Leistung und Performance von MVS-Großrechnern

von
Dipl.-Math. Klaus Becker

R. Oldenbourg Verlag München Wien 1996

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Becker, Klaus:
Leistung und Performance von MVS-Großrechnern / von Klaus Becker. - München ; Wien : Oldenbourg, 1996
ISBN 3-486-23767-5

Gesamtherstellung: WB-Druck, Rieden

ISBN 3-486-23767-5

Inhaltsverzeichnis

Vorwort

Die DV-Landschaft befindet sich Mitte der 90er Jahre in einem gewaltigen Umbruch. Zentral strukturierte Systemlösungen mit Mainframes als hard- und softwaretechnische Plattform für die DV-Abwicklung eines Unternehmens werden zunehmend durch dezentrale Strukturen ergänzt, teilweise auch ersetzt. Down-Sizing ist ein häufig zitiertes und gehörtes Schlagwort. Im Rahmen von Client/Server-Architekturen werden Applikationen auf mehrere Plattformen verteilt (verteilte Verarbeitung). Die Mensch-Maschine-Kommunikation wird zum Beispiel in Form von grafischen Benutzeroberflächen auf Arbeitsplatzsysteme verlagert. Als Server kommen UNIX-Systeme und/oder traditionelle Mainframes zum Einsatz. Gleichzeitig werden die Bemühungen zur Leistungssteigerung im Großsystembereich verstärkt. Das Stichwort heißt Parallelisierung. Es ist möglich, MVS-Großrechner über eine neue Coupling-Technologie zu einem System zu koppeln (Single-System-Image) mit gleichzeitigem Lastausgleich bis auf Transaktionsebene. (Parallel Sysplex-Architecture). Hinzu kommt, daß der Marktführer IBM die Entwicklung herkömmlicher auf bipolarer Chip-Technologie basierender Rechner aufgibt und zukünftig ausschließlich auf CMOS-basierende Prozessoren setzt. Eine rasante Entwicklung nimmt die Leistungsfähigkeit der UNIX-basierenden Systeme mit massiv paralleler Architektur (MPP von Massiv Parallel Processor im Gegensatz zu SMP von Symmetric Multiprocessor). Das Betriebssystem MVS selbst öffnet sich (Open Edition MVS) und unterstützt mehr und mehr heterogene Systemlandschaften. Wie paßt das alles zusammen? Welche Richtung nimmt die DV-Technologie? So komplex die Fragestellung ist, so wenig einfach kann die Antwort sein. Wir fassen die Entwicklungstendenzen zusammen und befassen uns kurz mit einigen Aspekten:

- Client/Server-Architekturen,
- Dezentralisierung/Down-Sizing,
- Leistungssteigerung im Großsystem- und UNIX-Umfeld und
- Open Edition MVS.

Client/Server-Architekturen

Client/Server-Architekturen unterstützen den Endbenutzer mit grafischen Oberflächen. Ziel ist die Effizienzsteigerung beim Endbenutzer (bessere Bedienbarkeit der Endgeräte, höhere Akzeptanz, we-

niger Schulungsaufwand). Diese Entwicklung unterstützt die als notwendig erkannte Erhöhung des Terminalisierungsgrades, um alle MitarbeiterInnen des Unternehmens mit Standardapplikationen wie Text- und Tabellenverarbeitung versorgen und die Unternehmensabläufe DV-technisch durchdringen zu können (Bürokommunikation, computerunterstützte Sachbearbeitung, Vorgangsbearbeitung). Zentralrechner werden in diesem Umfeld als

- Verwalter der Unternehmensdaten (Data-Base-Server),
- Transaktionsabwickler (Transaction-Server) und
- Abwickler komplexer Rechenwerke

eingesetzt.

A priori ist der Zentralrechner kein MVS-Rechner. Abhängig von der Größe des Unternehmens, nicht zuletzt auch aus Gründen des Investitionsschutzes, werden MVS-Systeme in diesem Umfeld ihre traditionelle Stärke hinsichtlich

- Verfügbarkeit,
- Datenintegrität und
- Sicherheit

ausspielen können.

In kleineren Systemumgebungen werden UNIX-Systeme als Hostserver im Einsatz sein. Eine dritte Systemebene zwischen MVS-Großsystem und Arbeitsplatzrechner, z. B. auf UNIX-Basis, wird sich im kommerziellen Umfeld vorrangig bei verteilten Unternehmensstrukturen durchsetzen können (Filialen, Außenstellen). In zentral organisierten Unternehmen ergibt sich dafür weniger eine Rechtfertigung. Kapazitätsprobleme im Netzbereich, aber auch wirtschaftliche Überlegungen und vom Markt angebotene Anwendungslösungen können allerdings eine Dezentralisierung in diesem Umfeld erzwingen.

Dezentralisierung/Down-Sizing

Mit Dezentralisierung und Down-Sizing verfolgen Unternehmen die Absicht, ihre DV-Dienstleistungen wirtschaftlicher anzubieten und flexibler auf die sich rasch verändernden Anforderungen reagieren zu können. Auch in diesem Zusammenhang spielt die Unternehmensgröße und damit der Umfang der Rechenwerke, der Umfang der Datenhaltung und der Umfang der Transaktionsverarbeitung sowie der Investitionsschutz eine entscheidende Rolle. Inzwischen scheint festzustehen, daß dezentralisierte DV-Lösungen a priori nicht

kostengünstiger arbeiten als zentrale Mainframe-Rechenzentren. Partiell wird auch schon über einen Stop, wenn nicht sogar über eine Umkehr, des Trends zur Dezentralisierung diskutiert. Die Wortschöpfung "Right-Sizing" kennzeichnet diese Entwicklung.

Leistungssteigerung im Großsystem- und UNIX-Umfeld

Seit Jahren beobachtet man eine von wirtschaftlichen Überlegungen getriebene Konsolidierung von Rechenzentren. Als Folge werden die zu beherrschenden Einheiten komplexer und der Leistungsanspruch höher. In diesem Umfeld kommt die Parallel Sysplex-Architecture zum Einsatz. Eine neue Kopplungstechnik macht derartige Großkomplexe erst handhabbar (Single-Image-Systeme) und eröffnet ein aus heutiger Sicht nahezu unbegrenztes horizontales Wachstum. In einem Sysplex-Verbund können über die neue Coupling-Facility bis zu 32 MVS-Systeme mit einander verbunden werden. Während den ersten Systemen mit CMOS-Technologie noch spezielle Aufgaben zugedacht waren, gelten die Nachfolgesysteme IBM 9672 der Modellreihe R als universale MVS-Rechner. Ein zur Zeit noch bestehendes Problem ist die relativ geringe Leistungsfähigkeit der CMOS-Prozessoren in der Größenordnung von ca. 20 Mips. Erwartet wird eine Verdoppelung der Leistung im Zweijahresrhythmus. Eine zweite Entwicklungsrichtung verläuft weg von den symmetrischen Multiprozessorsystemen (SMPs) in Richtung Massiv Paralleler Architekturen (MPPs), die häufiger auch schon im kommerziellen Umfeld eingesetzt werden. UNIX-basierende MPPs stellen inzwischen ein enormes Leistungspotential zur Verfügung, so daß unter reinen Leistungsgesichtspunkten die DV einer größeren Unternehmung über derartige Systeme abgewickelt werden kann. Das Hauptproblem ist der fehlende Schutz der in die Anwendungssysteme getätigten Investitionen. Es wird spannend bleiben, wohin die Entwicklung geht. Bleiben wird das Problem, daß die bestehende Anwendungslandschaft (Anwendungsarchitektur) bei Nutzung Massiv Paralleler Systeme geändert werden muß. Bei Nutzung der Parallel Sysplex-Architektur auf Basis von symmetrischen Multiprozessorsystemen können die herkömmlichen Anwendungen (beispielsweise IMS- und CICS-Applikationen) im Grundsatz unverändert eingesetzt werden. Die Entwicklungsrichtungen lassen sich wie in Abbildung 1 dargestellt zusammenfassen.

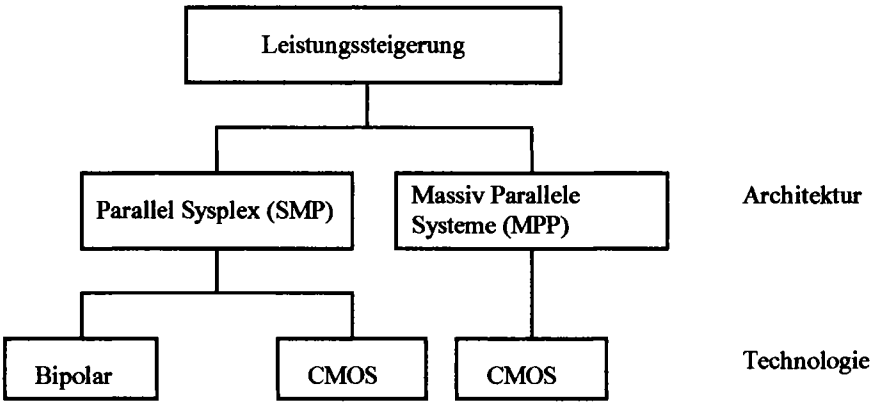


Abbildung 1: Entwicklung der Leistung von Rechnersystemen

Open Edition MVS

MVS unterstützt mehr und mehr heterogene Systemwelten, insbesondere die Industriestandards der UNIX-Welt und in seiner neuen Version sogar UNIX selbst. Diese Entwicklung wird auf eine Integration der Welten hinauslaufen. Wie sich die Entwicklung auf die zukünftige DV-Landschaft endgültig auswirkt, kann heute noch niemand mit Sicherheit voraussehen.

Zusammenfassend werden die heutigen Großsysteme mit MVS weiter - und zwar über einen noch nicht abzusehenden Zeitraum - ihre Rolle spielen. Das Wachstum in diesem Bereich wird allerdings moderater verlaufen, als es in der Vergangenheit der Fall war. Dabei werden die Ansprüche an die Verfügbarkeit und den Durchsatz der Systeme weiter zunehmen. Vor diesem Hintergrund hat eine zusammenfassende Darstellung der Performance-Aspekte eines MVS-Systems, wie sie mit der vorliegenden Arbeit beabsichtigt ist, ihre Berechtigung. Sämtliche Fragestellungen lassen sich auf andere Systemplattformen portieren, wenn auch die Modelle, die die neuen Architekturen zu berücksichtigen haben, im Detail anders ausfallen mögen.

Ich danke allen, die mich bei meiner Arbeit unterstützt haben, insbesondere meiner Frau, die viel Verständnis aufbringen mußte, sowie meinem Arbeitgeber, der Landesbank Rheinland-Pfalz, der mir den Zugriff auf die Systemdaten ermöglicht hat. Ich hoffe, daß ich dem Leser mit dieser Arbeit nützliche Hinweise geben kann.

1 Theoretische Grundlagen

1.1 Einleitung

Im vorliegenden Kapitel werden wir die wichtigsten Begriffe kennenlernen, die im Zusammenhang mit der Beurteilung der Leistung einer Datenverarbeitungsanlage stehen. Wir werden vorzugsweise - auch in späteren Kapiteln - mit Systemmodellen arbeiten. Diese dienen als Erklärungsmodelle der gedanklichen Durchdringung der Abläufe und dem Verständnis für die Begriffsbildung. Es ist keinesfalls beabsichtigt, ein komplettes Datenverarbeitungssystem zu modellieren, um daraus das Systemverhalten vorherzusagen bzw. zu berechnen. Modelle mit dieser Funktion findet man zum Beispiel in [6]. Später werden wir die in dem vorliegenden Kapitel 1 vorgestellten Modelle auf die Komponenten eines Datenverarbeitungssystems wie Prozessor, Ein-/Ausgabesystem und Netz anwenden und ergänzen.

1.2 Begriffsdefinitionen am Beispiel eines einfachen Systemmodells

Das zunächst benutzte Systemmodell ist äußerst einfach und abstrakt. Trotzdem sind wir damit in der Lage, die grundsätzlichen Probleme, die im Zusammenhang mit der Leistungsbeurteilung von Datenverarbeitungsanlagen stehen, zu erklären. Das System sehen wir dabei als einen Server, der aufgrund des Requests eines Terminalbenutzers (Eingabe) Information produziert (Verarbeitung) und diese dem Requestor zur Verfügung stellt (Ausgabe). Siehe Abbildung 1.2.1. Den genannten Prozeß von der Eingabe des Requests bis zur Ausgabe der Systemantwort am Endgerät (Terminal) nennen wir Transaktion. Wir versuchen zu klären, welche Leistungsanforderungen an ein solches System zu stellen sind. Dazu versetzen wir uns zunächst in die Lage des Benutzers. Für ihn ist die Leistung des Systems sicher auch verbunden mit der Qualität des Ergebnisses seiner Anfrage sowie mit der Verfügbarkeit des Systems. Diese Leistungskomponenten eines DV-Systems sind nicht Gegenstand

unserer Untersuchungen. Wir konzentrieren uns vielmehr auf die zeitgerechte Erledigung der Anfrage unseres Benutzers, nämlich auf die Antwortzeit des Systems. Diese bezeichnen wir mit t_r und definieren:

Response-Time

Unter der Response-Time t_r einer Transaktion versteht man die Zeitspanne zwischen dem Absenden des Requests an das System und dem Eintreffen des letzten Zeichens der Systemantwort am Terminal.

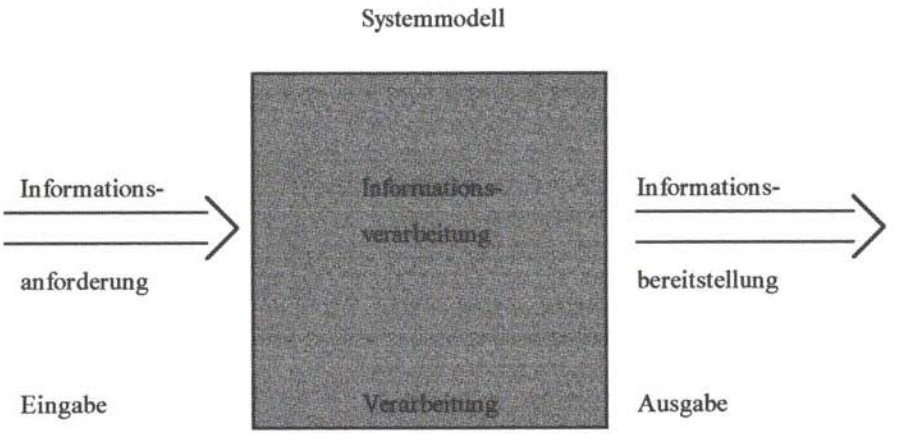


Abbildung 1.2.1: Einfaches Systemmodell

Die Response-Time t_r setzt sich zusammen aus der Bearbeitungszeit t_s, die das System für die Produktion der angeforderten Information benötigt - auch Service-Time - sowie aus der Wartezeit t_q - auch Queue-Time -, die im Laufe des Prozesses entsteht, wenn die Verarbeitung auf die Zuordnung von Ressourcen warten muß. Während unser Modellsystem nur über eine einzige Ressource (einen Server) verfügt, existieren in realen Systemen eine Vielzahl. Beispiele sind der oder die Zentralprozessoren und Magnetplatteneinheiten. Auf sogenannte Server-Netze, die diesen Sachverhalt besser berücksichtigen, werden wir in den folgenden Abschnitten noch eingehen. Im Augenblick genügt die Abbildung des Systems auf einen einzigen Server. Wir definieren nun die Größe Service-Time.

Service-Time

Unter der Service-Time t_s einer Transaktion versteht man die Zeitspanne, die das System (Server) für die Bearbeitung einer Anforderung benötigt.

Diese Zeitspanne entspricht der "Alleinelaufzeit" für die Bearbeitung der Anforderung, wobei Alleinelaufzeit umschreibt, daß sich der Request alleine im System befindet. Es wird also unterstellt, daß zu dieser Zeit keine anderen Anforderungen bearbeitet werden. Das System läuft im Einzelprogrammbetrieb, die Transaktion wird in ihrem Ablauf nicht durch andere Aktivitäten gestört. Es gilt dann $t_r = t_s$. Wir stellen nun eine Situation dar, bei der sich mehr als ein Request im System befindet. Im Modell bilden wir diese Situation dadurch ab, daß wir weitere Benutzer (=Terminals) - insgesamt vier - als Informationsanforderer zulassen. Die Ankunftszeit der Requests im System überlassen wir dem Zufall. Es gibt also keine Absprachen der Benutzer über den Eingabezeitpunkt der Anforderung (im Terminalbetrieb trivial). Die Situation könnte sich dann, wie in Abbildung 1.2.2 dargestellt, einstellen. Während einige der Requests das System ohne Wartezeiten mit einer Response-Time von $t_r = t_s$ verlassen, entstehen bei anderen Wartezeiten. Dabei wird unterstellt, daß die Requests in der Reihenfolge ihres Eintreffens im System bearbeitet werden. Wir definieren

Queue-Time

Unter der Queue-Time (Wartezeit) t_q einer Transaktion versteht man die Zeitspanne, während der sie auf Bearbeitung durch das System (Server) warten muß. Es gilt

$$t_r = t_q + t_s .$$

Die sich aus unserem Modell ergebenden Zeitabschnitte fassen wir pro User in Abbildung 1.2.3 zusammen. Dabei sind alle Zeitabschnitte in Einheiten von t_s angegeben. Danach liegt die mittlere Response-Time einer Transaktion bei $t_r = 1.7 \cdot t_s$, die mittlere Queue-Time bei $t_q = t_r - t_s = 0.7 \cdot t_s$.

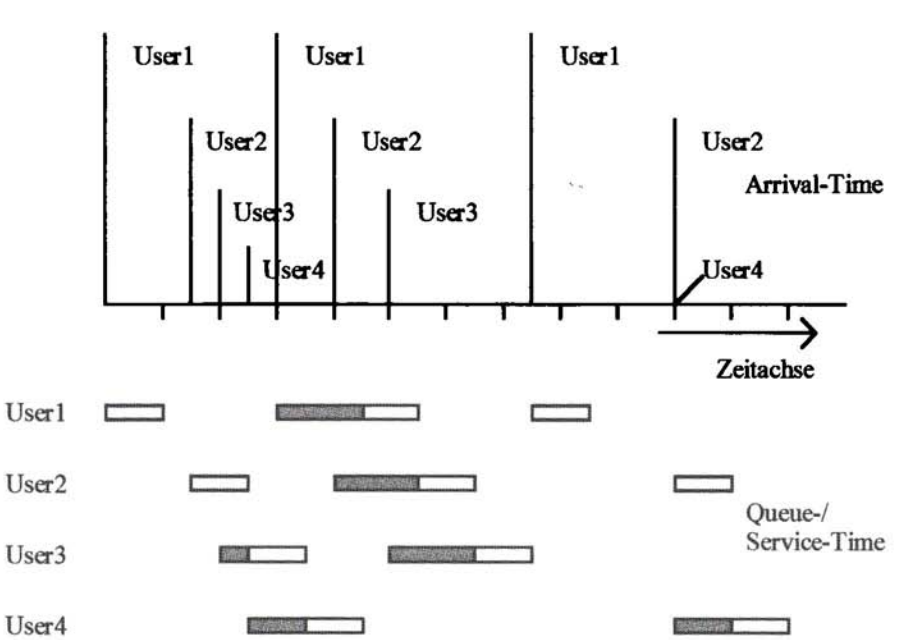


Abbildung 1.2.2: Einfaches Systemmodell

	t_r	t_q	t_s
User1	4.5	1.5	3.0
User2	4.5	1.5	3.0
User3	4.0	2.0	2.0
User4	4.0	2.0	2.0

Abbildung 1.2.3: Ergebnisse aus dem einfachen Systemmodell

Wir nehmen nun den Standpunkt des DV-Managers ein, der an einer möglichst hohen Auslastung der Datenverarbeitungsanlagen interessiert ist und definieren:

Transaction-Rate

Unter der Transaction-Rate r versteht man die Anzahl n der pro Zeiteinheit durchgesetzten Transaktionen. Werden n Transaktionen in einem Zeitintervall t durchgesetzt, so gilt $r = \frac{n}{t}$.

Kapazität

Unter der Kapazität c (Capacity) versteht man die maximal mögliche Transaction-Rate (auch Nennleistung).

In unserem Modellsystem wurden in dem betrachteten Zeitintervall von 12 t_s Zeiteinheiten 10 Transaktionen durchgesetzt. Die Transaction-Rate r liegt somit bei

$$r = \frac{n}{t} = \frac{10}{12 \cdot t_s} = 0.8\overline{3} \cdot t_s^{-1}.$$

Das entspricht beispielsweise ca. 8.3 Transaktionen pro Sekunde, wenn wir unterstellen, daß t_s in Zehntelsekunden dimensioniert ist. Maximal sind

$$c = \frac{1}{t_s} = t_s^{-1}$$

Transaktionen pro Sekunde durchsetzbar. Diese Feststellung gibt Gelegenheit, eine weitere performancerelevante Größe zu definieren, nämlich den Auslastungsgrad δ des Systems, genauer die Auslastung einer Ressource (= Server). Auf die Definition der Auslastung eines Systems, das aus mehreren Servern besteht, werden wir noch eingehen, wenn wir die entsprechenden Modelle behandelt haben. Im Augenblick begnügen wir uns mit dem Modell eines Ein-Server-Systems und verstehen unter "Server" also das Gesamtsystem.

Auslastungsgrad δ

Unter dem Auslastungsgrad δ eines Servers versteht man das Verhältnis zwischen Transaction-Rate r und Kapazität c. Es ist

$$\delta = \frac{r}{c} = \frac{r}{t_s^{-1}} = r \cdot t_s.$$

Unser Modellsystem hat einen Auslastungsgrad von

$$\delta = \frac{r}{c} = 0.8\overline{3} \cdot t_s^{-1} \cdot t_s = 0.8\overline{3}.$$

Prozentual ausgedrückt schreiben wir $\delta\% = 83\%$ und nennen $\delta\%$ die Auslastung des Systems bzw. des Servers. Es läßt sich leicht einsehen, daß eine hohe Auslastung des Systems nur mit einer zunehmenden Verschlechterung der Antwortzeiten zu erkaufen ist. Diesen Zusammenhang zwischen Transaktionsrate und Antwortzeit liefert das Gesetz von Little (siehe beispielsweise [7], [11] oder [45]). Dieses grundlegende Gesetz, das viele Phänomene im Leistungsverhalten von Datenverarbeitungsanlagen erklären hilft, werden wir im Folgenden formulieren. Wir werden später - bei der Behandlung von Warteschlangenmodellen - auf diese Gesetzmäßigkeit, die uns noch häufiger begegnen wird, zurückkommen. Vorher definieren wir noch:

Muliprogramming-Level

Unter dem Multiprogramming-Level m versteht man die mittlere Anzahl der im System verweilenden Transaktionen. Dabei kann sich eine Transaktion im Wait-State (die Transaktion wartet auf die Zuordnung einer Ressource) oder im Processsing-State (die Transaktion wird von einem Server bedient) befinden.

Wir formulieren nun das Gesetz von Little.

Gesetz von Little

Zwischen der Tranaction-Rate r, der Response-Time t_r und dem Multiprogramming-Level m gilt die Relation

$$m = r \cdot t_r .$$

In unserem Modellsystem ist

$$m = r \cdot t_r = 0.8\overline{3} \cdot 1.7 \approx 1.42.$$

Im Mittel sind also 1.42 Transaktionen "unterwegs". Diese warten auf Bedienung durch einen Server oder werden gerade von einem Server bedient.

Wir betrachten nun das Benutzer-Maschine-System als ein geschlossenes System. Den Begriff "geschlossenes System" werden wir später bei der Behandlung von Warteschlangenmodellen noch ge-

nauer definieren. Im Augenblick genügt die Vorstellung, die uns Abbildung 1.2.4 vermittelt: Eine Transaktion, die den Server verläßt, wird nach einer Zeit t_u (Benutzerdenkzeit, auch User-Think-Time) erneut dem Server zugeführt. Die Transaktionen verbleiben in dem geschlossenen Benutzer-Server-System.

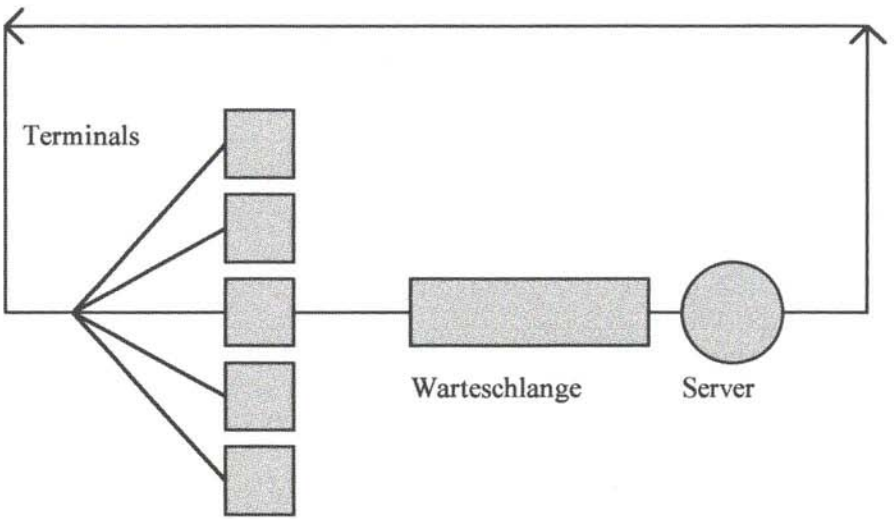


Abbildung 1.2.4: Geschlossenes Server-Netz

Sei nun k die Anzahl der Benutzer (= Terminals), dann gilt die Relation (Anwendung des Littleschen Gesetzes auf $t_r + t_u$)

$$k = r \cdot (t_r + t_u) \text{ oder } t_r = \frac{k}{r} - t_u \ .$$

Diese Relation wird uns noch einige Male beschäftigen. Sie dient uns beispielsweise im nächsten Kapitel zur Simulation von Lastzunahmen. Dabei sind zwei Ansätze möglich:

- Erhöhung der Anzahl Benutzer (=Terminalanzahl) k, aber Beibehaltung des Benutzerverhaltens (t_u bleibt unverändert) und dadurch eine indirekte Erhöhung der Transaktionslast r oder
- Veränderung des Benutzerverhaltens durch Reduzierung der User-Think-Time (t_u wird kleiner) bei unveränderter Benutzer-

anzahl (=Terminalanzahl) und dadurch indirekte Erhöhung der Transaktionslast.

Wir definieren weitere performancerelevante Größen und leiten Relationen her.

Queue-Length

Unter der Queue-Length n_q versteht man die mittlere Anzahl der auf Bedienung wartenden Transaktionen.

Die Anwendung des Littleschen Gesetzes ergibt

$$n_q = \frac{n \cdot t_q}{t} = r \cdot t_q .$$

Analog ist

$$n_s = r \cdot t_s = \delta$$

die mittlere Anzahl der Requests, die gerade durch einen Server bedient werden. Man nennt m auch die mittlere Anzahl aktiver Requests. Wir schreiben deshalb in Analogie zu n_q auch n_a. Es gilt

$$n_a = n_q + n_s = n_q + \delta .$$

Bezeichnen wir noch mit n_I die mittlere Anzahl der auf Benutzereingabe wartenden Requests (idle Transactions), so ist schließlich

$$k = n_I + n_a = n_I + n_q + n_s = n_I + n_q + \delta .$$

Das Verhältnis zwischen mittlerer Response-Time und mittlerer Service-Time einer Transaktion gibt Auskunft darüber, wie sich die Laufzeit einer Transaktion aufgrund eines Multi-User-Betriebs gegenüber der Alleinelaufzeit verändert.

Elapsed-Time-Multiplication-Factor

Unter dem Elapsed-Time-Multiplication-Factor e versteht man das Verhältnis zwischen der Response-Time und der Service-Time einer Transaktion. Es gilt

$$e = \frac{t_r}{t_s} = \frac{t_s + t_q}{t_s} = 1 + \frac{t_q}{t_s}.$$

Im Modellsystem ist $e = \frac{t_r}{t_s} = 1.7$.

Zwischen dem Multiprogramming-Level m und e gilt die Relation

$$m = r \cdot t_r = r \cdot e \cdot t_s = e \cdot \delta.$$

Den Eintreffprozeß der Transaktionen im System beschreibt man mit der sogenannten Zwischenankunftszeit oder Arrival-Time t_a.

Arrival-Time

Unter der Arrival-Time t_a versteht man die mittlere Zeitspanne zwischen dem Eintreffen der Requests im System. Der Kehrwert von t_a heißt Eintreffrate oder Arrival-Rate.

Wenn man voraussetzt, daß $t_a > t_s$ ist, folgt

$$\frac{1}{t_a} < \frac{1}{t_s}.$$

Da der "Ausstoß" (Bedienrate, Durchsatz, Transaction-Rate) nicht größer sein kann als die Anforderungsrate (Arrival-Rate), stimmen r und Arrival-Rate zahlenmäßig überein. Wir arbeiten ausschließlich mit Modellen, die die obige Voraussetzung erfüllen und machen deshalb keinen Unterschied zwischen den beiden Größen.

Wir fassen nun alle bisher definierten und abgeleiteten Größen sowie die wichtigsten Relationen tabellarisch in der Abbildung 1.2.5 zusammen.

Response-Time t_r: Zeitspanne zwischen dem Absenden eines Requests und dem Eintreffen des letzten Zeichens der Systemantwort am Terminal.
Queue-Time t_q: Zeit, während der eine Transaktion auf Bedienung wartet.
Service-Time t_s: Zeit, während der eine Transaktion von einem Server bedient wird.
Response-Time als Summe aus Queue- und Service-Time: $t_r = t_q + t_s$.
Transaction-Rate r: Anzahl der pro Zeiteinheit durchgesetzten Transaktionen.
Multiprogramming-Level n_a: Anzahl aktiver Requests.
Littles Gesetz für aktive Requests: $n_a = r \cdot t_r$.
Queue-Length n_q: Anzahl der auf Bedienung wartenden Requests.
Littles Gesetz für wartende Requests: $n_q = r \cdot t_q$.
Kapazität c: Maximal durchsetzbare Transaction-Rate.
Auslastungsgrad δ: Verhältnis zwischen Transaction-Rate r und Kapazität c.
Anzahl aktiver Requests als Summe aus wartenden und in Service befindlichen Requests: $n_a = n_q + n_s$.
Arrival-Time t_a: Zeitintervall zwischen dem Eintreffen der Requests im System.
Arrival-Rate $r = t_a^{-1}$: Kehrwert der Arrival-Time.
User-Think-Time t_u: Zeitspanne zwischen dem Ende und der Eingabe einer neuen Transaktion durch den Benutzer.
Anzahl User k: Anzahl der am Transaktionsprozeß beteiligten User (= Anzahl Terminals).
Response-Time in Abhängigkeit von r, k **und** t_u: $t_r = \frac{k}{r} - t_u$.
Elapsed-Time-Multiplication-Factor e: Verhältnis zwischen Response-Time t_r und Service-Time t_s.

Abbildung 1.2.5: Definierte und abgeleitete Größen

1.3 Analytische Systemmodelle

Grundsätzlich sehen wir die noch darzustellenden Rechnermodelle bzw. Modelle einzelner Systemkomponenten weniger unter dem Aspekt, das Verhalten eines realen Systems oder einer Systemkomponente berechenbar und vorhersagbar zu machen, als unter dem Aspekt, mit Hilfe dieser Modelle Zusammenhänge zu erkennen und zu durchdringen, um auf dieser Grundlage Systemverhalten und Systemleistung qualitativ beurteilen zu können. Wir werden also nicht den Versuch unternehmen, ein reales System mit all seinen Teilsystemen und Systemkomponenten in einem Modell abzubilden. Dazu gibt es komplexe Modellansätze (siehe zum Beispiel [7]) und umfangreiche Softwaresysteme (siehe [6]). Die analytischen Modelle lassen sich prinzipiell in zwei Kategorien einteilen, in deterministische und in stochastische Modelle (siehe[7]). Die Modelle der zweiten Kategorie werden auch häufig als Warteschlangenmodelle bezeichnet. Mit der ersten Modellkategorie werden wir uns nur beispielhaft befassen.

1.3.1 Deterministische Systemmodelle

Deterministische Modelle beschreiben in der Regel Einzelprogrammsysteme. Sie eigenen sich weniger zur Beschreibung und Analyse eines Multi-User-Betriebs. Sie arbeiten gewöhnlich mit einer kleinen Anzahl deterministischer Modellparameter und bestimmen nur einige wenige Leistungsindizes wie zum Beispiel die Job-Laufzeit eines Batch-Jobs. Ein Beispiel eines deterministischen Modells ist neben dem, das wir in diesem Abschnitt vorstellen, das "Blocksize-Modell", das wir im Zusammenhang mit Ein-/Ausgabe-Operationen im Kapitel 4 besprechen. Da deterministische Modelle nicht mit Zufallsvariablen arbeiten, können sie auch nicht das Auftreten von Warteschlangen vor den Servern berücksichtigen. Wie wir bereits bei unserem einfachen Systemmodell im Abschnitt 1.2 feststellen konnten, ist aber gerade dieser Aspekt äußerst wichtig für das Leistungsverhalten eines Systems. Unser Eingangsmodell verfügt über Elemente beider Modellkategorien. Während wir die Service-Time t_s als konstant angenommen hatten, wurden die Arrival-Time t_a und die User-Think-Time t_u als zufallsverteilt unterstellt. Eine analytische Formulierung der Verteilung dieser Größen werden wir

im nächsten Abschnitt vornehmen. Zunächst werden wir ein Modell der ersten Kategorie vorstellen. Dieses Modell ist angelehnt an das in [11] beschriebene "Model of CPU-I/O-Overlap". In ihm geht es um die Darstellung der Effekte der überlappenden Verarbeitung von Prozessor und Ein-/Ausgabesystem. Wir besprechen zunächst die Parameter des Modells.

Systemparameter

Das Modellsystem besteht aus einem Prozessor, einem oder mehreren Magnetplattenspeichern (siehe Fallunterscheidung) und einem oder zwei - je nach Fallunterscheidung - Ein-/Ausgabepuffern.

Workloadparameter

Es werden Datenblöcke der Länge blksize, bestehend aus bf logischen Records, von einem Magnetplattenspeicher in einen Pufferbereich gelesen. Zur Aufnahme der Datenblöcke stehen ein oder zwei Puffer der Länge blksize (je nach Fallunterscheidung) zur Verfügung. Die Daten werden vom Prozessor verarbeitet und anschließend auf einen Magnetplattenspeicher zurückgeschrieben, je nach Fallunterscheidung auf denselben oder auf einen zweiten.

Leistungsparameter

Für die Bearbeitung eines Datenblocks werden tcpublk, für einen Record tcpurec Zeiteinheiten Prozessorzeit benötigt. Der Ein- sowie der Ausgabevorgang dauert tioblk Zeiteinheiten.

Leistungsindizes

Das Modell dient der Demonstration der Effekte überlappender Verarbeitung von Prozessor und Ein-/Ausgabe-Subsystem. Als Leistungsindizes wählen wir die Verarbeitungszeit t_s pro Record sowie den Kehrwert von t_s, die Anzahl der pro Zeiteinheit verarbeiteten Records r (= Durchsatz).

Wir führen Fallunterscheidungen durch und wählen unterschiedliche Überlappungsschemata. Innerhalb der Überlappungsschemata unterscheiden wir nach I/O- bzw. processorbounded Verarbeitung. Dabei heißt die Verarbeitung I/O-bounded, wenn die Bearbeitung eines Datenblocks weniger Zeit beansprucht als der Transfer des Datenblocks vom Speichermedium in den Puffer und vom Puffer auf das

Speichermedium. Dauert die Bearbeitung eines Datenblocks durch den Prozessor länger als der Transfer, so sprechen wir von einer processorbounded Verarbeitung.

Serielle Verarbeitung

Prozessor und Ein-/Ausgabesystem arbeiten nicht überlappt. Für die Aufnahme der Datenblöcke steht ein Datenpuffer zur Verfügung (siehe Abbildung 1.3.1).

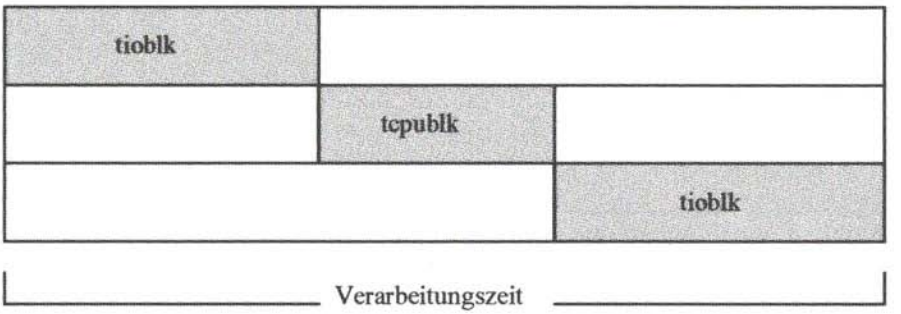


Abbildung 1.3.1: Serielle Verarbeitung

Für einen Verarbeitungszyklus (Verarbeitung eines Records) werden danach

$$t_s = \frac{\text{tioblk} + \text{tcpublk} + \text{tioblk}}{\text{bf}} = \frac{2 \cdot \text{tioblk} + \text{tcpublk}}{\text{bf}}$$

$$= \frac{2 \cdot \text{tioblk}}{\text{bf}} + \text{tcpurec}$$ Zeiteinheiten benötigt. Der Durchsatz r liegt bei

$$r = \frac{\text{bf}}{2 \cdot \text{tioblk} + \text{bf} \cdot \text{tcpurec}}.$$

Eine Unterscheidung zwischen processorbounded bzw. I/O-bounded erübrigt sich bei diesem Überlappungsschema. Wir kommen zum nächsten Verarbeitungstypus.

Zwei-Wege-parallele Verarbeitung

Prozessor und Ein-/Ausgabesystem arbeiten in diesem Falle überlappt. Die Ein-/Ausgabeoperationen erfolgen nach wie vor seriell. Es steht ein Ein-/Ausgabepuffer zur Verfügung. Wir treffen nun eine Fallunterscheidung.

a) I/O-bounded

Es gilt (tcpublk ≤ 2 tioblbk) und damit (siehe Abbildung 1.3.2)

$$t_s = \frac{2 \cdot \text{tioblk}}{\text{bf}} \text{ und } r = \frac{\text{bf}}{2 \cdot \text{tioblk}}.$$

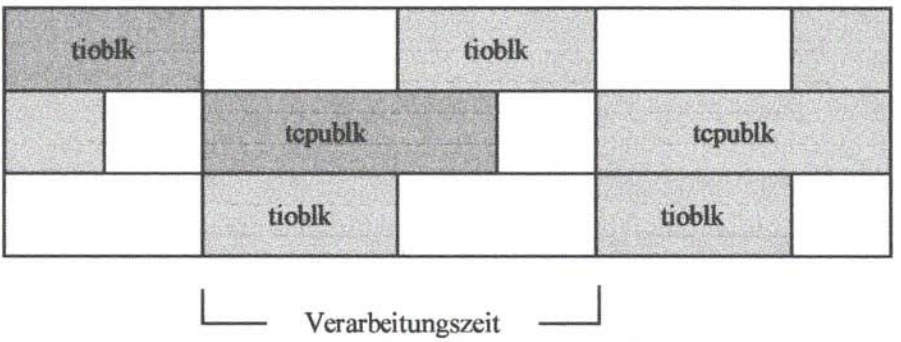


Abbildung 1.3.2: Zwei-Wege-parallel I/O-bounded

b) Processorbounded

Es gilt tcpublk ≥ 2 tioblbk und damit (siehe Abbildung 1.3.3)

$$t_s = \frac{\text{tcpublk}}{\text{bf}} = \frac{\text{bf} \cdot \text{tcpurec}}{\text{bf}} = \text{tcpurec} \text{ und } r = \frac{1}{\text{tcpurec}}.$$

Drei-Wege-parallele Verarbeitung

Bei diesem Verarbeitungsschema stehen zwei Puffer für die Aufnahme der Datenblöcke zur Verfügung. Die Konfiguration verfügt über zwei Magnetplatten.

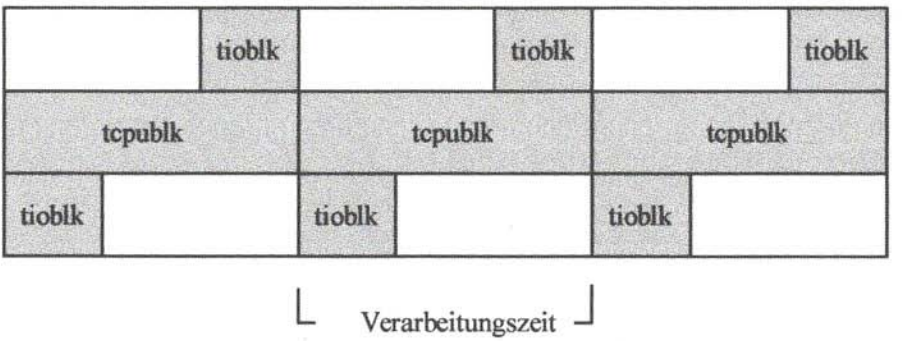


Abbildung 1.3.3: Zwei-Wege-parallel processorbounded

a) I/O-bounded

Es gilt tcpublk ≤ tioblk und damit (siehe Abbildung 1.3.4)

$$t_s = \frac{\text{tioblk}}{\text{bf}} \text{ und } r = \frac{\text{bf}}{\text{tioblk}}.$$

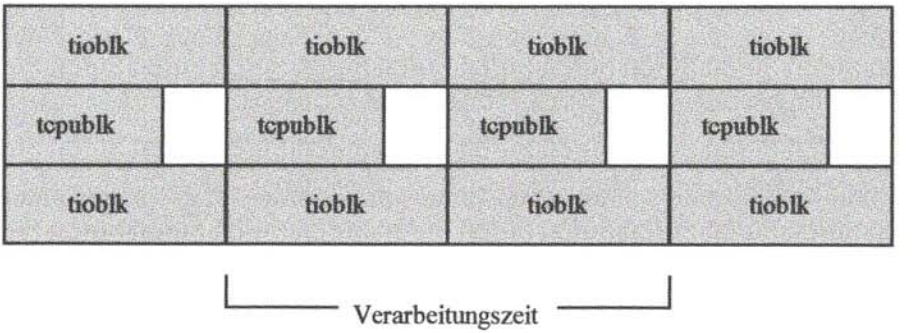


Abbildung 1.3.4: Drei-Wege-parallel I/O-bounded

b) Processorbounded

Es gilt tioblk ≤ tcpublk und damit (siehe Abbildung 1.3.5)

$$t_s = \frac{\text{tcpublk}}{\text{bf}} = \frac{\text{bf} \cdot \text{tcpurec}}{\text{bf}} = \text{tcpurec} \text{ und } r = \frac{1}{\text{tcpurec}}.$$

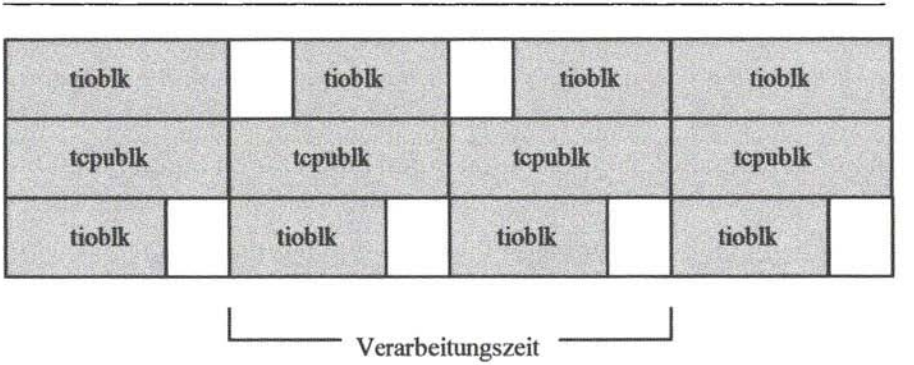


Abbildung 1.3.5: Drei-Wege-parallel processorbounded

Zusammenfassend stellen wir die Ergebnisse in Abbildung 1.3.6 tabellarisch dar.

Verfahren	**Verarbeitungszeit/ Record**	**Durchsatz**
seriell	$\frac{2 \cdot \text{tioblk}}{\text{bf}} + \text{tcpurec}$	$\frac{\text{bf}}{2 \cdot \text{tioblk} + \text{bf} \cdot \text{tcpurec}}$
zwei-Wege a)	$\frac{2 \cdot \text{tioblk}}{\text{bf}}$	$\frac{\text{bf}}{2 \cdot \text{tioblk}}$
zwei-Wege b)	tcpurec	$\frac{1}{\text{tcpurec}}$
drei-Wege a)	$\frac{\text{tioblk}}{\text{bf}}$	$\frac{\text{bf}}{\text{tioblk}}$
drei-Wege b)	tcpurec	$\frac{1}{\text{tcpurec}}$

Abbildung 1.3.6: Zusammenfassung der Modellergebnisse

Die Ergebnisse waren nicht schwer vorherzusehen. Je höher der Überlappungsgrad der Verarbeitung ist, um so leistungsfähiger mußte unser System im Sinne der definierten Leistungsindizes sein. Allerdings wird beim Übergang vom zweiten zum dritten Überlappungsschema genau dann keine Leistungssteigerung erreicht, wenn

der Verarbeitungsprozeß processorbounded ist. Soll in diesem Falle die Leistung gesteigert werden, muß die Leistungsfähigkeit des Prozessors erhöht werden. Diese eigentlich triviale Feststellung zeigt, daß die Leistung eines Rechnersystems nicht nur abhängig ist von den zur Verfügung gestellten Hard- und Softwareressourcen, sondern, wie wir noch bei mehreren Gelegenheiten sehen werden, auch von der Zusammensetzung der Arbeitslast (des Workloads). Auf diesen Aspekt der Leistung eines Rechnersystems werden wir in den nachstehenden Kapiteln noch häufiger zurückkommen. Er ist einer der wesentlichen Gründe für das Fehlen einer objektiven Leistungsdefinition eines Rechnersystems.

1.3.2 Warteschlangenmodelle

Da die Anforderungen an ein Rechnersystem im allgemeinen in nicht vorherbestimmbarer Weise (nicht deterministisch) variieren, nennt man Modelle, die zufallsbedingte Einflüsse auf das System berücksichtigen, Wahrscheinlichkeits- oder stochastische Modelle oder auch, die Auswirkung zufallsbedingter Anforderungen in Betracht ziehend, Warteschlangenmodelle. Dieser Zusammenhang wird im Laufe der nun folgenden Diskussion noch deutlich. Zunächst geht es darum, die Verteilung bestimmter Zufallsgrößen zu berücksichtigen. Dazu gehören die Verteilung der Arrival-Time t_a und die Verteilung der Service-Time t_s.

Single-Node-Wartesysteme

Ein Single-Node-Wartesystem ist definiert durch (siehe zum Beispiel [7], [11]oder [45])

- eine Quelle (Source), die die Anforderungen (Requests) generiert,
- eine Warteschlange (Queue), in die die Anforderungen, die nicht unmittelbar nach Ankunft im System bearbeitet werden können, eingereiht werden,
- eine oder mehrere Instanzen (Server), in denen die Anforderungen bearbeitet werden.

Ein Single-Node-Wartesystem mit beispielsweise zwei Servern hat damit den in Abbildung 1.3.7 skizzierten Aufbau.

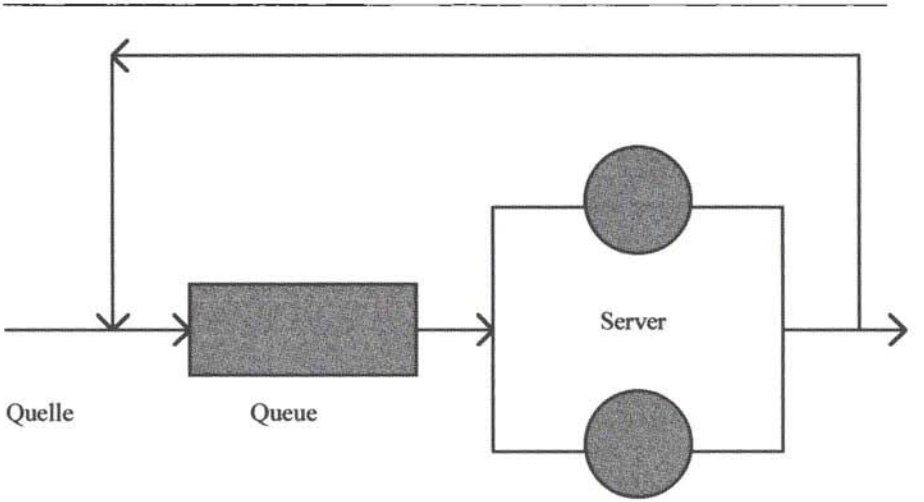


Abbildung 1.3.7: Beispiel eines Single-Node-Wartesystems

Nachdem ein Request vom Server bearbeitet worden ist, verläßt er das System oder wird erneut in die Warteschlange eingereiht (Preemption). Um Aussagen über das Verhalten eines solchen Wartesystems machen zu können, müssen zwei Eigenschaften über die Aufträge bekannt sein:

- die Wahrscheinlichkeitsverteilung für das Eintreffen der Requests ("Arrival- Process") und
- die Wahrscheinlichkeitsverteilung für die Bearbeitungszeit der Requests ("Service-Process").

Wenn noch der Scheduling-Algorithmus, das heißt, der Algorithmus, nach dem der nächste zur Bearbeitung anstehende Request aus der Warteschlange selektiert und der Bearbeitung zugeführt wird, bekannt ist, schreibt man zur Charakterisierung des Wartesystems "A/B/m-Scheduling-Algorithmus". Dabei steht A für die Wahrscheinlichkeitsverteilung des Ankunftsprozesses, B für die Wahrscheinlichkeitsverteilung des Bedienprozesses und m für die Anzahl der Instanzen für die Bedienung der Requests (Server). Die gebräuchlichsten Verteilungen A und B und deren Eigenschaften sind zum Beispiel in [7] beschrieben. Wir beschränken uns auf die Behandlung von Exponentialverteilungen sowohl für den Ankunfts- als auch für den Bedienprozeß. Während der Ankunftsprozeß in der Regel gut durch eine Exponentialverteilung modelliert werden kann, sind die

Bedienprozesse in realen Systemen eher nicht exponentialverteilt (siehe beispielsweise [11]). Wir geben für die Fälle einer konstanten Bedienzeit sowie einer allgemeinen Verteilung der Bedienzeit lediglich die aus dem Modell resultierenden Leistungsindizes an (siehe in diesem Abschnitt weiter unten). Die im übrigen auferlegte Beschränkung auf Exponentialverteilungen bedeutet für die Erklärung des Verhaltens von Rechnersystemen keinen Nachteil. Anders verhält es sich, wenn man das Verhalten realer Systeme mittels eines möglichst vollständigen Modellsystems vorhersagen und berechenbar machen will. Wir befassen uns nun zunächst mit den Eigenschaften der Exponentialverteilung beim Ankunftsprozeß und definieren, wann der Ankunftsprozeß exponential verteilt heißt.

Definition

Die Arrival-Time t_a heißt exponentialverteilt, wenn

$$p(t_a \le t) = 1 - e^{-\frac{t}{\bar{t}_a}}$$

gilt für die Wahrscheinlichkeit, daß der nächste Auftrag innerhalb der nächsten t Zeiteinheiten im System eintrifft. Dabei ist $\bar{t}_a$ der Mittelwert der Arrival-Time.

Wir fassen die Eigenschaften dieser Verteilungsfunktion zusammen (siehe zum Beispiel [45]):

- Der Ankunftsprozeß ist unabhängig vom Ursprung der Zeitmessung (der Prozeß hat kein "Gedächtnis").
- Die Aufträge sind unabhängig voneinander.
- Der Ankunftsprozeß ist vollständig bestimmt durch den Mittelwert der Arrival-Time $\bar{t}_a$.

Üblicherweise verwendet man für eine Verteilungsfunktion F der Zufallsvariablen X mit Werten $x \in W$ die Notation

$$F_X(x) = p(X \le x) = \int_{-\infty}^{x} f_X(x)dx.$$

Dabei heißt $f_X(x)$ die Dichte der Verteilung. Im Falle der Exponentialverteilung gilt

$$F_X(x) = \int_0^x a \cdot e^{-a \cdot x} dx = e^{-a \cdot x} \Big|_0^x = 1 - e^{-a \cdot x}.$$

Die Konstante a entspricht dem Mittelwert der Zufallsgröße X, so daß für den Arrival-Process

$$F_{t_a}(t) = \int_0^t \frac{1}{\bar{t}_a} \cdot e^{-\frac{1}{\bar{t}_a} \cdot t} dt = 1 - e^{-\frac{1}{\bar{t}_a} \cdot t}$$

gilt. Man setzt

$$\bar{r} = \frac{1}{\bar{t}_a}$$

und bezeichnet $\bar{r}$ als Arrival-Rate - auch Ankunftsrate oder Request-Rate. Für den Erwartungswert (= Mittelwert) E, die Varianz V und den Variationskoeffizienten C von t_a erhält man (siehe beispielsweise [7] und [45])

$$E(t_a) = \int_0^{+\infty} t \cdot f_{t_a}(t) dt = \int_0^{+\infty} t \cdot \bar{r} \cdot e^{-\bar{r} \cdot t} dt = \bar{r} \cdot \int_0^{\infty} t \cdot e^{-\bar{r} \cdot t} dt = \frac{1}{\bar{r}} = \bar{t}_a,$$

$$V(t_a) = E(t_a^2) - E(t_a)^2 = 2 \cdot E(t_a)^2 - E(t_a)^2 = E(t_a)^2 = \bar{t}_a^2 \quad \text{und}$$

$$C(t_a) = \frac{\sqrt{V(t_a)}}{E(t_a)} = \frac{\sqrt{E(t_a)^2}}{E(t_a)} = 1.$$

Exponentialverteilungen heißen auch Markow-Verteilungen. Man kürzt ab mit M. Für den Bedienprozeß erhält man in Analogie

$$F_{ts}(t) = \int_0^t \frac{1}{\bar{t}_s} \cdot e^{-\frac{1}{\bar{t}_s} \cdot t} dt = 1 - e^{-\frac{1}{\bar{t}_s} \cdot t}$$

Man setzt in diesem Falle

$$\bar{c} = \frac{1}{\bar{t}_s}$$

und nennt $\bar{c}$ die Kapazität - auch Service-Rate oder Bedienrate des Systems in bezug auf den durch $\bar{t}_s$ festgelegten Transaktionstyp. Auch hier gilt unter der Prämisse $\bar{t}_a > \bar{t}_s$, daß Arrival-Rate und Transaction-Rate (= Durchsatz) identisch sind, so daß wir keine unterschiedlichen Bezeichnungen für diese beiden Größen einführen müssen, solange wir ausschließlich mit der genannten Prämisse arbeiten. Für den Mittelwert, die Varianz und den Variationskoeffizienten gelten in Analogie zum Ankunftsprozeß

$$E(t_s) = \bar{t}_s,\ V(t_s) = \bar{t}_s^2 \text{ und } C(t_s) = 1.$$

Um unser Modellsystem zu vervollständigen, beschreiben wir noch einige der gebräuchlichsten Scheduling-Algorithmen (siehe zum Beispiel [7], [11] und [45]).

First-Come-First-Serve (FCFS)

Die Aufträge werden in der Reihenfolge ihres Eingangs im System bedient. Wir werden uns bis auf wenige Beispiele auf diesen Algorithmus beschränken. Falls im Folgenden kein Algorithmus explizit genannt ist, gilt deshalb FCFS.

Last-Come-First-Serve (LCFS)

Der zuletzt eingetroffene Auftrag wird zuerst bedient.

Round-Robin (RR)

Ist ein Auftrag in einer fest vorgegebenen Zeitscheibe noch nicht erledigt, so wird er aus der Verarbeitung verdrängt und erneut in die Warteschlange eingereiht, die dann nach FCFS abgearbeitet wird. Dieser Vorgang wird so oft wiederholt, bis der Auftrag erledigt ist.

Processor-Sharing (PS)

Dieser Algorithmus entspricht RR mit infinitesimal kleiner Zeitscheibe. Dadurch entsteht der Eindruck, daß alle Aufträge gleichzeitig mit entsprechend längerer Bedienzeit bearbeitet werden.

Externe Priorität

Die Aufträge sind mit einer externen Priorität versehen und werden in dieser Reihenfolge in die Warteschlange eingereiht und bearbeitet.

Wir können nun unser Standard Single-Node-Wartesystem wie folgt beschreiben.

M/M/m-FCFS bedeutet:

- der Ankunftsprozeß ist exponentialverteilt (markowverteilt),
- der Bedienprozeß ist exponentialverteilt (markowverteilt),
- das Wartesystem verfügt über m Server,
- der Scheduling-Algorithmus ist FCFS.

Wir werden neben den Leistungsindizes für dieses Standardmodell auch Leistungsindizes angeben für die Modelle M/D/1-FCFS und M/G/1-FCFS. Dabei steht D für deterministisch und bedeutet konstante Service-Time und G für general, das heißt für eine allgemeine Verteilung der Bedienzeit, deren Varianz V und Korrelationskoeffizient C bekannt sind.

Nachdem wir das Single-Node-Wartesystem durch die Modellparameter "A/B/m-Scheduling-Algorithmus" definiert haben, beschäftigen wir uns mit den aus dem Modell ableitbaren Leistungsgrößen. Wir verzichten auf eine Herleitung der Relationen und geben nur die Ergebnisse bekannt. Zur Herleitung siehe [7], [11] oder [45]). Vorausgesetzt wird in allen Fällen statistisches Gleichgewicht, d.h. die Anzahl der ankommenden und abgehenden Aufträge ist im Mittel identisch ($\bar{r} = \frac{1}{\bar{t}_a}$). Unter dieser Voraussetzung können alle Leistungsindizes aus der Zustandswahrscheinlichkeit $p(k)$ abgeleitet werden. Dabei bedeutet $p(k)$, daß sich genau k Aufträge im System befinden: $p(k) = p$ ("es befinden sich k Aufträge im System").

Leistungsindizes

a) Auslastungsgrad δ *des Servers*

Es gilt

$$\delta = \frac{\bar{t}_s}{\bar{t}_a} = \frac{\bar{r}}{\bar{c}} = 1 - p\ (0) = 1 - p\ (\text{"Wartesystem ist inaktiv"}) \text{ für } m=1$$

und

$$\delta = \frac{\bar{r}}{m \cdot \bar{c}} = 1 - \sum_{k=0}^{m-1} \frac{m-k}{m} \cdot p(k) \text{ für } m>1.$$

Dies entspricht der Anzahl der im Mittel aktiven Server. Im Gleichgewichtszustand muß $\delta < 1$ gelten, das heißt, im Mittel dürfen nicht mehr Aufträge eintreffen, als bedient werden können. Das ist die bereits bekannte Prämisse $\bar{t}_a > \bar{t}_s$.

b) Transaction-Rate $\bar{r}$ *(=Durchsatz)*

Im Gleichgewichtszustand ist die Transaction-Rate gleich der Arrival-Rate, also

$$\bar{r} = \frac{1}{\bar{t}_a}.$$

c) Response-Time (Verweilzeit) $\bar{t}_r$

Die Response-Time eines Auftrags entspricht der Aufenthaltsdauer des Auftrags im System. Bei nicht online orientierten Aufträgen (Batch-Jobs) spricht man in der Regel von der Verweilzeit (auch Execution-Time oder Elapsed Time). Da wir fast ausschließlich terminalgetriebene Aufträge behandeln, verwenden wir vorzugsweise den Terminus Response-Time. Die Response-Time setzt sich zusammen - wie im letzten Abschnitt bereits dargelegt - aus der Queue-Time $\bar{t}_q$ und der Service-Time $\bar{t}_s$

$$\bar{t}_r = \bar{t}_q + \bar{t}_s.$$

Da wir im Folgenden ausschließlich Mittelwerte betrachten, verzichten wir auf die Indikation als Mittelwert und schreiben statt

$\bar{t}_r$, $\bar{t}_q$ und $\bar{t}_s$ t_r , t_q und t_s. Sei nun $\bar{n}_a$ die mittlere Anzahl der Aufträge im System, die entweder auf Bedienung warten oder gerade bedient werden (aktive Requests), $\bar{n}_q$ die mittlere Anzahl der auf Bedienung wartenden Requests und $\bar{r}$ die Transaction-Rate, so gilt das Gesetz von Little

$$\bar{n}_a = \bar{r} \cdot \bar{t}_r \quad \text{bzw.} \quad \bar{n}_q = \bar{r} \cdot \bar{t}_q$$

für die aktiven bzw. für die wartenden Requests. Wir verzichten auch bei den Größen $\bar{r}$, $\bar{n}_a$ und $\bar{n}_q$ auf die Indikation als Mittelwert und schreiben

$$n_a = r \cdot t_r \ , \ n_q = r \cdot t_q \quad \text{und damit} \quad t_r = \frac{n_a}{r} \quad \text{und} \quad t_q = \frac{n_q}{r}.$$

Das Gesetz von Little gilt für alle Wahrscheinlichkeitsverteilungen und ist eines der grundlegenden Gesetze der Warteschlangentheorie. Aus dem Littleschen Gesetz und der Relation $r = \delta \cdot m \cdot t_s^{-1}$ folgt

$$n_a = r \cdot t_r = r \cdot \left(t_q + t_s\right) = n_q + \delta \cdot m .$$

Wir fassen zusammen. Für ein Single-Node-Wartesystem "A/B/m-Scheduling-Algorithmus" gelten die in Abbildung 1.3.8 zusammengestellten Leistungsindizes und Relationen. Diese gelten für alle Wahrscheinlichkeitsverteilungen A und B sowie für alle genannten Scheduling-Algorithmen. Für die Wartesysteme M/M/1-FCFS, M/D/1-FCFS und M/G/1-FCFS gelten außerdem die folgenden Relationen (siehe beispielsweise [7], [11] oder [46]).

M/M/1-FCFS

Es gilt

$$n_a = \frac{\delta}{1-\delta}.$$

Daraus folgt mit dem Gesetz von Little

$$t_r = \frac{n_a}{r} = \frac{\delta}{r \cdot (1-\delta)} = \frac{t_s}{1-\delta}.$$

Wegen $n_q = n_a - \delta$ folgt für die Queue-Length n_q bzw. die Queue-Time t_q

$$n_q = n_a - \delta = \frac{\delta}{1-\delta} - \delta = \frac{\delta^2}{1-\delta} \quad \text{bzw.}$$

$$t_q = t_r - t_s = \frac{t_s}{1-\delta} - t_s = t_s \cdot \frac{\delta}{1-\delta}.$$

Leistungsindex	**Relation**
Auslastungsgrad eines Servers	$\delta = \frac{r \cdot t_s}{m}$ (m=1) $\delta = 1 - \sum_{k=0}^{m-1} \frac{m-k}{m} \cdot p(k)$ (m>1)
Transaction-Rate	$r = \delta \cdot m \cdot t_s^{-1}$
Response-Time	$t_r = t_q + t_s = \frac{n_a}{r}$
Queue-Time	$t_q = \frac{n_q}{r}$
Multiprogramming-Level	$n_a = n_q + m \cdot \delta$
Queue-Length	$n_q = n_a - m \cdot \delta$

Abbildung 1.3.8: Leistungsindizes für Wartesysteme

M/D/1-FCFS

Es gilt

$$n_a = \delta + \frac{\delta^2}{2 \cdot (1-\delta)}.$$

Daraus folgt

$$n_q = n_a - \delta = \frac{\delta^2}{2 \cdot (1-\delta)}$$

für die mittlere Queue-Length,

$$t_r = \frac{n_a}{r} = \frac{t_s}{\delta} \cdot \left(\delta + \frac{\delta^2}{2 \cdot (1-\delta)} \right) = t_s \cdot \frac{2-\delta}{2 \cdot (1-\delta)} \text{ und}$$

$$t_q = t_r - t_s = t_s \cdot \frac{\delta}{2 \cdot (1-\delta)}$$

für die Response- bzw. Queue-Time.

M/G/1-FCFS

Es gilt

$$n_a = \delta + \frac{\delta^2}{2 \cdot (1-\delta)} \cdot \left[1 + \left(\frac{\sqrt{V(t_s)}}{E(t_s)} \right)^2 \right].$$

Daraus folgt

$$n_q = n_a - \delta = \frac{\delta^2}{2 \cdot (1-\delta)} \cdot \left[1 + \left(\frac{\sqrt{V(t_s)}}{E(t_s)} \right)^2 \right] \text{ und damit}$$

$$t_q = \frac{n_q}{r} = \frac{t_s}{\delta} \cdot n_q = t_s \cdot \frac{\delta}{2 \cdot (1-\delta)} \cdot \left[1 + \left(\frac{\sqrt{V(t_s)}}{E(t_s)} \right)^2 \right] \text{ und}$$

$$t_r = t_q + t_s = t_s \cdot (1 + \frac{\delta}{2 \cdot (1-\delta)}) \cdot \left[1 + \left(\frac{\sqrt{V(t_s)}}{E(t_s)} \right)^2 \right].$$

Bei der Bedienung peripherer Geräte gilt für den Variationskoeffizienten $C(t_s)$ der Bedienzeit t_s erfahrungsgemäß (siehe beispielsweise [7] und [11])

$$C(t_s) = \frac{\sqrt{V(t_s)}}{E(t_s)} \leq 1.$$

Damit gilt $1 + c(t_s)^2 \leq 2$. Somit kann die Exponentialverteilung als Worst Case betrachtet werden. Eine Ausnahme bildet der Bedienprozeß bei Zentralprozessoren. Für die Modellierung von Zentralprozessoren benutzt man besser Verteilungsfunktionen mit $C(t_s) > 1$. Dazu gehören die sogenannten Hyperexponentialverteilungen, die man interpretieren kann als eine Hintereinanderschaltung von gewichteten Exponentialverteilungen (siehe [7] und [11]). Wir fassen nun die Ergebnisse für die Leistungsindizes bei den Modellen M/M/1-FCFS, M/D/1-FCFS und M/G/1-FCFS tabellarisch zusammen (siehe Abbildung 1.3.9 - Abbildung 1.3.11)).

Leistungsindex	Relation
n_a	$n_a = \frac{\delta}{1-\delta}$
n_q	$n_q = \frac{\delta^2}{1-\delta}$
t_r	$t_r = \frac{t_s}{1-\delta}$
t_q	$t_q = t_s \cdot \frac{\delta}{1-\delta}$

Abbildung 1.3.9: Leistungsindizes für das M/M/1-FCFS-Modell

Leistungsindex	Relation
n_a	$n_a = \delta + \frac{\delta^2}{2 \cdot (1-\delta)}$
n_q	$n_q = \frac{\delta^2}{2 \cdot (1-\delta)}$
t_r	$t_r = t_s \cdot \frac{2-\delta}{2 \cdot (1-\delta)}$
t_q	$t_q = t_s \cdot \frac{\delta}{2 \cdot (1-\delta)}$

Abbildung 1.3.10: Leistungsindizes für das M/C/1-FCFS-Modell

Leistungsindex	Relation
n_a	$n_a = \delta + \frac{\delta^2}{2 \cdot (1-\delta)} \cdot \left[1 + C(t_s)^2\right]$
n_q	$n_q = \frac{\delta^2}{2 \cdot (1-\delta)} \cdot \left[1 + C(t_s)^2\right]$
t_r	$t_r = t_s \cdot \frac{2-\delta}{2 \cdot (1-\delta)} \cdot \left[1 + C(t_s)^2\right]$
t_q	$t_q = t_s \cdot \frac{\delta}{2 \cdot (1-\delta)} \cdot \left[1 + C(t_s)^2\right]$

Abbildung 1.3.11: Leistungsindizes für das M/G/1-FCFS-Modell

Am Beispiel eines peripheren Gerätes, das eine Magnetplatte sein kann, stellen wir t_r in Abhängigkeit vom Auslastungsgrad δ des Servers für die Modelle M/M/1 und M/C/1 dar (siehe Abbildung 1.3.12). Für t_s wählen wir $t_s = 0.020$ s. Dieser Wert entspricht in der Größenordnung der mittleren Bedienzeit einer Magnetplatte, wenn sie ohne Cache-Memory (siehe Kapitel 4) arbeitet. Das wesentliche Ergebnis nach Abbildung 1.3.12 ist, daß sich die Response-Time t_r mit zunehmender Last r exponentiell entwickelt.

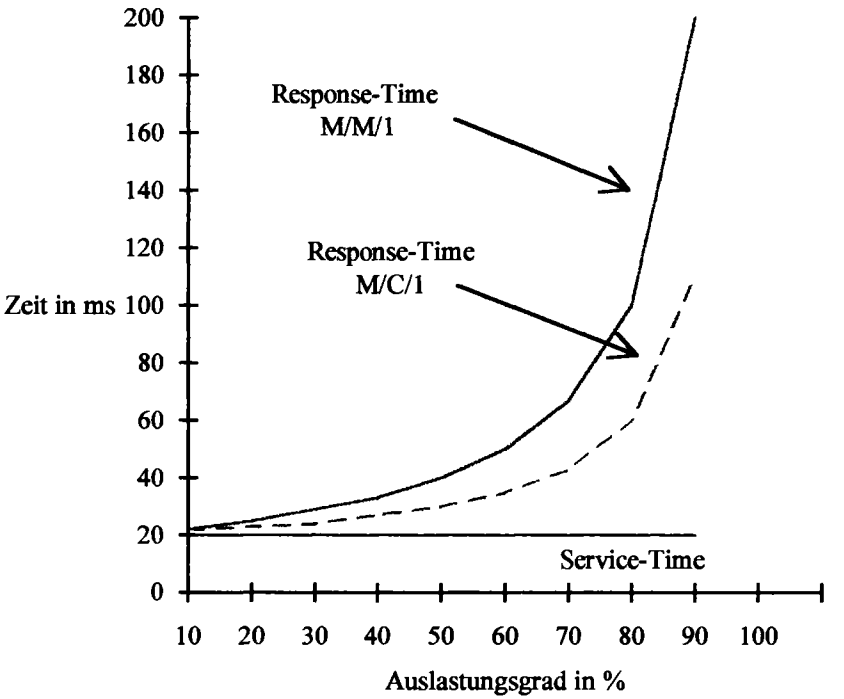


Abbildung 1.3.12: Response-Time einer Magnetplatte bei unterschiedlicher Verteilung der Bedienzeit

Wie die Abbildung außerdem zeigt, ist die Wartezeit t_q in der Warteschlange für dieses Verhalten verantwortlich. Die Entwicklung der Wartezeit nach

$$t_q = t_s \cdot \frac{\delta}{2(1-\delta)}\left[1 + C(t_s)^2\right].$$

wird verursacht durch die Zufälligkeit, mit der die Requests im System eintreffen und verarbeitet werden.

Wie schließen den Abschnitt ab mit einer Fehlerabschätzung für den Fehler bei der Response-Time t_r, wenn einerseits der Worst Case (exponentielle Verteilung beim Bedienprozeß), andererseits eine konstante Service-Time unterstellt wird (Best Case). Unter der

Annahme $C(t_s) \leq 1$ gilt für t_s bei Vorliegen einer allgemeinen Verteilung des Bedienprozesses

$$t_s \cdot \frac{2-\delta}{2\cdot(1-\delta)} \leq t_r \leq t_s \cdot \frac{1}{1-\delta}. \text{ Es folgt } \frac{t_s}{1-\delta}\cdot\left(1-\frac{\delta}{2}\right) \leq t_r \leq \frac{t_s}{1-\delta}$$

und damit für den relativen Fehler $\frac{\delta}{2-\delta}$.

Dieser ist demnach nur abhängig von der Auslastung des Servers. Bei einer Auslastung von beispielsweise 20% liegt der maximale relative Fehler bei ca. 11%. Im obigen Beispiel liegt die Response-Time bei einer Auslastung von 20% nach M/M/1 bei

$$t_r = \frac{t_s}{1-\delta} = \frac{0.020}{1-0.2} = 0.025$$

und nach M/C/1 bei

$$t_r = t_s \cdot \frac{2-\delta_s}{2-2\cdot\delta} = 0.020 \cdot \frac{1.8}{2\cdot 0.8} = 0.0225.$$

Schätzt man also die Response-Time wie hier die Response-Time einer Magnetplatte mit Hilfe des M/M/1-Modells ab, macht man einen Fehler von maximal 2.5 Millisekunden, wenn die Magnetplatte zu 20% ausgelastet ist. Bei üblichen Auslastungsgraden bis ca. 30% liegt der maximale relative Fehler bei ca. 18%.

Netze von Single-Node-Wartesystemen (Server-Netze)

In dem vorliegenden Abschnitt beschäftigen wir uns mit Netzen von Single-Node-Wartesystemen, auch als Server-Netze bezeichnet. Server-Netze bestehen aus mehreren Single-Node-Wartesystemen, die man die Knoten (Nodes) des Server-Netzes nennt. Wir beschränken unsere Betrachtung ausschließlich auf Nodes vom Typ M/M/m-FCFS. Als einzige Ausnahme lassen wir einen Knoten vom Typ M/M/∞ zu. Die Definition eines M/M/∞ - Knotens werden wir weiter unten vornehmen. Die Knoten des Server-Netzes dienen der Modellierung der einzelnen Systemkomponenten eines Rechner-

systems, wie beispielsweise Prozessor, Magnetplatteneinheiten und Terminals. Ein Auftrag (Transaktion, Job) durchläuft dabei die Knoten des Netzes und wird mit einer bestimmten Wahrscheinlichkeit - genannt Übergangswahrscheinlichkeit - von einem zum anderen Knoten durch das System geschleust. Er erzeugt dabei einen oder mehrere Requests an die Server. Man unterscheidet grundsätzlich zwischen offenen und geschlossenen Server-Netzen. Wir behandeln zunächst offene, im Anschluß geschlossene Netze.

Offene Server-Netze

Bei offenen Server-Netzen sind Aufträge von außen und Abgänge nach außen in jedem Knoten des Netzes zugelassen. Man definiert (siehe zum Beispiel [7]):

N	Anzahl der Knoten des Server-Netzes,
$r_{0,i}$	mittlere Ankunftsrate der Aufträge von außen im Knoten i,
r_i	mittlere Ankunftsrate der Requests im Knoten i (außen und innen),
m_i	Anzahl Server im Knoten i,
c_i	mittlere Service-Rate der Requests im Knoten i,
$p_{i,j}$	Wahrscheinlichkeit, daß ein im Knoten i bearbeiteter Auftrag zum Knoten j übergeht,
$p_{i,N+1}$	Wahrscheinlichkeit, daß ein im Knoten i bearbeiteter Auftrag das System verläßt,
$p_{0,i}$	Wahrscheinlichkeit, daß ein Auftrag von außen im Knoten i eintrifft.

Beispiel

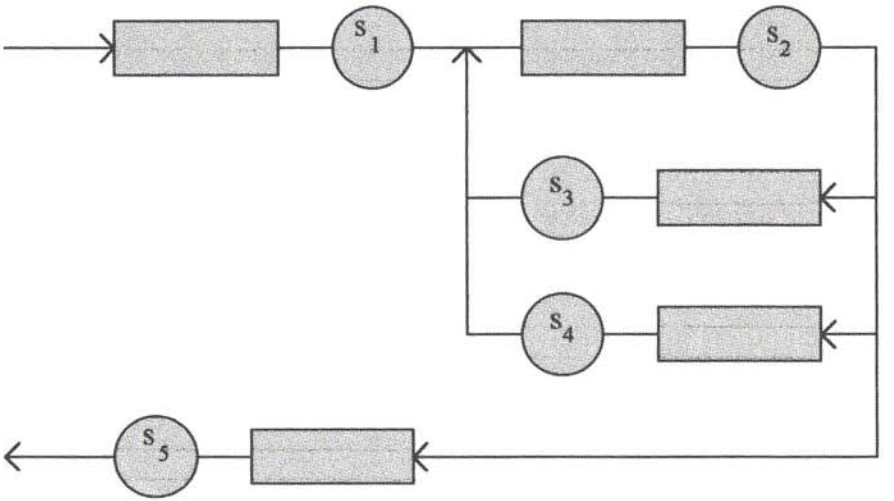
Wir betrachten ein Server-Netz bestehend aus N=5 Servern S_i, i=1,...5:

- Input-Server S_1: Eingabe-Device, über das alle Aufträge in das Server-Netz gelangen $(p_{0,1} = 1)$,
- Prozessor-Server S_2: CPU, die von jedem Auftrag, der über den Input-Server in das System gelangt, als erster Server frequentiert wird $(p_{1,2} = 1)$,
- DASD-Server S_3: Magnetplatte, die von jedem Auftrag viermal beansprucht wird. Nach jeder Ein-/Ausgabe geht der Auftrag

zurück zum Prozessor und verläßt nach der vierten schließlich das System ($p_{2,3} = 0.4, p_{3,2} = 1$),

- DASD-Server S_4: Magnetplatte, die von jedem Auftrag fünfmal beansprucht wird. Nach jeder Ein-/Ausgabe geht der Auftrag zurück zum Prozessor und verläßt nach der fünften schließlich das System ($p_{2,4} = 0.5, p_{4,2} = 1$),
- Output-Server S_5: Ausgabe-Device, über das alle Aufträge das System verlassen ($p_{2,5} = 0.1, p_{5,6} = 1$).

Wir stellen das beschriebene Server-Netz in Abbildung 1.3.13 schematisch dar und beschreiben die Parameter des Systems.



1.3.13: Beispiel eines offenen Server-Netzes

Modellparameter

Als Modellparameter bezeichnen wir alle Parameter, die das Modell vollständig beschreiben:

- Die Knoten des Server-Netzes durch "A_i / B_i / m_i -Scheduling-Algorithmus" ($i = 1,....,N$),
- den Netztyp - offen oder geschlossen und

- die Übergangswahrscheinlichkeiten $p_{i,j}$ $(i = 0,...,N; j = 1,...,N+1)$.

Lastparameter

Als Workload oder Lastparameter gilt die Requestrate r (=Gesamtdurchsatz):

$$r = \sum_i r_{o,i} .$$

Leistungsindizes

Leistungsindizes sind

- die mittlere Anzahl n_a der Aufträge im System,
- die mittlere Anzahl n_q der auf Bedienung wartenden Aufträge im System,
- die mittlere Response-Time t_r (Verweilzeit, Elapsed Time) eines Auftrages,
- die mittlere Queue-Time t_q eines Auftrages,
- die mittlere Anzahl n_{a_i} aktiver Aufträge im Server i,
- die mittlere Anzahl n_{q_i} der auf Bedienung wartenden Aufträge im Server i,
- die mittlere Verweilzeit t_{r_i} eines Requests im Knoten i,
- die mittlere Aufenthaltsdauer t_{q_i} eines Requests in der Warteschlange des Knotens i,
- die Transaction-Rate r,
- die Anzahl r_i der pro Zeiteinheit durchgesetzten Requests im Knoten i und der
- Auslastungsgrad δ_i des Servers i.

Wir formulieren nun den Satz von Jackson für den Spezialfall eines offenen Server-Netzes, bestehend aus N M/M/1-FCFS-Knoten (siehe [7]):

Theorem von Jackson

Falls das Gleichungssystem $r_i = r_{0,i} + \sum_j p_{j,i} \cdot r_j \quad (i = 1,\ldots,N)$ eindeutig lösbar ist und für die Lösungen r_i $r_i < t_{s_i}^{-1}$ gilt, ist die Wahrscheinlichkeit für den Gleichgewichtszustand $\vec{k} = (k_1,\ldots,k_N)$ durch das Produkt der Zustandswahrscheinlichkeiten der einzelnen Knoten (Randwahrscheinlichkeiten) $p_i(k_i) = p(k_i$ Aufträge im Knoten i) gegeben. Es gilt

$$p(\vec{k}) = p(k_1,\ldots,k_N) = \prod_{i=1}^{N} p(k_i) \quad k_i \in 0,\ldots,n_a .$$

Dabei steht $p(\vec{k}) = p(k_1,\ldots,k_N)$ für die Wahrscheinlichkeit, daß sich in den einzelnen Knoten i genau k_i Aufträge befinden (wartend oder gerade bedient werdend).

Die einzelnen Knoten können nach dem Theorem von Jackson als voneinander unabhängige M/M/1-FCFS-Knoten betrachtet werden mit der Ankunftsrate r_i und der Bedienrate $t_{s_i}^{-1}$, das heißt, für die Randwahrscheinlichkeiten $p_i(k_i)$ gelten die bekannten M/M/1-Relationen

$$n_{a_i} = \frac{\delta_i}{1-\delta_i},\ n_{q_i} = \frac{\delta_i^2}{1-\delta_i} \quad \text{und damit}$$

$$n_a = \sum_i n_{a_i} = \sum_i \frac{\delta_i}{1-\delta_i} \text{ und } n_q = \sum_i n_{q_i} = \sum_i \frac{\delta_i^2}{1-\delta_i}.$$

$$\text{Aus } t_{r_i} = \frac{n_{a_i}}{r_i} = \frac{t_{s_i}}{1-\delta_i} \text{ und } t_{q_i} = \frac{n_{q_i}}{r_i} = t_{s_i} \cdot \frac{\delta_i}{1-\delta_i} \text{ folgt}$$

$$t_r = \frac{n_a}{r} = \frac{1}{r}\sum_i n_{a_i} = \frac{1}{r}\sum_i \frac{\delta_i}{1-\delta_i} \text{ und } t_q = \frac{n_q}{r} = \frac{1}{r}\sum_i n_{q_i} = \frac{1}{r}\sum_i \frac{\delta_i^2}{1-\delta_i}.$$

Setzt man $d_i = \frac{r_i}{r}$, so ist d_i die mittlere Anzahl Requests, die von einem Auftrag im Knoten i generiert werden (auch Anzahl "Besuche" eines Auftrages im Knoten i). d_i läßt sich direkt aus den Übergangswahrscheinlichkeiten bestimmen. Wegen

$$r_{o,i} = r \cdot p_{o,i}$$

folgt

$$r_i = r_{o,i} + \sum_j r_j \cdot p_{j,i}\text{, damit } d_i = p_{o,i} + \sum_j d_j \cdot p_{j,i}$$

und schließlich

$$t_r = \frac{n_a}{r} = \frac{1}{r}\sum_i \frac{\delta_i}{1-\delta_i} = \sum_i \frac{r_i}{r} \cdot \frac{t_{s_i}}{1-\delta_i} = \sum_i d_i \cdot \frac{t_{s_i}}{1-\delta_i} = \sum_i d_i \cdot t_{r_i}\,.$$

Wir belegen nun das obige Beispiel mit Zahlen und berechnen für einige Fälle die Leistungsindizes nach dem Theorem von Jackson.

Es sei r = 5, $t_{s_1} = t_{s_5} = 0.020s$, $t_{s_2} = 0.015s$, $t_{s_3} = 0.020s$ und $t_{s_4} = 0.030s$. Die Matrix der Übergangswahrscheinlichkeiten $\left(p_{i,j}\right)$, i=2,..,5; j=2,..,5 habe die Werte ($(p_{i,o})$, i=1,..,5 und $(p_{5,j})$, j=1,..,5 sind 0)

1.0	0.0	0.0	0.0
0.0	0.4	0.5	0.1
1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0

Damit gehen wir in das Gleichungssystem $r_i = \sum_j p_{j,i} \cdot r_j$:

$r_1 = r$

$r_2 = r_1 \cdot p_{1,2} + r_3 \cdot p_{3,2} + r_4 \cdot p_{4,2}$

$r_3 = r_2 \cdot p_{2,3}$

$r_4 = r_2 \cdot p_{2,4}$

$r_5 = r_2 \cdot p_{2,5}$.

Als Lösung erhält man $r_1 = 5, r_2 = 50, r_3 = 20, r_4 = 25$ und $r_5 = 5$. Wir stellen die berechneten Leistungsgrößen $\delta_i, n_{a_i}, n_{q_i}, t_{r_i}$ und t_{q_i} für die einzelnen Knoten in Abbildung 1.3.14 tabellarisch zusammen.

S_i	δ_i	n_{a_i}	n_{q_i}	t_{r_i}	t_{q_i}
1	0.10	0.11	0.01	0.022	0.002
2	0.75	3.00	2.25	0.060	0.045
3	0.40	0.66	0.26	0.033	0.013
4	0.75	3.00	2.25	0.120	0.090
5	0.10	0.11	0.01	0.022	0.002

Abbildung 1.3.14: Serverbezogene Leistungsindizes für r =5

Aus den Werten nach Abbildung 1.3.14 folgt

$$n_a = \sum_i n_{a_i} = \sum_i \frac{\delta_i}{1-\delta_i} = 6.\overline{8},$$

$$n_q = \sum_i n_{q_i} = \sum_i \frac{\delta_i^2}{1-\delta_i} = 4.7\overline{8},$$

$$t_r = \frac{n_a}{r} = \frac{6.\overline{8}}{5} = 1.3\overline{7} \text{ und}$$

$$t_q = \frac{n_q}{r} = \frac{4.7\overline{8}}{5} = 0.95\overline{7}.$$

t_r läßt sich nach der oben abgeleiteten Relation auch direkt aus t_{r_i} bestimmen. Es ist

$$t_r = \sum_i d_i \cdot t_{r_i} = 0.0\overline{2} + 0.60 + 0.1\overline{3} + 0.60 + 0.0\overline{2} = 1.3\overline{7}.$$

Entsprechend gilt für t_q

$$t_q = \sum_i d_i \cdot t_{q_i} = \sum_i d_i \cdot \left(t_{r_i} - t_{s_i}\right)$$
$$= \sum_i d_i \cdot t_{r_i} - \sum_i d_i \cdot t_{s_i} = 1.3\overline{7} - 0.420 = 0.95\overline{7}.$$

Dabei ist $t_s = \sum_i d_i \cdot t_{s_i}$ die Service-Time der Anforderung über alle Server. Sie entspricht der Alleinelaufzeit.

Wir stellen die Behandlung offener Server-Netze abschließend das Verhalten des Workloads bei wachsender Transaktionslast r dar. Wir beschränken uns dabei auf die Bestimmung der Größen t_{r_i}, t_{q_i}, t_r und t_q. Die Werte für die Server S_1 und S_5 fassen wir zusammen. Die Ergebnisse (Werte in Millisekunden) stellen wir in Abbildung 1.3.15 tabellarisch und in Abbildung 1.3.16 grafisch dar.

r	$t_{q_{1/5}}$	t_{q_2}	t_{q_3}	t_{q_4}	t_q	t_s	t_r
1	1	3	2	5	64	420	484
2	2	6	4	13	143	420	563
3	3	12	6	25	272	420	692
4	3	23	9	45	494	420	914
5	4	45	13	90	956	420	1376
6	5	135	18	270	2777	420	3197

Abbildung 1.3.15: Response- und Queue-Time in Abhängigkeit von der Transaction-Rate r

Aus Abbildung 1.3.15 ist erkennbar, daß der Prozessor-Server S_2 und der DASD-Server S_4 den größten Anteil zur Queue-Time beitragen. Beispielsweise gilt für $r = 6$ (Werte gerundet in Millisekunden)

$$d_1 \cdot t_{q_1} \approx 1 \cdot 3 = 3,$$
$$d_2 \cdot t_{q_2} = 10 \cdot 135 = 1.350,$$

$$d_3 \cdot t_{q_3} = 4 \cdot 18 = 72,$$
$$d_4 \cdot t_{q_4} = 5 \cdot 270 = 1.350,$$
$$d_5 \cdot t_{q_5} \approx 1 \cdot 3 = 3 \text{ und}$$
$$t_q = \sum_i d_i \cdot t_{q_i} = 2.778.$$

Damit werden von diesen beiden Servern jeweils ca. 48.6% der Queue-Time bzw. 42.2% der gesamten Durchlaufzeit (Response-Time) der Transaktion in Anspruch genommen. Diese Server bilden einen Engpaß beim Durchfluß der Transaktionen. Der in diesem Kontext übliche Begriff Bottleneck-Server wird später noch exakt definiert werden.

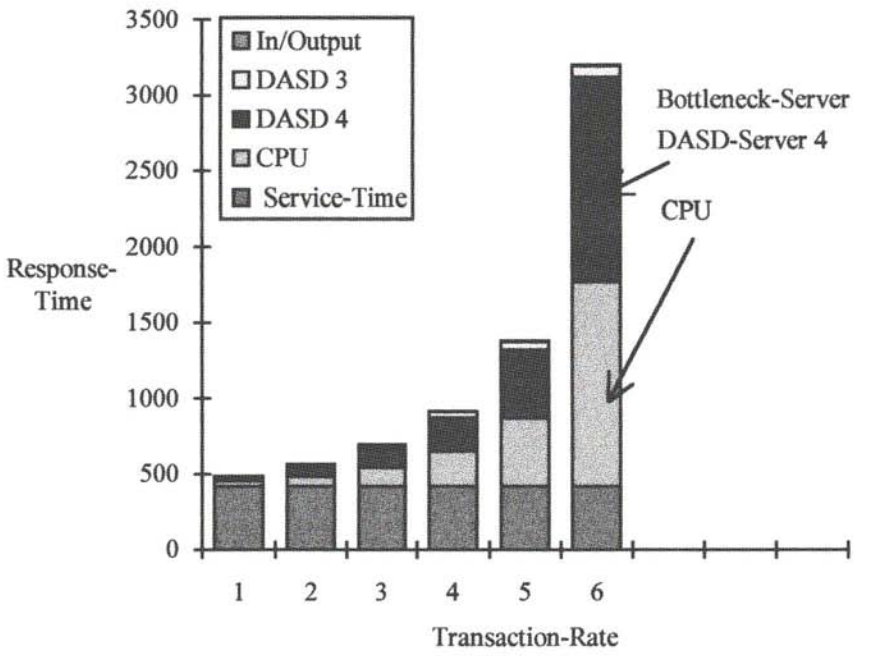


Abbildung 1.3.16: Response- und Queue-Time in Abhängigkeit von der Transaction-Rate r

Abbildung 1.3.16 zeigt sehr schön, wie sich die Queue-Time-Komponenten $d_i \cdot t_{q_i}$ der einzelnen Server und damit die Durchlaufzeiten $d_i \cdot t_{r_i}$ mit wachsender Transaktionslast exponentiell entwickeln.

Mit diesem für das Verhalten von Rechnersystemen wichtigen Ergebnis - Anstieg der Antwortzeiten bei zunehmender Last (Transaction-Rate) und zwar überproportional - schließen wir die Behandlung offener Server-Netze ab. Ziel eines Performance-Managements wird es sein - soviel kann an dieser Stelle schon gesagt werden -, die eventuell bestehenden Engpässe in einem System zu erkennen, das heißt, die Engpaßserver zu identifizieren. Auf einer so gesicherten Basis sind dann die richtigen Maßnahmen einzuleiten, um diese Engpässe unter wirtschaftlichen Gesichtspunkten zu entschärfen. Wir behandeln nun geschlossene Server-Netze.

Geschlossene Server-Netze

Ein geschlossenes Server-Netz ist dadurch charakterisiert, daß keine Zu- und Abgänge erfolgen. Aufträge verlassen, nachdem sie abgearbeitet sind, nicht das System, sondern werden nach ihrer Bearbeitung dem System als neuer Auftrag zugeführt. Geschlossene Netze eignen sich gut für die Modellierung von terminalgetriebenen Transaktionsklassen mit einer festen Anzahl von Benutzern (= Terminals). Die Terminals werden dabei durch einen Knotentyp modelliert (Knoten vom Typ M/M/∞), den wir in dem vorliegenden Kontext noch definieren werden. Zunächst stellen wir analytisch dar, welche Auswirkungen die Annahme "keine Zu- und Abgänge" auf die bisher dargestellten Relationen hat. "Keine Zu- bzw. Abgänge" läßt sich formulieren mit Hilfe der Wahrscheinlichkeiten $p_{o,i}$ und $p_{i,N+1}$. Es gilt

$$p_{o,i} = p_{i,N+1} = 0 \quad (i = 1,\ldots,N),$$

das heißt, die Anzahl der Aufträge, die durch das System zirkulieren, ist konstant ($k = \sum_i k_i$). Damit gilt für die mittlere Ankunftsrate im Knoten i

$$r_i = \sum_{j=1}^{N} r_j \cdot p_{j,i} \; .$$

Ist $d_i = \frac{r_i}{r}$ wieder die mittlere Anzahl von Besuchen pro Auftrag im Server i, so folgt

$$d_i = \frac{r_i}{r} = \frac{\sum_{j=1}^{N} r_j \cdot p_{j,i}}{r} = \sum_{j=1}^{N} \frac{r_j}{r} \cdot p_{j,i} = \sum_{j=1}^{N} d_j \cdot p_{j,i} \; .$$

Wir beschränken uns auch bei den geschlossenen Server-Netzen auf Knoten vom Typ M/M/1 bis auf den bereits erwähnten Fall eines Knotens vom Typ M/M/∞ (siehe weiter unten), den wir für die Modellierung der Terminals benötigen. Alle Leistungsindizes und Lastgrößen bleiben unverändert definiert. Zur Bestimmung der Leistungsgrößen in einem geschlossenen Netz gibt es eine Reihe von Algorithmen, die der Bestimmung der Normierungskonstanten G(k) dienen. G(k) ist definiert durch den Satz von Gordon/Newell (siehe [7] und [11]). Nach dem Satz von Gordon/Newell gilt für ein geschlossenes Server-Netz im Gleichgewichtszustand die Relation

$$p\left(\vec{k}\right) = p(k_1,\ldots,k_N) = \frac{1}{G(k)} \cdot \prod_{i=1}^{N} \frac{x_i^{k_i}}{k_i!} \quad (x_i = d_i \cdot t_{si}, \; i = 1,\ldots,N).$$

Zu den Algorithmen für die Bestimmung der Leistungsindizes gehören (siehe [7]) der Algorithmus von Buzen (1971/1973), der Algorithmus von Chandy, Herzog und Wov (1975) und der Algorithmus von Kobayashi (1978). Diese sind in [7] detailliert beschrieben und hergeleitet. Wir verzichten auf die Darstellung dieser Algorithmen und beschränken uns auf die sogenannte Mittelwertanalyse zur Bestimmung der Leistungsindizes t_r, t_q, n_a und n_q. Die Mittelwertanalyse basiert auf dem Gesetz von Little und dem nachfolgend formulierten Satz über die Verteilung beim Ankunftsprozeß (siehe [7]).

Satz über die Verteilung beim Ankunftsprozeß

Bei einem geschlossen Warteschlangennetz gilt für die Wahrscheinlichkeit $p(\vec{k})$, daß ein Auftrag, der in den Knoten i eintritt, den Netzzustand $\vec{k} = (k_1,...,k_N)$ mit $\sum_i k_i = k$ vorfindet

$$p(\vec{k}) = p(k_1,...,k_N) = p(k_1,...,k_i - 1,...,k_N).$$

Die rechte Seite dieser Relation beschreibt den Gleichgewichtszustand des Server-Netzes für einen Auftrag weniger im Knoten i $(k_i - 1)$.

Wir formulieren nun einen Iterationsprozeß zur Bestimmung der Mittelwerte t_r, r und n_a in einem geschlossenen Server-Netz von M/M/1- und M/M/∞ -Knoten. Vorher definieren wir noch, was wir unter einem Knoten vom Typ M/M/∞ verstehen.
Ein Knoten vom Typ M/M/∞ besitzt keine Warteschlange und besteht aus beliebig vielen Servern. Mit einem Knoten dieses Typs läßt sich ein Terminalsystem gut modellieren. Die Service-Time der Server eines M/M/∞ -Knotens entspricht dann der User-Think-Time t_u. Nach Ablauf der User-Think-Time erfolgt die Eingabe des neuen Requests. Dieser wird nach Ablauf dieser Zeit, ohne daß er in eine Warteschlange eingereiht wird, als im ersten "Systemserver" eingetroffen angenommen. Man kann sich auch vorstellen, daß der Knoten über unendlich viele Warteschlangen verfügt, die dann immer leer sind.
In Abbildung 1.3.17 zeigen wir die schematische Darstellung eines geschlossenen Server-Netzes mit einem M/M/∞-Terminalknoten. Dieser hat den Input- und den Output-Server aus unserem Beispiel für offene Netze ersetzt. Die restlichen Knoten entsprechen den Knoten aus unserem Beispiel für offene Netze. Die Übergangswahrscheinlichkeiten $p_{i,j}$ des in Abbildung 1.3.17 dargestellten Netzes haben folgende Werte:

$p_{1,2} = 1.0$; $p_{2,3} = 0.4$; $p_{2,4} = 0.5$; $p_{2,1} = 0.1$; $p_{3,2} = 1.0$ und $p_{4,2} = 1.0$.

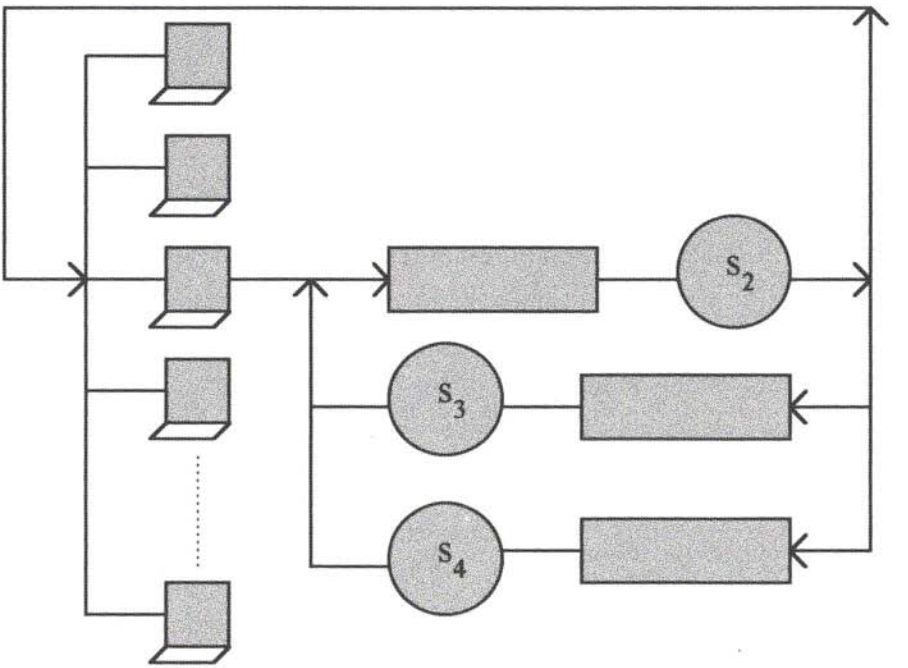


Abbildung 1.3.17: Beispiel eines geschlossenen Server-Netzes

Wir geben nun den Algorithmus für die Bestimmung der Leistungsgrößen an.

Algorithmus für die Berechnung der Leistungsindizes in einem geschlossenen Server-Netz mit Knoten vom Typ M/M/1-FCFS und M/M/∞ - FCFS (Mittelwertanalyse)

- Berechne $x_i = d_i \cdot t_{s_i}$ für i=1,.....,N
- Initiiere $k_i^0 = 0$ für i=1,.....,N
- Iteriere für j=1 bis k (k Anzahl der Aufträge im System)

Verweilzeiten: $t_{r_i}^j = \begin{cases} x_i \cdot \left(1 + k_i^{j-1}\right) & m_i = 1 \\ x_i & m_i = \infty \end{cases} \quad i = 1,....,N$

Durchsatz: $r^j = \frac{j}{\sum_{i=1}^{N} t_{r_i}^j}$ und $r_i^j = d_i \cdot r^j$, $i = 1,....,N$

Anzahl Aufträge: $k_i^j = r_i^j \cdot t_{r_i}^j$, $i = 1,....,N$

Die Größen $t_{r_i}^k, r^k, r_i^k$ $t_{r_i}^k$, r^k, r_i^k und k_i^k sind dann die gesuchten Leistungsindizes.

Beispiel

Wir rechnen das Modell nach Abbildung 1.3.17 ohne die Terminal-Server. Es gilt $d_1 = 10$, $d_2 = 4$ und $d_4 = 5$. Damit ist $\left(x_i = d_i \cdot t_{s_i}\right)$ für $i = 2,3,4$ $x_2 = 0.150$, $x_3 = 0.080$ und $x_4 = 0.150$. Wir rechnen dieses Beispiel für k=5 und iterieren j von 1 bis 5:

Initiiere $k_i^0 = 0$ für $i = 2,3,4$

1. Iterationsschritt (j=1)

Verweilzeiten: $t_{r_i}^j = x_i \cdot \left(1 + k_i^{j-1}\right)$: $t_{r_2}^1 = 0.150, t_{r_3}^1 = 0.080, t_{r_4}^1 = 0.150$.

Durchsatz : $r^j = \frac{j}{\sum_i t_{r_i}^j}$ und $r_i^j = d_i \cdot r^j$: $r^1 = \frac{1}{0.380} \approx 2.632$ und

$r_2^1 = 26.32, r_3^1 = 10.53, r_4^1 = 13.16$.

Anzahl Aufträge: $k_i^j = r^j \cdot t_{r_i}^j$: $k_2^1 = 0.395, k_3^1 = 0.211, k_4^1 = 0.395$.

2. Iterationsschritt (j=2)

Verweilzeiten: $t_{r_i}^j = x_i \cdot \left(1 + k_i^{j-1}\right)$: $t_{r_2}^2 = 0.209, t_{r_3}^2 = 0.097, t_{r_4}^2 = 0.209$.

Durchsatz: $r^j = \frac{j}{\sum_i t_{r_i}^j}$ und $r_i^j = d_i \cdot r^j$: $r^2 = \frac{2}{0.515} \approx 3.883$ und

$r_2^2 = 38.83, r_3^2 = 15.53, r_4^2 = 19.42$.

Anzahl Aufträge: $k_i^j = r^j \cdot t_{r_i}^j$: $k_2^2 = 0.812, k_3^2 = 0.377, k_4^2 = 0.812$.

3. Iterationsschritt (j=3)

Verweilzeiten: $t_{r_i}^j = x_i \cdot \left(1 + k_i^{j-1}\right) : t_{r_2}^3 = 0.272 , t_{r_3}^3 = 0.272 , t_{r_4}^3 = 0.272 .$

Durchsatz: $r^j = \frac{j}{\sum_i t_{r_i}^j}$ und $r_i^j = d_i \cdot r^j : r^3 = \frac{3}{0.654} \approx 4.587$ und

$r_2^3 = 45.87 , r_3^3 = 18.35 , r_4^3 = 22.94 .$

Anzahl Aufträge: $k_i^j = r^j \cdot t_{r_i}^j : k_2^3 = 1.248 , k_3^3 = 0.505 , k_4^3 = 1.248 .$

4. Iterationsschritt (j=4)

Verweilzeiten: $t_{r_i}^j = x_i \cdot \left(1 + k_i^{j-1}\right) : t_{r_2}^4 = 0.337 , t_{r_3}^4 = 0.120 , t_{r_4}^4 = 0.337 .$

Durchsatz: $r^j = \frac{j}{\sum_i t_{r_i}^j}$ und $r_i^j = d_i \cdot r^j : r^4 = \frac{4}{0.794} \approx 5.038$ und

$r_2^4 = 50.38 , r_3^4 = 20.15 , r_4^4 = 25.19 .$

Anzahl Aufträge: $k_i^j = r^j \cdot t_{r_i}^j : k_2^4 = 1.698 , k_3^4 = 0.605 , k_4^4 = 1.698 .$

5. Iterationsschritt (j=5)

Verweilzeiten: $t_{r_i}^j = x_i \cdot \left(1 + k_i^{j-1}\right) : t_{r_2}^5 = 0.405 , t_{r_3}^5 = 0.128 , t_{r_4}^5 = 0.405 .$

Durchsatz: $r^j = \frac{j}{\sum_i t_{r_i}^j}$ und $r_i^j = d_i \cdot r^j : r^5 = \frac{5}{0.938} \approx 5.330$ und

$r_2^5 = 53.30 , r_3^5 = 21.32 , r_4^5 = 26.65 .$

Anzahl Requests: $k_i^j = r^j \cdot t_{r_i}^j : k_2^5 = 2.159 , k_3^5 = 0.682 , k_4^5 = 2.159 .$

Im Ergebnis erhält man die in den Abbildungen 1.3.18 und 1.3.19 dargestellten Werte. In der ersten Tabelle werden die globalen Leistungsindizes, in der zweiten die serverbezogenen angezeigt. Die Iteration realisieren wir nun mit einem einfachen BASIC-Programm, dessen Coding wir im Anhang A.1 zur Verfügung stellen. Wir rechnen damit das Modell inklusive der Terminal-Server.

Leistungsindex	Wert
r	5.330
t_r	0.938
t_q	0.558
n_a	5.000
n_q	2.970

Abbildung 1.3.18: Systemweite Leistungsindizes

Leistungsindex	S_2	S_3	S_4
r_i	53.30	21.32	26.65
t_{r_i}	0.405	0.128	0.405
t_{q_i}	0.255	0.048	0.255
n_{a_i}	2.159	0.682	2.159
n_{q_i}	1.359	0.256	2.159
δ_i	0.800	0.430	0.800

Abbildung 1.3.19: Serverbezogene Leistungsindizes

Dabei nehmen wir $t_{s_i} = 20$ $(= t_u)$ und $k = 100$ an. Das Ergebnis stellen wir in den Abbildungen 1.3.20 und 1.3.21 dar.

Leistungsindex	Wert
r	4.740
t_r	21.10
t_q	0.720
n_a	100.0
n_q	3.400

Abbildung 1.3.20: Systemweite Leistungsindizes

Leistungsindex	S_1	S_2	S_3	S_4
r_i	4.740	47.40	18.96	23.70
t_{r_i}	20.00	0.490	0.130	0.490
t_{q_i}	0.000	0.340	0.050	0.340
n_{a_i}	94.80	2.160	0.610	2.300
n_{q_i}	0.000	1.360	0.230	1.590
δ_i	0.950	0.800	0.380	0.710

Abbildung 1.3.21: Serverbezogene Leistungsindizes

Bei dieser Darstellung ist die User-Think-Time t_u in der Gesamtverweilzeit der Transaktion enthalten, weil das Modellsystem die Terminalserver und deren Service-Time (t_u), die gerade der Denkzeit des Benutzers entspricht, als Systemkomponente betrachtet. Der Benutzer sieht aber nur das System mit den Servern S_2 bis S_4. Aus dessen Sicht gilt deshalb

$$t_r = \sum_{i=2}^{4} d_i \cdot t_{r_i} = 1.098 \text{ und } n_a = r \cdot t_r = 4.740 \cdot 1.098 = 5.20.$$

Wir unterscheiden im folgenden zwischen k und n_a und verstehen unter t_r stets die "interne" Systemverweilzeit. Damit gilt

$$r = \frac{k}{t_r + t_u} \text{ und im obigen Beispiel } r = \frac{k}{t_r + t_u} = \frac{100}{1.098 + 20} = 4.740.$$

Auf diese Relation werden wir im Kapitel 2 bei der Behandlung der Gesamtleistung eines Rechnersystems zurückkommen. Sie ermöglicht uns, bei Systemmodellen mit wenigen Servern, eine direkte Berechnung der Leistungsindizes.

Wir besprechen nun noch den Spezialfall eines geschlossenen Server-Netzes, das als Central Server-Modell bezeichnet wird. Dieses Modell wurde zum ersten Mal von Buzen (siehe [7] und [11])

entwickelt. Im Central Server-Modell hat jeder Knoten eine Bedieneinheit ($m_i = 1$). Es gelten die Übergangswahrscheinlichkeiten

$$p_{i,j} = \begin{cases} p_j & i = 1 \wedge j = 1,...,N \\ 1 & j = 1 \wedge i = 2,...,N \\ 0 & \text{sonst} \end{cases}.$$

Der Server S_1 ist dabei der Central Server. Aufträge, die diesen Server verlassen, gehen mit der Wahrscheinlichkeit p_i zum Server S_i (i=1,....,). Aufträge, die durch die Server S_i gelaufen sind, kehren immer zum Central Server zurück ($p_{i,1} = 1$). Mit dem Server S_1 modelliert man gewöhnlich eine Central Processing-Unit (CPU), mit den Servern S_i (i=2,....,N) periphere Geräte wie beispielsweise Magnetplatten. In Abbildung 1.3.22 stellen wir ein typisches Central Server-Modell dar. Die Berechnung der Mittelwerte der Leistungsindizes erfolgt nach derselben Methode wie bei dem allgemeinen geschlossenen Server-Netz.

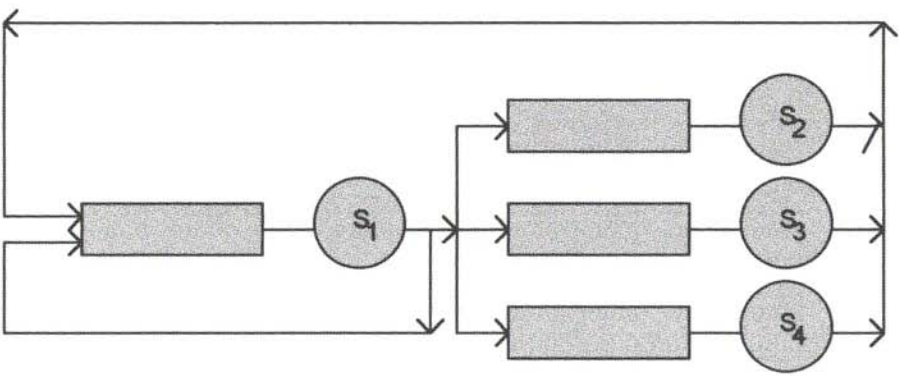


Abbildung 1.3.22: Beispiel eines Central Server-Modells

Das in Abbildung Abbildung 1.3.22 dargestellte Modell ist in [7] für k=6 mit $t_{s_1} = 0.020$, $t_{s_2} = 0.200$, $t_{s_3} = 0.400$, $t_{s_4} = 1.67^{-1}$ und $p_1 = 0.3$, $p_2 = 0.4$, $p_3 = 0.2$, $p_4 = 0.1$ gerechnet.