und der guten Prozessorauslastung.

Es ist aber auch bei *Supercomputern* eine den Mainframe-Systemen (beispielsweise von IBM) analoge Entwicklung zu beobachten: der Übergang von der reinen Batch-Verarbeitung zu einem gemischten Batch- und Dialogbetrieb. Der Dialogbetrieb erlaubt dem Benutzer einen direkten Zugriff auf das System. Dies ist beispielsweise bei der Fehlersuche in einem Programm von Vorteil.

Durch die zunehmende Verbreitung von Arbeitsplatzrechnern führt die Entwicklung zu sogenannten *Client-Server*-Strukturen. Hier wechselwirken mit dem Computer-Server mehrere, weniger leistungsfähige Arbeitsplatzrechner (*Workstations*), zur grafischen Vor- und Nachbearbeitung.

Solche Workstations werden in Zukunft im Zuge der Verwirklichung sogenannter offener Systeme die bisher eingesetzten Terminals mit direktem Zugriff auf einen Großrechner verdrängen.

Der Supercomputer wird weiterhin vorwiegend für Batch-Aufgaben eingesetzt werden, wobei allerdings dialogartige Anwendungen mit erhöhter Priorität bearbeitet werden. Dabei sind zwei konträre Ziele zu verfolgen, die nicht gleichzeitig optimal realisierbar sind:

- Hohe Auslastung des Supercomputers, insbesondere der CPU;
- Vertretbare Antwortzeiten bei den Dialogsitzungen der Benutzer beziehungsweise bei *Client-Server*-Anwendungen.

Die vorliegende Arbeit soll einen Ansatz zur Bewertung des Systemverhaltens im interaktiven Betrieb bzw. im quasi-interaktiven Betrieb einer *Client-Server*-Anwendung darstellen. Diese Bewertung ist für alle aufgeführten Architekturen möglich. Es können einerseits verschiedene Systeme im Hinblick auf Eignung für die vorgesehene Anwendung untersucht werden. Andererseits kann aber auch eine nähere Untersuchung eines Systems erfolgen. Hier kann eine Abschätzung zur sinnvollen Systemauslegung eines zu installierenden Systems durchgeführt werden. Ebenso wird für ein vorhandenes System

eine Aussage ermöglicht, welche interaktive Arbeitslast nicht überschritten werden darf, um dem Benutzer eine zumutbare Antwortzeit zu gewährleisten.

Anlaß zur Durchführung dieser Arbeit ist das Bestreben im Forschungszentrum Jülich, den interaktiven Betrieb auf einer installierten CRAY X-MP/416 neben der bisherigen Batch-Nutzung zu ermöglichen.

Die projektierten Merkmale für diesen interaktiven Betrieb sind im einzelnen:

- Eine maximale Speichergröße für einen einzelnen Benutzer von 2 Megaworten (MW);
- Eine Begrenzung der Rechenzeit eines Benutzers pro Sitzung (geplant sind hier 10 Minuten); Hierdurch soll eine Abgrenzung zwischen Batch-Aufgaben und sinnvollem interaktiven Zugriff definiert werden;
- Ein höherer Kostenfaktor für den interaktiven Betrieb. Bisher wurden Batch-Lasten die Kostenfaktoren 1/2, 1 oder 2 zugewiesen, entsprechend ihrer Priorität.

Kapitel 2 gibt einen Überblick über verschiedene Möglichkeiten zur Leistungsbeschreibung und -bewertung eines Systems und deren Vor- und Nachteile. Es wird eine Klassifizierungsmöglichkeit zur Beschreibung verschiedener Lastzustände aufgeführt. Zur Analyse der Messungen wird ein Modell eines Warteschlangennetzwerkes beschrieben und Parameter mit Hilfe der *Operational Analysis of Queuing Network Models* von Denning und Buzen [BuDe] berechnet. Dieses Modell führt zu einem Verfahren, welches die Abhängigkeiten des Systemdurchsatzes und der Antwortzeiten von der Zahl der aktiven Benutzer widerspiegelt.

In Kapitel 3 wird eine Übersicht der untersuchten Rechnersysteme gegeben. Diese Systeme wurden mit dem Ziel ausgewählt, die beschriebenen Architekturen (Supercomputer, RISC, PC) vergleichend gegenüberzustellen. Der Aufbau des Meßsystems und die Realisierung der Betriebsmittelbelastungen werden ebenso beschrieben wie die gewählte Meßmethode. Asynchrone Prozesse bilden einen weiteren Schwerpunkt dieses Kapitels. Es werden einige Besonderheiten der Kommunikation von asynchronen Prozessen aufgeführt, sofern sie zur Realisierung des Meßsystems notwendig waren oder Probleme aufwarfen. An diese Untersuchungen schließt sich die Präsentation der Struktur des hier erstellten Benchmarksystems *Interbench* und die Beschreibung der genutzten Funktionen des zugrundeliegenden UNIX-Systems an. Ebenso werden die Verfahren der Parametereingabe erläutert.

Kapitel 4 beschreibt die Analyse der Arbeitslast auf der im Forschungszentrum Jülich installierten CRAY X-MP/416. Ziel dieser Analyse ist es, eine der interaktiven Nutzung entsprechende Belastung und Abhängigkeiten zwischen benötigter Rechenleistung, Speicherbelastung und Ein-/Ausgabeverhalten zu finden.

Kapitel 5 und 6 beschäftigen sich mit den durchgeführten Untersuchungen der beschriebenen Rechnersysteme. Einerseits werden die durchgeführten Messungen und deren Ergebnisse dargestellt. Die in Kapitel 2 dargelegten Zusammenhänge werden hier auf mehreren Systemen verifiziert. Unter Berücksichtigung der Ergebnisse aus Kapitel 4 werden einheitliche Belastungsklassen gebildet und abgearbeitet. Das Verhalten der verschiedenen Systeme wird vergleichend gegenübergestellt. Andererseits werden an dieser

Stelle auch Schwachstellen des Benchmarksystems angeführt, die während der Anwendungen zutage getreten sind.

In Kapitel 7 werden als Ausblick einige mögliche Erweiterungen des Benchmarks umrissen, welche es ermöglichen, andere Aspekte interaktiver Belastungen zu untersuchen.

2.0 Leistungsbewertung von Rechnersystemen

2.1 Interaktive Lasten und Batch-Lasten

Die auf einem Rechnersystem abzuarbeitenden Lasten können in zwei Gruppen aufgeteilt werden:

1. Im interaktiven Betrieb setzt der Benutzer einen Befehl ab und erwartet unmittelbar eine Reaktion des Systems. Anschließend wird der Benutzer bis zur weiteren Eingabe eine gewisse Zeit verstreichen lassen. Diese 'Denkzeit' des Benutzers ist zur Verarbeitung der vom System gelieferten Informationen notwendig. Interaktive Lasten besitzen typischerweise einen geringen Bedarf an CPU-Zeit.

Als Beispiel soll hier die Programmentwicklung genannt werden, welche in Aufrufe eines Inhaltsverzeichnisses, Benutzung eines Editors, Anzeigen einer Datei und Compilation unterteilt werden kann.

In einem *Timesharing*-System kann eine größere Anzahl von Benutzern 'gleichzeitig' bedient werden, da sich viele dieser Benutzer in der 'Denkphase' befinden und nur wenige 'aktive' Benutzer um die Ressourcen konkurrieren.

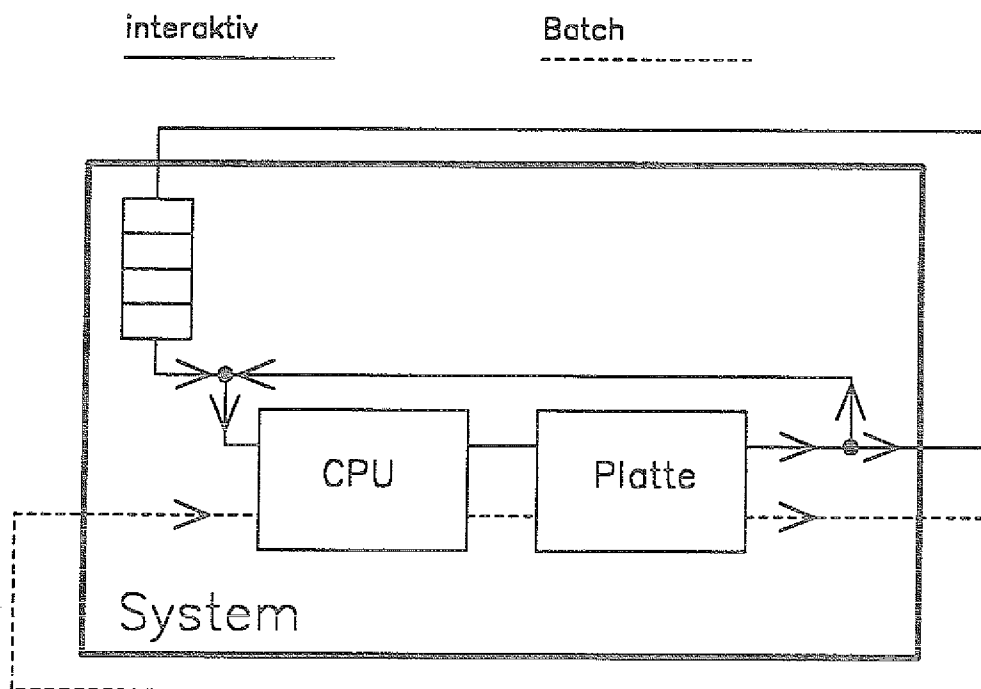


Abb. 1. Rechnersystem mit zwei Belastungsarten: Dargestellt ist das Modell eines geschlossenen Netzwerks. Dies ist durch die Rückkopplung des Ausgangs auf den Eingang charakterisiert. Offene Netzwerke besitzen diese Rückkopplung nicht (aus [BuDe]).

2. Batch-Lasten besitzen eine andere Charakteristik. Das vorrangige Ziel besteht nicht in der Realisierung einer möglichst geringen Antwortzeit, sondern in einer guten

Systemauslastung. Diese Lasten werden in eine Warteschlange eingereiht und abgearbeitet, sobald eine Recheneinheit zur Verfügung steht.

3. Mischbetrieb bedeutet die Abarbeitung von interaktiven Lasten und Batch-Lasten auf einem System (Abb. 1 auf Seite 5). Die interaktive Last wird hier bevorzugt behandelt, um kurze Antwortzeiten zu ermöglichen. Batch-Lasten werden im Hintergrund abgearbeitet.

Die Zuteilung der Recheneinheit erfolgt häufig auf Grundlage von sogenannten Prioritäten. Prozesse mit hoher Priorität werden bevorzugt behandelt. Durch entsprechende Zuordnung der Prioritäten zu den Prozessen ist es möglich, Batch-Lasten nur dann abzuarbeiten, wenn interaktive Benutzer das System nicht genügend auslasten. Die Strategie zur Vergabe der Prioritäten muß auf die Arbeitslast des Systems abgestimmt sein.

Im Rahmen dieser Arbeit wird die Gruppe der interaktiven Lasten näher betrachtet. Zur Untersuchung von Batch-Lasten kann an dieser Stelle auf [LINN] verwiesen werden.

2.2 Produktivität der Systembenutzer

Die Effektivität von interaktiven Sitzungen hängt, neben anderen zu betrachtenden Faktoren, vom Durchsatz des Prozessors, von der Komplexität der Software und der Benutzerschnittstelle ab.

Aufgrund der sinkenden Hardwarekosten und der steigenden Kosten für qualifiziertes Bedienungspersonal sollte ein Dialogsystem heute nicht mehr im Bereich der maximalen Maschineneffektivität arbeiten.

Verschiedene Studien stellen einen unmittelbaren Zusammenhang zwischen der Antwortzeit des Systems und der Denkzeit des Benutzers fest. Diese Studien kommen zu dem Ergebnis, daß die Benutzer bei kurzen Antwortzeiten des Systems produktiver arbeiten. Weiterhin bewirken längere Antwortzeiten verlängerte Denkzeiten des Benutzers. Somit ist der kosteneffektivste Kompromiß zwischen Maschinenleistung - und somit Kosten - und Produktivität des Benutzers zu finden.

Man kann die Antwortzeit eines Systems in Abhängigkeit der Hardwarekosten und Produktivität unterteilen (Abb. 2 auf Seite 7).

1. Im Bereich 1 mit den kürzesten Antwortzeiten ist die Produktivität nicht vom System begrenzt, also spiegelt sich eine erhöhte Rechenleistung nicht in erhöhter Produktivität wider. Die Kapazität der Hardware ist für diese Anwendungsklasse zu groß.
2. Der zweite Bereich zeigt für den Benutzer noch akzeptable Antwortzeiten. Unter der Voraussetzung eines unveränderten Benutzerverhaltens bewirkt eine Variation der Antwortzeit ebenso eine Veränderung der Produktivität.
3. Der Bereich 3 spiegelt die für den Benutzer nicht akzeptablen Antwortzeiten wider. Dieser Bereich kann auf den Benutzer zwei Effekte haben:
 - a. Die Produktivität des Benutzers nimmt deutlich ab.

- b. Der Benutzer ändert sein Nutzungsprofil mit dem Ziel, die Produktivität zu steigern.

Dies kann beispielsweise zur Eingabe eines komplexen Befehls anstelle mehrerer einfacher Befehle führen, was letztendlich wiederum zu einer Batch-artigen Arbeitsweise führt.

Die zu geringen Investitionen in eine Hardware werden mit steigenden Personalkosten bezahlt.

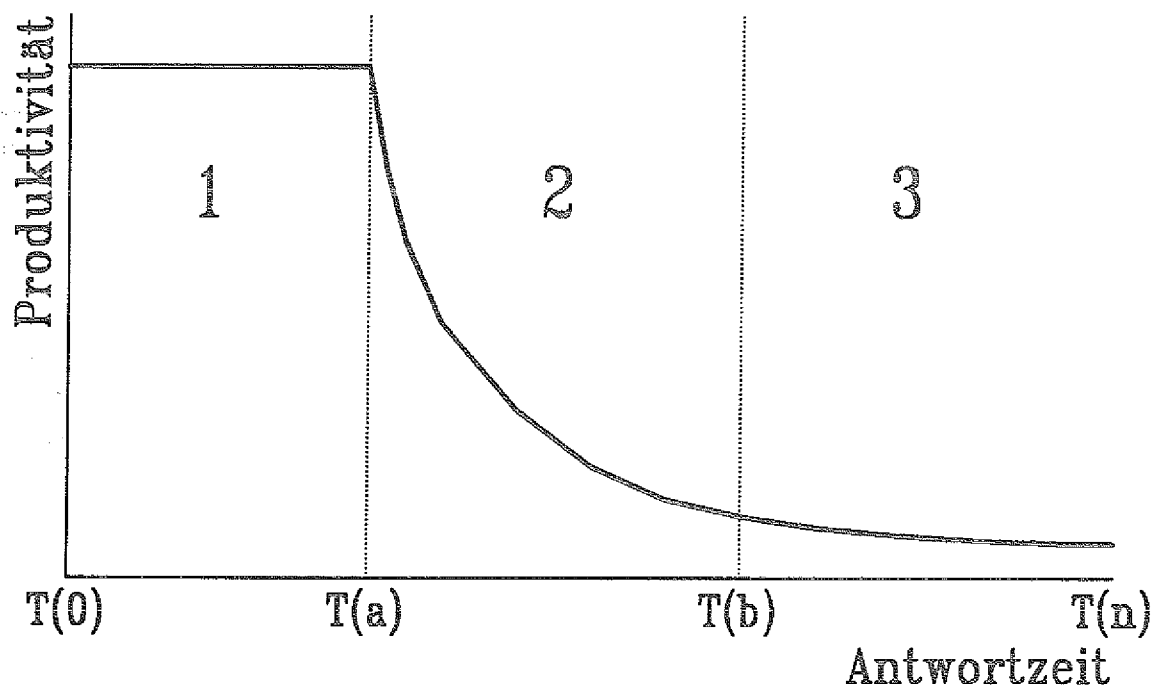


Abb. 2. Produktivität des Benutzers in Korrelation zur Antwortzeit: $T(b)$ muß in Abhängigkeit der jeweiligen Personalkosten und der Nutzungshäufigkeit des Systems im Einzelfall festgelegt werden ([Hei]).

Der Einfluß der Antwortzeit ist ebenfalls von der Nutzungsfrequenz abhängig. Es ist offensichtlich, daß ein Benutzer mit 2-3 Systemanfragen pro Stunde weniger von steigenden Antwortzeiten betroffen ist als ein intensiver Benutzer mit bis zu mehreren hundert Systemanfragen pro Stunde. Die freie Übersetzung eines Zitates von J. Martin [Mar] ergibt einen weiteren Anhaltspunkt:

"... Für komplexe Problemlösungen oder komplexe Aktivitäten am Terminal wird das Kurzzeitgedächtnis stark beansprucht. Unter Psychologen herrscht die übereinstimmende Meinung, daß der im Kopf zu lösende Komplexitätsgrad direkt von der Menge der Informationen abhängt, welche im Kurzzeitgedächtnis gespeichert werden können ..."

2.3 Ziele der Leistungsbewertung

Leistungsbewertungen werden sowohl für Batch- als auch für interaktive Rechnersysteme durchgeführt. Eine Leistungsbewertung kann aus verschiedenen Gründen notwendig werden. Die häufigsten sind:

- Vergleich von verschiedenen Rechnersystemen:
Diese Problemstellung tritt insbesondere auf, wenn ein neues System beschafft werden soll. Hierbei muß die erwartete Arbeitslast nachgebildet werden. Die zur Diskussion stehenden Systeme werden bewertet und die Ergebnisse in die endgültige Entscheidung für ein System einbezogen.
- Kapazitätsabschätzung eines einzelnen Systems:
Die Zielsetzung ist hier eine andere. Es soll eine Aussage darüber getroffen werden, welche Gesamtlast für den einzelnen Benutzer eines bestehenden Systems eine noch tolerierbare Antwortzeit zur Folge hat. Dies ist insbesondere deshalb wichtig, weil diese Antwortzeit nicht linear mit der Gesamtlast ansteigt, sondern eine exponentielle Abhängigkeit besteht (Abb. 3 auf Seite 9) [BuDe].
- Optimierung der Software:
Ebenso ist es möglich, die Leistung von Softwarepaketen zu beurteilen. Die Ergebnisse können sowohl dazu dienen, vergleichbare Software verschiedener Hersteller zu beurteilen als auch verschiedene Entwicklungsstufen eines Softwarepakets. Die Erfahrung zeigt, daß die Weiterentwicklung von Software nicht in jedem Falle zu einer Geschwindigkeitssteigerung führen muß. Somit ist es nicht immer sinnvoll, die neueste Version der benötigten Software zu installieren, wenn für den vorgesehenen Anwendungsfall eine ältere Version ausreichend ist, der Leistungsverlust der neuesten Version aber für die zu erwartende Anwendung in einem nicht akzeptablem Bereich ist.
Ergebnisse solcher Untersuchungen können z.B. zur Leistungssteigerung von Betriebssystemversionen, Compilern oder Datenbanksystemen berücksichtigt werden.

Zur Leistungssteigerung eines Rechners bei gegebener Arbeitslast können verschiedene Maßnahmen ergriffen werden. Zum einen kann der Rechner weiter ausgebaut werden, die maximal verfügbare Rechenleistung wird somit erhöht. Zum anderen kann die verfügbare Rechenleistung besser genutzt werden.

Falls eine Entscheidung für eines dieser Verfahren getroffen werden muß, ist zu beachten, daß die Kosten des Systems überproportional mit der Rechenleistung steigen. Es ist also unter Umständen sinnvoll, einen Teil der Investitionskosten des Gesamtsystems dafür aufzuwenden, die Engpässe des Systems zu untersuchen und einen Belastungszustand in diesem Bereich zu vermeiden.

Beispiel: Leistungssteigerung

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Leistungssteigerung durch

Antwortzeit

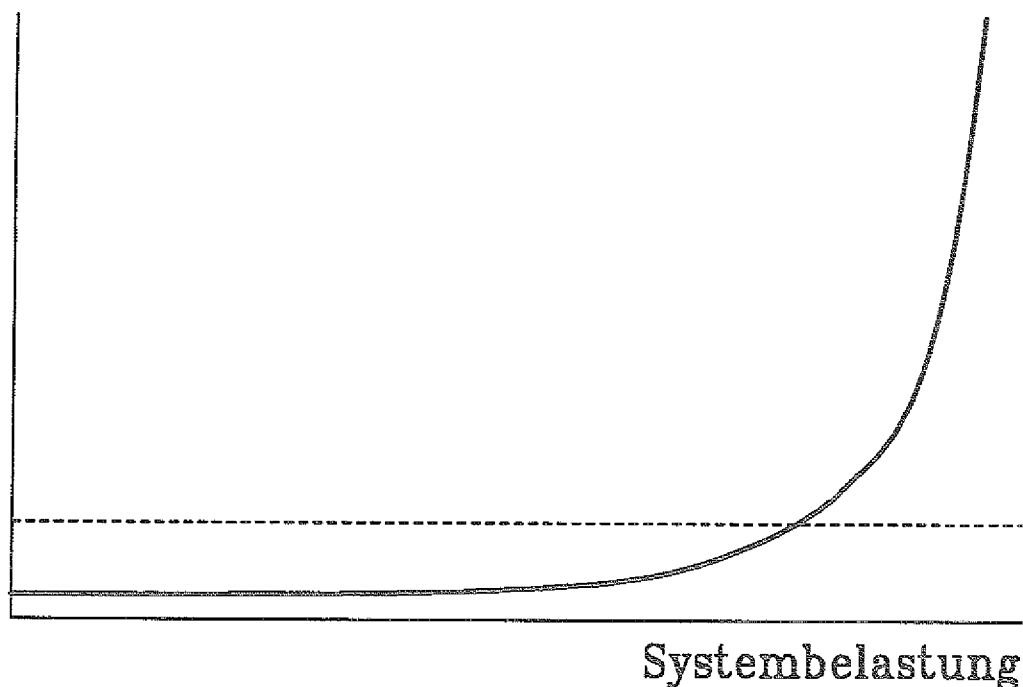


Abb. 3. Antwortzeit eines Prozesses in Abhängigkeit der Systembelastung: Eine mögliche Grenze der vertretbaren Antwortzeit ist eingezeichnet (gestrichelt). Diese Grenze muß im Einzelfall festgelegt werden.

2.4 Methoden der Leistungsbewertung

Zur Leistungsbewertung kann zwischen verschiedenen Vorgehensweisen unterschieden werden:

- Die Messungen werden auf realen Systemen durchgeführt. Zum einen kann die reale Arbeitslast abgearbeitet werden, zum anderen kann eine typische Arbeitslast ausgewählt werden. Weiterhin besteht die Möglichkeit zur Modellierung der Arbeitslast. Gute Leistungsbewertungen führen dann zu einer Ungenauigkeit der Ergebnisse von weniger als 5% [MORO].
- Das zu bewertende System wird simuliert. Dies kann notwendig werden, wenn das System in der zu untersuchenden Konfiguration nicht zur Verfügung steht. Die Ergebnisse einer Simulation können durchaus 15-20% von den tatsächlichen Gegebenheiten abweichen. Dies ist durch die erforderliche Modellbildung und die dadurch notwendigen Vereinfachungen zu erklären [MORO]. Hier werden zwei Schritte der Modellbildung benötigt:
 1. Modellierung des Rechnersystems;
 2. Modellierung der Arbeitslast.

Im Falle einer sehr detaillierten Modellbildung kann die erreichbare Ungenauigkeit in der Größenordnung der realen Systeme liegen, hierbei wird aber der notwendige Aufwand selten in vertretbaren Grenzen liegen.

Ergebnisse von Simulationen ermöglichen eine Abschätzung der zu erwartenden Leistungsfähigkeit.

Vor- und Nachteile der Simulation werden in [Ihrer] behandelt.

Für eine Leistungsbewertung stehen verschiedene Werkzeuge zur Verfügung. Im folgenden werden einige aufgeführt:

Das erste Verfahren stellt Untersuchungen im realen Systembetrieb an.

- **Monitorverfahren:**

Bei diesem Verfahren werden Zustandsänderungen aufgezeichnet und ausgewertet. Die Meßdaten können sowohl mit Hardware-Monitoren als auch mit Software-Monitoren ermittelt werden. Die Meßreihen werden im laufenden Betrieb aufgenommen und aufgezeichnet. Nach Abschluß der Messungen werden die Meßwerte aufgearbeitet und in eine aussagekräftige Form gebracht. Falls sich die Messung über einen längeren Zeitraum erstreckt, werden zur Begrenzung der Datenmenge oft nur stichprobenartig Messungen durchgeführt. Eine ausführliche Betrachtung der Monitorverfahren erfolgte in [Stim].

Die Verfahren der zweiten Gruppe beruhen auf künstlichen Arbeitslasten.

- **Mix-Verfahren:**

Diese Verfahren arbeiten eine Anzahl von einfachen, maschinennahen Befehlen ab, welche so ausgewählt sind, daß die 'typische Last' des Systems repräsentiert wird. Diese Befehle werden in der Maschinensprache des Zielsystems implementiert.

- **Kerne:**

Dies sind einfache Routinen, welche ebenfalls die 'typische Last' eines Systems nachbilden. Die Realisierung erfolgt allerdings nicht in Maschinensprache sondern in einer (systemunabhängigen) Hochsprache. Als Beispiel sollen hier Additionen, Wurzelberechnungen und Matrizenmultiplikationen angeführt werden. Einige Kerne können ebenso die Ein-/Ausgabeeinheiten testen. Diese Kerne lassen Rückschlüsse auf die maximale Leistungsfähigkeit des Systems zu. Ebenso können Engpässe im System analysiert werden [HeKa].

- **Benchmarks:**

Benchmarks sind Programme oder Programmsysteme, welche eine Last erzeugen, die möglichst große Übereinstimmung mit der tatsächlich zu erwartenden Last aufweisen. Benchmarks können in zwei Gruppen unterteilt werden: Synthetische Benchmarks und *Application*-Benchmarks. Letztere bestehen aus einer Menge realer Benutzerprogramme.

Das Ziel der Benchmarks besteht darin, das vorhandene oder erwartete Belastungsprofil möglichst genau nachzubilden, so daß die Ergebnisse Rückschlüsse auf den realen Betrieb zulassen. Für den *Application*-Benchmark werden ty-

pische, häufig benötigte Programmsysteme ausgewählt, so daß die echte Last gut angenähert wird. Das Hauptproblem besteht darin, eine repräsentative Stichprobe der Programme zu finden.

Synthetische Benchmarks bestehen in der Regel aus Kernen, welche verschiedene Betriebsmittel besonders stark belasten. Solche Kerne können Ein-/Ausgabeoperationen, Speicherzugriffe oder Nutzung der Recheneinheit beinhalten. Diese Kerne werden in der Regel parametergesteuert und erlauben nun eine flexible Nachbildung der einzelnen Lasten [Linn]. Die Schwierigkeit dieser Benchmarks besteht darin, die Kernparameter so festzusetzen, daß in der Gesamtheit die tatsächliche Last nachgebildet wird [Ben].

Die angeführten Verfahren sind in der Praxis nicht eindeutig zu trennen, so bestehen beispielsweise synthetische Benchmarks häufig aus einzelnen Routinen, welche nach obiger Einteilung den Kernen zugeordnet werden können.

Im gegebenen Fall wurde aus folgenden Gründen dem synthetischen Benchmark der Vorzug gegeben:

- Repräsentative interaktive Lasten sind kaum durch Anwenderprogramme nachzubilden, da die Reaktion des Benutzers ein grundlegender Bestandteil dieser Art der Systembelastung ist. Auch fehlen die Meßwerkzeuge zur Untersuchung des Benutzerverhaltens. Weiterhin ist das Benutzerverhalten, also die typische Rechnerbelastung, für große Systeme nicht stationär. Typische Programme können aufgrund der Heterogenität der Last nur schwer bestimmt werden.
- Eine Einteilung in Belastungsklassen ermöglicht eine gewisse Reproduzierbarkeit der erzeugten Last. Es wird die Rechnerbelastung analysiert und in Bereiche unterteilt. Für jeden dieser Bereiche wird eine typische Belastung bestimmt. Hieraus ergeben sich die Belastungsklassen.
Da der synthetische Benchmark auf diese Belastungsklassen zurückgreift, ist die erzeugte Last in Grenzen reproduzierbar.

2.5 Modell eines Transaktionssystems

Im folgenden wird ein Ansatz zur Leistungsbewertung von Dialogsystemen entwickelt. Der Ansatz geht von durchschnittlichen Systemanfragen aus, welche als Transaktionen oder Benutzeraufträge bezeichnet werden.

Definition der Antwortzeit

Der Begriff der Antwortzeit bezieht sich ausschließlich auf interaktive Prozesse. (Im Batch wird die Zeit zwischen Absenden des Jobs und Beendigung der Ausgabe als *turnaround-time* bezeichnet.)

Die Antwortzeit eines Systems kann in zwei Weisen definiert werden (Abb. 4 auf Seite 12):

1. Interne Antwortzeit:

Hier wird die Antwortzeit aus der Sicht des *schedulers* festgestellt. Es ist die Zeit zwischen der Eingabe des letzten Eingabezeichens und dem Beginn der Ausgabe der

Ergebnisse. Bei der Ausgabe sind keine rein informativen Meldungen, wie z.B. 'processing started ...', angesprochen. In der Regel liefern Software-Monitore diese internen Antwortzeiten als Ergebnis. Verzögerungen durch Netzwerksoftware (beispielsweise bei *Client-Server-Systemen*) werden nicht berücksichtigt.

2. Externe Antwortzeiten:

Diese spiegelt das Systemverhalten aus der Sicht des Benutzers wider. Dies ist die Zeitspanne welche zur Übertragung der Eingabedaten und deren Bearbeitung benötigt wird. Die für die Datenübertragung zum Endgerät benötigte Zeit wird hier nicht berücksichtigt. Hier wird zusätzlich zur internen Antwortzeit die Leistungsfähigkeit der Netzwerksoftware berücksichtigt.

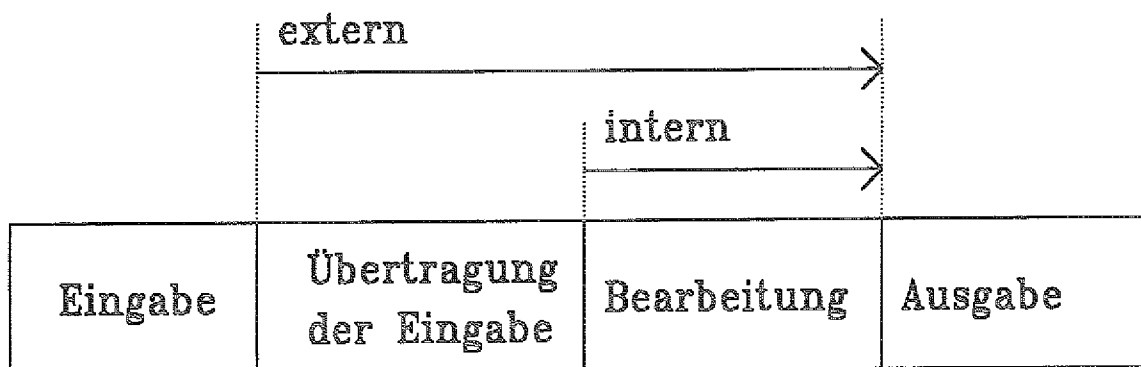


Abb. 4. Interne und externe Antwortzeit: Der Unterschied zwischen interner und externer Antwortzeit wird verdeutlicht.

Die unterschiedlichen Auswirkungen dieser Definitionen werden in [GuMe] untersucht.

2.6 Warteschlangenmodell nach Denning und Buzen

P.J. Denning und J.P. Buzen stellten Ende der siebziger Jahre ein Modell zur Beschreibung von Warteschlangennetzwerken vor [BuDe]. Dieses Modell kann sowohl zur Beschreibung von Batch-Lasten als auch von interaktiven Lasten eingesetzt werden. Im folgenden Zusammenhang wird nur der interaktive Betrieb betrachtet, obwohl das Modell auch Aussagen für den Batch-Betrieb ermöglicht. Denning und Buzen wendeten sich von den bis dahin zur Analyse notwendigen Markov-Ketten ab. Alle Gesetzmäßigkeiten beruhen in diesem Modell auf drei Voraussetzungen:

1. Alle verwendeten Größen müssen durch Messungen unmittelbar zugänglich sein. Die Gültigkeit der Ergebnisse sollte ausschließlich auf Annahmen beruhen, welche durch Beobachtungen des Systems unmittelbar verifiziert werden können.
2. Das System muß sich im Gleichgewicht befinden (*job flow balance*). Dies bedeutet für den Beobachtungszeitraum, daß die Zahl der eintreffenden Aufträge der Zahl der bearbeiteten Aufträge entspricht.

3. Die aktiven Betriebsmittel (*devices*) des Systems müssen homogen sein. Dies bedeutet, daß die Bearbeitung der Aufträge unabhängig ist von dem Füllungsgrad lokaler Warteschlangen und die mittlere Zeit der Bearbeitung nicht von der Warteschlangenlänge anderer Betriebsmittel abhängig ist.

Denning und Buzen haben gezeigt, daß diese Voraussetzungen letztlich zu denselben Aussagen über den Zustand des Systems führen, wie sie durch die traditionelle Betrachtung aufgrund von Markov-Ketten hergeleitet werden können [Klein]. Die Methode von Denning und Buzen, die sie *Operational Analysis of Queueing Network Models* nannten, liefert aber nur Mittelwerte über bestimmte Zustandsparameter (beispielsweise Durchsatz oder Antwortzeit), während die statistische Methode auch Aussagen über Verteilungen der Aufträge im Warteschlangennetzwerk bei vorgegebenen Bedingungen liefern. Für die praktische Anwendung reichen aber Mittelwerte im allgemeinen aus.

Es existieren verschiedene Anwendungsgebiete der vorgestellten Analysemethode:

1. Berechnung und Vorhersage des Durchsatzes:
Ausgehend von Meßwerten können weitere, nicht gemessene Größen, ermittelt werden. Die erzielten Ergebnisse können zur Vorhersage des Systemverhaltens für Zeiträume herangezogen werden, für die keine Meßwerte vorliegen.
2. Plausibilitätskontrollen:
Ein Widerspruch der vorliegenden Daten mit der entwickelten Theorie weist auf fehlerhafte Daten, fehlerhafte Meßwerkzeuge oder Nichtbeachtung einer kritischen Hypothese hin.
3. Analyse von Systemengpässen:
Aufgrund der gemessenen und berechneten Ergebnisse kann das Betriebsmittel bestimmt werden, welches den Engpaß des Gesamtsystems darstellt.
4. Bestimmung von Antwortzeiten:
Bei Kenntnis der Benutzerzahl, der durchschnittlich für einen Benutzerauftrag benötigten Rechenleistung und der Verwaltungszeiten des Systems kann eine Abschätzung der zu erwartenden mittleren Antwortzeit erfolgen.

Das von Denning und Buzen entwickelte Warteschlangenmodell wurde hier für die Betrachtung von Mehrprozessorsystemen erweitert. Die angeführten Zusammenhänge sind nur im Zustand des Gleichgewichtszustand eines Systems gültig. Dieser Gleichgewichtszustand wird auch als *steady state* bezeichnet.

Zur Beschreibung des Modells werden folgende Größen eingeführt, wobei sich der Index '0' stets auf das Gesamtsystem bezieht, während sich Indizes $i > 0$ auf die einzelnen Betriebsmittel beziehen. Größen ohne Index können sowohl auf einzelne Betriebsmittel als auch auf das Gesamtsystem angewendet werden.

Definition der verwendeten Parameter:

- X Die Anzahl der beendeten Transaktionen pro Zeiteinheit, auch als Durchsatz bezeichnet. Als Transaktion ist in Dialogsystemen jedes einzelne Benutzerkommando zu verstehen.

- V V_i beschreibt den Durchsatz des Betriebsmittels i bezogen auf den gesamten Systemdurchsatz.

$$V_i = \frac{X_i}{X_0}$$

- Hieraus läßt sich das *Gesetz des erzwungenen Durchsatzes* ableiten. Dies besagt, daß der Durchsatz in jeder Komponente des Systems den Durchsatz in jeder anderen Komponente bestimmt.
- T Beobachtungszeitraum. Dies ist die Zeit, in der das System dem Benutzer zur Verfügung steht. Es sind weder Wartungs- oder Reparaturzeiten noch Ausfallzeiten eingeschlossen.
- A Verwaltungszeiten des Betriebssystems (*Administration time*). Diese werden unter anderem durch die Prozeßverwaltung hervorgerufen.
- p Zahl der Recheneinheiten (CPU's) pro System
- N Zahl der Systembenutzer im Dialogbetrieb. Die Benutzer, welche auf eine Antwort des Systems warten, werden als aktive Benutzer¹ N_A bezeichnet. In gleicher Weise werden die denkenden Benutzer N_D charakterisiert. Diese belasten das System nicht.

$$N = N_A + N_D$$

- S Bearbeitungszeit eines Prozesses in einem Betriebsmittel des Systems (*Servicezeit*). Wartezeiten sind hier nicht eingeschlossen.

$$S_0 = \sum_i V_i \times S_i$$

- C Bearbeitungszeit eines Prozesses in der CPU. Der besseren Übersicht wegen wird im folgenden die *Servicezeit* der CPU mit C bezeichnet. C kann maximal $C_{\max} = p \times T$ betragen, falls während der gesamten Produktionszeit p Prozessoren zur Verfügung stehen.
- D mittlere Denkzeit des Benutzers. Dies ist der Zeitraum, in welchem das System auf eine Eingabe des Benutzers wartet. Diese Denkzeit ist ein typisches Merkmal des interaktiven Betriebes.
- R Zeit zwischen Eingabe des Benutzers und Fertigmeldung des Datenverarbeitungssystems (*Responsetime, Antwortzeit*).

$$R_0 = \sum_i V_i \times R_i + A(N) \quad i \geq 1$$

Im Bereich der Volllastung der CPU's gilt unter der Voraussetzung, daß alle Aufträge in der Reihenfolge des Eintreffens abgearbeitet werden:

$$R_0 = \frac{N}{p} \times C + A(N) + \sum_i V_i \times R_i \quad i: \text{alle Komponenten mit Ausnahme der CPU's}$$

¹ aus der Sicht des Systems

Hierbei kann die Verwaltungszeit A auch von der Zahl der aktiven Benutzer abhängig sein: $A = f(N)$.

U Auslastungsgrad des Betriebsmittels (*Utilization*).

$$U_i = \frac{N \times V_i \times S_i}{T}$$

Denning und Buzen geben für die aufgeführten Größen folgende Zusammenhänge an:

$$N = X_0(D + R_0) \quad (1)$$

Dieser Zusammenhang wird als *Antwortzeitgesetz* bezeichnet. Er gilt ausschließlich unter der Voraussetzung des Gleichgewichtszustandes.

Der Durchsatz des Gesamtsystems bei p Prozessoren kann durch Gleichung (2) bestimmt werden.

$$X_0 = \frac{U_0}{S_0} \quad (2)$$

Bei ausschließlicher Betrachtung der CPU gilt

$$X_0 = \frac{U_{CPU}}{C} \quad (2a)$$

Falls das System nicht ausgelastet ist, kann die Antwortzeit im sequentiellen Fall, also für nicht parallele Programme, aus

$$R_{\min} = S_0 + A(N) \quad N \leq p \quad (3)$$

bestimmt werden. Hieraus folgt bei ausschließlicher CPU-Belastung:

$$R_{\min} = C + A(N) \quad N \leq p \quad (3a)$$

Auf Mehrprozessorsystemen läßt sich die Auslastung der Recheneinheiten mittels

$$U_{CPU} = \frac{N \times C}{D + R} \times \frac{1}{p} \quad (4)$$

berechnen.

Diese Beziehungen können im Falle paralleler Programme nicht mehr angewendet werden. Es lassen sich aus den aufgeführten Modellvariablen verschiedene Dehnungsfaktoren bestimmen, der wichtigste ist der

TTSF (*transaction time stretch-factor, Verweilzeitdehnungsfaktor*). Dieser ist das Verhältnis der insgesamt benötigten Bearbeitungszeit zu der minimal möglichen Zeit. Betrachtet man das Betriebsmittel CPU, so gibt er an, welcher Bruchteil der Antwortzeit auf CPU-Belastung zurückzuführen ist. Einbenutzersysteme sollten im Idealfall eine TTSF von 1 besitzen. Sowohl für Einprozessorsysteme als auch für Mehrprozessorsysteme wird definiert:

$$TTSF = \frac{R}{R_{\min}}$$

Aus (3) folgt für die minimal mögliche TTsf von Mehrprozessorsystemen bei Vernachlässigung der Verwaltungszeit A:

$$R_{\min} = C \quad \text{bei } N \leq p$$

Falls die Antwortzeit R nicht bekannt ist, kann durch Umformung von Gleichung (4) der Verweilzeitdehnfaktor TTsf auch aus

$$TTsf = \frac{N}{U \times p} - \frac{D}{C} \quad (5)$$

bestimmt werden.

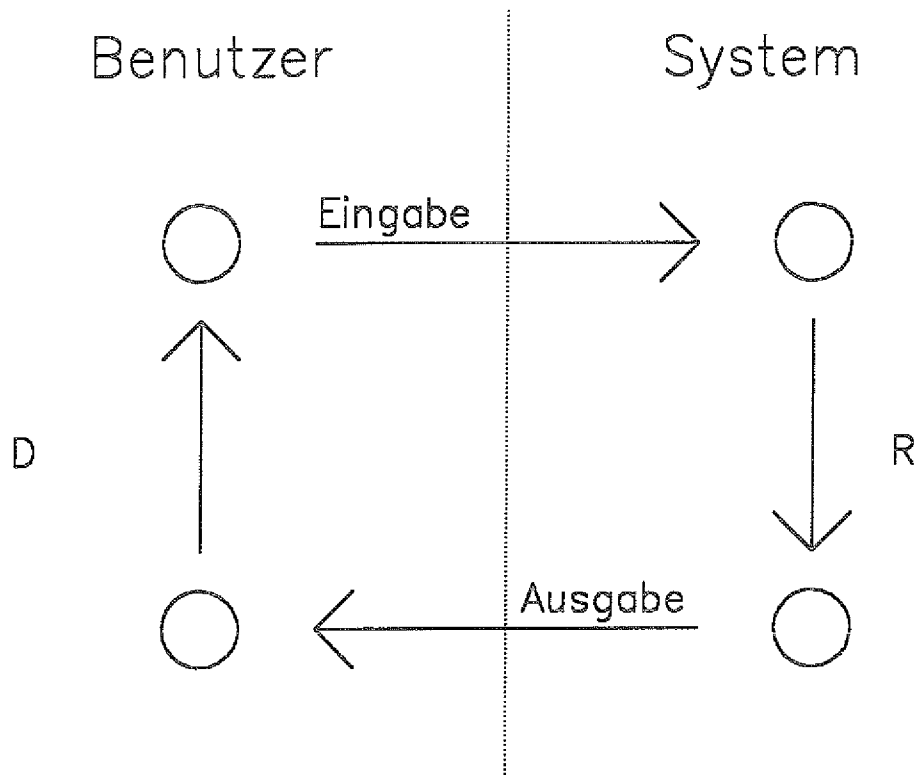


Abb. 5. Allgemeines Modell zur Beschreibung des Dialogbetriebes

Analyse von Systemengpässen

Das Modell nach Denning und Buzen kann ebenso zur Analyse von Engpässen in einem Rechnersystem eingesetzt werden. Das Rechnersystem wird als Netzwerk aufgefaßt. Dieses Netzwerk besteht aus verschiedenen Betriebsmitteln, sogenannter *devices*. Jeder dieser Betriebsmittel, beispielsweise CPU oder die Ein-/Ausgabepheripherie, besitzen eine eigene Warteschlange. In der Warteschlange des Betriebsmittels i warten L_i Prozesse. Die Zahl der aktiven Prozesse im Gesamtsystem bestimmt sich zu $N_A = \sum L_i$. Die Warteschlangenlänge eines Betriebsmittels kann durch *Little's Law* bestimmt werden:

$$L_i = X_i \times R_i$$

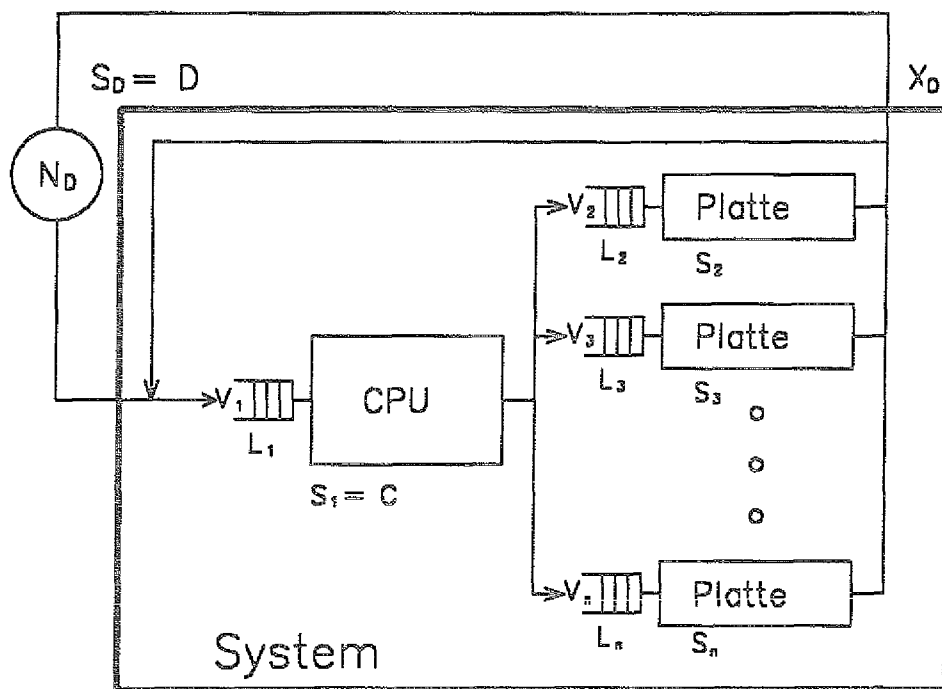


Abb. 6. Modell eines interaktiven Rechnersystems nach Denning und Buzen

Jeder eintreffende Prozeß wird von einer oder mehreren Komponenten bearbeitet. In einer Komponente i entstehen nun bei der Bearbeitung des Prozesses sowohl Wartezeiten in der Warteschlange als auch Servicezeiten S_i . In diesem Modell wird vorausgesetzt, daß keine gleichzeitige Belastung mehrerer Komponenten auftritt.

In der Realität treten solche Überlappungen auf², allerdings in vernachlässigbarem Umfang.

Es wird hierbei angenommen, daß die Besuchsrates und die mittlere Servicezeit invariant in bezug auf die Zahl der Benutzer N ist. Mit dieser Annahme läßt sich einerseits der Durchsatz und andererseits die Antwortzeit geschlossener Systeme analysieren. Die Komponente i ist gesättigt, falls die Auslastung $U_i = 1$ ist. Somit ist der Durchsatz zu bestimmen durch $X_i = \frac{1}{S_i}$. Diese Komponente arbeitet an ihrer Kapazitätsgrenze.

In jedem System kann nun eine Komponente B bestimmt werden, welche aufgrund ihres Sättigungspunktes der Engpaß (*bottleneck*) des Gesamtsystems ist. Dies ist jene Komponente, welche den größten Wert für $V_i \times S_i$ aufweist. Der Durchsatz des Gesamtsystems wird dann durch das Betriebsmittel B mit dem Engpaßverhalten bestimmt.

$$X_0 = \frac{1}{V_B \times S_B}$$

Dieses Betriebsmittel ist für große N voll ausgelastet ($U_B = 1$).

² insbesondere zwischen Recheneinheit und Ein-/Ausgabeperipherie

Durchsatz X_0

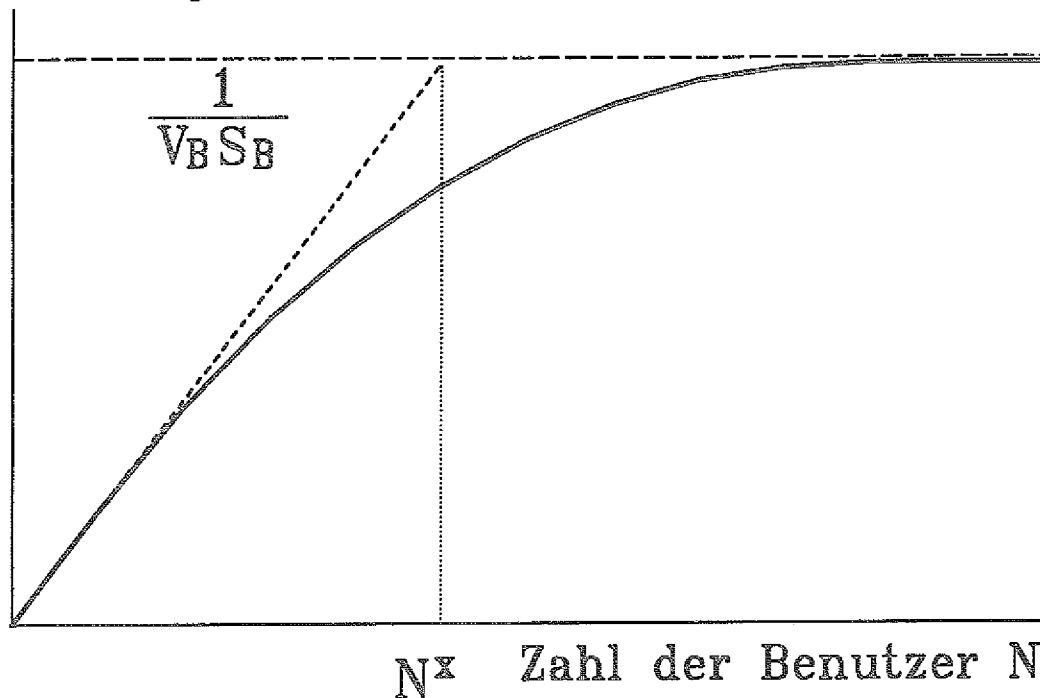


Abb. 7. Systemdurchsatz: Der gesamte Systemdurchsatz X_0 ist durch die Leistungsfähigkeit einer Komponente B beschränkt. ([BuDe])

$V_i \times S_i$ beschreibt die gesamte Belastung der Komponente i , welche von einem Prozeß hervorgerufen wird. Die kleinste vom System erreichbare Antwortzeit $R_{0,\min}$ ist zu bestimmen durch:

$$R_{0,\min} = \sum_i V_i \times S_i$$

Für den Spezialfall einer ausschließlichen Belastung der CPU mit $V_{CPU} = 1$ und $S_{CPU} = C$ folgt

$$R_{0,\min} = C$$

Unter der Bedingung, daß k Betriebsmittel in der Lage sind, k Prozesse so zu bearbeiten, daß diese sich nicht gegenseitig verzögern, gilt für den Sättigungspunkt N^* :

$$N^* = \frac{R_0}{V_B \times S_B}$$

Falls nicht die CPU den Systemengpaß darstellt, kann deren Rechengeschwindigkeit verringert werden, ohne daß die Antwortzeit sich spürbar verändert. Diese Maßnahme bewirkt eine erhöhte CPU-Auslastung U_{CPU} .

2.7 Das Kennlinienverfahren

Das Ziel des Kennlinienverfahrens besteht darin, das Last-/ Leistungsverhältnis bei unterschiedlicher Anzahl von Benutzern möglichst genau zu bestimmen.

Dies wird erreicht, indem während einer Meßreihe die Zahl der auf dem System vorhandenen Prozesse erhöht wird. Es kann jeder Last ein bestimmtes Systemverhalten zugeordnet werden.

Voraussetzung für dieses Verfahren ist eine homogene Belastung. Dies bedeutet die Spezifizierung möglichst weniger unterschiedlicher Benutzereingaben. Häufig reduziert sich dies auf eine, sogenannte *spezifische*, Last. Hierzu wird das heterogene Benutzerverhalten auf *eine*, homogene, Belastung komprimiert, welche im Mittel das Benutzerverhalten repräsentiert.

Im Gegensatz hierzu steht die heterogene Belastung, welche möglichst reproduzierbare und repräsentative Lasten definiert. Es werden mehrere Benutzer mit unterschiedlichem Verhalten nachgebildet.

Bei Dialogsystemen kann ein typisches Systemverhalten festgestellt werden, falls das Benutzerverhalten sich in den Beobachtungszeiträumen nicht ändert. Dies wird bei diesem Verfahren durch die homogene Belastung gewährleistet. Dieses Systemverhalten wird im folgenden beschrieben.

2.7.1 Antwortzeiten bei der Anwendung des Kennlinienverfahrens

Antwortzeiten stellen aus Benutzersicht ein wichtiges Merkmal eines Dialogsystems dar. Aus diesem Grunde werden im folgenden die zu erwartenden Kennlinien qualitativ untersucht.

Im einfachsten Fall besteht die Kennlinie aus zwei Bereichen (Abb. 8 auf Seite 20):

1. Die Antwortzeit ist nahezu unabhängig von der Zahl der Benutzer. Hier ist offensichtlich die Recheneinheit durch die anstehende Belastung nicht ausgelastet. Dieser Zustand kann beispielsweise bei Mehrprozessorsystemen durch nicht belegte CPU's bestehen. Im interaktiven Betrieb ist der Prozessor während der Denkzeit inaktiv, in diesem Zeitraum kann somit ein Auftrag eines anderen Benutzers bearbeitet werden ohne daß sich die Antwortzeit des Systems vergrößert. Der Durchsatz des Systems, X_0 , steigt mit der Zahl der Benutzer.
2. Die Antwortzeiten steigen proportional mit der Zahl der Benutzer. Für die Recheneinheit bedeutet dies den höchsten Auslastungsgrad, $U = 1$, während der Durchsatz konstant bleibt.

Der Sättigungspunkt bestimmt sich aus

$$N^* = \left(\frac{1 + D}{C} \right) \times p$$

Für die angegebenen Asymptoten gilt:

$$TTSF = \begin{cases} 1 & \text{für } N \leq p \\ \frac{N}{p} - \frac{D}{C} & \text{für große } N \end{cases} \quad (6)$$

Antwortzeit R

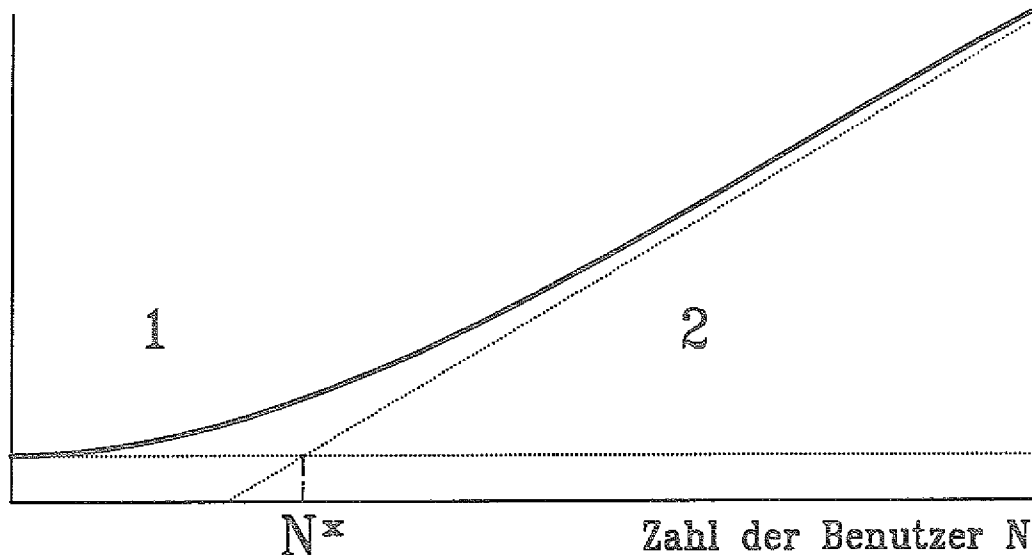


Abb. 8. Antwortzeit bei steigender Benutzerzahl und einem Engpaß

Vergleichbare Kennlinien entstehen durch Belastung des Speichers oder der Ein-/Ausgabekanäle.

Dieses erste Modell beruht auf der Voraussetzung, daß ausschließlich ein Betriebsmittel, in diesem Beispiel die Recheneinheit, für die Antwortzeiten verantwortlich ist.

Eine Voraussetzung hierfür ist die Auslegung der übrigen Betriebsmittel, wie beispielsweise Speicher, für erheblich höhere Belastungen.

Im folgenden sollen diese Betriebsmittel gleichzeitig untersucht werden (Abb. 9 auf Seite 21). Zur Vereinfachung werden hier nur Recheneinheit und Speicher betrachtet, die Ergebnisse sind allerdings leicht übertragbar.

Es bilden sich drei Bereiche aus:

1. Die Antwortzeit des Systems ist unabhängig von der Benutzerzahl.
Wie schon bekannt, ist dies auf die Nichtauslastung der Recheneinheit zurückzuführen.
2. Hier entsteht wiederum der Zustand der Vollast eines Betriebsmittels (z.B. CPU).
3. Dieser Bereich trat in den bisherigen Betrachtungen nicht auf. Der Durchsatz sinkt. Ursache für diesen erhöhten Anstieg der Antwortzeiten ist in der Volllastung des Speichers zu suchen; dieser ist nun der Engpaß im Gesamtsystem. Das System ist somit überlastet. Eine Ursache dieser Überlast kann beispielsweise die erhöhte *Paging-Rate* infolge eines zu kleinen Speichers sein.

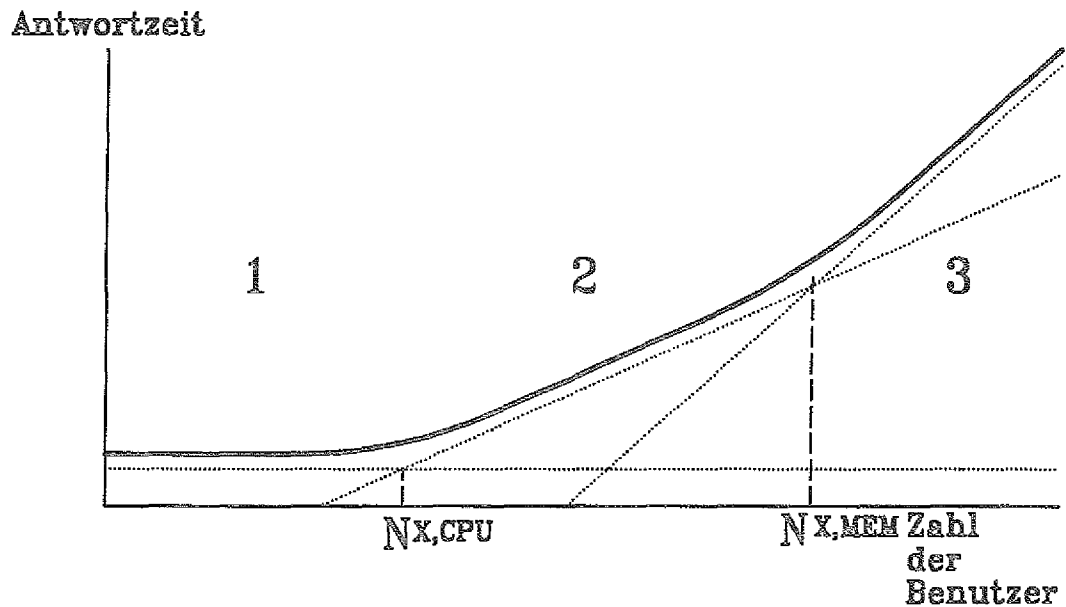


Abb. 9. Antwortzeit bei steigender Benutzerzahl und zwei Engpässen

2.7.2 Systemauslastung

Aus den bisherigen Untersuchungen läßt sich der Auslastungsgrad U des Systems ableiten. Wiederum sind drei Bereiche feststellbar (Abb. 10). Diese stimmen mit der bisherigen Einteilung überein.

$(1 - U_0)$ ist die Verwaltungszeit $A(N)$. Der dritte Bereich ist im Rahmen des gewählten Modells nicht analytisch zu erfassen. Für die Bereiche 1 und 2 gilt:

$$U = \begin{cases} U_0 & \text{für } N > (1 + \frac{D}{C}) \times U_0 \times p \\ \frac{N}{1 + \frac{D}{C}} \times p & \text{sonst} \end{cases}$$

In der Regel kann $U_0 \approx 1$ gesetzt werden. Da der Begriff der Auslastung sich hier ausschließlich auf die Arbeitslast³, welche die Verwaltungszeiten nicht einschließt, bezieht, sinkt die Auslastung im Bereich 3 mit steigender Benutzerzahl. Dies kann einerseits auf das angesprochene *Paging* zurückzuführen sein, andererseits verstärkt sich mit steigender Benutzerzahl der Zeitaufwand für die Prozeßverwaltung.

Die Auslastung U kann aus der Steigung der TTSTF-Kennlinien bestimmt werden:

$$\frac{1}{U} = \frac{\partial TTSTF}{\partial N}$$

³ die vom Benutzer nachgefragte Arbeit

Auslastung U

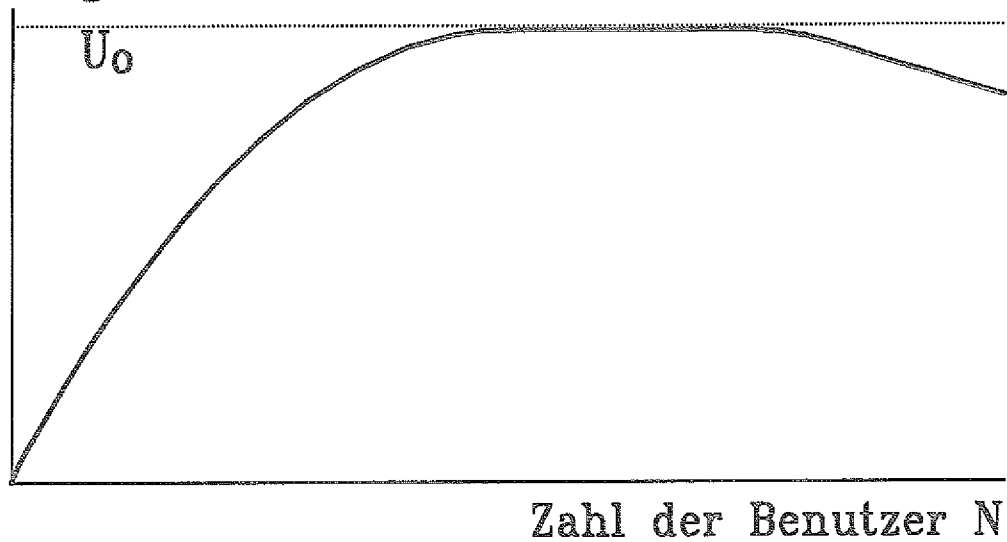


Abb. 10. Auslastung des Systems bei steigender Benutzerzahl: Der Wert U_0 läßt sich aus der Steigung der TTSTF-Kennlinien bestimmen.

2.7.3 Einfluß von Kenngrößen

Es wird nun der Einfluß verschiedener Parameter auf den Verlauf der Kennlinie betrachtet (Abb. 11):

1. Der mögliche Durchsatz der CPU, X_{CPU} , steigt.
Dies bewirkt neben einer Verminderung der minimalen Antwortzeit eine Parallelverschiebung der Asymptote in Richtung steigender Benutzerzahlen. Die Leistungssteigerung der CPU kann durch Betrachtung des Verhältnisses von Antwortzeit und CPU-Zeit bestimmt werden. Aus Gleichungen (2) und (5) folgt:

$$TTSTF = \frac{D}{C} \times \left(\frac{N}{X \times p} - 1 \right)$$

Bei Betrachtung zweier unterschiedlich schneller CPU's a und b ergibt sich unter der Bedingung gleicher Benutzerzahl N und gleicher Prozessorzahl p:

$$X_a \times \left(\frac{R_a}{D_a} + 1 \right) = X_b \times \left(\frac{R_b}{D_b} + 1 \right)$$

Unter der Voraussetzung gleicher Denkzeiten ($D_b = D_a$) und der Annahme $X_b = n \times X_a$ mit $n > 1$ folgt für die Antwortzeit R_b bei $N \geq N^*$:

$$R_b = \frac{R_a}{n} + \frac{D_a}{n} - D_a$$

Da $\frac{D_a}{n} > 0$ ist, sinkt die Antwortzeit bei einer schnelleren CPU nicht proportional der Leistungssteigerung der CPU.

2. Die Denkzeit steigt bei konstanter CPU-Belastung. Die TTSTF-Kennlinie wird in Richtung steigender Benutzerzahlen verschoben. Dieses Verhalten ist unmittelbar durch Gleichung (5) zu erklären. Steigende Denkzeiten bewirken eine Verminderung des TTSTF.
3. Sinkender System-Overhead
Hier tritt der Sättigungspunkt später auf und die Steigung der Asymptote nimmt ab. Die Beziehung $1/U_0 = \partial TTSTF / \partial N$ führt unmittelbar zu der Erkenntnis, daß flachere TTSTF-Kurven auf einen geringeren Systemoverhead hindeuten. In diesem Fall steigt U_0 an, wodurch zwangsweise der Systemoverhead $(1 - U_0)$ absinkt.

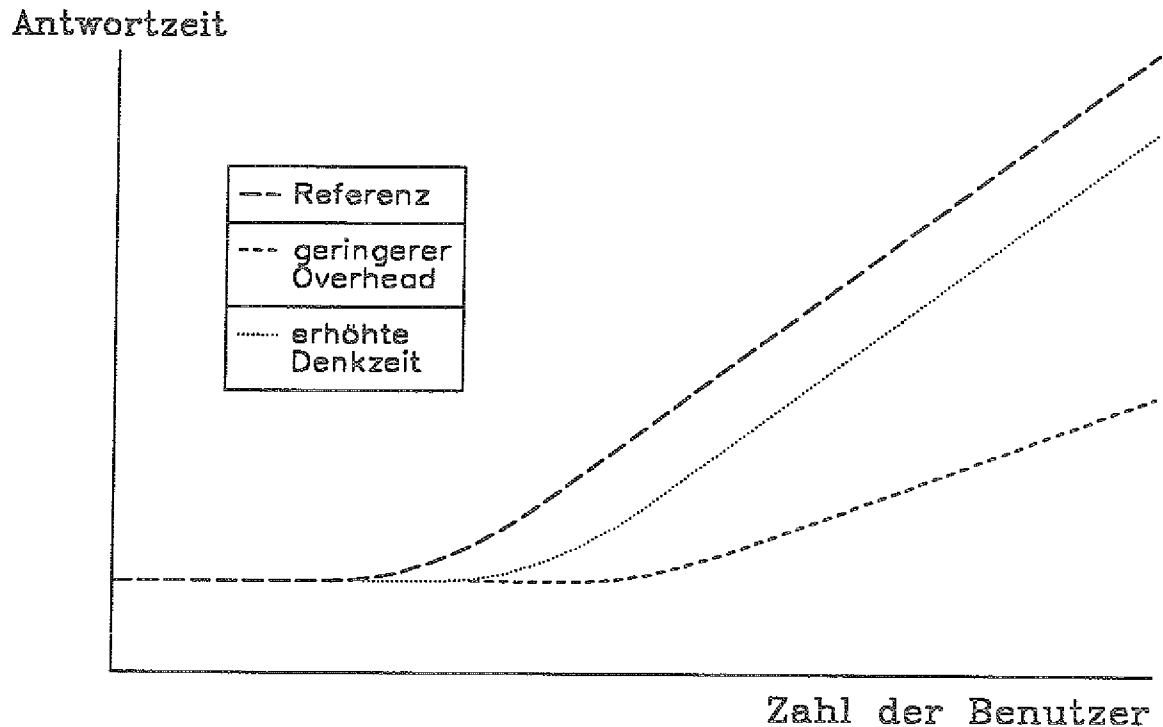


Abb. 11. Einfluß verschiedener Parameter auf die Kennlinie

24

25

26

27

28

29

30

3.0 Aufbau des Meßsystems

Ziel dieser Arbeit war es, ein Werkzeug zur Bewertung von Rechensystemen im interaktiven Betrieb zu entwickeln. Infolge der beschriebenen Überlegungen wurde die Methode des synthetischen Benchmarking gewählt, da diese für die Aufgabenstellung die beste Eignung aufwies. Das hier entwickelte System *Interbench* erlaubt es, dateigesteuert eine interaktive Last zu generieren und Antwortzeiten der einzelnen Prozesse zu ermitteln. Diese interaktive Last kann aus beliebigen Einzellasten bestehen. Alle erforderlichen Einstellungen können vor der Messung programmgesteuert abgespeichert werden. Wegen der flexiblen Konzeption können beliebig viele Dateien angelegt werden, welche diese Informationen speichern.

Aufgrund der weiten Verbreitung und der Möglichkeit des *multiprocessing* wurde das vorliegende System *Interbench* für das Betriebssystem UNIX entwickelt. Dieses Betriebssystem wurde seit 1969 von Ken Thompson bei den Bell Laboratories entwickelt. Zielsetzung war hierbei ein portierbares, dialogorientiertes Betriebssystem.

Zur Gewährleistung der Übertragbarkeit auf andere Rechnersysteme wurde die Programmiersprache B entwickelt, welche später von Dennis Ritchie zu C weiterentwickelt wurde.

Eine offene Lizenzpolitik der *American Telephone & Telegraph Corp.* (AT&T) und die Verwendung des Betriebssystems an den Universitäten für Forschung und Lehre führte zu seiner Verbreitung im wissenschaftlichen Bereich. Zu Beginn der achtziger Jahre entstanden mehrere, zum Teil untereinander nicht kompatible, UNIX-Versionen. In der Zwischenzeit entwickelten die maßgebenden Hersteller einen UNIX-Standard, heute kann UNIX System V als ein solcher angesehen werden [Bach], [Bou].

Für das Forschungszentrum Jülich bildet das Betriebssystem UNIX den Kern für ein offenes Workstationkonzept. Dies war ein weiterer Grund zur Realisierung des Benchmarksystems unter UNIX.

3.1 Prozeßverwaltung

Jedes Betriebssystem, welches mehrere Prozesse gleichzeitig verwalten kann, muß die Zuteilung der vorhandenen Ressourcen (z.B. CPU-Zeit oder Zugriff auf Massenspeicher) verwalten. Dieser Teil des Betriebssystems wird häufig als *scheduler* bezeichnet. Je nach Anwendungen können die Zielsetzungen der Prozeßverwaltung unterschiedliche Schwerpunkte besitzen:

- Gute Auslastung der Ressourcen;
- Minimierung der Zeit, in welcher der CPU keine Aufgaben zugewiesen sind;
- Hoher Systemdurchsatz;
- Vertretbare Antwortzeiten des Systems;
- Keine unendlich lange Blockierung von Prozessen.

In der Regel ist es nicht möglich, eine Strategie zu finden, die alle diese Anforderungen gleichzeitig erfüllt. Folgendes Beispiel soll die Problematik verdeutlichen:

Ein Prozeß großen Umfangs soll eine Ressource, z.B. die Druckausgabe, ausschließlich besitzen. Hieraus folgt, daß andere Prozesse nicht auf die Druckausgabe zugreifen können, solange sich dieser Prozeß im System befindet. Falls die Prozeßverwaltung nun sehr

kurze Prozesse stark bevorzugt, ist ein Zustand denkbar, in welchem dem umfangreichen Prozeß keine Rechenzeit zugewiesen wird, da ständig neue, kurze Prozesse auftreten (Abb. 12). Dies bedeutet wiederum, daß die Druckausgabe faktisch stillgelegt wird, da der einzige Prozeß mit Zugangsberechtigung keine Rechenzeit zugewiesen bekommt [CoKI], [TsBern].

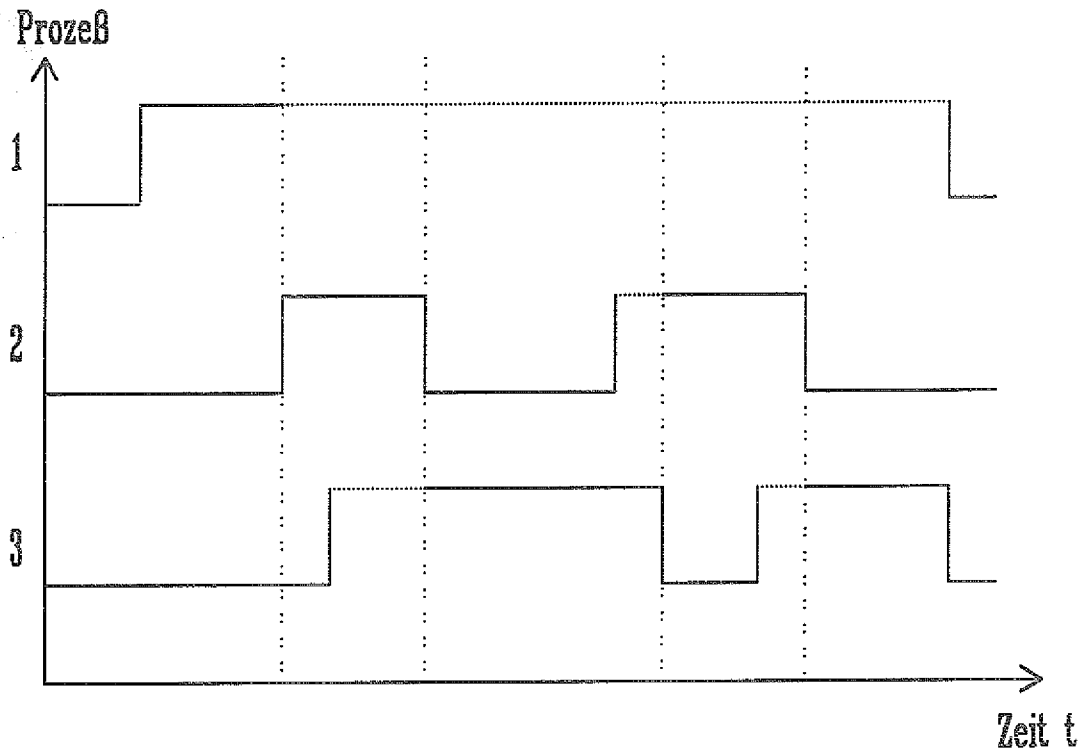


Abb. 12. Blockade eines langen Prozesses: Jeder der drei dargestellten Prozesse besitzt drei verschiedene Zustände: passiv, aktiv und blockiert (gestrichelt). Es wird deutlich, daß kurze Prozesse einen langen Prozeß (und somit auch die belegten Ressourcen) stilllegen können, falls das Betriebssystem dieses Problem nicht berücksichtigt. Ressourcen, die nur exklusiv zugeteilt werden, können somit den anderen Prozessen nicht zur Verfügung gestellt werden.

Modell zur Beschreibung der Prozeßzustände

Ein Ansatz zur Entwicklung von Verwaltungsstrategien soll im folgenden beschrieben werden (Abb. 13 auf Seite 27):

Jeder Prozeß im System nimmt genau einen der folgenden Zustände an:

1. Dem Prozeß wird Rechenzeit zugewiesen, das heißt er führt seine Aufgabe aus.
2. Der Prozeß ist blockiert. Dieser Zustand kann eintreten, falls Zugriff auf eine momentan belegte Ressource angefordert wird.
3. Der Prozeß wartet auf Zuteilung der CPU.

Eine Aufgabe der Prozeßverwaltung besteht darin, aus den wartenden Prozessen den auszuwählen, welcher auf der CPU bearbeitet werden soll. Spezielle Strategien der Prozeßverwaltung können [TsBern] entnommen werden.

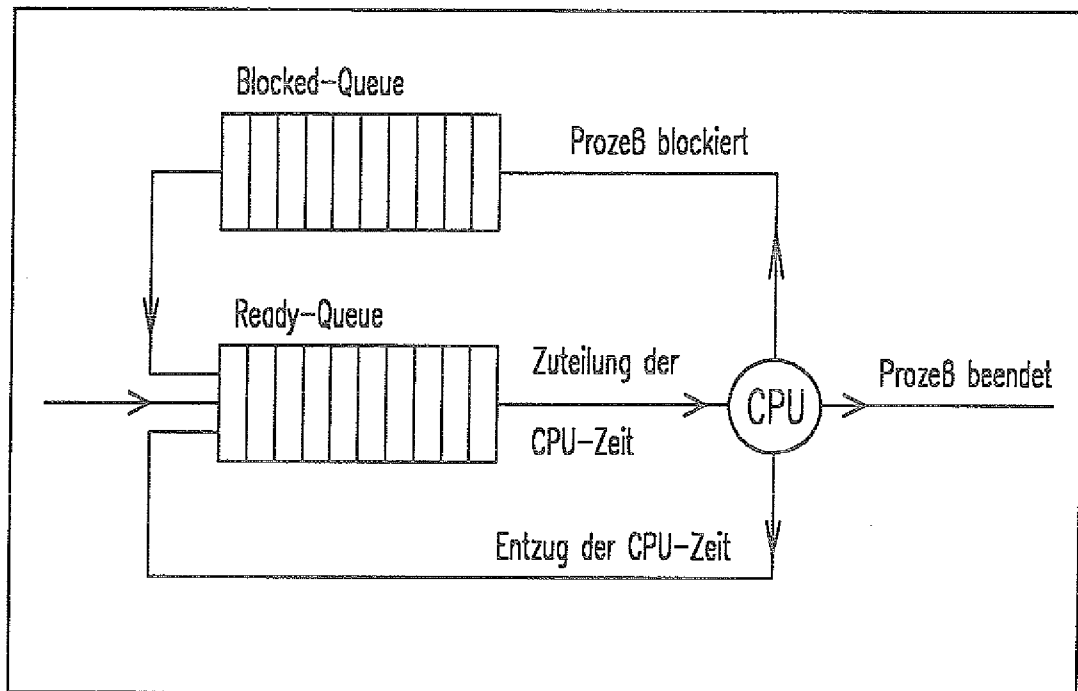


Abb. 13. Allgemeines Modell zur Beschreibung der Prozeßzustände: Blockierte Prozesse werden nach Ablauf dieser Blockade in die Warteschlange eingefügt (aus [TsBern]).

3.2 Asynchrone Prozesse

Asynchrone Prozesse sind nur in Betriebssystemen möglich, welche die Möglichkeit bieten, mehrere Tasks⁴ gleichzeitig⁵ ablaufen zu lassen. Dies kann auf unterschiedlichen Wegen erreicht werden:

1. Es ist nur ein Benutzer zugelassen. Dieser Benutzer kann aber mehrere Prozesse generieren. Die Prozesse können entweder vollkommen unabhängig voneinander sein - in diesem Fall ist eine Kommunikation der Prozesse nur auf Betriebssystemebene notwendig um Ressourcen wie Drucker oder Bildschirm zu verteilen - oder aufeinander aufbauen, so daß ein Datenaustausch zwischen den Prozessen notwendig wird. Dieses Verhalten wird oft mit *single-user, multitasking* bezeichnet.

⁴ Eine Task besteht aus einem Datenteil und einem Programmteil. Die Abarbeitung des Programmteils führt, unter Zuhilfenahme des Datenteils, zu einer Problemlösung.

⁵ gleichzeitig bedeutet im folgenden, daß jeder Task zyklisch eine gewisse Rechenzeit zugewiesen wird. Diese 'Zeitscheibe' ist so kurz, daß dem Benutzer ein Eindruck der Gleichzeitigkeit vermittelt wird.

2. Andere Betriebssysteme ermöglichen mehreren Benutzern gleichzeitig den Zugriff auf das Rechnersystem. Jeder dieser Benutzer darf aber nur einen Prozeß generieren. Diese Gruppe kann als *multi-user, singletasking* beschrieben werden.
3. Eine dritte Gruppe von Betriebssystemen ermöglicht mehreren Benutzer gleichzeitig den Zugriff, wobei jeder dieser Benutzer wiederum mehrere Prozesse generieren kann. Somit werden die Eigenschaften der beiden oben beschriebenen Gruppen vereint. Diese sehr leistungsfähige Gruppe kann als *multi-user, multitasking* charakterisiert werden.

3.2.1 Möglichkeiten zur Interprozeß-Kommunikation

Ein Betriebssystem, welches nach obiger Bewertung in die Gruppe der *multi-user, multitasking* Systeme eingeordnet werden muß, ist UNIX und alle daran angelehnte Betriebssysteme wie UNICOS oder AIX.

Jedes dieser Betriebssysteme bietet auch mindestens eine der folgenden Möglichkeiten zur Kommunikation der Prozesse untereinander [Bach] [IBM,a] [CRAY,b] :

- Einrichten von Nachrichtenkanälen, sogenannte *PIPES*:
PIPES sind Strukturen, welche es zwei Prozessen erlauben, Mitteilungen auszutauschen, ohne daß diese Prozesse synchron ablaufen müssen. Prozeß 1 schreibt in die *PIPE* sobald Informationen anliegen, während Prozeß 2 diese Information erst dann lesen muß, wenn er sie benötigt. Eine genauere Beschreibung dieser *PIPES* folgt in „Eigenschaften einer PIPE“.
- Senden und Empfangen von Signalen: Diese sind mit Interrupts vergleichbar, können aber vom Benutzer gesteuert werden.
- Benutzung von Semaphoren:
Diese verhindern einen gleichzeitigen Zugriffes mehrerer Prozesse auf eine Speicheradresse. Dies ist nur für sogenannte *Shared memory* Systeme sinnvoll und somit nicht auf allen UNIX-Systemen realisiert.
- Benutzung sogenannter 'Messages':
Dies ist vergleichbar mit einem Briefkastensystem, bei dem mehrere Prozesse Nachrichten in einem Briefkasten ablegen, wobei andere Prozesse die für sie relevanten Nachrichten abarbeiten können und alle anderen Nachrichten unbeeinflusst bleiben. Es besteht somit keine *FIFO*-Warteschlange aller Nachrichten mehr.

Alle oben aufgeführten Methoden ermöglichen eine asynchrone Kommunikation der Prozesse untereinander.

3.2.2 Eigenschaften einer PIPE

Da *Interbench* insbesondere stark auf die *PIPE*-Struktur zurückgreift, soll diese im folgenden genauer beschrieben werden:

Eine *PIPE* stellt in gewisser Weise eine Warteschlange mit FIFO-Charakteristik⁶ dar. Ein Element dieser Warteschlange ist eine beliebig geartete Meldung. Die Form und der Inhalt dieser Meldungen werden nicht vom Betriebssystem vorgeschrieben. Somit ist es ausschließlich Aufgabe des Anwenders, ein für seine Zwecke sinnvolles Protokoll zu entwerfen.

Diese Meldungen werden in die Eingabeseite der *PIPE* geschrieben und aus der Ausgabeseite herausgelesen. Hier gibt es wiederum keine Einschränkungen bezüglich der lesenden und schreibenden Prozesse. Im folgenden sollen einige der möglichen Kombinationen aufgeführt werden:

1. Im einfachsten Fall sind der lesende und schreibende Prozeß identisch, die *PIPE* dient als Datenpuffer.
Das *FIFO*-Verhalten der *PIPE* steht hier im Gegensatz zu einem Stack⁷, welcher ein *LIFO*⁸-Verhalten aufweist.
2. In der Regel kommunizieren genau zwei Prozesse über eine *PIPE* miteinander. Ein Prozeß schreibt Daten, sobald diese anfallen, der andere Prozeß liest, sobald er diese Daten benötigt. Dabei ist allerdings zu beachten, daß diese Daten ausschließlich in jener Reihenfolge gelesen werden können, in der sie vorher geschrieben wurden.
3. Ebenfalls ist es möglich, daß mehrere Prozesse auf die gleiche *PIPE* schreiben und diese Daten von genau einem Prozeß gelesen werden (Abb. 14 auf Seite 30). Das Betriebssystem ist in der Lage, mögliche Kollisionen, die durch den gleichzeitigen Zugriff mehrerer Prozesse auftreten können, zu verhindern. Dies bedeutet, daß während des Zugriffs eines Prozesses alle anderen Prozesse, welche ebenfalls Informationen in die Pipe schreiben wollen, blockiert werden, bis der erste Prozeß seinen Zugriff abgeschlossen hat.
4. In ähnlicher Weise können mehrere Prozesse von der *PIPE* lesen, hierbei nimmt aber der Kommunikationsaufwand zu, da jeder Prozeß erst eine Nachricht einlesen muß, danach erst überprüfen kann, ob er der vorgesehene Empfänger ist und gegebenenfalls diese Nachricht wieder auf die *PIPE* schreiben muß. Falls mehrere Prozesse gleichzeitig Information lesen wollen, werden wie in obigem Falle einige Prozesse blockiert. Hier sollten, falls möglich, andere Verfahren angewendet werden.

In der Regel wird die maximale Länge einer *PIPE* vom Betriebssystem begrenzt. Ein in eine gefüllte *PIPE* schreibender Prozeß wird suspendiert solange nicht genügend Platz geschaffen worden ist.

Falls die schreibenden Prozesse keine Verzögerung erfahren dürfen, sollte also darauf geachtet werden, daß der lesende Prozeß die Daten nicht dauerhaft langsamer verarbeitet als sie erzeugt werden. Andernfalls besteht aufgrund einer Überfüllung der *PIPE*, in ge-

⁶ First In First Out

⁷ Ein Stack ist eine Datenstruktur, welche im Falle eines Lesevorganges die zuletzt abgelegten Daten verfügbar macht. Dieses Verhalten wird insbesondere bei Implementierung von Rechenoperationen auf unteren Systemebenen häufig angewendet.

⁸ Last In First Out

wissem Grad eine zeitliche Abhängigkeit der Prozesse. Es ist wiederum ausschließlich Aufgabe des Anwenders, solche Probleme zu vermeiden.

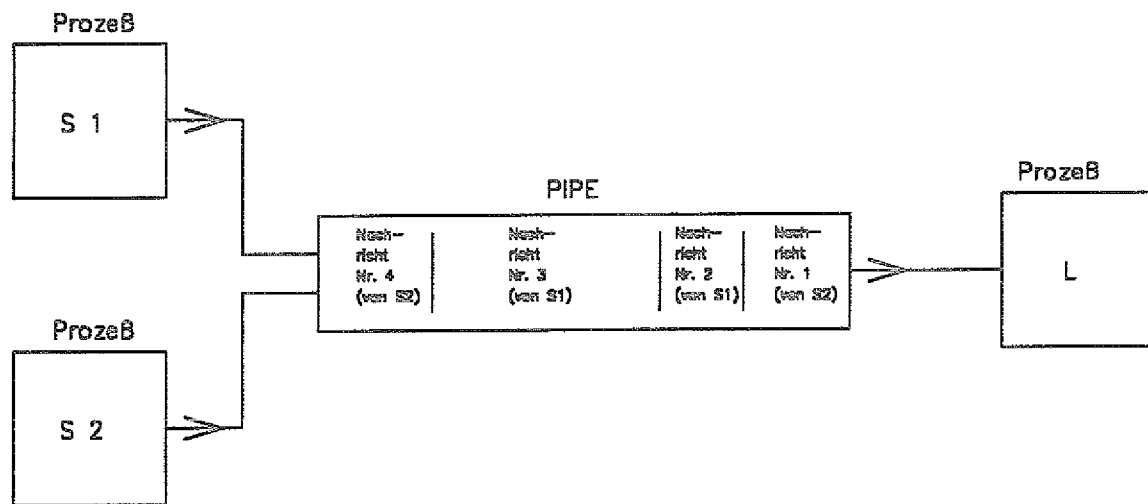


Abb. 14. Anwendung der PIPE-Struktur: Es besteht auch die Möglichkeit, daß mehrere Prozesse in eine PIPE schreiben. Die einzelnen Nachrichten können unterschiedliche Länge besitzen.

3.3 Zielsysteme

Für die Entwicklung leistungsfähiger Benchmarks werden oft spezielle Betriebssystemaufrufe eingesetzt, welche in dieser Form nur auf einem speziellen oder einer kleinen Gruppe von Rechnersystemen verfügbar sind. Je nach Zielsetzung des zu entwickelnden Benchmarks können nun verschiedene Strategien angewendet werden:

- Falls der Benchmark nur für eine Maschine entwickelt wird, können natürlich alle möglichen Betriebssystemaufrufe eingesetzt werden.
- Eine andere Art Benchmarks verfolgt das Ziel, verschiedene Rechner mit vergleichbaren Betriebssystemen zu untersuchen. Hier müssen die Systemaufrufe auf Verfügbarkeit überprüft werden und gegebenenfalls einige spezielle Aufrufe durch allgemein verfügbare ersetzt werden.

Das Benchmark-System *Interbench* ist dieser Gruppe zuzuordnen. Ein Problem bestand darin, die Prozeßkommunikation und Messung der Antwortzeit einheitlich zu gestalten, so daß der hierdurch bedingte Overhead im Verhältnis zur erzeugten Rechenlast auf allen Systemen in vergleichbaren Größenordnungen liegt.

Insbesondere unter AIX werden einige Möglichkeiten angeboten, welche der UNIX-Standard nicht zuläßt. Als Beispiel für eine solche nicht standardmäßige Erweiterung kann das Message-System angeführt werden.

Da bei Entwicklung des Systems auf Portabilität Rücksicht genommen wurde, konnten solche Erweiterungen nicht angewendet werden.

- Bei Vergleich grundsätzlich verschiedener Betriebssysteme muß der Benchmark auf das jeweilige Zielsystem abgestimmt werden. Es muß also für jedes Zielsystem ein eigener Benchmark entwickelt werden, wobei darauf zu achten ist, daß die ausgeführte Last vergleichbar bleibt. Auftretende Unterschiede beziehen sich insbesondere auf die Zeitmessung und die zur Verfügung stehenden Möglichkeiten der Prozeßkommunikation.

Interbench kann auf Betriebssystemen eingesetzt werden, welche sich an 'UNIX System V' anlehnen. Entwickelt wurde er auf einem 'AIX'-System unter PS/2, die Messungen auf den Systemen des Herstellers CRAY wurden unter UNICOS 6.0 durchgeführt. Weiterhin wurde ein RISC-Rechner der Serie RS/6000 des Herstellers IBM untersucht. Für einen Einsatz auf diesen Rechnertypen sind keine Änderungen des Benchmarksystems erforderlich.

Da ausschließlich standardisierte Systemaufrufe eingesetzt wurden, sollte die Portierung auf weitere Systeme keine grundlegenden Probleme bereiten.

IBM-System PS/2 Modell 70-A21

Das verwendete System stellt einen *Cluster* aus vier einzelnen Rechnern mit unterschiedlicher Leistungsfähigkeit dar. Einer dieser Rechner ist eine IBM 4381, die sogenannte *Primary Site*. Die übrigen Maschinen innerhalb des Clusters sind PS/2-Systeme unterschiedlicher Leistungsfähigkeit und Ausbaustufe. Diese Systeme arbeiten mit der Intel-CPU 80386 und gehören nicht zur Gruppe der Multiprozessorsysteme.

Alle vier Systeme sind innerhalb des Clusters selbstständige Rechner, wobei dem Benutzer aber nicht unmittelbar bekannt ist, auf welchem System seine Daten und Benutzerprogramme abgespeichert sind. Die hierzu notwendige Verwaltungsarbeit ist automatisiert und nicht unmittelbar zu beeinflussen.

Der beschriebene Cluster ist in der KFA zu Testzwecken installiert.

Die Messungen wurden auf einem Modell 70-A21 durchgeführt. Dieses System verfügt über einen 80386 Hauptprozessor und zusätzlich über einen Fließkommaprozessor 80387. Die Taktrate beträgt 25 MHz. Die installierte Festplatte stellt 120 MB Speicherplatz zur Verfügung.

IBM-System RS/6000-320

Herkömmliche Rechnersysteme besitzen einen umfangreichen Befehlssatz hoher Komplexität. Diese Befehlssätze sind auf Hochsprachenfunktionen ausgerichtet.

Untersuchungen der statistischen Zusammensetzung des Instruktionsmixes führten zu folgendem Ergebnis: 80% der abgearbeiteten Befehle greifen auf 20% des Befehlssatzes zurück. Die führte zur Entwicklung der RISC-Architekturen (Reduced Instruction Set Computer). Das Ziel dieser Entwicklung war es, wenige, einfache Befehle mit höherer Geschwindigkeit abzuarbeiten.

Das untersuchte System gehört zur Systemfamilie RS/6000 und ist der Gruppe der RISC-Systeme zuzuordnen. Die Zykluszeit beträgt 50 ns. Die Architektur erlaubt $4 \cdot 10^{15}$ Byte virtuellen Adreßraum. Ein Merkmal dieser Serie sind drei parallel arbeitende Einheiten:

1. Ein Prozessor ist für Sprungbefehle zuständig.
2. Ein weiterer Prozessor bearbeitet Festkommaprobleme.
3. Der dritte Prozessor führt Fließkommaoperationen durch.

Zur weiteren Systembeschleunigung werden einige effektive Maßnahmen getroffen:

- Einsatz hochoptimierender Compiler;
- Installation eines Festplatten-Subsystems;
- Unterteilung des Hauptspeichers in Speicherbänke.

Als Ergebnis dieser Maßnahmen ergibt sich laut Herstellerangaben eine Leistungsfähigkeit von 8 MFLOPS⁹ bei Untersuchung mit Hilfe des Linpack-Benchmarks, welcher von Dongarra entwickelt wurde [Don].

CRAY-Rechner

Die im Rahmen der Arbeit ebenfalls untersuchte CRAY X-MP/416 des Forschungszentrums Jülich wird überwiegend für Batch-Lasten eingesetzt. Im normalen Betrieb wird dem Benutzer zur Zeit keine Möglichkeit des interaktiven Betriebes zur Verfügung gestellt. Die allgemeine Nutzung im interaktiven Betrieb ist geplant. *Interbench* soll hier auch zu Untersuchungen über den möglichen Rahmen der interaktiven Nutzung eingesetzt werden. Tiefergehende Betrachtungen dieser Rechnersystemen werden in [Thom] und [BEGK] angestellt.

An dieser Stelle soll nur darauf hingewiesen werden, daß diese Systeme kein *Paging* durchführen, sondern ausschließlich *Swap*. Dies bedeutet, daß im Bedarfsfall der gesamte Prozeß aus dem Hauptspeicher entfernt und auf hierfür vorgesehenen Festplattensystemen zwischengespeichert wird.

3.4 Betriebsmittelbelastungen

Die Belastung des Dialogbetriebes kann sowohl durch statische als auch dynamische Analysen erfaßt werden.

Die statische Analyse beruht auf einer Auswertung der Abrechnungsdaten. Hierbei werden beispielsweise Häufigkeit des Aufrufes, verbrauchte CPU-Zeit, benötigter Speicher oder Zahl der Ein-/Ausgabeoperationen berücksichtigt.

Der Hersteller CRAY-Research bietet hierzu für seine Rechensysteme ein Soft- und Hardwarepaket an, welches diese Daten im Betrieb mißt und zur Verfügung stellt.

Die dynamische Nutzung kann durch folgendes Dialogsystem erfaßt werden:

⁹ Millionen Fließkommaoperationen pro Sekunde

- Eingabe
- Bearbeitung durch Datenverarbeitungssystem
- Ausgabe der Ergebnisse
- weitere Bearbeitung
- Ausgabe der neuen Ergebnisse
- Fertigmeldung des Systems
- Denkzeit des Benutzers, z.B. durch Erfassung und Umsetzung der Ausgabedaten
- erneute Eingabe

Ein ablaufender Prozess kann verschiedene Bereiche des Rechners unterschiedlich stark belasten. Im allgemeinen wird eine Unterscheidung getroffen zwischen Belastung der Recheneinheit, Belegung des zur Verfügung stehenden Speicherplatzes und Belastung der Ein-/Ausgabekanäle.

Im vorliegenden Fall wurden aufgrund der im interaktiven Betrieb überwiegenden Zustände folgende Belastungen nachgebildet und näher untersucht:

- Speicherzugriff auf den gesamten zugewiesenen Speicher und Teilbereiche hiervon;
- Belastung der Recheneinheit durch Fließkommaoperationen;
- Belastung der Ein-/Ausgabeeinheiten durch File-Zugriff (sowohl schreibend als auch lesend);
- Nachbildung der Denkzeit des Benutzers durch zeitweise Suspendierung des Prozesses. In diesem Zeitraum wird dem Prozeß keine Rechenzeit zugewiesen.

Diese Belastungssequenz kann mehrfach durchlaufen werden, wobei die angegebenen Belastungen einzeln durch Parameter von außen gesteuert werden können. Bei mehrfacher Abarbeitung obiger Sequenz wird das Benutzerverhalten insofern nachgebildet, daß in der Regel nicht intensiv mit ausschließlich einer Belastungsart gearbeitet sondern diese häufig gewechselt wird.

Belastung der Recheneinheit

Insbesondere die Realisierung der CPU-Belastung erforderte ein genaues Konzept zur Verhinderung compilerabhängiger Optimierung. Hierzu wurde die Analyse der einzelnen Optimierungskonzepte notwendig. Insbesondere der Hersteller CRAY-Research bietet Compiler mit ausgereiften Optimierungskonzepten an. Es wurde folgende Belastung gewählt:

```
dummy2=10*PI; dummy3 = dummy2 * (PI-1);
for (count=0; count < (Daten.cpupar); count++) {
    dummy2=dummy2*PI-dummy3;
    dummy2=dummy2*PI-dummy3;
    ...
    ...
    dummy2=dummy2*PI-dummy3;
    dummy2=dummy2*PI-dummy3; }
```

Dieser Belastungskern verfolgt mehrere Ziele:

1. Aufgrund der Voreinstellung der Variablen *dummy2* und *dummy3* verändert die Zuweisung

$$dummy2 = dummy2 \times PI - dummy3$$

nicht den Inhalt der Variablen *dummy2*.

Somit wird eine erneute Initialisierung der Variablen nach jedem Schleifendurchlauf überflüssig. Diese erneute Zuweisung hätte einen Speicherzugriff zur Folge, was vermieden werden sollte.

2. Die Einbeziehung der Kreiszahl *PI* in die Zuweisung bewirkt eine Belastung der Fließkommarecheneinheiten. Manche Rechnersysteme arbeiten Fließkommaoperationen erheblich langsamer ab, als dies bei Festkommaoperationen der Fall ist.
3. Innerhalb der Schleife wird die Zuweisung insgesamt zwanzigfach ausgeführt. Im praktischen Betrieb stellte sich heraus, daß in der Befehlssequenz mit nur einer Anweisung innerhalb der Schleife

```
for (count=0; count < (Daten.cpupar); count++)  
    dummy2=dummy2*PI-dummy3;
```

der Aufwand zur Verwaltung der for-Schleife bei manchen Systemen bis zu 30% der durch die gesamte Schleife hervorgerufenen Belastung beträgt. Die Wiederholung der Zuweisung innerhalb der Schleife vermindert diesen störenden Einfluß auf unter 2%.

4. CRAY-Compiler sind in der Lage, mehrere gleichlautende Anweisungen innerhalb einer For-Schleife zusammenzufassen. Aus diesem Grunde werden innerhalb der Schleife Variablen und keine Konstanten bearbeitet. In der aktuellen Version¹⁰ ist der Compiler nicht in der Lage, die Schleife zu optimieren. Dies ist in der Datenabhängigkeit der Schleifenelemente begründet. Der Compiler ist allerdings in der Lage, eine Schleife der Form

```
for (count=0; count < (Daten.cpupar); count++) {  
    dummy1=dummy2*PI-dummy3;  
    dummy1=dummy2*PI-dummy3;  
    ...  
    ...  
    dummy1=dummy2*PI-dummy3;  
    dummy1=dummy2*PI-dummy3; }  
zu optimieren. Das Ergebnis lautet:
```

```
dummy1 = dummy2*PI-dummy3;
```

¹⁰ dies ist z.zt Version 3.0.0

Die For-Schleife existiert nicht mehr. Eine Optimierung dieser Art verhindert eine automatische Skalierung wie sie durch *Interbench* realisiert wurde.

Bei Portierung des Benchmarksystems ist in jedem Fall zu untersuchen, ob eine Optimierung dieser Belastungsschleife erfolgt. Gegebenenfalls sind entsprechende Maßnahmen zu treffen.

5. Die Datenabhängigkeit der Belastungsschleife verhindert außerdem eine Vektorisierung. Dieser Effekt war im gegebenen Fall erwünscht.

Im Bereich der Belastung des Speichers und der Ein-/Ausgabeeinheiten wurde eine tiefergehende Analyse der Abhängigkeiten nicht erforderlich.

3.5 Meßmethode

Zur Aufnahme der Meßwerte wurde die Multitaskingfähigkeit der zur Verfügung stehenden Systeme ausgenutzt. Unter Beachtung der früher beschriebenen Einteilung wird als Mindestanforderung ein Betriebssystem der Gruppe *Single-user, multitasking* benötigt. Das System *Interbench* stellt sich während der Messung nicht als ein Prozeß im Speicher dar, sondern als eine Menge ähnlicher Prozesse eines Benutzers. Die generierten Prozesse können in drei Gruppen eingeteilt werden (Abb. 15 auf Seite 36):

- Der Steuerprozeß, welcher alle anderen Prozesse generiert, die Meßwerte sammelt und speichert.
- Der Dialogprozeß, welcher für die Kommunikation mit dem Benutzer zuständig ist und die abzuarbeitende Befehlsdatei einliest.
- Die Meßprozesse, welche im folgenden Tochterprozesse genannt werden. Diese führen die spezifizierte Arbeitslast und die notwendigen Messungen durch.

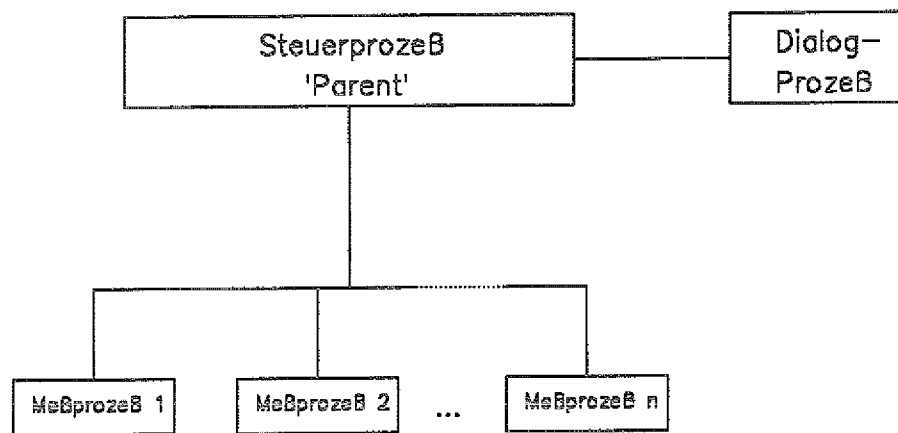


Abb. 15. Steuerung der Tochterprozesse: Zu verschiedenen Zeitpunkten kann eine Gruppe von Tochterprozessen mit unterschiedlichen Charakteristiken gestartet werden

3.6 Genereller Aufbau eines Tochterprozesses

Der Ablauf eines Tochterprozesses kann in drei Phasen aufgeteilt werden (Abb. 16):

- Die erste Phase folgt unmittelbar nach der Generierung des Prozesses und beinhaltet eine Bearbeitung der übergebenen Parameter.
- In der zweiten Phase ist der Prozeß suspendiert, er wartet auf das Startsignal durch den Steuerprozeß.
- Die dritte Phase übernimmt die eigentliche Messung und ist somit die wichtigste Phase. Die zugewiesene Arbeitslast wird ausgeführt und, falls dies spezifiziert wurde, Messungen durchgeführt. Diese letzte Phase wird durch ein Stoppsignal des Steuerprozesses abgebrochen, woraufhin der Tochterprozeß sofort terminiert.

Die Meßschleife der dritten Phase ist in Abb. 17 dargestellt.

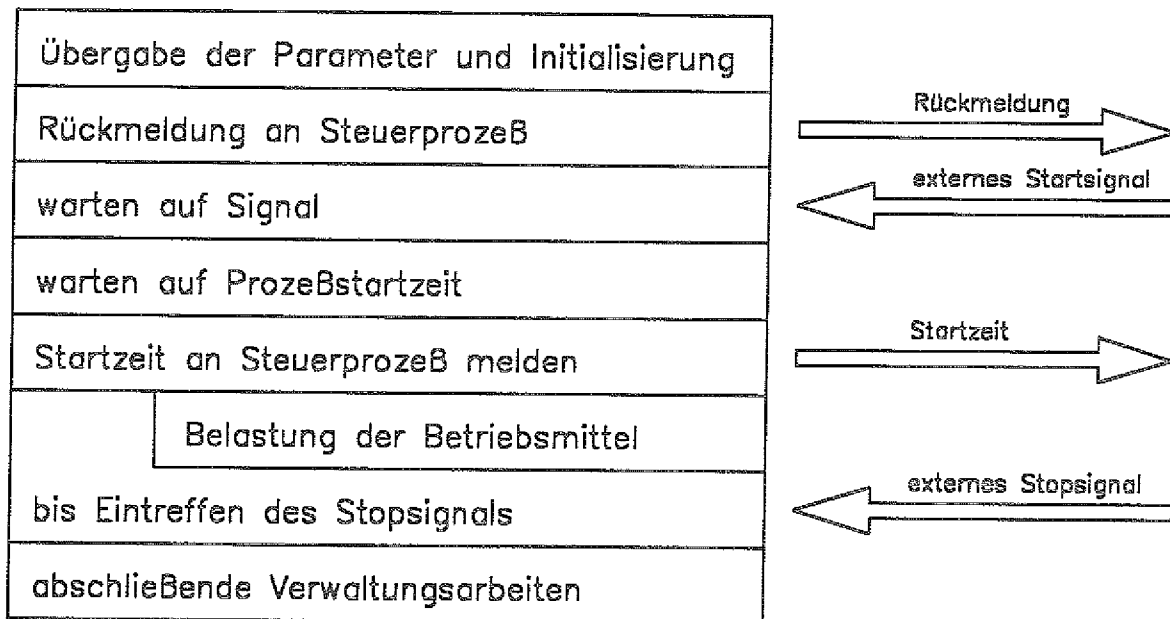


Abb. 16. Aufbau eines Tochterprozesses: Jeder Tochterprozeß bildet eine interaktive Last mit einer bestimmten Charakteristik nach. Diese Charakteristik wird durch die bei Generierung vorliegenden Parameter gesteuert.

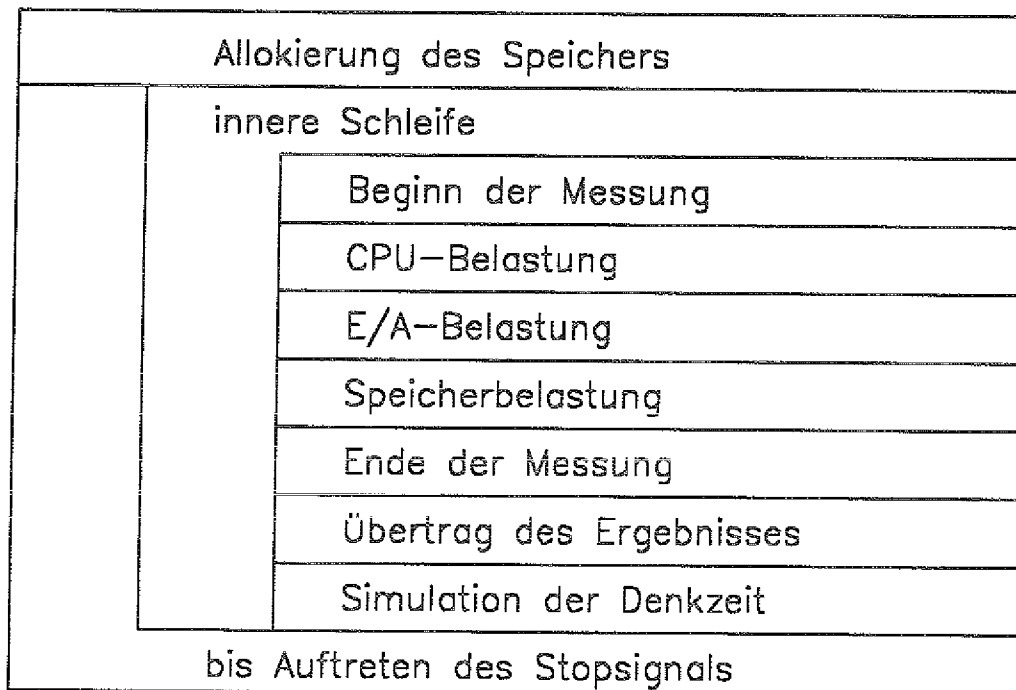


Abb. 17. Ablauf der Belastungsschleife: Nach jedem Schleifendurchlauf wird eine Messung durchgeführt

3.7 Meßmethode für einen interaktiven Benchmark

Ein Ziel des Systems *Interbench* bestand darin, den Einfluß einer großen Zahl von gleichzeitig interaktiv arbeitenden Benutzer auf den Durchsatz und die Antwortzeiten des benutzten Rechners zu untersuchen. Es mußte also eine Möglichkeit geschaffen werden, mehrere interaktive Benutzer gleichzeitig nachzubilden. Dies wurde erreicht, indem verschiedene Belastungskerne gleichzeitig abgearbeitet werden können. Vor der Messung kann der Anwender die Zahl der zu erzeugenden Prozesse einer Klasse bestimmen. Dies geschieht in Form einer Befehlsdatei des Filetyps 'BPROG'. Nachdem alle dort spezifizierten Prozesse generiert sind, simuliert jeder dieser Prozesse einen interaktiven Benutzer und mißt die Ausführungszeiten der zugewiesenen Arbeitslast. Somit dürfen zwischen diesen Meßprozessen keine direkten Abhängigkeiten untereinander auftreten, dies würde nicht dem gewählten Modell entsprechen.

Diese Abhängigkeiten der Prozesse untereinander können zwei Ursachen haben:

1. Die Abhängigkeiten werden durch das Benchmarksystem verursacht. Solche Abhängigkeiten sind nicht rechnerspezifisch und müssen daher vermieden werden, um eine Verfälschung der Meßergebnisse zu verhindern.
2. Die Abhängigkeiten werden durch das zu untersuchende Rechnersystem hervorgerufen. Es ist ein Ziel des Benchmarks, solche Abhängigkeiten aufzuzeigen.

Die Abhängigkeiten der ersten Gruppe wurden beispielsweise vermieden, indem jeder Meßprozeß bei Realisierung des E/A auf eine eigene Datei zugreift und nicht alle Prozesse dieselbe Datei benutzen. Falls das Betriebssystem alle diese Dateien auf derselben Festplatte speichert, entstehen aus diesem Grunde Abhängigkeiten, welche aber nicht durch den Benchmark verursacht werden, sondern durch das zu untersuchende Rechnersystem. Falls eine Untersuchung dieser Einflüsse nicht erfolgen soll, müssen entsprechende Maßnahmen vom Systemverwalter durchgeführt werden.

Eine durch den Benchmark bedingte Abhängigkeit ist durch die Meßwertübermittlung über eine *PIPE* gegeben. Der negative Einfluß hierdurch kann aber aus drei Gründen vernachlässigt werden:

1. Die Verwaltung der *PIPE* wird, auch aufgrund der gewählten Realisierung, auf einer niedrigen Systemebene durchgeführt. Dies hat einen entsprechend geringen *Overhead* zur Folge.
2. Die zu übertragende Datenmenge wurde möglichst gering gehalten.
3. Die *PIPE* ist physikalisch im Hauptspeicher realisiert. Dies hat eine sehr hohe Zugriffsrates zur Folge.

3.8 Realisierung systeminvarianter Parameter

Interbench wurde auch mit dem Ziel der problemlosen Portabilität entwickelt. Einerseits bezieht sich dies auf das Programmsystem. Andererseits aber sollten ebenso die Parameter so gestaltet sein, daß einzelne Parameterwerte auf verschiedenen Systemen vergleichbare Belastungen abbilden. Der Anwender muß keine systeminternen Zusammenhänge kennen.

Dies war im Falle der Angabe einer Denkzeit ohne zusätzliche Maßnahmen möglich. Im UNIX-System ist die Möglichkeit gegeben, die Dauer dieser Denkzeit direkt in Sekunden anzugeben. Die Transformation dieser Angabe in systembedingte Wartezeiten erfolgt durch das Betriebssystem selbst.

Ähnlich ist die Definition der Speicherbelastung möglich. Hier erfolgt die Angabe seitens des Benutzers in Kilobyte.

Die Belastung der Ein-/Ausgabeeinheit erfolgt durch Angabe der Zahl der gewünschten Ein-/Ausgabeoperationen. Diese Angabe ist wiederum maschinenunabhängig.

Diese Maschinenunabhängigkeit bezieht sich ausschließlich auf den logischen E/A. Die physikalische Realisierung des E/A ist maschinenabhängig. Nachteile der physikalischen Realisierungen können mit dem Benchmarksystem untersucht werden.

3.8.1 Skalierung der CPU-Belastung

Bei der Parametrisierung der CPU-Belastung muß grundsätzlich anders vorgegangen werden: *Interbench* benötigt intern eine Variable, welche die Zahl der benötigten Schleifendurchläufe festlegt (vgl. „Belastung der Recheneinheit“ auf Seite 33). Systeme unterschiedlicher Leistungsfähigkeit benötigen hier auch eine unterschiedliche Anzahl von Schleifendurchläufen. Die direkte Angabe dieser Anzahl seitens des Anwenders widerspräche der ursprünglichen Zielsetzung.

Ein Weg zur Entkopplung des Anwenders von systemspezifischen Parametern ist die Angabe der Rechenzeit in Sekunden. Hierzu muß jedes System Informationen darüber besitzen, wie diese Angabe abgebildet werden kann.

Im vorliegenden Fall geschieht dies durch Abarbeitung eines Skalierungsdurchlaufes: Auf jedem System muß als erstes diese Skalierung durchgeführt werden. Der Bereich einstellbarer CPU-Zeiten wird in mehrere Abschnitte unterteilt und entsprechende Stützpunkte festgelegt. Für jeden dieser Stützpunkte wird nun unter Zuhilfenahme des Pegasusverfahrens die notwendige Anzahl der Schleifendurchläufe ermittelt, welche genau die eingestellte CPU-Belastung hervorrufen.

Das Verfahren von Pegasus eignet sich für diese Aufgabenstellung besonders [EMR]. Es hat sich gezeigt, daß Iterationsverfahren höherer Konvergenzordnung nicht in jedem Falle konvergieren und somit für diese Anwendung ungeeignet sind [Jel].

Die gesamte Skalierung benötigt etwa 20-30 Minuten und muß für jedes System nur einmal durchgeführt werden. Die durch Skalierung ermittelten Werte werden in einer Datei namens 'bench.SCALE' abgelegt. Bei jeder Aktivierung des Benchmark-Systems wird diese Datei eingelesen. Anschließend erfolgt eine Abbildung der Eingabeparameter auf die benötigten internen Parameter.

Das Pegasus-Verfahren berechnet eine Nullstelle einer stetigen Funktion f auf einem abgeschlossenen Intervall $[a,b]$. Dieses Verfahren bestimmt eine Nullstelle des Intervalls durch fortgesetzte Verkleinerung desselben.

Es konvergiert immer, wenn die Voraussetzung $f(a) \times f(b) < 0$ erfüllt ist. Das Verfahren kann in 4 Schritte unterteilt werden:

1. Bestimmung der Sekanten
Es wird die Steigung der Sekante zwischen den Punkten a und b bestimmt. Anschließend kann der Schnittpunkt c dieser Sekanten mit der Abszisse berechnet werden.
2. Berechnung des Funktionswertes
Der Wert $f(c)$ wird bestimmt. Falls $f(c) = 0$ gilt, ist die gesuchte Nullstelle gefunden.
3. Festlegung eines neuen Einschlußintervalls
Falls $f(c) \times f(b) < 0$ gilt, wird $a = c$ gesetzt, andernfalls wird $b = c$ gesetzt.
4. Überprüfung der Abbruchbedingung
Die Iteration wird abgebrochen falls für eine vorgegebene Fehlerschranke ε gilt:

$$\left| \frac{f(b) - f(a)}{2} \right| < \varepsilon$$

Andernfalls wird eine weitere Iteration mit dem verkleinerten Intervall $[a,b]$ durchgeführt.

Für die gegebene Anwendung besitzt dieses Verfahren einerseits den Vorteil, daß nur innerhalb des Ausgangsintervalls $[a,b]$ eine Nullstelle gesucht wird. Andererseits wird eine Nullstelle sicher gefunden, falls eine solche innerhalb des Ausgangsintervalls existiert.

Die Fehlerschranke ε wurde hier auf 50 Millisekunden festgesetzt. Die Konvergenzordnung dieses Verfahrens beträgt 1.642.

3.8.2 Skalierung der E/A-Belastung

Es hat sich herausgestellt, daß auf einigen Systemen eine E/A-Belastung ebenso eine CPU-Belastung bewirkt. Zur Vermeidung einer Verfälschung des zugewiesenen Lastprofils wird auch hier eine Skalierung durchgeführt: Für einige ausgewählte E/A-Eingabeparameter wird die zusätzliche CPU-Belastung festgestellt.

Während der Simulation wird diese zusätzliche CPU-Belastung so berücksichtigt, daß die gesamte CPU-Belastung die Vorgabe des Anwenders nicht übersteigt.

3.9 Erzeugung von Prozessen mit verschiedenen Klassen

Der Benutzer bestimmt das charakteristische Verhalten der Meßprozesses durch Angabe eines Klassennamens. Der Steuerprozeß muß nun diesem Klassennamen die zugehörigen Parameter zuordnen. Hierzu wird auf eine Datei *bench.CLASSES* zurückgegriffen, welche jedem Klassennamen die entsprechenden Parameter zuordnet.

Im einzelnen sind dies:

- Belegter Speicher in Kilobytes;
- Benutzter Teilbereich des Speichers in Kilobytes;
- CPU-Belastung in Sekunden;
- Anzahl der Ein-/Ausgabeeinheiten;
- Simulierte Denkzeit des Benutzers in Sekunden;

- Startverzögerung des Prozesses bezogen auf den Startzeitpunkt;
- Maximale Streuung der Startverzögerung.

3.10 Befehlsdatei zur Benchmark-Steuerung

Da auf die Reproduzierbarkeit der Meßergebnisse Wert gelegt wird, muß es möglich sein, eine Messung mehrmals unter identischen Bedingungen durchzuführen. Daher ist es sinnvoll, eine Datei abzuarbeiten, welche das Lastprofil definiert. *Interbench* benutzt hierzu eine Datei mit dem Filetyp 'BPROG'. Alle notwendigen Parameter für eine Prozeßgruppe werden in einer Kommandozeile angegeben. Jede Kommandozeile der Datei enthält folgende Angaben:

- Die Zahl der zu erzeugenden Prozesse dieser Klasse;
- Der Name der abzuarbeitenden Klasse;
- Der relative Startzeitpunkt dieser Prozesse, bezogen auf den Meßbeginn;
- Der Anteil der Prozesse, welcher Messungen durchführt.

Da der Dateiname vor Beginn der Messung von *Interbench* erfragt wird, können mehrere solcher Befehlsdateien existieren.

Eine solche Befehlsdatei ist beispielhaft im Anhang dargestellt.

3.11 Ausgabedatei

Aufgrund der während einer Messung auftretenden Datenmenge ist eine Zwischenspeicherung der Daten durch den Steuerprozeß in einer Datei erforderlich. Hierdurch kann infolge der Erhöhung der E/A-Belastung die Messung verfälscht werden. Zur Vermeidung einer übermäßigen Verfälschung der Meßergebnisse werden diese Daten in möglichst kurzer Form abgespeichert, so daß der entstehende Fehler keinen entscheidenden Einfluß gewinnt.

Nach Abschluß der Messungen bereitet der Steuerprozeß diese Zwischendatei in eine sinnvolle Form auf. Insbesondere findet hier eine Formatierung der Ausgabedaten und Berechnung des Verweilzeitdehnungsfaktors TTSF statt.

Dem Benutzer werden folgende Informationen zur Verfügung gestellt:

- Relativer Zeitpunkt des Transaktionsstartes;
- Klasse des Prozesses;
- Zahl der aktiven Prozesse zu diesem Zeitpunkt;
- Benötigte CPU-Zeit C;
- Antwortzeit R;
- Verweilzeitdehnungsfaktor TTSF, aus R/C berechnet.

Ein Ausschnitt einer während der Untersuchungen entstandenen Ausgabedatei ist in Anhang C dargestellt.

3.12 Übergabe der benötigten Daten und Parameter

Zur Datenübergabe wurden im vorliegenden Fall zwei verschiedene Möglichkeiten realisiert. Der Steuerprozeß besitzt als einziger Prozeß alle Informationen über definierte Klassen und die entsprechenden Parameter. Das Betriebssystem kopiert bei der Generierung eines Tochterprozesses die Daten des Steuerprozesses in den Datenbereich des Tochterprozesses (Abb. 18). Die Daten des Steuerprozesses sind so strukturiert, daß jedem Tochterprozeß nur die relevanten Parameter kopiert werden. Eine Speicherbelastung durch einen zu großen Datenbereich des Tochterprozesses wird vermieden. Dieser Datensatz besteht nun in doppelter Ausführung und eine Änderung des Datenbereichs durch den Steuerprozeß bewirkt keine Änderung des Tochter-Datenblocks. Nach Erzeugung des Tochterprozesses greift dieser ausschließlich auf seinen eigenen Datenbereich zu.

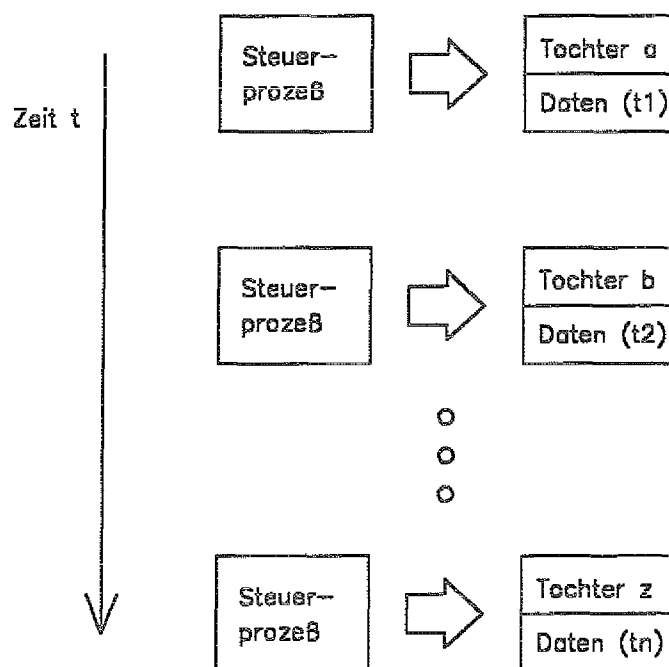


Abb. 18. Übergabe der Daten: Der Tochterprozeß erhält zum Zeitpunkt seiner Generierung eine Kopie der Daten des Steuerprozesses. Auf die Kopie hat ausschließlich diese Tochter Zugriff

Nachdem der Tochterprozeß generiert ist und seine notwendigen Parameter wie erläutert zugewiesen bekommen hat, erwartet er keine weiteren Daten des Steuerprozesses, sondern arbeitet ausschließlich seine zugewiesene Last ab.

Informationen, welche die Tochterprozesse dem Steuerprozeß übermitteln, werden grundsätzlich anders behandelt. Sie werden über eine *PIPE*-Struktur übertragen. Jeder Tochterprozeß schreibt die anfallenden Informationen in die *PIPE*, welche der Steuerprozeß abarbeitet, nachdem alle früheren Informationen in der *PIPE* - welche sowohl

von diesem als auch allen anderen generierten und aktiven Tochterprozessen stammen können - abgearbeitet wurden. Diese Vorgehensweise ist in der FIFO-Struktur der *PIPE* begründet und wirft im gegebenen Anwendungsfall keine Nachteile auf.

Der Tochterprozeß übermittelt auf diese Art sowohl Statusinformationen, wie zum Beispiel eine Rückmeldung über die vollendete Abarbeitung der zugewiesenen Last als auch die ermittelten Meßwerte.

Es bestand somit die Notwendigkeit zur Entwicklung eines sinnvollen Protokolls. Der maximale, einer *PIPE* zugeordnete Speicher ist begrenzt. Zur Vermeidung eines undefinierten Wartezustandes einzelner Meßprozesse aufgrund der Überfüllung der *PIPE* mußte die notwendige, zu übermittelnde Information auf ein Minimum begrenzt werden. Dies wurde insbesondere erreicht, indem die Meßwertübermittlung sehr kurz gehalten wurde.

3.13 Start der erzeugten Prozesse

Da die Meßergebnisse reproduzierbar sein sollten, mußte ein Weg gefunden werden, alle Tochterprozesse gleichzeitig zu starten. Andernfalls werden die Tochterprozesse zu nicht reproduzierbaren Zeitpunkten gestartet. Diese zufällige Beeinflussung ist auf den *Scheduler* zurückzuführen. Die Reproduzierbarkeit wurde ermöglicht, indem der Steuerprozeß nach Generierung des Tochterprozesses auf eine Rückmeldung wartet. Der Tochterprozeß wiederum führt unmittelbar nach seiner Generierung einige Verwaltungsarbeiten durch. Diese dürfen ebenso keinen Einfluß auf die Meßergebnisse nehmen. Nach dem Abschluß dieser Arbeiten wird die vom Steuerprozeß erwartete Rückmeldung gegeben. Anschließend suspendiert sich der Tochterprozeß und wartet auf das Startsignal des Steuerprozesses. In diesem Wartezustand wird dem Prozeß keine Rechenzeit zugewiesen.

Nachdem der Steuerprozeß nun die Rückmeldung erhalten hat, erzeugt er entsprechend der vorliegenden Daten weiterhin Tochterprozesse nach demselben Verfahren. Nach Ablauf dieser Generierungsphase sendet der Steuerprozeß ein Startsignal an alle erzeugten Tochterprozesse gleichzeitig. Diese beginnen nun mit dem definierten Lastzyklus und führen die entsprechenden Messungen durch.

3.14 Kommunikation der asynchronen Prozesse

Während der Messung sind drei unterschiedliche Prozeßarten im Speicher aktiv:

1. Der Steuerprozeß:
Dieser verwaltet die anfallenden Meßwerte und speichert sie in einem File.
2. Die Tochterprozesse:
Diese führen die Messungen durch.
3. Der Dialogprozeß:
Er stellt die Schnittstelle zum Benutzer dar und arbeitet zusätzlich eine Programmdatei für den Steuerprozeß auf. Da er für den Fall einer Benutzereingabe suspendiert ist, also keine Rechenleistung angefragt wird, werden die Meßwerte nur in geringem Maße verfälscht.

Zur Kommunikation der Prozesse untereinander werden 2 *PIPES* benötigt. *PIPES* sind in Kapitel „Eigenschaften einer *PIPE*“ auf Seite 28 beschrieben. Eine *PIPE* ist für die Kommunikation des Dialogprozesses mit dem Steuerprozeß reserviert. Hier werden vom Dialogprozeß die aufgearbeiteten Benutzeranweisungen an den Steuerprozeß übermittelt.

Auf die zweite *PIPE* greifen alle Meßprozesse zu. Die ermittelten Ergebnisse aller Meßprozesse werden über diese *PIPE* an den Steuerprozeß übermittelt. Es muß gewährleistet sein, daß diese *PIPE* ausreichend dimensioniert ist, die anfallenden Ergebnisse aufzunehmen. Dies ist die einzige Möglichkeit, die asynchrone Eigenschaft der einzelnen Prozesse beizubehalten.

Es gab mehrere Gründe nur eine *PIPE* zur Übermittlung einzusetzen:

1. Da alle Meßprozesse auf eine *PIPE* schreiben, werden die Ergebnisse in der zeitlichen Reihenfolge bearbeitet, in der sie ermittelt wurden. Bei Installation einer *PIPE* für jeden Meßprozeß ist es ohne zusätzlichen Aufwand nicht möglich, diese Reihenfolge zu bestimmen. Dieser zusätzliche Aufwand führt zu einer Verfälschung der Meßergebnisse.
2. Der Steuerprozeß muß nur eine *PIPE* verwalten. Die Verwaltung einer größeren Anzahl bedeutet einerseits einen erhöhten Rechenaufwand. Im Rahmen der vorliegenden Arbeit wurden für eine Meßreihe bis zu 300 Prozesse generiert. Dies hätte die Installation von ebensovielen *PIPES* zur Folge gehabt. Der Speicherbedarf einer *PIPE* ist statisch, das heißt es wird die durch das System vorgesehene Speichermenge belegt, unabhängig vom Füllungsgrad der *PIPE*. Dies führt zu einem deutlich gestiegenen Speicherbedarf des Benchmarks *Interbench*, wodurch weitere Verfälschungen der Meßergebnisse bedingt sind. Durch die gewählte Konfiguration wird dies vermieden.

4.0 Analyse der Arbeitslast

Aufgrund der früher angesprochenen Gründe wurde *Interbench* als synthetischer Benchmark konzipiert. Dies erfordert einerseits die Analyse der vorhandenen Systembelastung zur realistischen Nachbildung der tatsächlichen Last. Andererseits müssen zur Abgrenzung der interaktiven Lasten von Batch-Lasten eindeutige Parameter gefunden werden (Abb. 19).

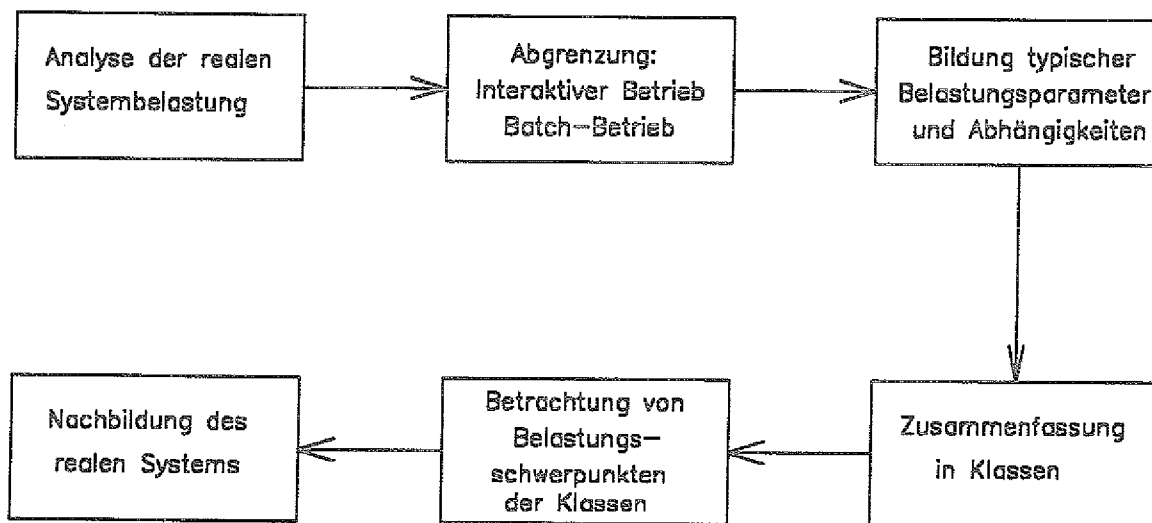


Abb. 19. Schritte der Modellbildung: Die Analyse der vorliegenden Belastung erfordert eine starke Abstraktion. Die anschließende Umsetzung dieser Abstraktion bewirkt eine erneute Annäherung an die Realität

Die folgende Analyse bezieht sich auf eine im Forschungszentrum Jülich installierte CRAY X-MP/416. Jede Portierung auf andere Rechnersysteme, auch auf vergleichbare CRAY-Systeme mit anderem Belastungsprofil, erfordert eine erneute Analyse. Bei vielen Systemen kann diese Analyse in vergleichbarer Form unter Zuhilfenahme der aufgeführten Parameter durchgeführt werden. Es ist jedoch in jedem Falle zu prüfen, ob dieses Modell übernommen werden kann oder eine Analyse aufgrund anderer Parameter vorgenommen werden muß.

Weiterhin muß der Auslastungsgrad der CRAY bestimmt werden. Dies ist erforderlich, da der Benchmark auch in diesem Punkt die reale Last modellieren soll. Eine überhöhte Auslastung des Systems hat zwangsweise Zugriffskonflikte zur Folge. Dies bezieht sich nicht nur auf den zur Verfügung stehenden Speicher, sondern auch auf die Zuteilung der Prozessoren auf die einzelnen Prozesse. In Grenzbereichen können nun in Abhängigkeit von der Strategie des *Schedulers* grundsätzlich unterschiedliche Ergebnisse auftreten.

Die Analyse der Systembelastung erfolgte aufgrund einer statistischen Auswertung der Belastung einer CRAY X-MP vom April 1991.

Die Produktionszeit der CRAY X-MP im Beobachtungszeitraum betrug 659,7 Stunden. Unter Beachtung der Tatsache, daß hier 4 Prozessoren zur Verfügung gestellt werden, ergibt sich eine Auslastung des Systems von 89 Prozent. Es ist anzumerken, daß der interaktive Zugriff auf dieses System eingeschränkt war. Ein normaler Benutzer besaß keinen interaktiven Zugriff. Auch für typische interaktive Tätigkeiten, wie Aufruf einer Dateiliste, war die Generierung eines Batch-Programmes notwendig. Da somit keine statistisch aussagekräftigen Daten von interaktiven Aufträgen zur Verfügung standen, wurden Batch-Lasten mit geringen CPU-Zeiten als quasi-interaktiv betrachtet und analysiert. Es bleibt anzumerken, daß bei der Batchverarbeitung auf der X-MP keine Denkzeiten des Benutzers, ein typisches Merkmal des interaktiven Betriebes, entstehen. Die eigentlich interaktiven Tätigkeiten werden auf einem vorgeschalteten System ausgeführt.

4.1 Ausgangsdaten

Im Beobachtungszeitraum wurden unter Zuhilfenahme eines Softwaremonitors (SAR) die Systembelastung der CRAY X-MP ermittelt. Im einzelnen wurden für jeden Prozeß ermittelt:

- Antwortzeit des Systems R;
- Benötigte CPU-Zeit;
- Mittlerer Speicherbedarf;
- Ein-/Ausgabeintensität.

Es muß nun eine sinnvolle Grenze gefunden werden, um Batch-Lasten von interaktiven Lasten zu trennen. Einerseits sollen vergleichsweise kurze Batch-Lasten abgegrenzt werden, andererseits muß beachtet werden, daß einige interaktive Anwendungen ähnliche Rechenzeiten aufweisen. Insbesondere sollen Compilerläufe als interaktiv betrachtet werden. Eine Analyse der zugrundeliegenden Statistik weist Compilerlaufzeiten des Fortran77-Compilers von durchschnittlich 16 Sekunden auf. Zur weiteren Betrachtung müssen allerdings noch andere Parameter gefunden werden. Der interaktive Benutzer erwartet eine Systemantwort des gestarteten Prozesses in vertretbaren Zeitabständen. Es bietet sich also die Betrachtung der Verweilzeit im System an, die *transaction-time*. Da, wie schon erwähnt, Batch-Aufträge mit geringer CPU-Zeit als quasi-interaktiv betrachtet werden, ist diese *transaction-time* hier gleichzusetzen mit der *turnaround-time* des Batch-Betriebes. Die alleinige Betrachtung der *transaction-time* führt allerdings zu keinem aussagekräftigen Ergebnis, da Batch-Lasten einerseits aufgrund der *Scheduler*-Strategie, andererseits aufgrund des Charakteristikums der hohen Rechenzeiten entsprechende Verweilzeiten im System besitzen.

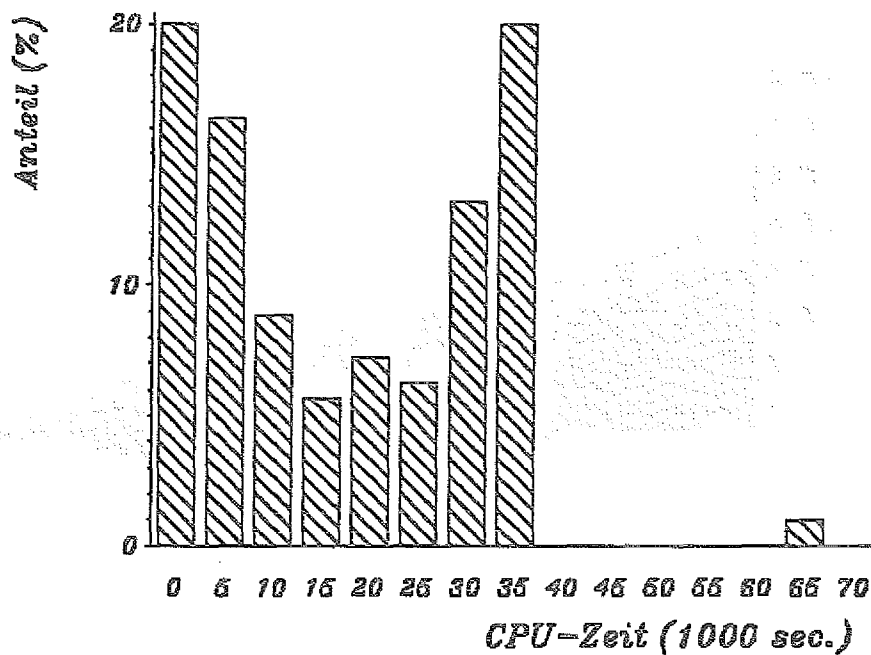


Abb. 20. Verteilung der Rechenzeit der CRAY X-MP des ZAM: Die zugrundeliegenden Daten wurden im April 1991 erhoben. Es ist der Anteil an der gesamten im Beobachtungszeitraum benötigten CPU-Zeit aufgetragen.

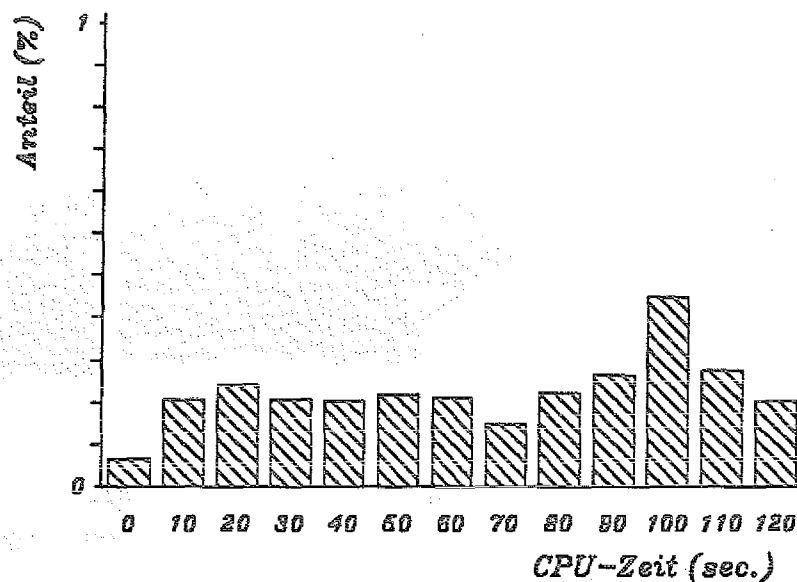


Abb. 21. Darstellung des relevanten Bereiches: Die CRAY X-MP wird nur in sehr geringem Maße interaktiv belastet. Die gesamte interaktive Belastung beträgt weniger als ein Prozent der verfügbaren Rechenleistung. Dies ist für einen Vektor-Supercomputer typisch.

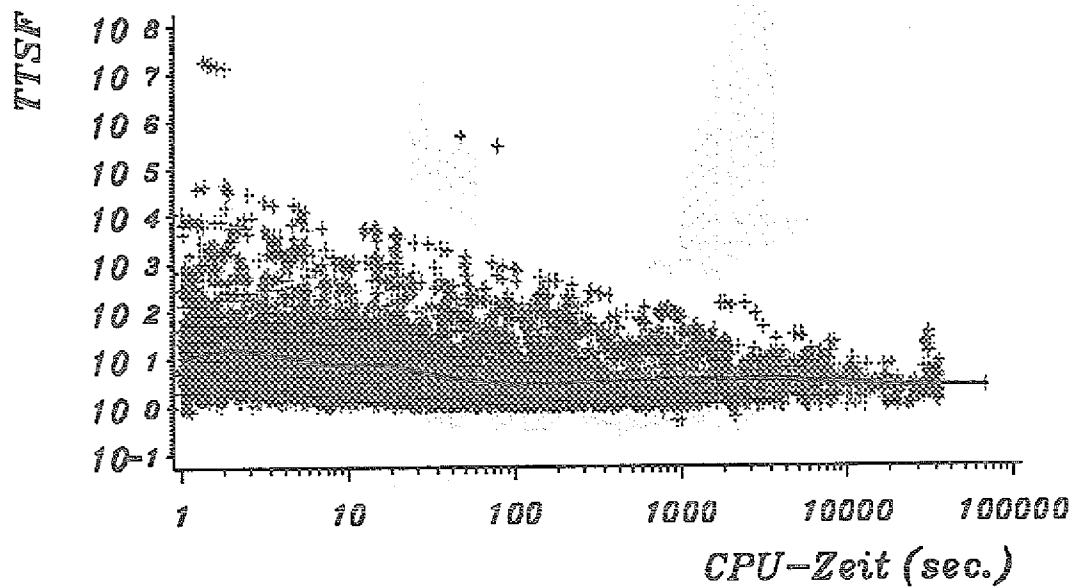


Abb. 22. Darstellung des TTSF: Aufgetragen ist der für jeden Prozeß im Analysezeitraum der TTS-Faktor über die benötigte CPU-Zeit. Die durchgezogene Linie stellt den Mittelwert aller TTS-Faktoren mit gleicher CPU-Zeit dar.

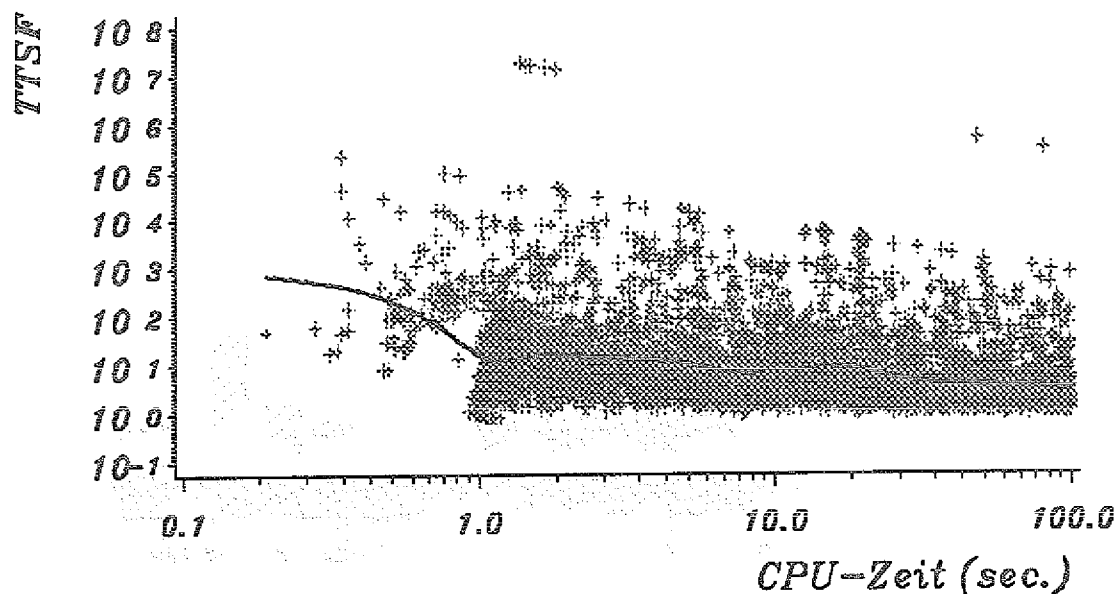


Abb. 23. Ausschnittweise Darstellung des TTSF: Nur wenige Prozesse besitzen eine CPU-Zeit kleiner 1 Sekunde.

Ebenso kann das Verhältnis von aufgewendeter CPU-Zeit zur Verweilzeit im System, also der *transaction-time stretch-factor* TTSF betrachtet werden.

Die Benutzeroberfläche, eine sogenannte *Shell*, besitzt eine hohe Verweilzeit im System. Da jeder Befehlsaufruf die Generierung eines neuen Prozesses bewirkt, ist die Rechenzeit einer solchen *Shell* sehr gering.

Weiterhin existieren *Dämonen*-Prozesse im System. Diese Prozesse besitzen die Eigenschaft wenig CPU-Zeit zu benötigen aber ständig im System zu verweilen. Dies kann beispielsweise bei Softwaremonitoren durchaus sinnvoll sein. Derart gestaltete Prozesse besitzen ebenfalls hohe TTSF-Werte, dürfen aber in der Untersuchung nicht berücksichtigt werden.

In Abb. 22 sind die Prozesse mit einem $TTSF > 10^7$ hier nicht zu betrachten, da sie auf Fehler in der statistischen Erhebung zurückzuführen sind.

Gut erkennbar ist die untere Schranke des TTSF. Offenbar benutzen nur sehr wenige Prozesse konsequent mehrere CPU's gleichzeitig. Die theoretische untere Grenze des TTSF bei vier Prozessoren liegt bei $TTSF = 0,25$. Trotz der Verzerrung durch die logarithmische Darstellung sind sowohl die rechenintensiven Batch-Lasten als auch die angesprochenen *Dämonen*-Prozesse mit geringer CPU-Lastung und hohen TTS-Faktoren gut erkennbar. Für weitere Betrachtungen ist eine Vergrößerung des relevanten Bereiches erforderlich (Abb. 23).

4.2 Speicherbedarf

Zur weiteren Analyse mußte festgestellt werden, ob eine Abhängigkeit zwischen der Rechenzeit eines Prozesses und dem zugeordneten Speicher besteht (Abb. 24). Es stellte sich heraus, daß rechenintensive Prozesse regelmäßig mehr Speicher benötigen, als weniger rechenintensive. In dem zu analysierenden Bereich geringer Rechenzeiten kann eine eindeutige Zuordnung nicht erfolgen, da keine Häufungspunkte erkennbar sind (Abb. 25).

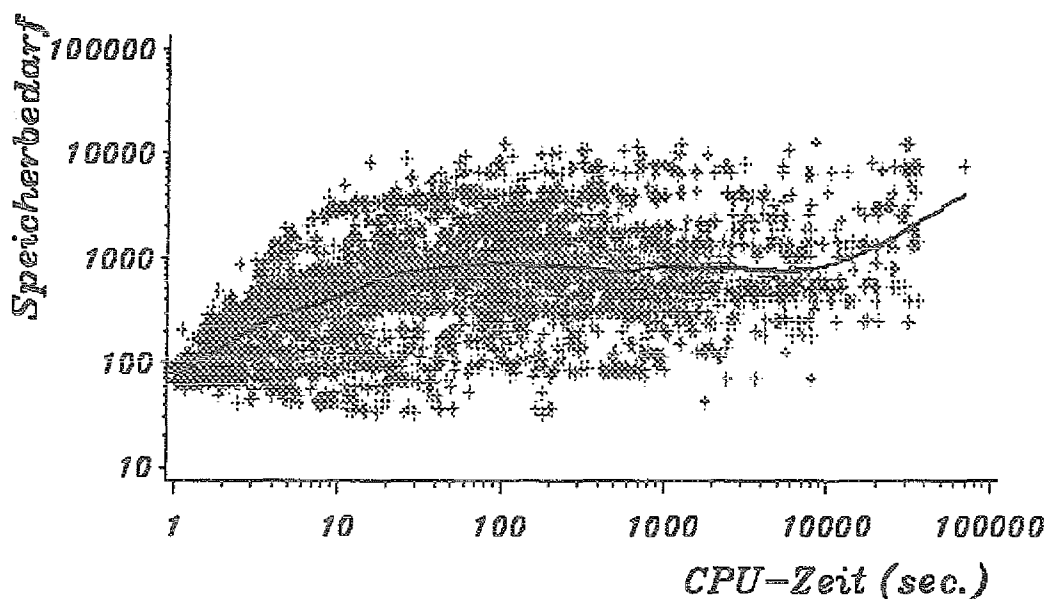


Abb. 24. Speicherbedarf in Kiloworten bei unterschiedlicher CPU-Lastung: Auffällig ist die Konstanz des Speicherbedarfs für $30 \text{ sec.} < C < 5000 \text{ sec.}$

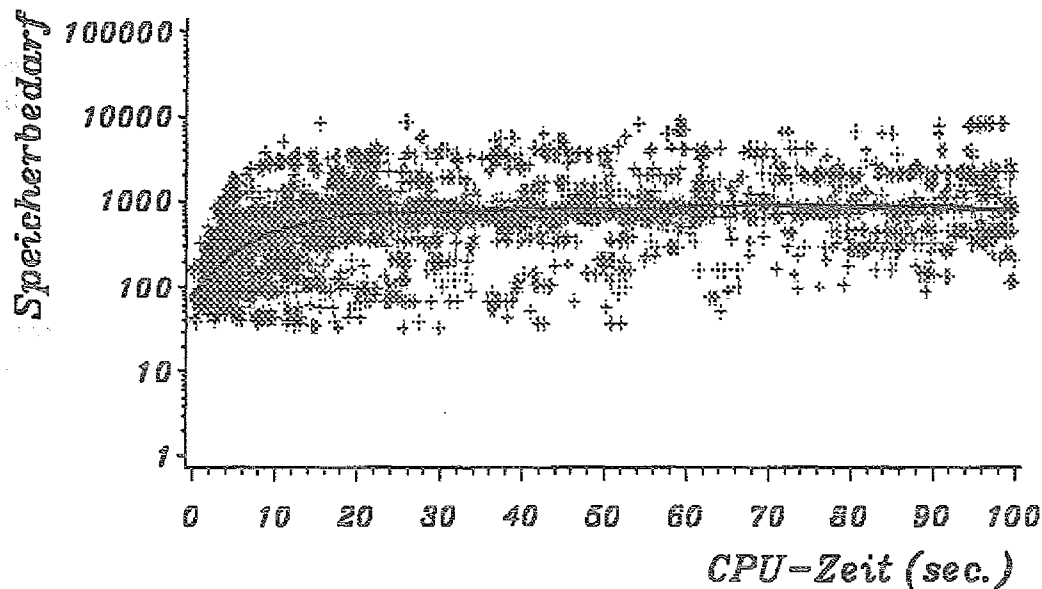


Abb. 25. Gewichtung des Speicherbedarfs

Der Speicherbedarf wurde mit zwei Faktoren gewichtet. Einerseits muß die Häufigkeit des Auftretens berücksichtigt werden. Andererseits aber muß auch eine Gewichtung mit der Verweilzeit im System erfolgen, da der angeforderte Speicher während der Prozeßlaufzeit nicht freigegeben wird.

4.3 Belastung der Ein-/Ausgabeeinheiten

In vergleichbarer Weise mußte untersucht werden, ob eine Abhängigkeit der Ein-/Ausgabeintensität von den bisher betrachteten Parameter besteht (Abb. 26, Abb. 27). Durch die Einteilung der mittleren CPU-Zeit in Klassen kann jeder dieser Klassen eine mittlere E/A-Intensität zugeordnet werden.

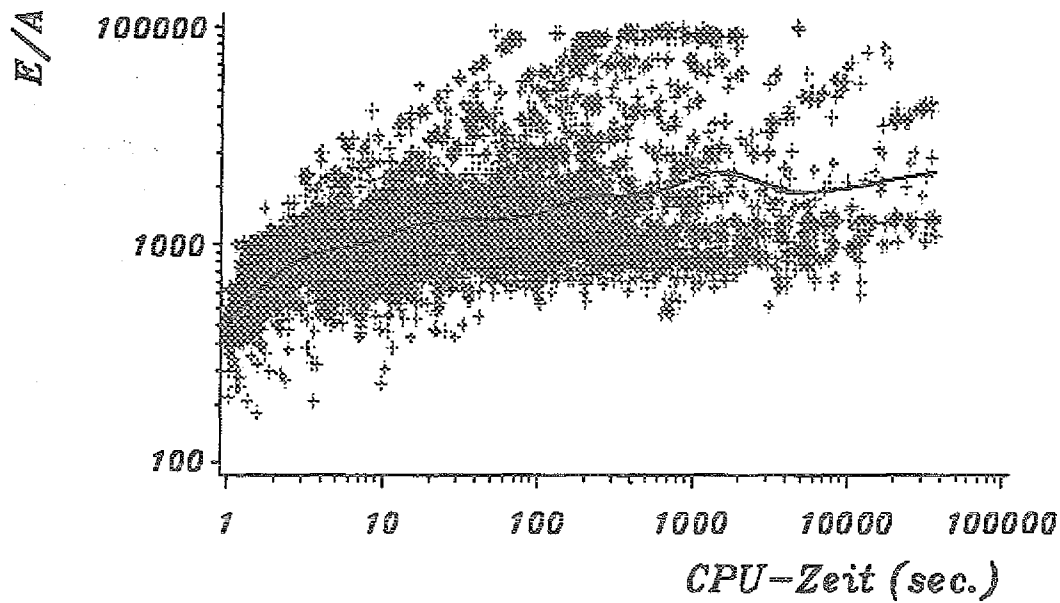


Abb. 26. Belastung der Ein-/Ausgabeeinheit in Abhängigkeit der Rechenintensität: Jeder Rechenintensität kann eine typische E/A-Belastung zugeordnet werden. Die maximal erreichte E/A-Rate beträgt 100000.

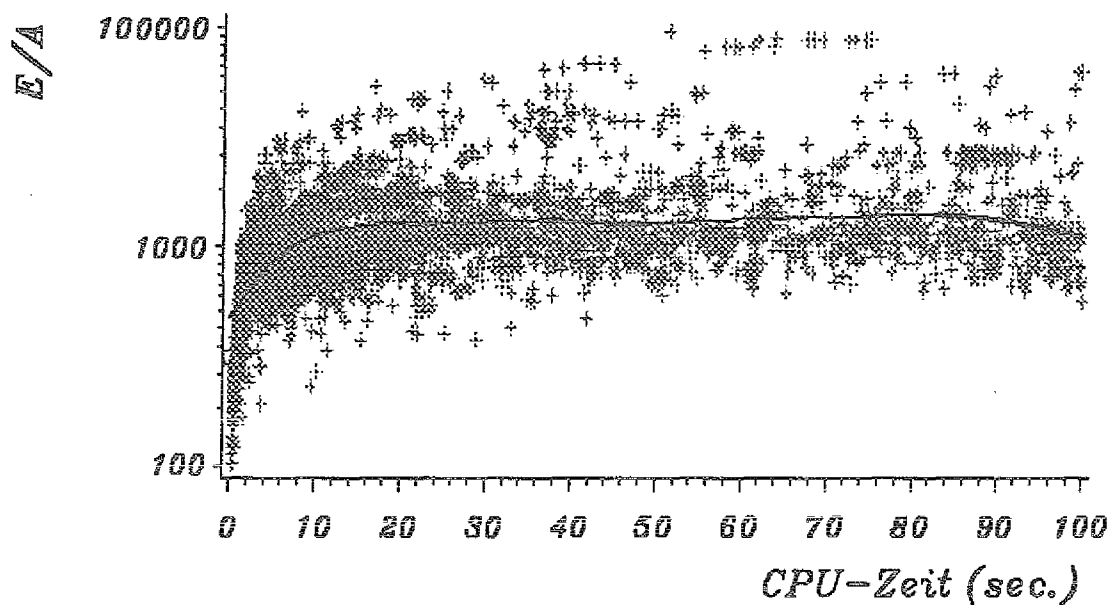


Abb. 27. Belastung der Ein-/Ausgabeeinheit: Dargestellt ist der Bereich interaktiver Belastung. Prozesse, deren CPU-Zeit unter 2 Sekunden liegt, führen wenige E/A-Operationen aus. Ein Teil dieser Prozesse sind *shells*.

4.4 Spezifizierung von Lastschränken

Zur besseren Übersicht wurden typische Lasten festgelegt und in Klassen zusammengefaßt. Eine solche Klasse beinhaltet nun die zum Programmablauf notwendigen Parameter, welche wiederum so festgesetzt werden können, daß die Maschine in einer Form belastet wird, wie dies auch im realen Betrieb aufgrund der Eingabe eines Befehls der Fall sein würde. Es können jederzeit neue Klassen mit einer anderen Zusammensetzung der Parameter definiert und abgearbeitet werden.

Falls der reale Betrieb eines Systems nachgebildet werden soll, muß diese Klasseneinteilung individuell auf das System abgestimmt werden. Der Übergang auf ein anderes System erfordert in jedem Falle eine erneute Analyse der Last und entsprechende Einteilung der Klassen. Die Notwendigkeit hierzu zeigt sich schon beim Übergang zur ebenfalls in der KFA installierten CRAY Y-MP. Eine Übernahme der Klasseneinteilung verursacht eine Benchmark-Belastung, welche stark von den tatsächlichen Gegebenheiten abweicht.

Die in diesem Kapitel vorgestellten Parameterdiagramme erlauben es, für jede Klasse eine typische CPU-Zeit festzulegen und anschließend diesen CPU-Zeiten Speicherbedarf und E/A-Raten zuzuordnen.

In den folgenden Kapiteln wird auf diese Parameterdiagramme zurückgegriffen.

5.0 Anwendung des Meßsystems

Der Schwerpunkt der durchgeführten Untersuchungen betrachtet die Recheneinheit als wichtigste Ressource. Die generierte Belastung wurde mit dem Ziel ausgewählt, zu untersuchen, wann bei steigender Last ein Engpaß auftritt, also keine dem Benutzer zumutbare Antwortzeit mehr erzielt werden kann.

Häufig werden zur Ausführung eines Benutzerkommandos zusätzliche Prozesse gestartet. Für die folgenden Untersuchungen wird davon ausgegangen, daß jeder Benutzer nur einen Prozeß startet.

Die Vor- und Nachteile dieser Arbeitsweise wurden im Rahmen dieser Arbeit nicht betrachtet. Diese Annahme wurde getroffen, da keine Analysemethode gefunden werden konnte, welche eine Aussage über die Zahl der von einem Benutzer erzeugten Prozesse erlaubt. Falls eine solche Analyse vorliegt, kann diese durch *Interbench* auch umgesetzt werden. Bei tiefergehenden Betrachtung wird in den entsprechenden Kennlinien die Abszissenbezeichnung 'Zahl der Benutzer' zu korrigieren sein.

Diese Annahme stellt keine Einschränkung dar, da der *Scheduler* des UNIX-Systems keinen Unterschied zwischen den Prozessen sieht, die von unterschiedlichen Benutzern erzeugt wurden und jenen, welche von einem Benutzer generiert wurden. Alle im System vorhandenen Prozesse konkurrieren gleichberechtigt um das Betriebsmittel CPU.

5.1 Kalibrierung

Für jedes System, welches mittels *Interbench* untersucht werden soll, muß eine Skalierung durchgeführt werden. Das Skaliervorgehen wurde in „Realisierung systeminvarianter Parameter“ auf Seite 38 beschrieben. Die Skaliergenauigkeit des gewählten Verfahrens mußte überprüft werden um festzustellen, ob eine nachträgliche, manuelle Korrektur notwendig war.

Diese Überprüfung erfolgte nur für Stützpunkte, welche in den späteren Messungen benötigt wurden. Zur Überprüfung wurden Sollwert, Mittelwert und Standardabweichung betrachtet.

Es zeigte sich, daß die Standardabweichung bei dieser Vorgehensweise für alle untersuchten Systeme unter zwei Prozent des Sollwertes liegt.

Der Mittelwert weist allerdings größere Abweichungen vom Sollwert auf. Hier existieren zwei Möglichkeiten der Korrektur:

1. Einführung eines Korrekturfaktors

Interbench bietet die Möglichkeit, alle Skalierungsfaktoren mit einer Konstanten zu multiplizieren. Somit wird die Abweichung des Mittelwertes vom Sollwert für alle Stützstellen gleichmäßig korrigiert.

2. Korrektur des Skalierungsfiles.

Falls nur ein Stützpunkt stark von der Vorgabe abweicht, kann im Skalierungsfile *bench.SCALE* die Skalierung entsprechend korrigiert werden.

Eine Verbesserung der Genauigkeit während des Skaliervorganges bewirkt eine deutliche Erhöhung der Dauer für die Berechnung der Skalierung. Hier treten bei geringer Erhöhung der Genauigkeit schon Laufzeiten von 90 Minuten auf.

Prinzipiell kann die Genauigkeit nicht beliebig gesteigert werden. Das gewählte Iterationsverfahren nach Pegasus setzt voraus, daß der Funktionswert eines Argumentes bei jeder Berechnung konstant bleibt. Da der Funktionswert im gegebenen Fall das Ergebnis einer Messung ist, welches eine Abweichung vom Mittelwert aufweist, wird die Genauigkeit des Iterationsverfahrens eingeschränkt. Ein Grund für diese Abweichungen ist durch die Systemverwaltung gegeben. Außerdem ist auch ein Einfluß des Meßwerkzeuges selbst denkbar.

Die eingestellten Skalierungsfaktoren sind ein sinnvoller Kompromiß zwischen maximaler Annäherung an die Vorgabe einerseits und tragbaren Skalierzeiten andererseits. Um die Skaliergenauigkeit der verschiedenen Systeme vergleichen zu können, wurden keine manuellen Korrekturen durchgeführt.

IBM-System PS/2 Modell 70-A21

Die gemessenen Daten werden statistisch ausgewertet. Hier wurden aus den Meßwerten der Mittelwert und die Standardabweichung bestimmt. Außerdem wurde die Abweichung des erzielten Mittelwertes vom Sollwert, also der Vorgabe betrachtet.

Sollwert	1 Sekunde	5 Sekunden	10 Sekunden	15 Sekunden	20 Sekunden
Mittelwert	1.03	5.16	10.06	15.10	19.89
Standardabweichung	0.0124	0.0276	0.0294	0.058	0.0399
Abweichung vom Sollwert	2.75%	3,32%	0.62%	0.73%	-0.53%

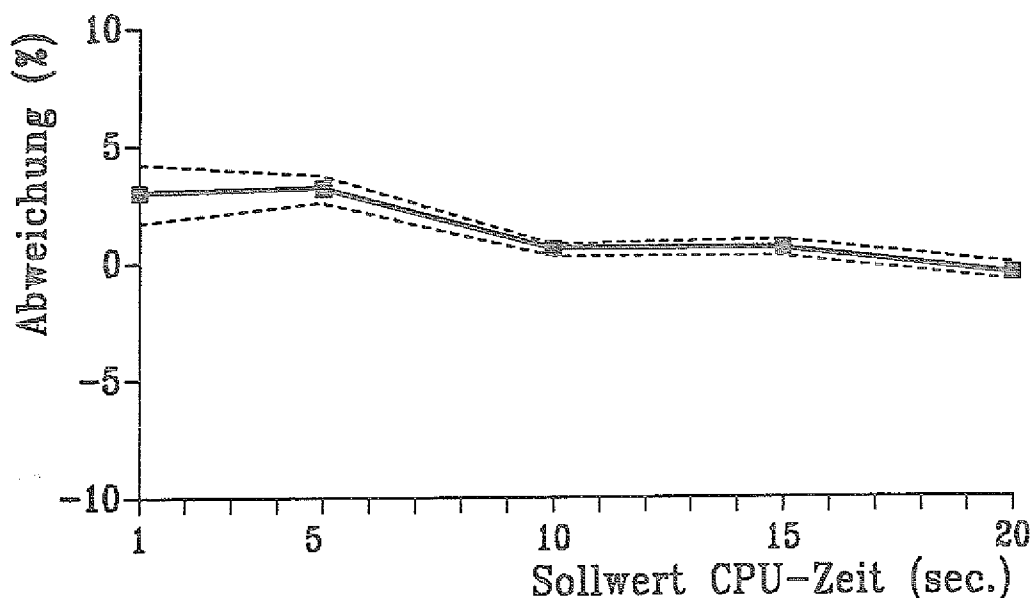


Abb. 28. Skaliergenauigkeit bei einem PS/2-System: Aufgetragen ist die Abweichung des Mittelwertes der Meßreihe vom Sollwert in Prozent (durchgezogene Linie). Der Bereich der Standardabweichung ist ebenfalls dargestellt (gestrichelte Linie).

Es mußte überprüft werden, ob die Vorgaben zuverlässig eingehalten werden. Zur Angleichung der Stützpunkte '1 Sekunde' und '5 Sekunden' empfiehlt sich eine Änderung der Skalierungsdatei.

Aufgrund der geringen Abweichungen wurde in diesem Falle keine Korrektur durchgeführt.

IBM-System RS/6000-320

Analoge Untersuchungen wurden auch für das RS/6000-System durchgeführt.

Sollwert	1 Sekunde	5 Sekunden	10 Sekunden	15 Sekunden	20 Sekunden
Mittelwert	1.01	4.80	9.59	14.40	19.41
Standardabweichung	0.0211	0.1098	0.1285	0.1328	0.2485
Abweichung vom Sollwert	1.00%	-4.00%	-4.10%	-4.00%	-2.95%

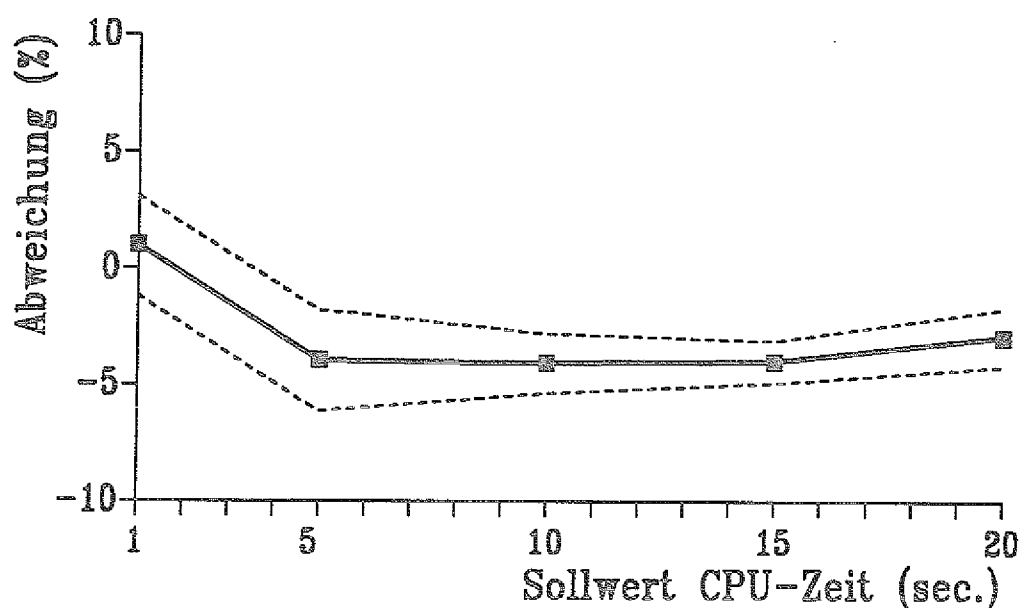


Abb. 29. Skaliergenauigkeit eines IBM-Systems RS/6000: Aufgetragen ist die Abweichung vom Mittelwert (durchgezogen) und der Bereich der Standardabweichung (gestrichelt)

Hier zeigt sich eine Tendenz zur Unterschreitung des Sollwertes. Es können wiederum beide aufgeführten Wege zur Korrektur der Stützpunkte eingeschlagen werden. Aufgrund der geringen Abweichungen wurden hier keine Korrekturen durchgeführt.

CRAY X-MP/416

Sollwert	1 Sekunde	5 Sekunden	10 Sekunden	15 Sekunden	20 Sekunden
Mittelwert	0.99	4.98	9.97	14.92	19.91
Standardabweichung	0.005	0.037	0.063	0.086	0.083
Abweichung vom Sollwert	0.50%	-0.26%	-0.23%	-0.50%	-0.45%

Die Skalierung dieses Systems erweist sich erheblich genauer als bei den bisher untersuchten Systemen. Eine Korrektur erscheint hier nicht sinnvoll.

Ein Grund für die erzielte Genauigkeit kann die erhöhte Meßgenauigkeit dieses Systems sein. Im Gegensatz zu den bisherigen Systemen, welche die CPU-Zeit in Vielfachen von 10^{-3} Sekunden angeben, wird die CPU-Zeit hier mit einer Genauigkeit von 10^{-6} Sekunden zur Verfügung gestellt. Aus diesem Grunde kann mit Hilfe des Verfahrens von Pegasus entsprechend genauer skaliert werden.

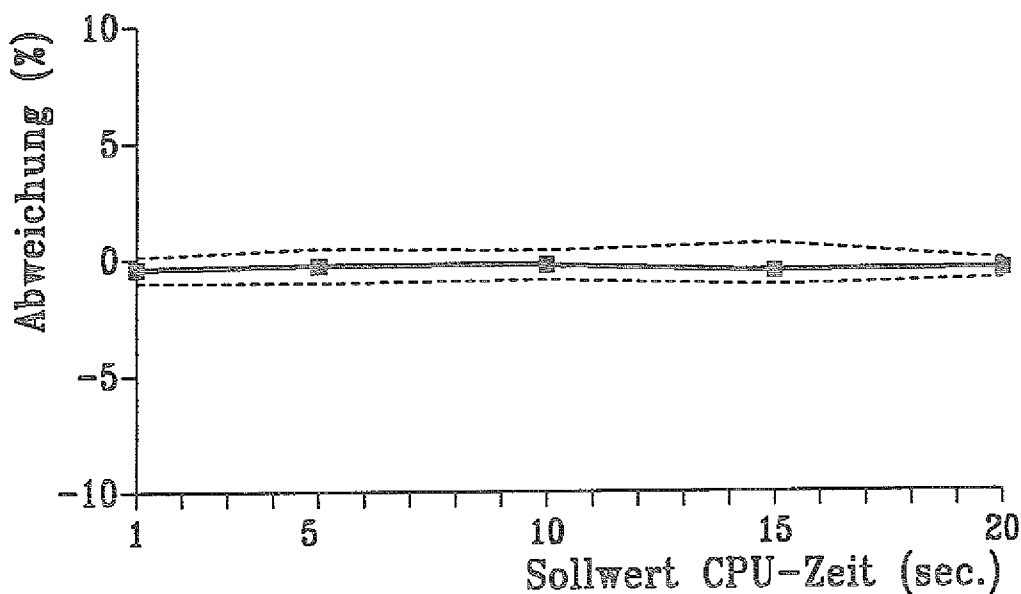


Abb. 30. Skaliergenauigkeit der CRAY X-MP/416: Aufgetragen ist die Abweichung vom Mittelwert (durchgezogen) und der Bereich der Standardabweichung (gestrichelt)

5.2 Einfluß des Benchmarksystems auf die Meßergebnisse

Ebenfalls untersucht werden mußte der Einfluß des Benchmarksystems *Interbench* auf die Meßergebnisse. Hierzu wurde der *Overhead* des Meßsystems bestimmt.

Der Benchmark *Interbench* wurde ohne Generierung einer Belastung gestartet. Die Parameter für CPU-Belastung, Speicherbedarf und E/A wurden hier auf Null gesetzt.

Diese Untersuchung wurde nur auf der CRAY X-MP/416 durchgeführt, da hier ein Werkzeug zur Untersuchung der Systembelastung zur Verfügung steht (SAR). Dieses bietet die Möglichkeit, die zur Verfügung stehende Rechenzeit daraufhin zu untersuchen,

ob sie von dem Benutzer, dem System oder gar nicht belegt wird. Diese Untersuchung erlaubt nun eine Aussage, in welchem Umfang *Interbench* die zur Verfügung stehenden Ressourcen belegt.

Die Messung wurde exemplarisch für 50 Benutzer durchgeführt. Es ergab sich eine Systemauslastung aufgrund des Benchmarks *Interbench* von weniger als zwei Prozent der zur Verfügung stehenden Rechnerleistung. Hierin eingeschlossen ist auch die von SAR selbst erzeugte Last. Das Meßsystem hat also nur geringen Einfluß auf die Ergebnisse. Es konnte kein Unterschied festgestellt werden, ob von den 50 generierten Prozessen alle Prozesse Messungen durchführten oder nur ein Prozeß. Hieraus kann geschlossen werden, daß die Verwaltung der *PIPE* das System ebenfalls nur in geringer Weise belastet.

6.0 Ergebnisse

6.1 Durchführung des Kennlinienverfahrens

Das Kennlinienverfahren bestimmt die Antwortzeit eines Rechnersystems bei steigender Benutzerzahl. Entsprechend [BuDe] werden Kennlinien für die wesentlichen Belastungsparameter im System gemessen beziehungsweise aus gemessenen Werten berechnet. Diese Kennlinien können zur Kapazitätsabschätzung herangezogen werden.

Die folgende Untersuchung verfolgte das Ziel, eine Abhängigkeit der Antwortzeit von der Belastung des Systems zu bestimmen. Es wurde die CPU-Zeit variiert während die Denkzeit D konstant gehalten wurde.

Folgende Tabelle enthält die relevanten Parameter der einzelnen Belastungsklassen.

		SLOW	LOW	MEDIUM	HIGH	VHIGH
CPU	sec.	1	5	10	15	20
Denkzeit	sec.	60	60	60	60	60
E/A	Anzahl	10	10	10	10	10
Speicher	KByte	10	10	10	10	10

Für jede Klasse wurde eine eigene Meßreihe durchgeführt. Um eine Vergleichbarkeit der Meßergebnisse der verschiedenen Systeme zu gewährleisten, wurde auf jedem System eine Untersuchung mit den tabellarisch aufgeführten Parametern durchgeführt. Dies schränkt die Wahl der Parameter ein, da diese auf allen Systemen eine meßbare Belastung darstellen sollen und gleichzeitig bei keinem System durch nur einen Prozeß ein Engpaßverhalten entstehen soll. Zum einen stellt der zur Verfügung stehende Speicherplatz eine Einschränkung dar: Das verwendete PS/2-System besitzt 4 Megabyte Hauptspeicher während die CRAY X-MP über 64 Megabyte Hauptspeicher und zusätzlich 1.6 Gigabyte Plattenspeicher zur Auslagerung der Prozesse (*Swap*) verfügt.

Zum anderen ist die Zahl der auf einem System maximal vorhandenen Prozesse begrenzt. Diese Begrenzung ist auf eine intern geführte Tabelle zurückzuführen, welche Informationen über jeden Prozeß im System beinhaltet. Die Größe dieser Tabelle ist festgelegt und kann auch nicht seitens des Systemverwalters verändert werden. Sie reicht von 23 möglichen Prozessen des untersuchten PS/2-Systems bis zu 650 Prozessen einer CRAY X-MP/416.

6.1.1 Berechnung der Kenngrößen

Da hier nur die Kennlinien für das CPU-Verhalten untersucht werden, können für die weiteren Betrachtungen einige der in Kapitel 2 erläuterten Zusammenhänge vereinfacht werden: Die Kennlinien können durch zwei Asymptoten grob angenähert werden (Abb. 31):

$$(a) \quad TTSF = TTSF_{\min}$$

Im Idealfall gilt $TTSF_{\min} = 1$.

$$(b) \quad TTSF = \frac{N}{U_B} - \frac{D}{C}$$

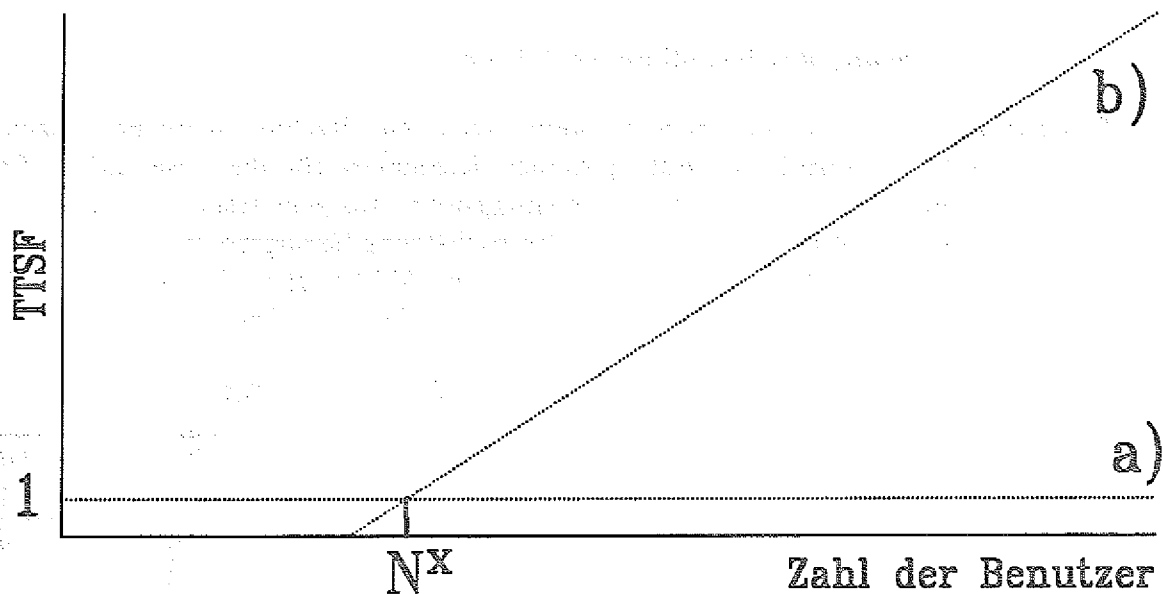


Abb. 31. Asymptoten der Kennlinien: Der Schnittpunkt der Asymptoten a) und b) ergibt den Sättigungspunkt N^* .

Der Schnittpunkt der beiden Asymptoten kann als die Benutzerzahl N^* betrachtet werden, an dem die CPU in die volle Auslastung, also in die Sättigung, übergeht. Dieser Sättigungspunkt läßt sich aus dem Schnittpunkt der beiden Asymptoten a) und b) bestimmen:

$$N^* = (TTSF_{\min} + \frac{D}{C}) \times U_B \quad (5a)$$

Bei Denning und Buzen wird für $TTSF_{\min} = \frac{R}{R_{\min}} = 1$ angenommen, wenn die Antwortzeit nur von dem Betriebsmittel CPU bestimmt wird.

Stehen zusätzlich mehr Prozessoren zur Verfügung, so wird

$$N^* = (TTSF_{\min} + \frac{D}{C}) \times p \times U_B \quad (7)$$

Wenn keine anderen Betriebsmittel die volle Kapazitätsausnutzung verhindern (z.B. bei einem Speicherengpaß), steigt die Auslastung bis auf $U_B = 1$.

Für $TTSF_{\min} = 1$ folgt unmittelbar aus Gleichung (7)

$$N^* = (1 + \frac{D}{C}) \times p \quad (8)$$

Die Gleichung (8) wird auf die Meßergebnisse angewendet und auf Übereinstimmung mit den abgelesenen Werten geprüft.

$TTSF_{\min}$ ergibt sich aus den durchgeführten Messungen. Der theoretisch mögliche, maximale Systemdurchsatz kann aus

$$X_0 = \frac{p}{C} \quad \text{mit } V_B = 1, \quad U_B = 1$$

bestimmt werden. Aufgrund der gewählten Charakteristika der Messungen ist die *Besuchshäufigkeit* für alle Messungen $V_{CPU} = 1$. Der Sättigungspunkt N^* läßt sich aus den TTSF-Kennlinien nach Erstellung der Asymptoten ablesen. Für den Bereich der Sättigung der TTSF-Kennlinien, also für $N \geq N^*$, wurde eine Regressionsgerade erstellt. Der Schnittpunkt dieser Geraden mit der Asymptote des Bereiches $TTSF = TTSF_{\min}$ ergibt den Sättigungspunkt N^* .

Im Bereich $N \geq N^*$ wird die maximale Auslastung bestimmt durch

$$\frac{1}{U_B} = \frac{\partial TTSF}{\partial N}$$

Auf allen untersuchten Systemen wurden die gleichen Steuerdateien bearbeitet. Die Vergleichbarkeit der Ergebnisse ist somit gewährleistet.

Die Messungen wurden auf einer Maschine des beschriebenen Clusters durchgeführt. Die Zahl der gleichzeitig vorhandenen Prozesse in einer Sitzung ist hier auf 23 Benutzer beschränkt. Diese Zahl wird durch die notwendige Benützeroberfläche, die *Shell*, den Steuerprozeß und den Dialogprozeß des Benchmarks *Interbench* auf 20 Meßprozesse reduziert.

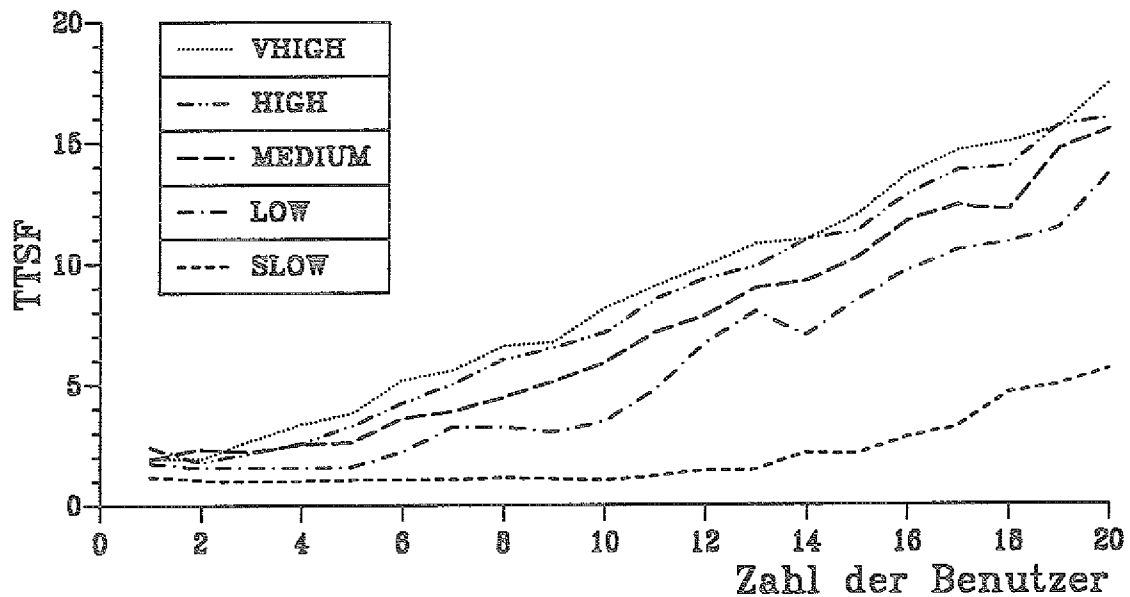


Abb. 32. TTSF-Kennlinien des PS/2-Systems

Die Abbildung 32 stellt die Meßergebnisse dar, welche für das PS/2-System ermittelt wurden. Aufgetragen sind die TTSF-Werte, welche bei einer steigenden Zahl von Prozessen einer Klasse gemessen werden konnten. Diese Untersuchung wurde für die Klassen VHIGH, HIGH, MEDIUM, LOW und SLOW durchgeführt. Einige charakteristischen Merkmale der TTSF-Kennlinien können hier festgestellt werden (Abb. 33):

1. Der minimale Wert der TTSF-Kennlinien bewegt sich in dieser Meßreihe zwischen 1 und 2,5.
2. Oberhalb des Sättigungspunktes N^* erfolgt ein nahezu linearer Anstieg.

Es soll eine Gegenüberstellung der gemessenen Kennlinien und den nach Denning und Buzen berechneten erfolgen. Hierzu kann Gleichung (6) aus dem Kapitel „Antwortzeiten bei der Anwendung des Kennlinienverfahrens“ auf Seite 19 herangezogen werden. Es wird beispielhaft die Klasse MEDIUM betrachtet. Für dieses System zeigt sich eine qualitative Übereinstimmung der ermittelten mit der aus Gleichung (6) berechneten Kennlinie. Allerdings sind die gemessenen TTSF-Werte grundsätzlich höher als die nach der Theorie erwarteten. Die Strategie des *Schedulers* erscheint als nicht optimal.

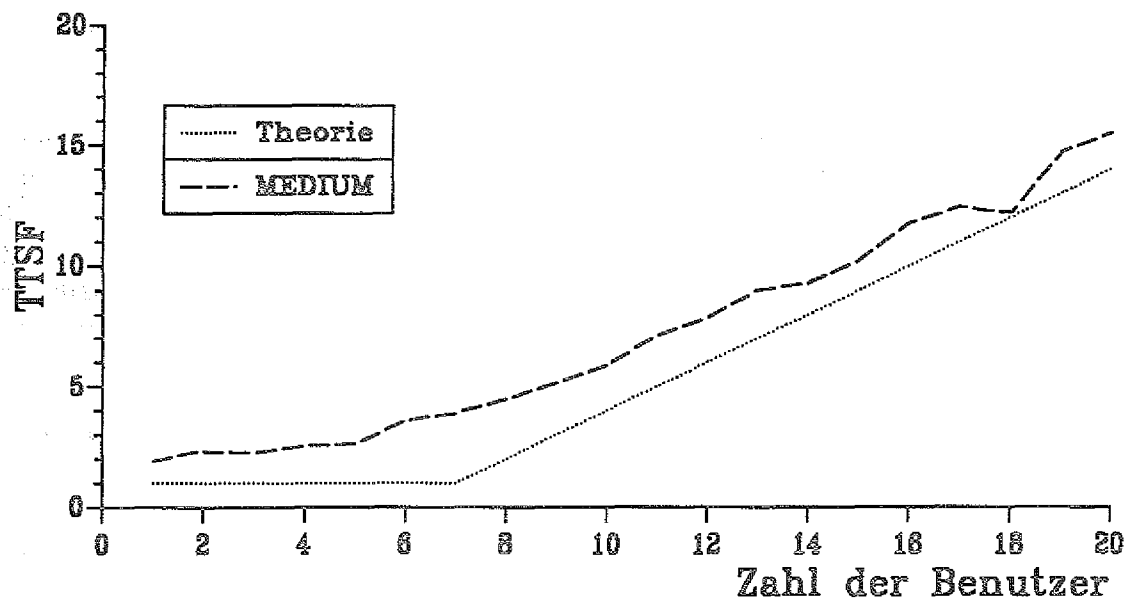


Abb. 33. Gegenüberstellung der Messung mit der Theorie

Weiterhin kann untersucht werden, ob eine Parallelität der Asymptoten verschiedener Belastungsklassen gegeben ist (Abb. 34). Aus der Theorie nach Denning und Buzen ergibt sich diese Parallelität für Belastungsklassen gleicher Denkzeit aber unterschiedlicher CPU-Zeit im Bereich der Sättigung.

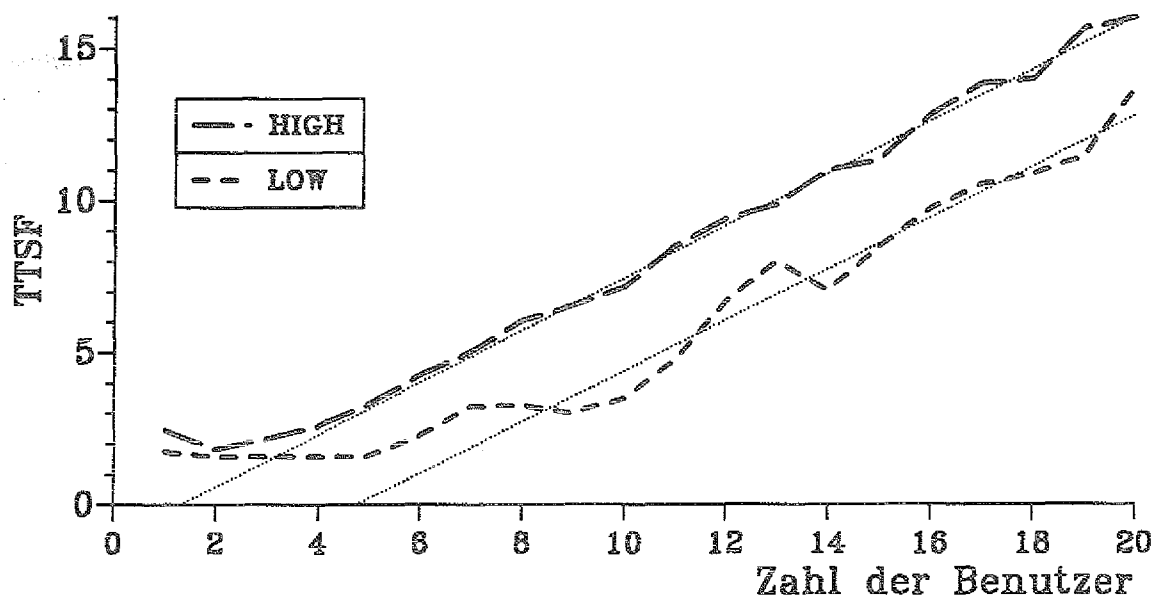


Abb. 34. Asymptote im Bereich der Sättigung: Beispielhaft wird an zwei Meßkurven eine Sekante angelegt.

Deutlich erkennbar ist die Parallelverschiebung dieser Sekanten im Bereich der Sättigung für verschiedene Klassen. Somit kann das gewählte Modell in bezug auf diese Aussage verifiziert werden.

Aus den gemessenen TTSTF-Kennlinien kann die Zahl der Benutzer im Sättigungspunkt N^* entnommen werden. Hierzu wird für den Bereich der Sättigung ($N > N^*$) eine Ausgleichsgerade nach dem Prinzip der kleinsten Fehlerquadrate bestimmt. Der Schnittpunkt dieser Ausgleichsgeraden mit der Geraden $TTSTF_{min} = 1$ ergibt den Sättigungspunkt N^* . Hierdurch wird ein, dem Eindruck des Benutzers entsprechender, Sättigungspunkt bestimmt, bei dessen Überschreitung die *transaction-time* des Systems spürbar ansteigt. Andere Verfahren können zwar höhere Werte für den Sättigungspunkt ergeben, spiegeln aber nicht den Eindruck des Benutzers vom System wider.

Diese Ausgleichsgerade entspricht im allgemeinen nicht einer Asymptoten der Steigung 1, so daß der abgelesene Sättigungspunkt nur in weiten Toleranzbereichen mit dem Sättigungspunkt von Denning und Buzen übereinstimmen kann.

Die Ergebnisse dieser Vorgehensweise werden im folgenden den rechnerisch ermittelten Sättigungspunkten gegenübergestellt:

	SLOW	LOW	MEDIUM	HIGH	VHIGH
$N^* \text{ ablesen}$	9.68	4.72	3.43	1.31	1.19
$N^* = (1 + \frac{D}{C}) \times p$	61.0	13.0	7.0	5.0	4.0
$X_0 = \frac{N}{D + R(N)}$	0.306	0.16	0.095	0.066	0.050

Die Ergebnisse der Klasse SLOW besitzen keine Relevanz, da keine volle Auslastung erreicht wird.

Die Ergebnisse der Gleichung (8) weisen zu starke Abweichungen von den Meßwerten auf, als daß sie zur Kapazitätsabschätzung verwendet werden könnten. Gründe sind einerseits in dem langsamen Übergang der Teilauslastung zur Vollauslastung zu suchen, andererseits können die im Vergleich zur Theorie erhöhten TTSTF-Werte genannt werden (vgl. Abb. 33).

IBM-System RS/6000-320

Es werden die bisherigen Betrachtungen wie beim PS/2-System auch hier durchgeführt. Auf dem System RS/6000-320 konnten aber bis zu 35 Benutzer simuliert werden.

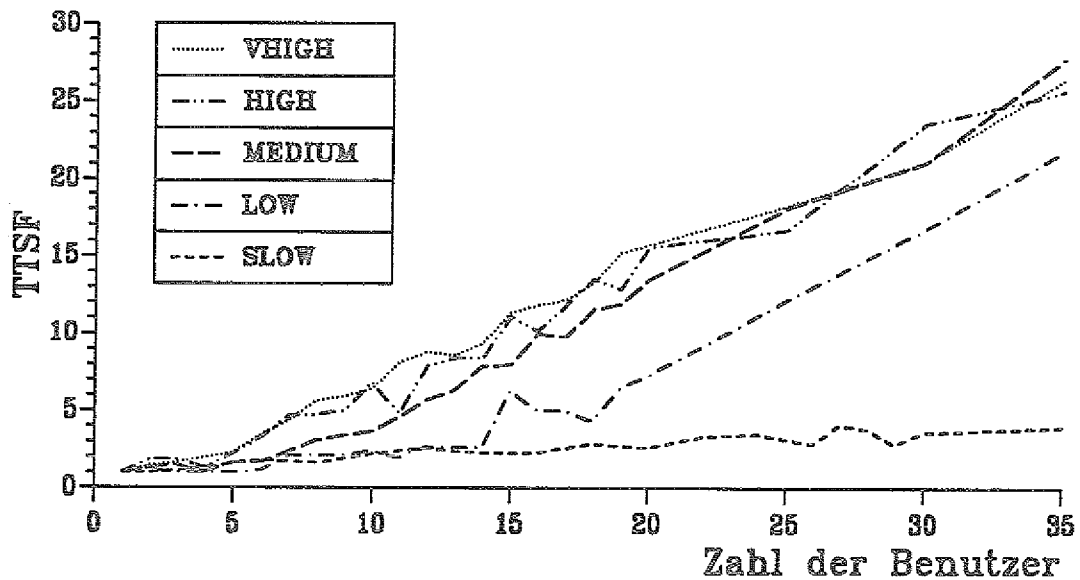


Abb. 35. TTSF-Kennlinien des IBM RS/6000-Systems

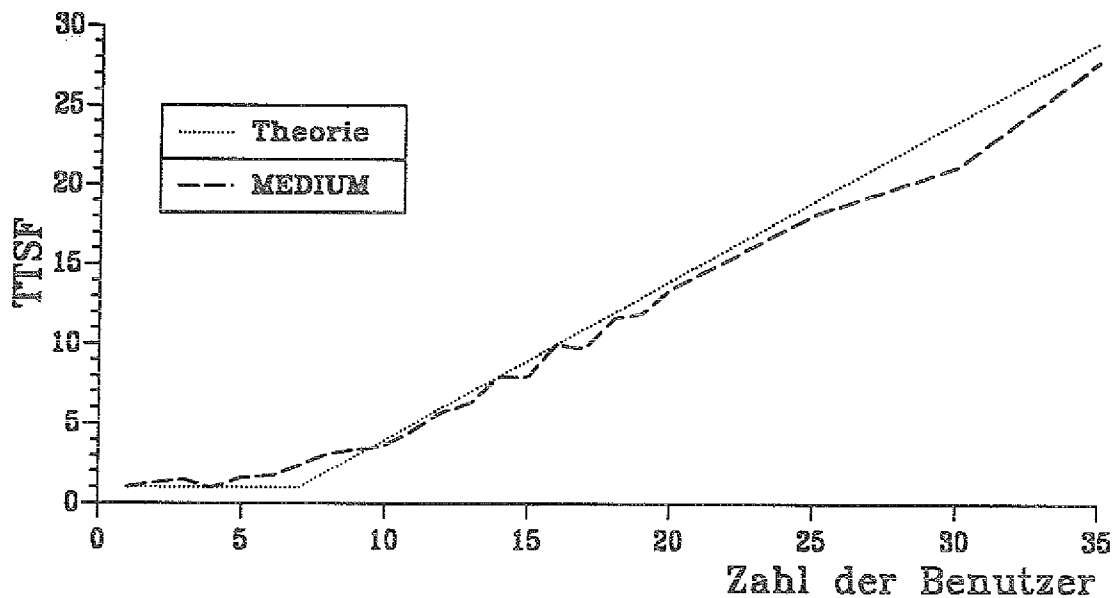


Abb. 36. Gegenüberstellung der Messung mit der Theorie

Das Verhalten der untersuchten RS/6000 entspricht deutlicher dem Modell von Denning und Buzen. Die drei signifikanten Parameter $TTSF_{min}$, N^* und die Steigung für $N \geq N^*$ zeigen sehr gute Übereinstimmung. Dies spiegelt sich auch in folgender Tabelle wider:

	SLOW	LOW	MEDIUM	HIGH	VHIGH
$N^*_{ablesen}$	--	13.59	6.58	4.91	2.25
$N^* = (1 + \frac{D}{C}) \times p$	61.00	13.00	7.00	5.00	4.00
$X_0 = \frac{N}{D + R(N)}$	0.54	0.21	0.10	0.08	0.06

Aus den Skalierungsdaten läßt sich unter Betrachtung des Kernels zur CPU-Belastung ein Durchsatz von 6,6 MFLOPS ermitteln. Der Hersteller gibt einen maximalen Durchsatz von 8 MFLOPS an. Ohne besondere Maßnahmen wurden somit 83% Prozent der maximal verfügbaren Leistung erreicht. Hier spiegelt sich die Auslegung dieses Systems für allgemeine Datenverarbeitungsaufgaben wider.

Auffällig ist der erreichte Systemdurchsatz der Klasse HIGH. Dieser liegt über dem theoretisch möglichen Wert von $X_{theor} = 0,067$.

Da durch Kontrollmessungen vergleichbare Ergebnisse erzielt werden konnten, deutet dies auf Unstimmigkeiten hin. Abbildung 36 läßt die Annahme zu, daß diese nicht in einer fehlerhaften Modellbildung zu suchen sind, sondern auf Ungenauigkeiten der Skalierung und Einhaltung der Denkzeit zurückzuführen sind.

Denkbar ist aber auch ein Einfluß der CPU, die durch starke Überlappung bei der Verarbeitung einiger Befehle eine gewisse Parallelität in der Abarbeitung der Befehle erzeugt, so daß $p > 1$ wird.

Die Ergebnisse der Belastungsklasse VHIGH weisen zu starke Abweichungen auf, wohingegen die Abweichungen der anderen Belastungsklassen unter 10% liegen. Die Gleichung (8) kann auf diesem System zur Kapazitätsabschätzung herangezogen werden. Das gewählte Modell nach Denning und Buzen zeigt grundsätzliche Übereinstimmung mit den Meßergebnissen.

CRAY X-MP/416

Es stehen in diesem Fall vier Prozessoren zur Verfügung. Hierdurch verlagert sich der Sättigungspunkt deutlich zu höheren Benutzerzahlen.

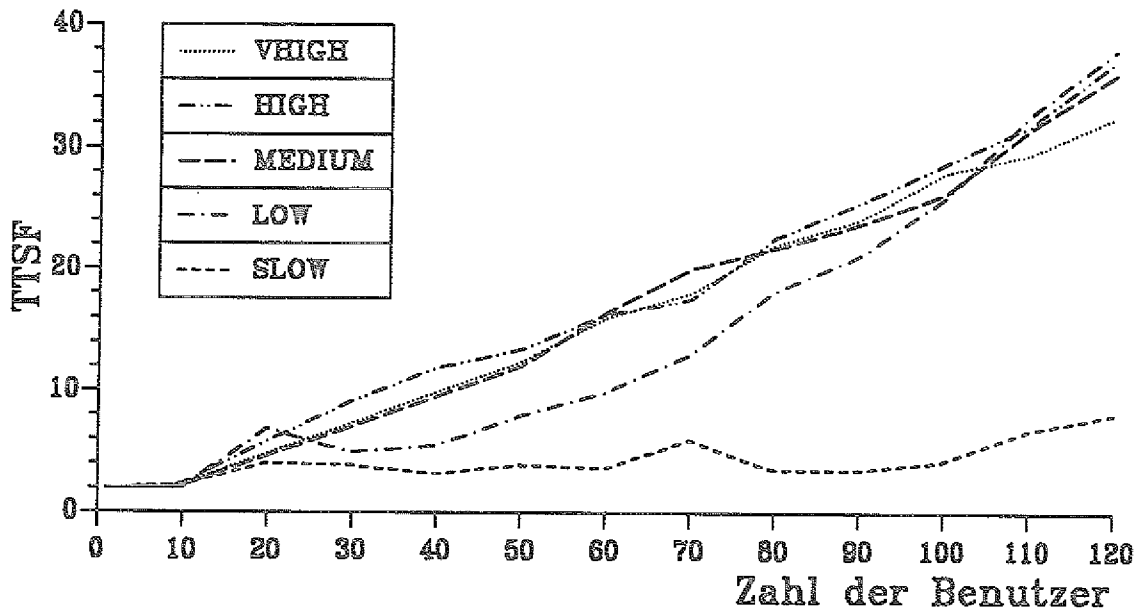


Abb. 37. TTSF-Kennlinie der CRAY X-MP/416

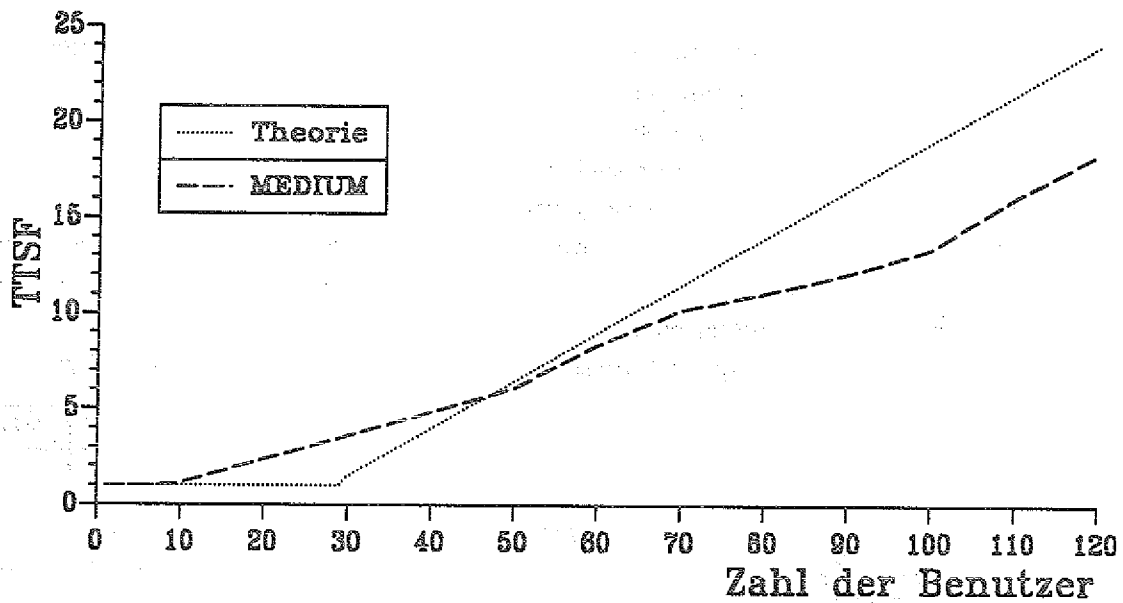


Abb. 38. Gegenüberstellung der Messung mit der Theorie

Dieses System begrenzt ebenfalls die maximal mögliche Prozeßzahl. Die Begrenzung hat hier allerdings keine Einschränkung verursacht. Es sind maximal 650 Prozesse möglich.

Die Begrenzung der Meßreihe auf 120 Prozesse wurde gewählt, um die Vergleichbarkeit mit den bisher erhaltenen Ergebnissen zu gewährleisten. Es muß beachtet werden, daß die CRAY X-MP/416 vier Prozessoren zur Verfügung stellt und somit zur Erzielung der gleichen Auslastung die vierfache Benutzerzahl notwendig wird. Auffällig ist das Systemverhalten für mehr als 160 Benutzer (Abb. 39).

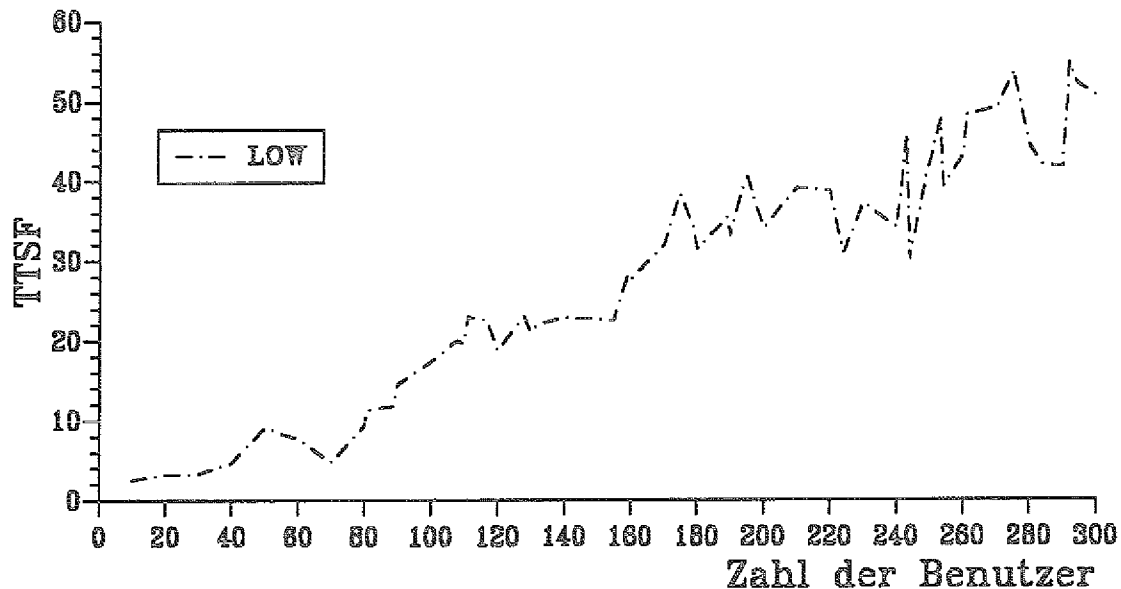


Abb. 39. TTSF-Kennlinie der CRAY X-MP/416: Es wurden maximal 300 Prozesse der Klasse LOW generiert.

Die größere Varianz entspricht nicht den Erwartungen. Nach dem Modell von Denning und Buzen sollte die Steigung der Kennlinie konstant bleiben oder, bei Auftreten eines weiteren Engpasses, sich vergrößern.

Hier entwickelte sich der Hauptspeicher zu einem weiteren Engpaß, das System lagert vermehrt Prozesse aus (Swap), es wird zusätzlicher E/A induziert.

Da bei Generierung von Benchmarkläufen mit höheren E/A-Raten ähnliche TTSF-Kennlinien festgestellt werden, wird diese Vermutung untermauert, insbesondere da für mehr als 220 Benutzer ein Engpaß erwartet werden kann.

Die ermittelten Sättigungspunkte dieses Systems sind im folgenden aufgeführt:

	SLOW	LOW	MEDIUM	HIGH	VHIGH
$N^*_{ab\ddot{a}lesen}$	--	40.0	14.0	16.21	9.0
$N^* = (1 + \frac{D}{C}) \times p$	244	52	28	20	16
$X_0 = \frac{N}{D + R(N)}$	1.87	0.76	0.49	0.34	0.30

Bei diesem System zeigt sich eine stärkere Abweichung des berechneten Sättigungspunktes N^* von den gemessenen Werten. Diese Abweichung ist nicht auf eine Belastungsklasse beschränkt. Das Modell nach Denning und Buzen ist hier nur mit Einschränkungen anwendbar. Die Gleichung (8) läßt keine Kapazitätsabschätzung zu.

Die aufgetretenen Differenzen weisen auf eine Vereinfachung des Modells im Vergleich zu den tatsächlichen Gegebenheiten hin. Zur Verbesserung des Modells müssen weitere Parameter berücksichtigt werden.

6.1.2 Kapazitätsabschätzung

Unter Zuhilfenahme der vom Benchmark-System ermittelten TTSTF-Kennlinien kann eine Kapazitätsabschätzung vorgenommen werden. Es wird eine obere Grenze für die zumutbare Antwortzeit vorgegeben, die nicht überschritten werden sollte. Den Kennlinien kann nun die Zahl Benutzer entnommen werden, die diese, noch zumutbare, Antwortzeit hervorrufen. Diese Untersuchung wurde beispielhaft für einen Verlängerungsfaktor $TTSTF = 10$ durchgeführt. Hieraus ergaben sich folgende Sättigungspunkte:

Klasse / System	IBM-System PS/2 Modell 70-A21	IBM-System RS/6000-320	CRAY X-MP/416
LOW	16,5	22,5	60
MEDIUM	15	17	42
HIGH	13	14	41
VHIGH	12	13	33

Da die Kennlinien Unregelmäßigkeiten aufweisen, können diese Abschätzungen nur als Anhaltspunkt dienen. Ebenso ist fraglich, ob ein TTSTF von 10 für alle Anwendungen tragbar ist. Insbesondere für kurze CPU-Zeiten kann die Arbeitseffektivität durch diese Verzögerungen eingeschränkt werden. Hier erwartet der Benutzer sehr geringe Antwortzeiten.

6.2 Bestimmung der Auslastung

Aus den TTSTF-Messungen läßt sich die Auslastung des Systems bestimmen. Aus dem Antwortzeitgesetz folgt:

$$U = \frac{N}{TTSF + \frac{D}{C}} \times \frac{1}{P}$$

IBM-System PS/2 Modell 70-A21

Die Klasse SLOW erzeugt keine Volllastung des Systems im untersuchten Bereich und besitzt somit keine Relevanz. Die Klassen MEDIUM, HIGH und VHIGH erzeugen annähernd eine Volllastung des Systems. Die Klasse LOW erzeugt eine geringere Auslastung des Systems als Klassen mit höheren CPU-Zeiten. Eine Begründung für dieses Verhalten kann eine erhöhte Verwaltungszeit sein, welche durch Prozesse mit geringerer CPU-Zeit hervorgerufen wird. Ein weiterer Grund kann durch das Benchmarksystem *Interbench* selbst gegeben sein: Nach Ablauf einer Messung wird ein geringes Maß an Bearbeitungszeiten der Meßergebnisse notwendig. Dies ist im wesentlichen eine Berechnung der Meßdauer. Außerhalb der Messung müssen der Startpunkt und Endpunkt der Messung durch Systemaufrufe ermittelt werden.

Diese Arbeiten sind vollkommen invariant in bezug auf die Dauer der Messung. Der störende Einfluß dieser Arbeiten verringert sich also prozentual mit steigender CPU-Zeit, die Systemaufrufe ergeben den größten Einfluß somit für die Klasse LOW.

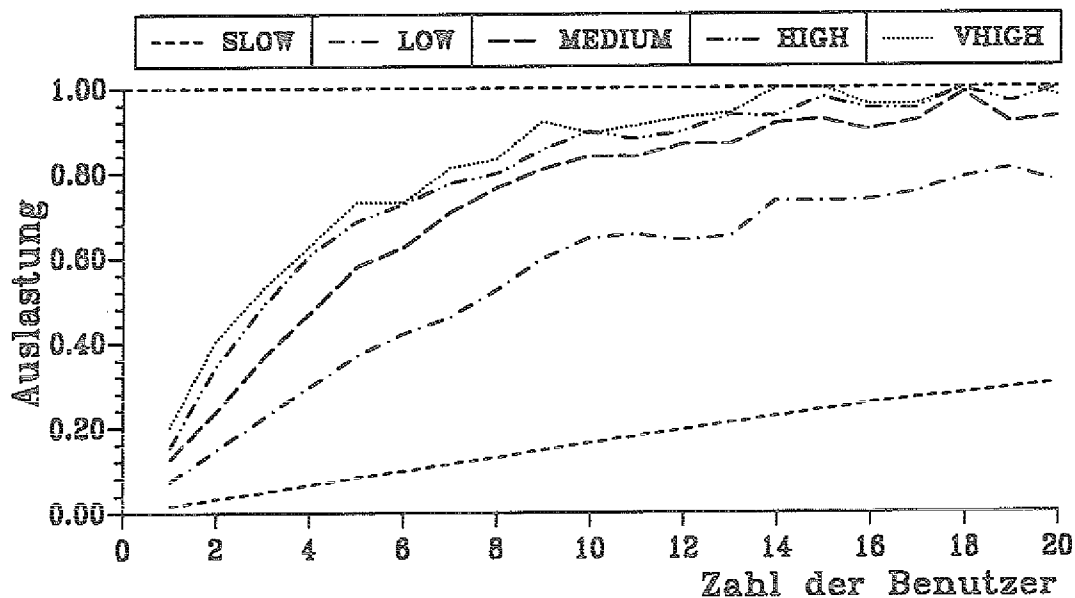


Abb. 40. Auslastung des PS/2-Systems durch verschiedene Belastungsklassen

Weitere Untersuchungen ergaben, daß die geringe Auslastung durch die Benutzerklasse LOW nicht auf das Meßsystem zurückzuführen ist. Eine Erklärung für diese geringe Auslastung kann im *Scheduler* gesucht werden. Dieser zeigt bei der Verwaltung von Prozessen mit geringer CPU-Zeit offenbar Schwächen.

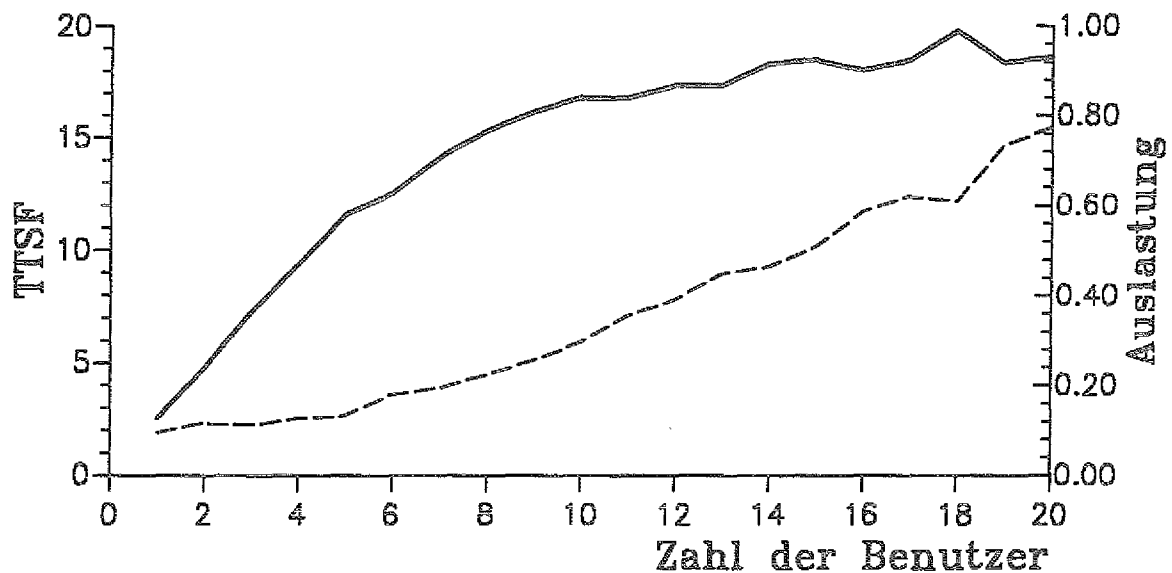


Abb. 41. Gegenüberstellung einer ausgewählten TTSF-Kennlinie (gestrichelte Linie) und der daraus resultierenden Systemauslastung (durchgezogene Linie). Es wurde beispielhaft die Klasse MEDIUM dargestellt.

IBM-System RS/6000-320

Die Auslastung U_b auch für Klassen mit geringerem CPU-Bedarf¹¹ liegt in der gleichen Größenordnung wie bei Klassen mit höherem CPU-Bedarf (Abb. 42). Dies deutet auf eine bessere Auslegung des *Schedulers* für solche Lasten hin. Die maximale Benutzerzahl ist hier wiederum systembedingt. Die Auslastungen $U_b > 1$ sind auf Ungenauigkeiten der Messung zurückzuführen. In dem Kapitel „IBM-System RS/6000-320“ auf Seite 55 konnte eine Tendenz des Meßsystems zur Unterschreitung der Sollwerte festgestellt werden. Die Auslastung wurde allerdings unter der Voraussetzung einer genauen Einhaltung dieser Sollwerte ermittelt. Unter Berücksichtigung dieser Abweichungen ergeben sich Auslastungen von etwa 99%.

¹¹ hier bezieht sich diese Aussage auf die Klasse LOW mit 5 Sekunden CPU-Zeit

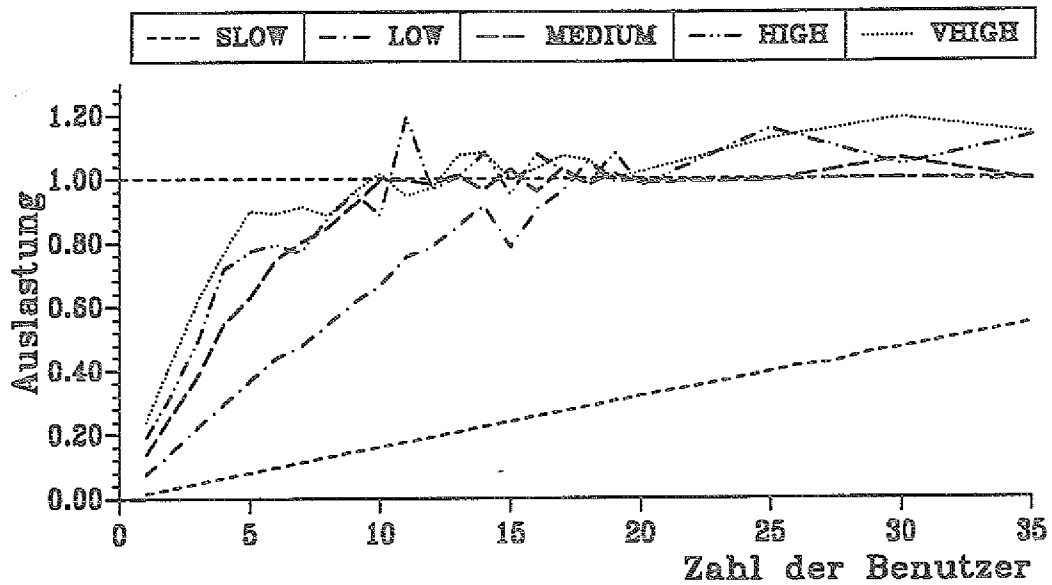


Abb. 42. Auslastung des IBM RS/6000-Systems bei verschiedenen Belastungen

CRAY X-MP/416

Die Auslastung $U_B > 1$ widersprechen der zugrundegelegten Theorie. Bei Kontrollmessungen konnten vergleichbare Ergebnisse erzielt werden. Somit muß das Modell von Denning und Buzen für dieses System als ungeeignet angesehen werden. Insbesondere das Antwortzeitgesetz (5) erscheint in diesem Fall nicht anwendbar.

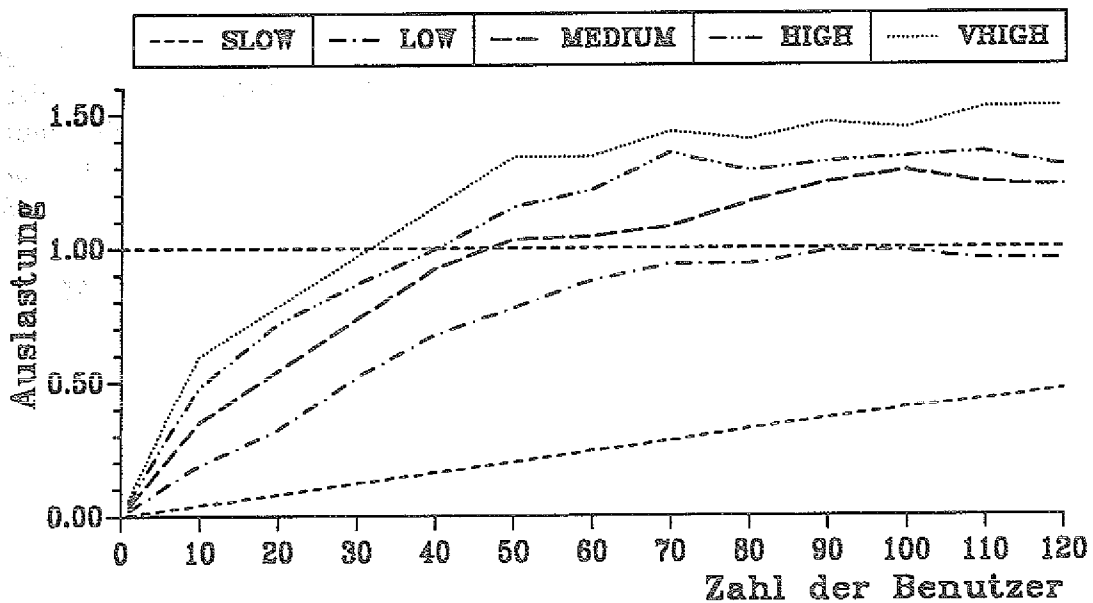


Abb. 43. Auslastung der CRAY X-MP bei verschiedenen Belastungen: Die maximale Benutzerzahl ist hier wiederum systembedingt.

Offenbar sind hierzu andere, von Denning und Buzen nicht berücksichtigte, Einflüsse auf die TTS-Faktoren verantwortlich. Parallelitäten bei der Abarbeitung der Belastungskerne könnten solche Ergebnisse hervorrufen.

Zur Kapazitätsabschätzung einer Cray X-MP in der hier betrachteten Konstellation können nur die TTSF-Kennlinien herangezogen werden. Aus dem gewählten Modell können keine aussagekräftigen Informationen bestimmt werden.

Bestimmung der Auslastung aus den Meßdaten

Die Auswertung der Meßergebnisse zeigt für eine X-MP/416 eine Abweichung des gemessenen Sättigungspunktes N^* von den Werten, welche sich durch Anwendung der Gleichung (8) ergeben.

Auffällig sind die zu hohen berechneten Auslastungen U_b die bei der Betrachtung der X-MP auftraten. Auf der CRAY X-MP standen andere Werkzeuge zur Bestimmung der Auslastung zur Verfügung (SAR). Eine Anwendung des SAR ergab in der Sättigung eine Auslastung zwischen 97 und 98%.

Zur Erklärung dieses Widerspruchs wurde aus den gemessenen Daten die Auslastung ohne Anwendung der Gleichungen nach Denning und Buzen bestimmt.

Im ersten Schritt wurde zu jedem Zeitpunkt innerhalb der Meßperiode die Auslastung bestimmt. Hierzu wurden die Zahl der Prozesse ermittelt, welche zu diesem Zeitpunkt Rechenleistung anforderten. Hier werden die Prozesse nicht berücksichtigt, welche die Denkzeit des Benutzers simulieren. Falls die Zahl der Prozesse, welche Rechenleistung anfordern, die Zahl der zur Verfügung stehenden Prozessoren übersteigt, wird eine Auslastung von 100% angenommen (Abb. 44).

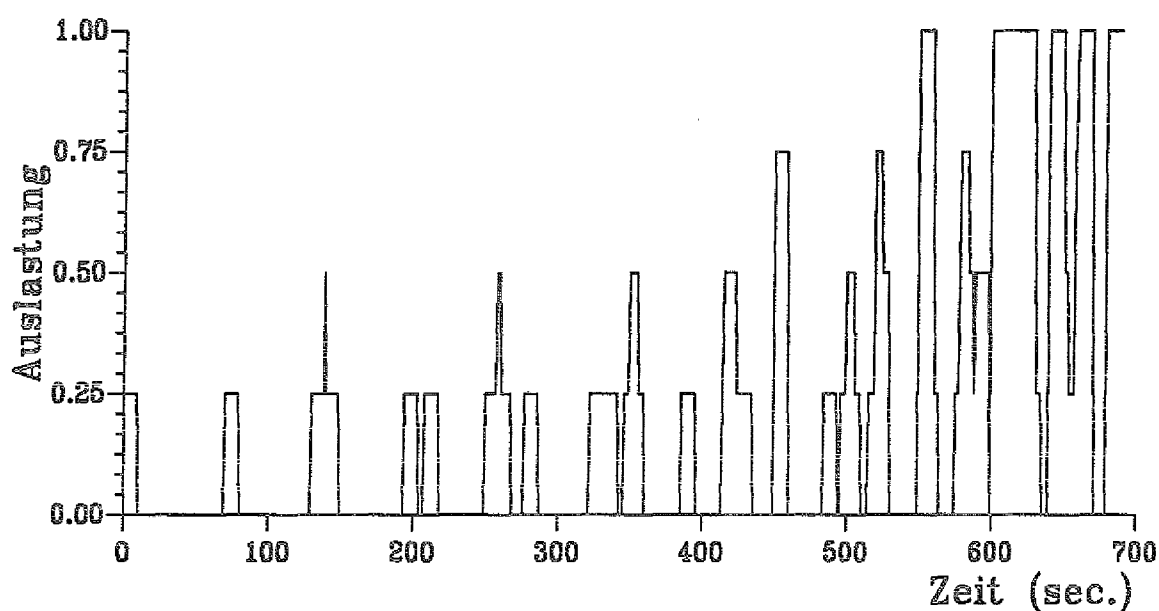


Abb. 44. Auslastung im zeitlichen Verlauf einer Messung: Es wird die Benutzerklasse LOW auf einer CRAY X-MP betrachtet.

Dieses Ergebnis wird auf die Zahl der Benutzer abgebildet. Hierzu wird für jeden Zeitpunkt die Zahl der Benutzer und die Auslastung bestimmt. Nun kann jeder Benutzerzahl ein Mittelwert der Auslastung zugeordnet werden (Abb. 45).

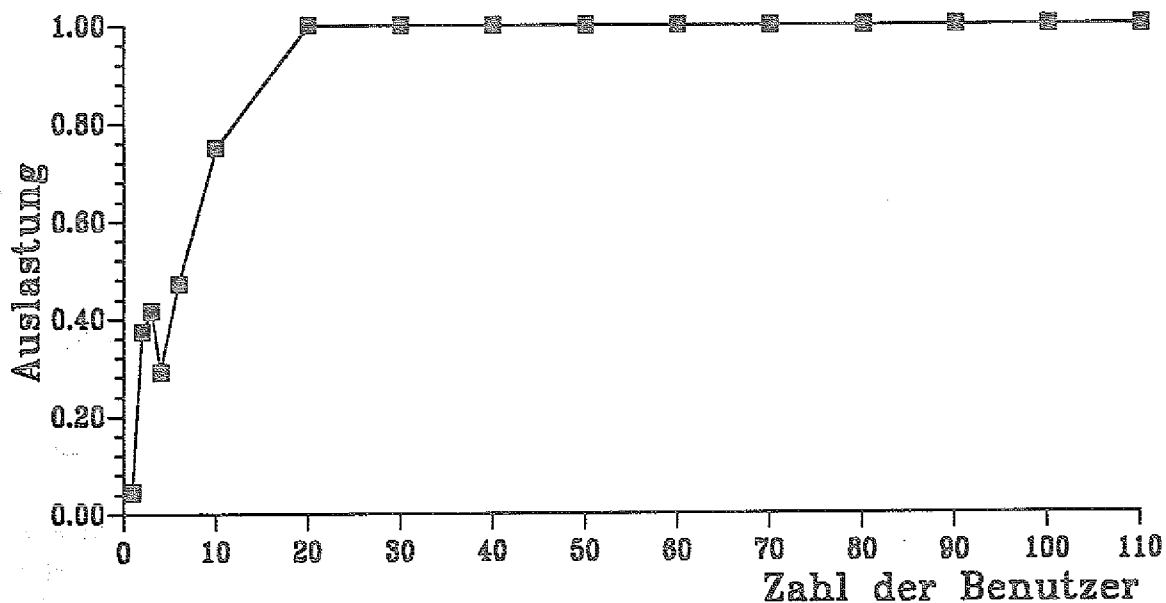


Abb. 45. Auslastung einer CRAY X-MP durch Benutzerklasse LOW

Hier erkennt man im Gegensatz zu Abb. 43 auf Seite 72, daß die CPU-Vollast bei etwa 20 Benutzern der Klasse LOW beginnt. Ein Absinken der TTSF-Werte bei hohen Benutzerzahlen kann hier aufgrund des gewählten Verfahrens nicht erfolgen da nicht unterschieden wird, ob ein Benutzer oder das System CPU-Zeit anfordert. Es wird ausschließlich festgestellt, ob ein Benutzer aktiv ist oder nicht. Hierdurch wird beispielsweise der durch *Paging* verursachte Aufwand als Antwortzeit des Prozesses betrachtet.

6.3 Konstanz des Denkzeitdehnfaktors

Diese Meßreihe untersucht die unterschiedlichen Reaktionen des *Schedulers* auf verschiedene Rechenzeiten bei konstanten Denkzeitdehnfaktor DDF.

$$DDF = \frac{D}{C}$$

Aufgrund der Konstanz des Denkzeitdehnfaktors ist für Auslastung des Systems im Bereich der Sättigung bei gleicher Benutzerzahl kein Unterschied zu erwarten. Die TTSF-Kennlinien müssen sich überlagern. Dies sollte überprüft werden.

Der DD-Faktor betrug bei allen Messungen $DDF = 12$. Es wurden jeweils zwei Meßreihen durchgeführt. Die CPU-Zeiten betrugen 1 bzw. 10 Sekunden.

IBM-System PS/2 Modell 70-A21

Für die CPU-Zeiten '1 Sekunde' und '10 Sekunden' treten keine relevanten Unterschiede bei der Steigung auf.

Die gemessenen Daten entsprechen aber weder dem theoretischen Modell noch den Erwartungen eines Benutzers, der bei diesem Rechnersystem höhere Antwortzeiten bei vielen kurzen Transaktionen erleidet als bei geringerer Anzahl von Transaktionen mit höheren CPU-Anforderungen. Dies weist wiederum auf Unzulänglichkeiten des *Schedulers* für kurze CPU-Zeiten hin.

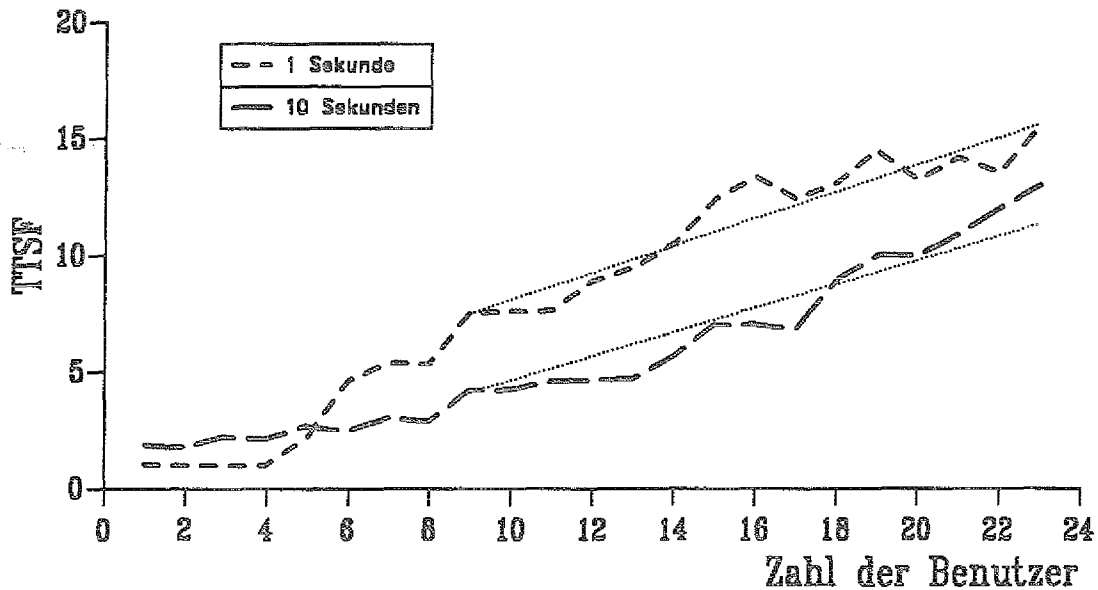


Abb. 46. Konstanter Denkzeitdehnfaktor auf PS/2: Das Verhältnis von CPU-Zeit zu Denkzeit wurde konstant gehalten.

IBM-System RS/6000-320

Eine Überlagerung der Kennlinien ist hier zwar auch nicht festzustellen, aber die Differenzen sind deutlich geringer als bei dem PS/2-System. Der *Scheduler* dieses Systems besitzt offenbar eine erheblich bessere Strategie zur Verwaltung von Prozessen mit kurzer CPU-Zeit.

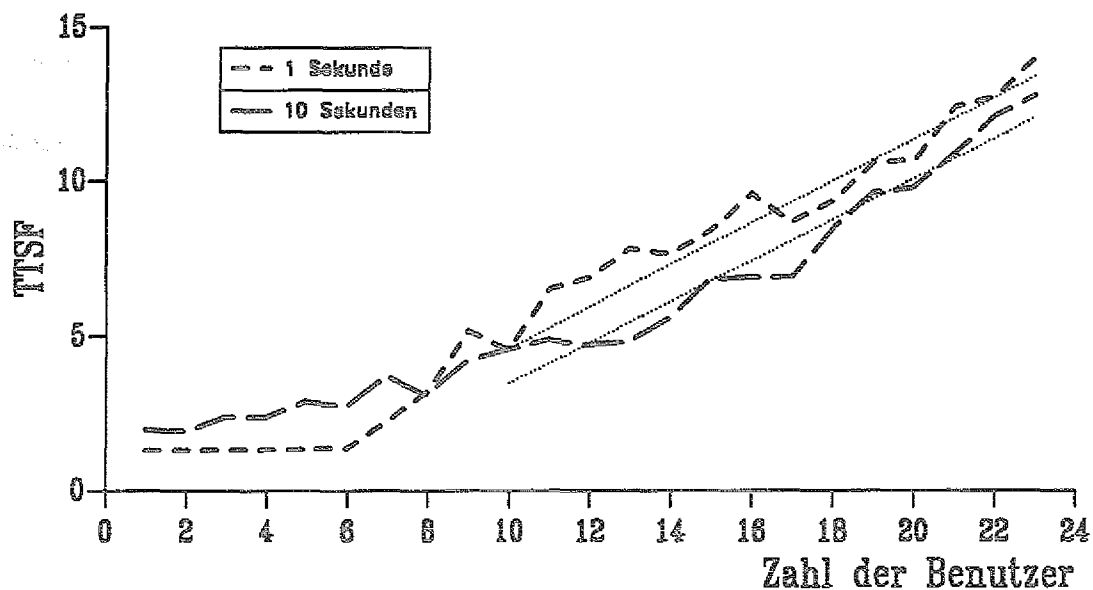


Abb. 47. Konstanter Denkzeitdehnfaktor auf IBM RS/6000: Das Verhältnis von CPU-Zeit zu Denkzeit wurde konstant gehalten, während die CPU-Zeit steigt.

Cray X-MP/416

Auf der Cray X-MP kann die beste Übereinstimmung beider Kennlinien festgestellt werden. Dieser *Scheduler* besitzt offenbar von den untersuchten Systemen die günstigste Strategie zur Verwaltung von Prozessen mit kurzer CPU-Zeit.

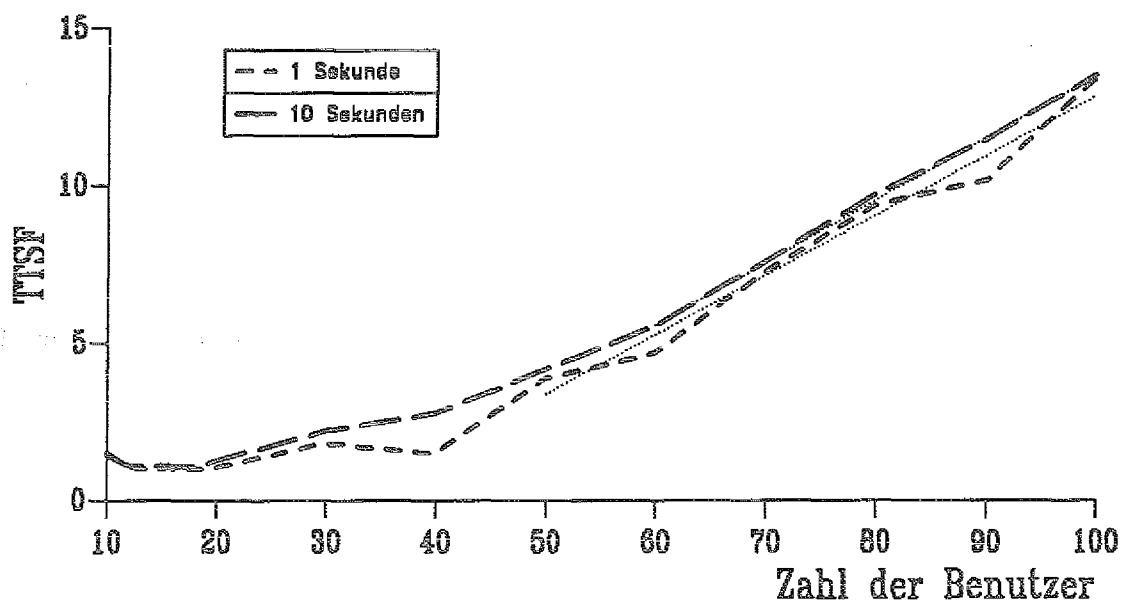


Abb. 48. Konstanter Denkzeitdehnfaktor auf einer X-MP: Das Verhältnis von CPU-Zeit zu Denkzeit wurde konstant gehalten.

6.4 Simulation interaktiver Benutzer im realen Betrieb der CRAY X-MP

Die bisherigen Untersuchungen wurden auf einer dedizierten CRAY X-MP durchgeführt. Somit konnten Aussagen über die Kapazität des Systems bei ausschließlich interaktiven Anwendungen getroffen werden. Ziel der folgenden Untersuchung war es, das Systemverhalten im laufenden Betrieb zu betrachten. Die Parameter der bisher verwendeten Klassen wurden unter Berücksichtigung der Ergebnisse aus Kapitel 4 verändert. Das System wurde somit zusätzlich zur realen Batch-Last mit einer steigenden Zahl simulierter Benutzer belastet.

Aus den Diagrammen in Kapitel 4 kann bei jeder vorgegebenen CPU-Zeit ein mittlerer Speicherbedarf und eine mittlere Anzahl Ein-/Ausgabecooperationen ermittelt werden. Im Rahmen der Messung wurde der Speicherbedarf von *Interbench* durch Korrektur der Ausgangsparameter berücksichtigt. Es ergeben sich somit folgende Parameter:

		SLOW	HIGH
CPU	sec.	1	15
Denkzeit	sec.	60	60
I/O	Anzahl	500	1000
Speicher	KByte	200	1800

Die hier untersuchten Klassen wurden mit der Zielsetzung ausgewählt, signifikante Differenzen der Parameter zu erhalten. Daher wurden die CPU-Zeiten und die Denkzeiten wie bei den bisher verwendeten Klassen SLOW und HIGH gewählt.

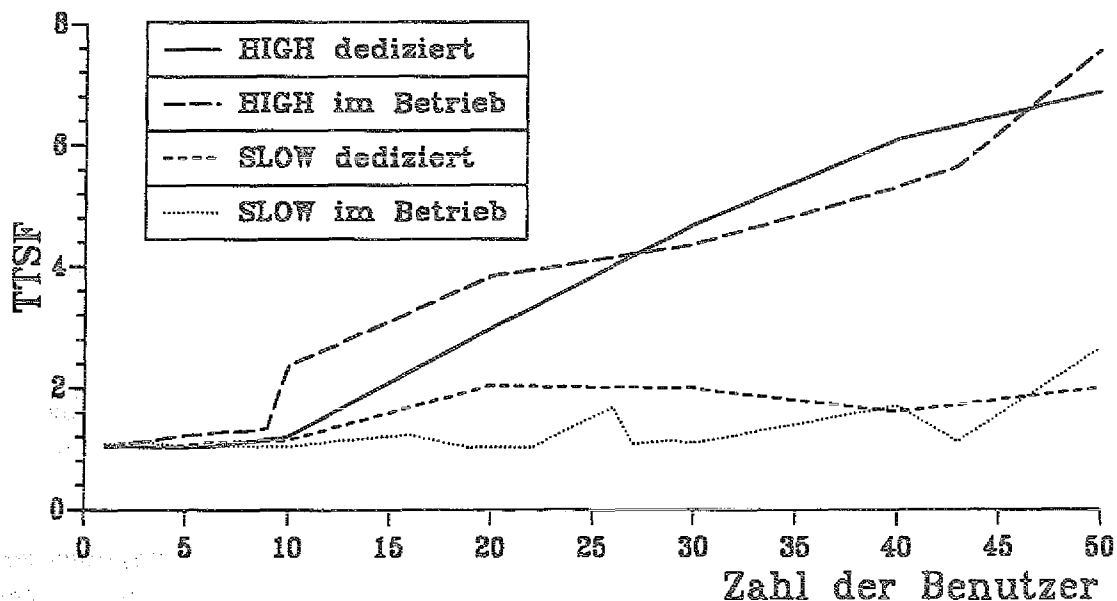


Abb. 49. Simulation entsprechend der Arbeitslastanalyse

In Abb. 49 werden die Ergebnisse einer Meßreihe im normalen Rechnerbetrieb den entsprechenden dedizierten Messungen gegenübergestellt.

In den untersuchten Klassen zeigt sich kein grundsätzlicher Unterschied zwischen den Ergebnissen der dedizierten Messung und der Messung im laufenden Betrieb. Die Ab-

weichungen der Kennlinien für $N \geq N^*$ könnten zum Teil auf die zusätzlich vorhandenen Batch-Lasten im realen Betrieb und den hierdurch nicht genau definierten und insbesondere nicht konstanten Lastzustand zurückzuführen sein. Eine fundierte Aussage kann nur unter Zuhilfenahme weiterer Untersuchungen getroffen werden.

Insbesondere fallen zwei Ergebnisse besonders auf:

1. Im Bereich der Teilauslastung ($N \leq N^*$) zeigt sich kein Unterschied zwischen der dedizierten Messung und dem normalen Betrieb. Offenbar werden interaktive Lasten durch die von der Systemverwaltung eingestellten Prioritäten stark bevorzugt.
2. Der Sättigungspunkt $N^* = 7$ der untersuchten Klasse HIGH zeigt keinen relevanten Unterschied zwischen beiden Meßreihen. Für die Klasse SLOW konnte wie bei früheren Untersuchungen kein Sättigungspunkt im untersuchten Bereich $N < 50$ Benutzer bestimmt werden.

Eine Analyse der Meßreihen im laufenden Betrieb mittels SAR ergab ähnliche Ergebnisse: Bei steigender Zahl interaktiver Prozesse wurden diese derart stark bevorzugt, daß keine Batch-Lasten mehr abgearbeitet werden konnten.

Diese Bevorzugung interaktiver Prozesse ist darauf zurückzuführen, daß während der normalen Dienstzeit die Prioritäten der Batch-Lasten reduziert werden. Außerhalb der Dienstzeit wird die Reduzierung der Prioritäten automatisch zurückgenommen. Diese Maßnahme soll eine vollständige Blockade der Batch-Lasten durch interaktive Anwender verhindern. Der Zeitraum der Absenkung liegt z.Zt.¹² zwischen 8:00 und 18:00 an Werktagen.

6.5 Vergleich der untersuchten Systeme

Bei Vergleich der untersuchten Systeme können mehrere Ergebnisse festgehalten werden:

1. Das Benchmarksystem *Interbench* verfälscht infolge von unvermeidlichen Verwaltungszeiten die Ergebnisse nicht in einem Maße, welches deren Aussagekraft einschränkt.
2. Das RISC-System IBM RS/6000-320 erreicht einen Großteil der vom Hersteller angegebenen maximalen Leistungsfähigkeit ohne besonderen Maßnahmen seitens des Benutzers. Hierdurch wird die Eignung für allgemeine Anwendungen unterstrichen.
3. Das Antwortzeitgesetz (5) zeigte sich als nicht geeignet zur Anwendung bei einer CRAY X-MP. Das Modell von Denning und Buzen erscheint für diese Architektur ungeeignet. Zur Beschreibung dieses Systems müßte das Modell durch Einführung weiterer Parameter detaillierter werden.
4. Die ermittelten TTSTF-Kennlinien können zur Kapazitätsabschätzung herangezogen werden. Sie beruhen nicht auf der Theorie von Denning und Buzen, sondern spiegeln ausschließlich das Systemverhalten während der Messungen wider.

¹² im August 1991

Leistungsfähigkeit der untersuchten Systeme

Es wird die Leistungsfähigkeit der hier untersuchten Systeme dargestellt (Abb. 50 auf Seite 79). Berücksichtigt wurden ausschließlich die Skalierungswerte der CPU. Aus der Zahl der Durchläufe des Belastungskerns pro Sekunde und der in jedem Durchlauf ausgeführten Fließkommaoperationen kann ein Anhaltspunkt zur Bestimmung der Leistungsfähigkeit gewonnen werden. Andere Betriebsmittel, wie beispielsweise Ein-/Ausgabeeinheiten, werden hier nicht betrachtet.

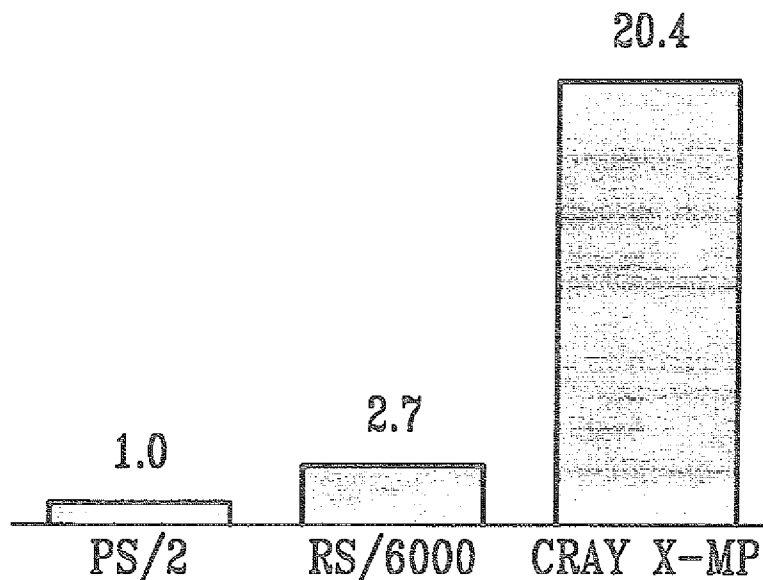


Abb. 50. Leistungsfähigkeit: Die skalare Leistung der untersuchten Systeme in vergleichender Darstellung. Die Leistung des PS/2-Systems wird mit 1 angesetzt und die übrigen Systeme entsprechend skaliert.

Die Aussagen beziehen sich ausschließlich auf den untersuchten Belastungskern. Es sind keine Rückschlüsse auf andere Anwendungen sinnvoll, da ausschließlich die Geschwindigkeit der Fließkommaeinheiten untersucht wurde. Bei diesem Vergleich sind mehrere Faktoren zu berücksichtigen:

1. Das untersuchte PS/2-System verfügt ebenso wie das IBM-System RS/6000-320 über einen zusätzlichen Koprozessor für Fließkommaoperationen. Somit ergibt sich eine besondere Eignung für den abgearbeiteten Belastungskern.
2. Der Belastungskern wurde so ausgelegt, daß die Vorteile einer CRAY X-MP nicht zum Tragen kommen. Die *Pipelining*-Einheiten werden nicht angesprochen. Somit wird der unmittelbare Vergleich des IBM RS/6000 mit der CRAY X-MP verzerrt.
3. Dieser Vergleich ermöglicht eine Aussage über die Abarbeitungsgeschwindigkeit eines Prozesses welcher ausschließlich skalare Operationen durchführt. Er berücksichtigt die Möglichkeit, auf der untersuchten X-MP vier Prozesse gleichzeitig abarbeiten zu können.

6.6 Erfahrungen mit dem Programmsystem Interbench

Im Verlauf der Untersuchungen traten einige signifikante Eigenschaften des Systems *Interbench* zutage. Diese werden im folgenden aufgeführt:

- Im ursprünglichen Programm wurde die Denkzeit konstant gehalten. Bei Generierung einer größeren Prozeßzahl ergab sich jedoch eine Häufung vieler Prozesse, welche gleichzeitig die CPU belasteten und anschließen wieder inaktiv wurden. Die CPU wurde zeitweise gar nicht belastet. Dies führte in Testmessungen zu deutlichen Fehlern, insbesondere konnten aus den TTSTF-Kennlinien nur sehr niedrige Werte für N^* ermittelt werden.

Der Grund für die angesprochenen Häufungspunkte war die Verlängerung der Antwortzeit durch die gleichzeitige Aktivierung mehrerer Prozesse. Bei geringer Prozeßzahl kann eine solche Überlappung durch entsprechende Auslegung der Befehlsdateien verhindert werden, bei einer größeren Prozeßzahl ist dies nicht mehr möglich. Aufgrund der konstanten Parameter traten diese Häufungspunkte periodisch auf und zeigten Tendenz zur Vergrößerung.

Ein Weg zur Vermeidung dieser Häufungspunkte war die Einführung der Streuung für die Denkzeit. Hier wurde eine Streuung von 25% der vorgegebenen Denkzeit für jeden Prozeß realisiert. Die mittlere Denkzeit der Prozesse bleibt allerdings weiterhin konstant.

Alle hier aufgeführten Untersuchungen wurden mit dieser Streuung der Denkzeit durchgeführt.

- Insbesondere die Auslastungs-Kennlinien des IBM-Systems RS/6000-320 offenbarten die Folgen von Ungenauigkeiten der Auswertungen. Für weitere Untersuchungen empfiehlt sich einerseits die manuelle Korrektur der Skalierung, welche hier aus Gründen der Vergleichbarkeit nicht vorgenommen wurden. Andererseits zeigten Kontrollmessungen eine Abweichung der Denkzeit von der Vorgabe bis zu 0.5 Sekunden. Diese Abweichung ist durch das jeweilige Betriebssystem bedingt und kann nicht beeinflußt werden. Eine Bestimmung der realen Denkzeit während der Messungen und Berücksichtigung in der Auswertung könnte diesen Fehlereinfluß minimieren. Insbesondere für kurze Denkzeiten würde sich eine deutliche Verbesserung ergeben.

In den durchgeführten Messungen mit Denkzeiten von 60 Sekunden liegt die aufgetretene Schwankung der Denkzeit bei etwa 1%.

- Jede Abarbeitung einer Steuerdatei muß manuell beendet werden.
Für die Zukunft kann es somit sinnvoll sein, einen Mechanismus zu entwickeln, welcher es gestattet, automatisch mehrere Meßreihen zu starten und zu beenden. Dann können Untersuchungen auch in Zeiten durchgeführt werden, zu denen nur wenige andere Systembenutzer gestört werden (z.B. nachts).
- Die Betrachtung der Ein-/Ausgabeeinheiten (E/A) als Engpaß des Systems führten zu keinen eindeutigen Aussagen. Die Bestimmung eines Sättigungspunktes konnte nicht erfolgen.
Interbench berücksichtigt die CPU-Zeit, welche durch E/A verursacht wird, so daß die gesamte CPU-Zeit, auch aufgrund von E/A-Operationen, nicht die Vorgabe

übersteigt. Die Untersuchungsergebnisse zeigten auf den verschiedenen Systemen grundsätzlich andere Charakteristika:

Das PS/2-System wickelt offenbar den gesamten E/A über die Recheneinheit ab. Es zeigten sich keine Unterschiede zwischen Meßreihen gleicher CPU-Belastung, unabhängig von der zusätzlich generierten Belastung der E/A-Einheiten.

Andere Ergebnisse zeigten sich auf dem RISC-System. Es konnten keine grundsätzlichen Abhängigkeiten zwischen der Zahl der generierten Prozesse und dem TTSF gefunden werden. An einigen Punkten konnten deutliche Spitzenwerte festgestellt werden, diese waren aber nicht grundsätzlich einer erhöhten E/A-Belastung zuzuordnen. Eine Erklärung kann in dem Festplatten-Subsystem dieser Architektur gesucht werden.

7.0 Ausblick

Im Rahmen dieser Arbeit konnten nur einige der möglichen Aspekte interaktiver Systeme betrachtet werden. In der Zukunft sind verschiedene Ergänzungen denkbar, welche die Problematik tiefer beleuchten können. Einige der möglichen Aspekte sollen hier umrissen werden:

- Für die untersuchten Systeme wurden die internen Antwortzeiten bestimmt. Ein wesentliches Merkmal von *Client-Server*-Systemen ist das Netzwerk zwischen dem Arbeitsplatzrechner und dem Computer-Server (Supercomputer). Eine Implementierung des Benchmarks *Interbench* auf dem Arbeitsplatzrechner würde es erlauben, die externe Antwortzeit des *Compute-Servers* zu bestimmen.
- Zur Überwachung eines Rechnersystems kann ein Sondenverfahren eingesetzt werden. Hier wird etwa ein Meßprozeß periodisch gestartet und seine Antwortzeit ermittelt. Aus den Meßwerten können Zeiträume bestimmt werden, in denen hohe oder niedrige Belastungen des Rechnersystems vorliegen. *Interbench* kann für eine solche Sondenmessung benutzt werden.

Die Realisierung dieser Ansätze kann in Verbindung mit der vorliegenden Arbeit ein flexibles Werkzeug zur Leistungsbeurteilung interaktiver Rechnersysteme und -netzwerke bieten.

The second part of the document focuses on the role of the accounting department in providing accurate and timely financial information. It outlines the various responsibilities of the accounting team, including the preparation of financial statements, the management of accounts payable and receivable, and the oversight of budgeting and forecasting activities. The document also discusses the importance of maintaining strong relationships with external auditors and regulatory bodies.

8.0 Literaturverzeichnis

- [Bach] M.J. Bach
The Design of the UNIX Operating System
Prentice-Hall, Englewood Cliffs, New Jersey (1986)
- [Ben] N. Benwell (ed.)
Benchmarking
John Wiley & Sons, New York (1975)
- [BEGK] T. Bönninger, R. Esser, J. Gehm und D. Krekel
Comparative Description of High-Performance Computers based on a
Catalog of Classificatory Characteristics
Angewandte Informatik, Vol. 30 (1988), pp 51-70
- [Bou] S.R. Bourne
Das UNIX System V
Addison-Wesley, Bonn (1989)
- [BuDe] J.P. Buzen, P.J. Denning
The Operational Analysis of Queueing Network Models
Computing Surveys, Vol.10, No.3, September 1978, pp. 225-261
- [CoKI] E.G. Coffman and L. Kleinrock
Computer Scheduling Measures and their Countermeasures,
Proc AFIPS 32, SJCC, pp. 11-21
- [CRAY,a] C Library Reference Manual
CRAY Research Inc., SR-0136 5.0, (1989)
- [CRAY,b] UNICOS System Calls Reference Manual, Vol.4
CRAY Research Inc., SR-2012 5.0, (1989)
- [CRAY,c] CRAY Y-MP, CRAY X-MP EA, and CRAY X-MP Multitasking
Programmer's Manual
CRAY Research Inc., SR-0222 F, (1989)
- [Don] J.Dongarra
The Linpack Benchmark
Proceedings of the First International Conference on Supercomputers,
(1987), pp.454-474
- [DimIy] R.T. Dimpsey and R.K. Iyer
Performance Degradation due to Multiprogramming and System
Overheads in Real Workloads: Case Study on a Shared Memory
Multiprocessor
Proceedings of 1990 International Conference on Supercomputing, pp.
227-238

- [EMR] G. Engeln-Müllges, F. Reutter
Numerische Mathematik für Ingenieure
BI-Wissenschaftsverlag, Mannheim (1987)
- [Goff] N.S. Goff
The Case for Benchmarking
Computers and Automation, May 1973, pp. 23-25
- [GuMe] W. Gürich, B. Mertens
Comparisons of Internal and External Interactive Response Times
Computer Performance, Vol. 4 (1983), pp. 13-18
- [GuTo] N.J. Gunther and M.T. Toga
A Benchmark for Shared-Memory Architectures
Computer Architecture News 17, (1989), pp. 54-61
- [Hei] Ms.R. Heidel (ed.)
Computer Management and Evaluation, Vol. 4
Share Inc. Chicago, Illinois 60601 (1982), pp. 316-323
- [HeKa] J. Helin and K. Kaski
Performance analysis of high-speed computers II
Tampere University of Technology, Electronics Laboratory Report 3-89
(1989)
- [HoJe] R.W. Hockney, C.R. Jesshope
Parallel Computers 2
Hilger, Bristol (1988)
- [Hoß] F. Hoßfeld
Vector-Supercomputers
Parallel Computing 7 (1988), pp. 373-385
- [Ihrer] F.C. Ihrer
Benchmarking vs. Simulation
Computer and Automation, November 1972, pp. 8-10
- [IBM,a] IBM
AIX Operating System, Technical Reference, Vol. 1
Danbury, USA (1989)
- [IBM,b] IBM
AIX Operating System, Technical Reference, Vol. 2
Danbury, USA (1989)
- [Jel] R. Jeltsch
Numerische Mathematik für Ingenieure, Teil A
Institut für Geometrie und Praktische Mathematik der RWTH Aachen
(1987)
- [KePo] A. Kelley, I. Pohl

C: Grundlagen und Anwendungen
Addison-Wesley, Bonn (1987)

[Klein] L. Kleinrock
Queueing Systems, Volume 1: Theory
John Wiley & Sons, New York (1975)

[Linn] M.A. Linn
Eine Programmierungsumgebung zur Messung der wechselseitigen Einflüsse
von Hintergrundlast und parallelem Programm
*Forschungszentrum Jülich, Zentralinstitut für Angewandte Mathematik,
Berichte Nr. 2416* (1990)

[Lor] H. Lorin
Introduction to Computer Architecture and Organization
John Wiley & Sons, New York (1982)

[Mar] J. Martin
System Analysis for Data Transmission
Prentice-Hall, Englewood Cliffs, New Jersey (1972), pp. 64-66

[Mart] J.L. Martin
Performance Evaluation: Applications and Architectures
Proceedings of the Second International Conference on Supercomputers
Vol. III, (1987), pp. 369-373

[MoRo] M.F. Morris, P.F. Roth
Computer Performance Evaluation, Tools and Techniques for Effective
Analysis
Van Nostrand Reinhold, New York (1982)

[Schw] H.R. Schwarz
Numerische Mathematik
B.G.Teubner, Stuttgart (1988)

[Stim] S. Stimler
Leistungsbewertung, Leistungsmessung und Leistungsverbesserung von
Datenverarbeitungssystemen
Oldenbourg Verlag, München (1976)

[Thom] R. Thomas
Parallelisierung des Max-Flow-Algorithmus von Shiloach/Vishkin auf
Cray-Rechnern
Forschungszentrum Jülich, Jül-Spez 557, (1990)

[TsBern] D.C. Tsichritzis and P.A. Bernstein
Operating Systems
Academic Press, New York (1974)

THEORY OF THE EARTH

CHAPTER I. OF THE ORIGIN OF THE EARTH.

1.1

1.2

1.3

1.4

1.5

1.6

1.7

1.8

1.9

1.10

1.11

1.12

1.13

1.14

Anhang A. Beispiel einer Befehlsdatei

LOW	1	0S	1
LOW	1	30S	0
LOW	1	60S	0
LOW	1	90S	0
LOW	1	120S	0
MEDIUM	1	150S	1
MEDIUM	1	350S	0
MEDIUM	1	550S	0
MEDIUM	1	750S	0
HIGH	1	1000S	1
LOW	5	1500S	0
MEDIUM	1	2000S	0
MEDIUM	1	2500S	0
MEDIUM	1	3000S	0
MEDIUM	1	3500S	0
MEDIUM	1	4000S	0
HIGH	1	4500S	0
HIGH	1	5000S	0

Die erste Spalte gibt die Klasse des zu startenden Prozesses an. In der zweiten Spalte wird die Zahl der erzeugten Prozesse festgelegt. Die Verzögerung des Prozeßstartes in Sekunden wird durch die vierte Spalte festgesetzt, während die fünfte Spalte angibt, wieviele der gestarteten Prozesse Messungen durchführen.

Anhang B. Parameter in den verwendeten Klassen

Klassenname	Speicherbedarf (KB)	Speichernutzung	CPU-Zeit (sec.)	Anzahl E/A	Durchlaufen der inneren Schleife	Denkzeit (sec.)	Startverzögerung (sec.)	Streuung der Startverzögerung
KALIB1	0	100	1	10	1	1	0	0
KALIB5	0	100	5	10	1	1	0	0
KALIB10	0	100	10	10	1	1	0	0
KALIB15	0	100	15	10	1	1	0	0
KALIB20	0	100	20	10	1	1	0	0
SLOW	10	100	1	10	1	60	0	0
LOW	10	100	5	10	1	60	0	0
MEDIUM	10	100	10	10	1	60	0	0
HIGH	10	100	15	10	1	60	0	5
VHIGH	10	100	20	10	1	60	0	0
THF1	1	0	1	0	1	12	0	0
THF5	1	0	5	0	1	60	0	0
THF10	1	0	10	0	1	120	0	0
NULL	0	0	0	0	1	60	25	10
SLOWB	0	100	1	500	1	60	0	0
HIGHB	1500	100	15	1000	1	60	0	0

THEORY OF THE EARTH AND ITS HISTORY

The theory of the earth and its history is a branch of geology which deals with the origin and development of the earth and its various parts. It is a science which seeks to explain the processes which have shaped the earth and its features, and to determine the sequence of events which have taken place since the earth was first formed. The theory of the earth and its history is based on the study of the rocks and the fossils which they contain, and on the principles of geology which govern the formation and development of the earth. The theory of the earth and its history is a science which is constantly developing, and which is based on the latest discoveries and research in geology.

Anhang C. Beispiel einer Ausgabedatei

Messung von Fri Jul 12 18:48:14 1991

Operating-System: AIX

Hardware: i386

Class	Process-ID	rel.time (sec.)	CPU (msec.)	elapsed (sec.)	Factor	active (Stk.)
MEDIUM	1200435	181.0	1.003e+04	2.0e+01	2.0e+00	2
MEDIUM	1200436	197.0	1.005e+04	2.0e+01	2.0e+00	2
MEDIUM	1200435	261.0	1.002e+04	2.3e+01	2.3e+00	2
MEDIUM	1200436	272.0	1.005e+04	2.2e+01	2.2e+00	2
MEDIUM	1200437	326.0	1.003e+04	1.8e+01	1.8e+00	3
MEDIUM	1200435	344.0	1.005e+04	2.7e+01	2.7e+00	3
MEDIUM	1200436	349.0	1.008e+04	2.5e+01	2.5e+00	3
MEDIUM	1200437	417.0	1.008e+04	2.5e+01	2.5e+00	3
MEDIUM	1200436	429.0	1.000e+04	3.1e+01	3.1e+00	3
MEDIUM	1200435	431.0	1.005e+04	3.0e+01	3.0e+00	3
MEDIUM	1200438	479.0	1.005e+04	1.8e+01	1.8e+00	4
MEDIUM	1200437	516.0	1.003e+04	3.6e+01	3.6e+00	4
MEDIUM	1200436	515.0	1.007e+04	3.7e+01	3.7e+00	4
MEDIUM	1200435	521.0	1.007e+04	3.4e+01	3.3e+00	4
MEDIUM	1200438	563.0	1.002e+04	1.7e+01	1.7e+00	4
MEDIUM	1200441	1039.0	1.003e+04	3.9e+01	3.8e+00	7
MEDIUM	1200436	1039.0	1.003e+04	4.0e+01	4.0e+00	7
MEDIUM	1200440	1049.0	1.007e+04	4.6e+01	4.6e+00	7
MEDIUM	1200437	1086.0	1.005e+04	4.1e+01	4.1e+00	7
MEDIUM	1200441	1131.0	1.005e+04	3.3e+01	3.3e+00	7
MEDIUM	1200436	1133.0	1.007e+04	3.3e+01	3.3e+00	7
MEDIUM	1200440	1144.0	1.005e+04	2.9e+01	2.9e+00	7
MEDIUM	1200438	1173.0	1.002e+04	3.2e+01	3.2e+00	8
MEDIUM	1200440	1221.0	1.007e+04	5.5e+01	5.4e+00	8
MEDIUM	1200436	1222.0	1.007e+04	5.5e+01	5.4e+00	8
MEDIUM	1200438	1271.0	1.007e+04	2.1e+01	2.1e+00	8
MEDIUM	1200435	1299.0	1.002e+04	3.1e+01	3.1e+00	8
MEDIUM	1200439	1300.0	1.003e+04	3.2e+01	3.2e+00	8
MEDIUM	1200440	1323.0	1.015e+04	6.4e+01	6.3e+00	9
MEDIUM	1200441	1328.0	1.002e+04	7.1e+01	7.0e+00	9
MEDIUM	1200435	1391.0	1.005e+04	4.4e+01	4.4e+00	9
MEDIUM	1200439	1393.0	1.005e+04	4.2e+01	4.2e+00	9
MEDIUM	1200440	1436.0	1.013e+04	2.4e+01	2.3e+00	9
MEDIUM	1200441	1452.0	1.002e+04	3.3e+01	3.2e+00	9
MEDIUM	1200436	1458.0	1.008e+04	4.0e+01	3.9e+00	9

Diese Arbeit wurde als Diplomarbeit am Lehrstuhl für Technische Informatik und Computerwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen erstellt.

Herrn Prof. Dr. F. Hoßfeld danke ich für die Möglichkeit, die Arbeit am Zentralinstitut für Angewandte Mathematik des Forschungszentrums Jülich (KFA) anzufertigen. Die gewährte technische und personelle Unterstützung erwies sich als vorbildlich. Insbesondere hervorheben möchte ich das Verständnis, welches den Problemen eines Studenten entgegengebracht wurde.

Weiterhin möchte ich allen Institutsangehörigen danken, welche die Entstehung dieser Arbeit gefördert haben. Mein besonderer Dank gilt Herrn W. Gürich für die Betreuung dieser Arbeit und den Herren O.Büchner, Hel. Schumacher, V. Sander, und H.R. Berrendorf sowie Frau M.E. Sichelschmidt für die gewährte Unterstützung.

Handwritten text, likely a list or notes, starting with "Handwritten text" and "Handwritten text".

Handwritten text, likely a list or notes, starting with "Handwritten text" and "Handwritten text".

JÜL-2597

März 1992

ISSN 0366-0885