RECHENZENTRUM TH MÜNCHEN
ARBEITSGRUPPE BETRIEBSSYSTEME

INTERNSCHRIFT Nr. 40

THEMA: **Überblick über Fragen der Leistungsmessung in einem Betriebssystem**

VERFASSER: **Wolf, Sediva**

DATUM: 6.5.1970

FORM DER ABFASSUNG

- ENTWURF
- AUSARBEITUNG
- ENDFORM

SACHLICHE VERBINDLICHKEIT

- ALLGEMEINE INFORMATION DISKUSSIONSGRUNDLAGE X
- ERARBEITETER VORSCHLAG
- VERBINDLICHE MITTEILUNG
- VERALTET

ÄNDERUNGSZUSTAND

BEZUG AUF BISHERIGE INTERNSCHRIFTEN
Vorkenntnisse aus:
Erweiterung von:
Ersatz für:

BEZUG AUF KÜNFTIGE INTERNSCHRIFTEN
Vorkenntnisse zu:
Erweiterung in:
Ersetzt durch:

ANDERWEITIGE LITERATUR

siehe Literaturhinweise am Ende dieser IS

4o/o6o57o/Wo Se

Inhaltsverzeichnis

Überblick über Fragen der Leistungsmessung in einem Betriebssystem

1. Zweck von Leistungsmessungen

   1.1 Abrechnung, Benutzer- und Betriebsstatistik
   1.2 Regelung
   1.3 Quantitative Untersuchungen im eigentlichen Sinn
   1.4 Allgemeine Gesichtspunkte

2. Methoden und Aufwand

   2.1 Analytische Methode

      2.1.1 Voraussetzungen
      2.1.2 Durchführung der Messungen
      2.1.3 Sicherstellung und Verarbeitung der gesammelten Daten

   2.2 "Stimulus"-Methode (Benchmark-Technik)

      2.2.1 Grundsätzliches
      2.2.2 Meßtechniken

   2.3 Vergleich beider Methoden

3. Implementierung von Meßmethoden, mögliche Untersuchungen (Beispiele)

   3.1 MTS (Michigan Terminal System)

      3.1.1 Bemerkungen zum System und zur Meßmethode
      3.1.2 Bemerkungen zur Implementierung
      3.1.3 Mögliche Untersuchungen

./. 2

3.2 RTOS (Apollo Real-Time Operating System)

3.2.1 Bemerkungen zum System
3.2.2 Untersuchungen und Methoden
3.2.3 Bemerkungen zur Implementierung

3.3 Multics (Multiplexed Information and Computing System)

3.3.1 Bemerkungen zum System
3.3.2 Anwendungen der analytischen Methode
3.3.3 Anwendungen der Stimulus-Methode (Benchmark-Technik)

3.4 ADEPT

3.4.1 Bemerkungen zum System
3.4.2 Anwendungen der analytischen Methode
3.4.3 Anwendungen der Stimulus-Methode (Benchmark-Technik)

Anhang:

Die Verwendung des exponentiellen Mittelwertes bei der Aufbereitung von Meßwerten.

Überblick über Fragen der Leistungsmessung in einem Betriebssystem

Im folgenden soll ein Überblick über die Probleme im Zusammenhang mit der Leistungsmessung in einem Betriebssystem gegeben werden, ohne bereits konkret auf den TR 440 bzw. das BSM einzugehen. Dabei erscheint interessant: Zu welchem Zweck sollen Leistungen des Systems gemessen werden? Mit welchen Methoden kann gemessen werden und welchen Aufwand erfordern diese Methoden? Wie lassen sich solche Meßmethoden implementieren und welche Fragen können damit untersucht werden (Beispiele)?

1. Zweck von Leistungsmessungen

Die Gewinnung von Information in einem System kann zum Zweck der Abrechnung mit den Benutzern und für eine Benutzer- und Betriebsstatistik, zum Zweck der Regelung des Systems oder zur Untersuchung spezieller Fragen erfolgen, die etwa in Zusammenhang mit dem Ausbau oder der Änderung eines Systems geklärt werden müssen.

1.1 Abrechnung, Benutzer- und Betriebsstatistik

Das Betriebssystem teilt den Programmen (und damit den durch diese Programme vertretenen Benutzern) die Betriebsmittel zu. Es muß daher darüber wachen, daß ein Benutzer "sein Konto nicht überzieht" und es muß dafür sorgen, daß eine gerechte Abrechnung über die Betriebsmittel erfolgt. In Verbindung damit kann laufend in einer Benutzer- und Betriebsstatistik über die insgesamt vom System erbrachten Dienstleistungen Buch geführt werden. Die Gewinnung und Verarbeitung solcher "Abrechnungsinformation" ist ein Teil des notwendigen Verwaltungsaufwandes des Systems; die entsprechenden Programmstücke sind daher permanenter Bestandteil des Systems und während der gesamten Betriebszeit aktiv. Die Gewinnung und Auswertung solcher Informationen darf selbst nicht zuviel Betriebsmittel

./. 2

brauchen, hängt daher stark von der Hardwareunterstützung (und der Struktur des Betriebssystem) ab und ist in ihrem Umfang nach oben beschränkt. Die gesammelten Informationen sollten jedoch nicht zu dürftig sein: Die Abrechnung sollte einigermaßen fair sein und den Benutzer zu einem möglichst effektiven Einsatz der Betriebsmittel erziehen. Außerdem sollte die aus der Abrechnungsinformation gewonnene Betriebsstatistik eine hinreichende Kontrolle des Systemverhaltens ermöglichen, wenn das System soweit ausgebaut ist, daß keine wesentlichen Änderungen mehr vorgenommen werden. Dazu sollte die Abrechnungsinformation durch entsprechende Auswertungsprogramme möglichst weitgehend analysiert und in verständlicher Form ausgegeben werden. Beispiele für solche Abrechnungsinformation sind etwa: CPU-Zeit und Gesamtlaufzeit eines Programmes und der virtuelle Speicher des Programms, integriert über die CPU-Zeit und über die Gesamtlaufzeit (Treppenfunktion bei paginiertem Speicher!). Neben der Verwendung zur Abrechnung kann man solche Größen wie den Mittelwert der Laufzeit, der CPU-Zeit und den Mittelwert des (virtuellen) Speicherbedarfs benutzen, um abzuschätzen, wo die sinnvolle Maximalbelastung für die Anlage liegt, und diese mit der aktuellen Belastung vergleichen (vgl. [2]).

1.2 Regelung

Die unter 1.1 betrachteten Meßwerte werden (in aufbereiteter Form) in Abrechnung und Betriebsstatistik nach außen sichtbar. Im Gegensatz dazu werden die Informationen, die über die entsprechenden Algorithmen das Verhalten des Systems steuern ("Regelinformation") im allgemeinen nicht nach außen abgegeben. Der Bereich dieser Informationen reicht von 1-Bit-Größen (etwa für den paging-Algorithmus) bis zu Schätzwerten, etwa für die Antwortzeit an einer Konsole. Im letzterem Fall handelt es sich um eine Größe, die für jeden Benutzer interessant sein kann, der ein Wechselgespräch eröffnen will;

./. 3

insbesondere kann dieser Wert über die Zulassung des Gesprächs entscheiden. Die Gewinnung und Verwertung solcher Regelinformation gehört wesentlich zum System; es wäre jedoch nützlich, diese Informationen darüber hinaus auch zur Untersuchung des Systemverhaltens zugänglich zu machen (vgl. 1.4). Für die Gewinnung von Regelinformation ist man umso stärker auf die Unterstützung durch die Hardware angewiesen, je häufiger man darauf zugreifen muß (z.B. paging-Algorithmus). Die Regelung des Systems sollte durch möglichst einfache Algorithmen realisierbar sein. Trotzdem sollten die Regelmechanismen eine dem jeweiligen Zweck angepaßte "Reaktionszeit" haben, d.h. sie sollten zwar nicht zu träge arbeiten, aber auch nicht zu empfindlich auf mehr oder minder zufällige Schwankungen reagieren. Zu diesem Zweck erscheint es sinnvoll, Regelinformation bereits zu einer Form zu ermitteln oder - sofern dies mit vertretbarem Aufwand möglich ist - in eine Form überzuführen, in der sie ein geeignetes und nicht zu stark schwankendes Maß für die zu regelnde Größe darstellt.(Wegen der in diese Richtung ziehenden statistischen Methoden vgl. den Anhang.) Man kann im allgemeinen übersehen, wie ein einzelner Algorithmus arbeitet, der einen bestimmten Regelmechanismus realisiert. Dagegen läßt sich die gegenseitige Beeinflussung von Regelmechanismen nicht ohne weiteres durchschauen und man läuft daher Gefahr, in vielen Fällen "lokal zu optimieren". Das Ideal wäre, daß sich das Verhalten des Systems durch einige wenige charakteristische Maße beschreiben ließe; diese Maße könnte das System selbst laufend ermitteln und sich durch Änderung von Parametern, Strategien und (soweit möglich) des job-mix auf optimales Verhalten einregeln (wobei noch zu definieren wäre, was man unter "optimalem" Verhalten verstehen will).

1.3 Quantitative Untersuchungen im eigentlichen Sinn

Ist ein System bereits hinreichend getestet und soll nur noch geringen Modifikationen unterworfen werden, so dürfte im wesentlichen eine hinreichendausführliche Betriebsstatistik

./. 4

genügen, um die benötigten Aussagen über das Systemverhalten zu gewinnen. Wenn jedoch ein System wie das BSM neuentwickelt und von einer notwendigerweise primitiven 1. Ausbaustufe aus stufenweise erweitert und verbessert werden soll, so erscheint es zweifelhaft, ob die vorhandenen Informationsquellen und die "Erfahrung von Systemprogrammierern" eine ausreichende Entscheidungsgrundlage bilden. Es wäre daher wünschenswert, über begrenzte Zeiträume Messungen in größerem Umfang durchzuführen, um sich für Verbesserungen und Erweiterungen auf gesicherte quantitative Aussagen stützen zu können. Da solche Messungen zu einer Belastung des Systems führen, die kaum zum allgemeinen Verwaltungsaufwand des Systems gerechnet werden kann, werden die entsprechenden Programmstücke i.a. nicht permanenter Bestandteil an jeder Installation des Systems sein und (auch wegen der Menge der gelieferten Daten) nicht während der gesamten Betriebszeit arbeiten. Auch sollte der Aufwand für die Implementierung solcher Meßprogramme und der Programme zur Auswertung der gelieferten Daten nicht unterschätzt werden. Daher sollte bereits von der Struktur und von der Implementierung des Systems her dafür gesorgt werden, daß sich Meßgrößen möglichst einfach und in möglichst geeigneter Form ermitteln lassen. Als Beispiele für Problemkreise, die untersucht werden können, seien genannt: Verhalten der Hardware (z.B. Feststellung konfigurationsbedingter Engpässe, Überprüfung technischer Daten u.ä.), Verhalten der Software (z.B. Untersuchung der Häufigkeit des Aufrufs und des Zeitbedarfs bestimmter Moduln, Fehlersuche usw.) und das Verhalten der Benutzer (vor allem im Dialogbetrieb).

1.4 Allgemeine Gesichtspunkte

Es wurde bereits betont, daß es für den Ausbau und die Verbesserung eines Betriebssystems notwendig erscheint, gezielt durch spezielle Messungen die dazu benötigten quantitationen Aussagen zu erhalten. Die weiteren Darlegungen beziehen sich

./. 5

vorwiegend auf solche "Messungen im eigentlichen Sinn". Daß auf Abrechnung, Benutzer- und Betriebsstatistik und auf Fragen der Regelung in diesem Zusammenhang eingegangen wurde, hat folgende Gründe: Die laufend gesammelte Abrechnungsinformation ist nützlich als Ergänzung und zur Kontrolle weiterer Messungen. Weiterhin sollen einerseits die Messungen zeigen, auf Grund welcher Informationen und wie das System reagiert, es sollen also die Regelvorgänge im System sichtbar gemacht werden; anderseits erscheint es möglich und sinnvoll, Größen, die zunächst nur gemessen werden ("offene Schleife") später in geeignet aufbereiteter Form zur Steuerung des System zu verwenden (Rückkopplung).

2. Methoden und Aufwand

Es gibt im wesentlichen zwei Methoden, um Aussagen über das Verhalten eines Systems zu gewinnen. Im einen Fall ("analytische Methode") werden per Hardware oder Software die entsprechenden Stellen der Rechenanlage bzw. des Betriebssystems "angezapft" und dort werden die entsprechenden Meßwerte bgenommen. Im anderen Fall ("Stimulus-Methode") betrachtet man das System als "schwarzen Kasten" mit einer Anzahl bekannter Funktionen, aktiviert diese Funktionen durch entsprechende Eingaben und beobachtet die Reaktionen.

2.1 Analytische Methode

2.1.1 Voraussetzungen

Das Problem besteht darin, das System an einer Reihe von Stellen anzuzapfen, ohne es wesentlich in seiner Arbeit zu stören. Praktisch ohne Störung kann nur gemessen werden, wenn spezielle Hardware zur Verfügung steht. Diese Möglichkeit wird im folgenden nicht ausführlich diskutiert, weil von der Natur der Sache her und aus finanziellen Gründen hier für Untersuchungen relativ enge Grenzen gesetzt sind. Messen per

./. 6

Software bedeutet, daß das System an den entsprechenden Stellen modifiziert werden muß. Die Arbeit des Systems muß also an bestimmten Stellen oder Zeitpunkten unterbrochen werden und es muß u.U. auf bestimmte Informationen (z.B. Listen des Systems) zugegriffen werden. Das System sollte daher von seiner Struktur her auch solche Gesichtspunkte berücksichtigen, um die Durchführung von Messungen nicht unnötig zu erschweren. Insbesondere ist hier die Behandlung von Unterbrechungen von Interesse: Eine Messung (Registrierung eines Meßwerts, Hochzählen einer Zählgröße) kann an die Verarbeitung einer Unterbrechung gekoppelt sein, die im normalen Betriebsablauf auftritt oder es kann absichtlich und zusätzlich eine Unterbrechung verursacht werden, um eine Messung durchzuführen.

2.1.2 Durchführung der Messungen

Es wird im allgemeinen kaum möglich sein, während eines längeren Zeitraums an allen zugänglichen Stellen des Systems zu messen, weil das anfallende Material zu umfangreich wäre und das Verhalten des Systems entsprechend gestört würde. Es zeichnen sich folgende zwei extemen Vorgehensweisen ab: Im einen Fall benutzt man eine einheitliche Meßtechnik, alle gelieferten Daten haben dieselbe Struktur, die entsprechenden Programme bzw. Programmstücke, die diese Daten liefern, erscheinen nach außen als eine Einheit (vgl. 2.1.2.1); in diesem Fall muß es eine Möglichkeit geben, diese Einheit nur teilweise zu aktivieren. Im anderen Fall hat man von vornherein mehrere "Pakete", die in ihrer Meßtechnik und daher auch in der Form, in der sie ihre Ergebnisse liefern, unterschiedlich und auf spezielle Zwecke zugeschnitten sind (vgl. 2.1.2.2 - 2.1.2.6).

2.1.2.1 Generelles "Event Recording"

Will man das Verhalten des Systems so aufzeichnen, daß man möglichst viel Freiheit (allerdings auch Aufwand!) bei der

./. 7

Auswertung des Materials hat, so erscheint es zweckmäßig, den zeitlichen Ablauf als eine Folge von "Ereignissen" anzusehen. Das Problem besteht darin, eine geeignete Menge charakteristischer Ereignisse festzulegen (evtl. mit der Möglichkeit, diese Menge später in gewissen Grenzen zu erweitern). Das Eintreten dieser Ereignisse wird dann mit Angabe des Zeitpunkts und evtl. zusätzlicher Information festgehalten. Was als Ereignis definiert wird, hängt ab von der Hardware und Software des Systems und von den Schwerpunkten der durchgeführten Untersuchungen. Da selbst bei einer nicht allzugroßen Menge von Ereignissen das anfallende Datenmaterial sehr umfangreich werden kann, wird man in einem bestimmten Zeitraum nur Ereignisse registrieren, die zu einem bestimmten job und/oder zu einer bestimmten Gruppe von Ereignissen gehören. Ein solches generelles "event recording" gestattet eine Reihe verschiedener Untersuchungen am selben Datenmaterial (und die Verwendung diese Materials als Eingabe für Simulationsprogramme!), jedoch ergibt sich ein ziemlich hoher Aufwand für die Auswertungsprogramme (wegen eines Beispiels vgl. 3.1).
Die folgenden Methoden sind zwar weniger universell, erfordern jedoch auch weniger Aufwand, vor allem für die Auswertung.

2.1.2.2 "Sampling Measurement"

Interessiert man sich etwa für den Anteil einer bestimmten Routine an der CPU-Zeit, so ist es u.U. zu aufwendig, jeden Eintritt und jedes Verlassen dieser Routine zu registrieren. Man kann den entsprechenden Zeitanteil näherungsweise ermitteln, indem man Stichproben macht: Man unterbricht zu bestimmten Zeitpunkten und stelltfest, ob gerade ein Befehl aus der betreffenden Routine ausgeführt wurde. Solche Unterbrechungen können "zufällig" verteilt sein oder periodisch erfolgen (Wecker!), müssen jedoch (statistisch) unabhängig vom Eintreten des untersuchten Ereignisses sein.

./. 8

2.1.2.3 "Trace Measurement"

In vielen Fällen möchte man nicht das Verhalten des gesamten Systems in einer Folge von Ereignissen festhalten, sondern lediglich die "elementaren Schritte", die zu einem bestimmten Vorgang (z.B. Ausführung eines Kommandos) gehören, in ihrer zeitlichen Reihenfolge ermitteln. Man wird also alle elementaren Ereignisse spezifizieren, die zu dem zu untersuchenden Vorgang gehören oder gehören könnten und dann das zeitliche Auftreten dieser Ereignisse registrieren. Die Methode ist dazu geeignet, die Zusammenarbeit von Systemteilen (insbesondere auf Fehler) zu untersuchen (vgl. etwa 3.4.2).

2.1.2.4 "Accounting Measurement"

Ein wichtiger Bereich für Messungen ist der Verbrauch an Betriebsmitteln. Diese Messungen dienen vor allem der Abrechnung mit den Benutzern (vgl. 1.1). Sie können jedoch auch zu Untersuchungen über das Verhalten des Systems und über das Verhalten der Benutzer verwendet werden. Darüberhinaus sind einige dieser Größen auch für die Regelung des Systems interessant (vgl. 1.2 und den Anhang).

2.1.2.5 "Logical Measurement"

Diese Methode besteht darin, gezielt auf Grund des Inhalts und/oder der Lage einen Ausschnitt aus dem Speicher auszuwählen und zu verfolgen, wie sich der Inhalt dieses Ausschnitts ändert, sofern diese Änderungen nicht zu schnell erfolgen (vgl. 3.3.2).

2.1.2.6 "Playback Measurement"

Eine relativ selten angewandte, weil aufwendige Technik besteht darin, sich über einen bestimmten Zeitraum die Eingabeinformation ganz oder teilweise und mit Zeitangaben aufzuheben, um

unter den gleichen Voraussetzungen mit dem System oder mit Teilen des Systems zu experimentieren. Die Methode wurde praktisch nur im militärischen Bereich angewandt (SAGE air defense system). Sie kommt bereits in die Nähe der in 2.2 angegebenen Methoden. Weniger aufwendig ist es, die Eingabeinformation in stark reduzierter Form aufzuheben, um sie in Simulationsmodellen zu verwenden.

2.1.3 Sicherstellung und Verarbeitung der gesammelten Daten

Die Meßergebnisse fallen (sofern nicht mit spezieller Hardware gemessen wird) zunächst im Kernspeicher an und müssen auf einen Hintergrundspeicher hinreichender Kapazität (Band, Platte) transportiert werden, wozu man sie zurVerringerung des Umfangs evtl. bereits vorverarbeitet. Später erfolgt die eigentliche Aufbereitung und Verarbeitung der Daten. Es wird nicht nur Platz auf der Platte bzw. auf Magnetband benötigt, sondern es ergibt sich eine zusätzliche Belastung des Plattenbetriebs bzw. die Belegung mindestens eines Magnetbandgeräts und eine zusätzliche Belastung der Kanäle zu diesen Geräten. Man kann überlegen, ob man diese Anforderungen - allerdings auf Kosten zusätzlicher CPU-Zeit - vermindern will. In Frage kommt eine Kompression der Daten, etwa durch Umcodierung, Verwendung spezieller Codes für häufig registrierte Werte oder Umpacken der Daten, eine Vorauswahl der Daten durch Weglassen unnötig erscheinender Eintragungen oder auch bereits eine Vorverarbeitung wie etwa die Reduktion der Angabe von Anfangs- und Endzeitpunkten auf die Angabe von Zeitintervallen oder die Angabe von Mittelwerten statt der Werte von Einzelmessungen. In den meisten Fällen dürfte der erzielte Gewinn den Nachteil der höheren CPU-Belastung nicht rechtfertigen. Es erscheint sinnvoller, bereits beim Sammeln der Daten auf nicht zu großen Umfang des anfallenden Materials zu achten und wenn möglich zu versuchen, Leerzeiten der Kanäle zu benutzen, um den Transport mit dem übrigen Betrieb zu überlappen. Der Aufwand für

./.10

Messungen läßt sich nur rechtfertigen, wenn auch entsprechend leistungsfähige Programme für die Auswertung der etwa auf Band gespeicherten Daten verfügbar sind. Als Minimum ist zu fordern, daß die Information in lesbarer, verständlicher und übersichtlicher Form gedruckt werden kann. Für umfangreiche Datenmengen braucht man Programme, die aus den Daten eine Auswahl treffen (z.B. job-spezifisch) und/oder einfache Reduktionen der Daten durchführen. Weiterhin sollte es möglich sein, Zählungen durchzuführen und Histogramme zu erzeugen. Dies kann dann die Grundlage und ersten Anhaltspunkte für eine weitergehende statistische Untersuchung der Daten bilden. Man wird auch daran denken, die so gewonnenen Aussagen (etwa über Häufigkeitsverteilungen) und evtl. die Daten selbst in Simulationsmodellen zu verwenden.

2.2 Stimulus-Methode (Benchmark-Technik)

2.2.1 Grundsätzliches

Man kann ein System als schwarzen Kasten betrachten, von dem lediglich bekannt ist, daß er eine Reihe von Funktionen ausführen kann. Durch Anstoß ("Stimulus") von außen (Eingabe) wird dieser schwarze Kasten aktiviert, er reagiert (indem er die entsprechenden Funktionen ausführt) und gibt eine Antwort (Ausgabe). Von diesem Standpunkt aus betrachtet, ist das System gekennzeichnet durch einige wenige wesentliche Funktionen, die man richtig erkennen und verstehen muß. Man möchte soviel Einsicht in das Verhalten des Systems gewinnen, daß man das System in kontrollierbarer und meßbarer Weise so einregeln kann, daß es (bei im wesentlich vorgegebenen Benutzeranforderungen) möglichst effiziert arbeitet. Das Problem besteht vor allem darin, welche Funktionen des Systems man als wesentlich ansehen und bei den Untersuchungen berücksichtigen will. Dabei möchte man einerseits in soviele Funktionen aufgliedern, daß das System hinreichend genau charakterisiert ist, andererseits wird durch eine zu starke Detaillierung der Aufwand für die Untersuchung zu hoch

In einer solchen Menge der wesentlichen Funktionen sollten etwa folgende Tätigkeiten des Systems berücksichtigt werden: scheduling (Verwaltungsaufwand bei der Betriebsmittelzuteilung), paging bzw. swapping, sonstiger E/A-Verkehr (evtl. geeignet aufgeteilt), Rechnerkernbelastung, Belastung durch Konsolverkehr. Man muß entscheiden, wie und in welcher Zusammenstellung man die Funktionen anstoßen will. Zu diesem Zweck kann man Objektprogramme entwickeln, die in gewisser Weise "typische" Benutzer bestimmter Klassen repräsentieren. Solche "Benchmark-Programme" (im folgenden auch als "Normprogramme" oder "Belastungsprogramme" bezeichnet) können zur Gewinnung von Aussagen über das Systemverhalten benutzt werden, wobei man jedoch im wesentlichen nur Angaben über Durchsatz und Antwortzeiten erthält.

2.2.2 Meßtechniken

Es gibt drei Möglichkeiten, ein solches Normprogramm einzusetzen: allein im System, in einer Gruppe von Normprogrammen oder parallel zum normalen Betrieb. Ist ein Normprogramm allein im System, so erhält es den bestmöglichen Service. Die so erhaltenen Meßwerte sind ein Maßstab, mit dem man die Ergebnisse von Läufen unter anderen Bedingungen vergleichen kann. Daneben können spätere Läufe des Normprogramms allein im System objektive Aussagen über die Auswirkung von Systemänderungen liefern. Läßt man eine Gruppe von Normprogrammen laufen, so kann man eine bestimmte Gesamtheit von Benutzern simulieren. Damit hat man für den Test eines Systems reproduzierbare Versuchsbedingungen. Außerdem hat man die Möglichkeit, ein im Normalbetrieb noch schwach ausgelastetes System unter höherer Belastung zu testen. Schließlich kann man ein Normprogramm auch neben dem normalen Betrieb als "Pseudobenutzer" starten. Man wird über mehrere solche Läufe mitteln, um zufällige Schwankungen auszuschalten und dann diese Mittelwerte mit den Standardwerten für einen Lauf dieses Normprogramms allein im System vergleichen. So kann man z.B. Aussagen über Änderungen im Verhalten des Systems bzw. Änderungen in den Anforderungen der Benutzer gewinnen. (Beispiele siehe 3.3.3 und 3.4.3)

./. 12

2.3. Vergleich beider Methoden

Die in 2.1. und 2.2. skizzierten Methoden unterscheiden sich erheblich hinsichtlich des erforderlichen Entwicklungsaufwandes, des Verbrauchs an Betriebsmitteln, des Umfangs und Detaillierungsgrades der Meßergebnisse und hinsichtlich des Aufwandes für die Auswertung: Die analytische Methode erfordert einen relativ hohen Entwicklungsaufwand und eine genaue Kenntnis des Systems; außerdem können sich Programmierfehler auf den gesamten Betrieb auswirken. Neben der Erhöhung des Verwaltungsaufwandes während der Zeit der Messungen ist vor allem der Bedarf an Hintergrundspeicher zu berücksichtigen. Auch der Aufwand für die Auswertung der Messungen ist relativ hoch (evtl. Einsatz statistischer Methoden), dafür können umfangreiche und ins einzelne gehende Untersuchungen durchgeführt werden. Normprogramme können auch ohne Kenntnisse der Feinheiten des Systems geschrieben werden; Fehler haben im allgemeinen nur lokale Auswirkungen. Um eine Grundlage für Vergleiche zu haben, braucht man für jedes Normprogramm kurze Zeit allein das System; außerdem belegt man während der Messungen in vielen Fällen mindestens eine Konsole (vgl. 3.3.3). Die Meßwerte sind praktisch sofort und ohne größeren Auswertungsaufwand verwendbar; es können jedoch nicht alle Probleme und nicht alle Einzelheiten mit dieser Methode untersucht werden. Beide Methoden ergänzen sich vorteilhaft und sollten daher zweckmäßigerweise nebeneinander angewandt werden.

3. Implementierung von Meßmethoden, mögliche Untersuchungen (Beispiele)

In den vorangegangenen Abschnitten wurde nur gestreift, wie sich die angeführten Meßmethoden implementieren lassen und welche Untersuchungen mittels solcher Messungen durchgeführt werden können und sollen. Da diese Fragen stark von der zur Verfügung

./. 13

stehenden Hardware, von der Struktur und dem Verwendungszweck des Systems abhängen, erscheint es zweckmäßig, sie nicht allgemein zu diskutieren, sondern an einzelnen Beispielen zu erläutern.

3.1 MTS (Michigan Terminal System)

3.1.1. Bermerkungen zum System und zur Meßmethode

Das MTS läuft auf einer IBM 36o/67 mit zwei Rechnerkernen. Genauer gesagt besteht das System aus zwei Schichten, dem UMMPS (University of Michigan Multiprogramming Supervisor) als unterer Schicht und dem MTS "job program", das im wesentlichen erst Konsolverkehr und langfristige Datenhaltung ermöglicht. Zunächst sei angemerkt, daß durch permanente Bestandteile des Systems Abrechnungsinformation gesammelt wird und daß Programme existieren, um diese Informationen, aufgegliedert nach Stapel- und Dialogbetrieb, für die Untersuchung des Systems auszuwerten und in übersichtlicher Weise auszugeben. Das wichtigste Instrument für Messungen ist jedoch ein als "Data Collection Facility" (DCF) bezeichneter Komplex von Programmen. Die angewandte Methode ist das in 2.1.2.1 erwähnte generelle "event recording". Kritisch bei dieser Methode ist, ob es gelingt, einerseits das Verhalten eines komplexen Systems zu erfassen und andererseits durch die Messungen selbst keine zu großen Störungen des Verhaltens hervorzurufen. Das zwingt insbesondere dazu, möglichst viel Arbeit in die Auswertungsprogramme zu verlegen.

3.1.2 Bemerkungen zur Implementierung

Es werden eine Reihe von Ereignissen (im Supervisor) identifiziert. Das Auftreten eines solchen Ereignisses wird mittels Anruf eines Unterprogrammes im Supervisor (SVC) in einem Puffer im Kernspeicher registriert.

./. 14

Auch in ein Benutzerprogramm kann ein entsprechender SVC eingefügt werden. Die Puffer (3 x 1 Seite) werden von einem Programm, das wie ein Benutzerprogramm behandelt wird, auf Magnetband geschrieben. Die Information über ein Ereignis besteht aus einem standardisierten Kopf von 2 Worten (= 8 Byte) und 0 - 6 Worten zusätzlicher Information. Der Kopf enthält Angaben über die Länge der Zusatzinformation, die Nummer des Ereignisses, die Nummer des betreffenden jobs und den Zeitpunkt des Eintritts des Ereignisses. Etwa 25 solcher Ereignisse sind definiert und werden auch von den Auswertungsprogrammen berücksichtigt. Um die Belastung durch die Messungen zu vermindern, kann man für einen Zeitraum etwa nur Ereignisse registrieren lassen, die sich auf bestimmte jobs beziehen und/oder in eine bestimmte Gruppe von Ereignissen fallen. (Dieselbe Auswahlmöglichkeit besteht in den Auswertuhgsprogrammen). Wenn aus irgendeinem Grund (z.B. Fehler) so viele Ereignisse so rasch aufeinanderfolgen, daß nicht rechtzeitig ein freier Puffer zur Verfügung steht, so werden diese Ereignisse lediglich gezählt und die Zahl dieser Ereignisse (in Form eines Ereignisses) in den ersten freien Puffer gesetzt. Obwohl die Anzahl der registrierten Ereignisse relativ hoch ist (z.B. etwa 57o Ereignisse für ein Wechselgespräch, das nur aus den "signon- und "signoff"-Kommandos besteht) läßt sich (teilweise durch Addition der Ausführungszeiten von Befehlen, teilweise aus den von der DCF selbst gelieferten Daten) feststellen, daß die Belastung durch die Messungen in ziemlich engen Grenzen bleibt. Wenn man davon absieht, daß während der Dauer der Messungen 1 Magnetbandgerät belegt ist, so erscheint der sonstige Aufwand, nämlich die Belastung der Kanäle (etwa o,5%), der CPU (etwa o,3%) und des Kernspeichers (etwa 4%) durchaus vertretbar.

./. 15

3.1.3 Mögliche Untersuchungen

Die DCF wurde im Zusammenhang mit Untersuchungen über das Verhalten von Programmen in Systemen mit virtuellem Speicher konzipiert (vgl. [1]). Die Ereignisse stehen also vielfach in Verbindung mit der Speicherverwaltung (etwa Veranlassung und Abschluß eines Seitentransports von der Trommel in den Kernspeicher und umgekehrt), geben aber auch Änderungen im Zustand eines jobs bezüglich der CPU (z.B. Übergang in den Wartezustand) wieder. Außerdem kann man in einem beliebigen Programm durch Einfügen eines entsprechenden SVC ein (in gewissen Grenzen) beliebiges Ereignis registrieren lassen; ein spezielles solches Ereignis ist bereits definiert und wird auch in den Auswertungsprogrammen berücksichtigt. Die Auswertungsprogramme erbringen im wesentlichen die folgenden Leistungen:

a) Ausgabe der Daten in lesbarer und interpretierter Form;
b) Reduktion der Folge von Ereignissen auf eine Folge von Intervallen, etwa zwischen Zustandsübergängen für einen job, evtl. mit Zusatzinformation über den Grund des Übergangs;
c) Ermittlung von Häufigkeitsverteilungen (Histogramme) und Erzeugung graphischer Darstellungen über Speicherausnutzung, Wartezeiten u.ä.
d) Bereitstellung von Eingabedaten für Simulationsprogramme

An Problemen, die untersucht werden, seien genannt:

a) Fragen bezüglich des paging-Algorithmus, insbesondere, wie die für paging benutzte Trommel unter verschiedenen Belastungen und bei verschiedenen Strategien arbeitet. Da das System zur Zeit der Untersuchungen nur schwach ausgelastet war, wurde zusätzlich ein job eingeführt, der in rascher Folge auf seine (zahlreichen) Seiten zugreift und damit die Anzahl der page defaults bei den anderen jobs erhöht.

./. 16

b) Untersuchungen über die Länge von CPU-Intervallen, was insbesondere Aussagen über die Dauer von Unterbrechungsbehandlungen liefert. Die Messung der Leerzeit erfolgt, indem man einen fiktiven job Nr. 0 mittels der DCF untersucht.
c) Wartezeiten auf den Abschluß von E/A-Vorgängen, aufgeschlüsselt nach Geräten. (Wegen der zwei Prozessoren und der damit verbundenen zwei Uhren ergaben sich Probleme, wenn Anfang und Ende eines Vorgangs auf verschiedenen Prozessoren registriert wurden).

Abschließend sei erwähnt, daß die mittels der DCF ermittelten Daten auch zu mehreren Untersuchungen an Simulationsmodellen verwendet wurden, etwa für die Frage der Verwendung eines Massenkernspeichers statt der Trommel für paging oder die Frage nach einer fairen Abrechnungsformel für die Benutzer.

3.2 RTOS (Apollo Real-Time Operating System)

3.2.1 Bemerkungen zum System

Dieses System wurde entwickelt, um die notwendigen Berechnungen bei tatsächlichen und bei simulierten Raumflügen durchzuführen. Bei diesem System handelt es sich um eine Modifikation von OS/360 für einen Komplex von 5 IBM 360/75, ergänzt vor allem durch einen großen Massenkernspeicher. Gefordert war eine hohe Zuverlässigkeit des Systems und garantierte Höchstwerte für Antwortzeiten während bestimmter kritischer Phasen. Dazu wurden umfangreiche Untersuchungen teils am System selbst (durch Messungen während simulierter Raumflüge), teils an entsprechenden Simulationsmodellen durchgeführt. Infolge der speziellen Anwendung des Systems (Echtzeitbetrieb) stehen die Messungen unter einem anderen Aspekt als bei Systemen für den allgemeinen wissenschaftlich-technischen Rechenbetrieb: Ein System für Echtzeitbetrieb

./. 17

muß im Einsatz noch hinreichende Leistungsreserven haben, weshalb man den Aufwand für Messungen ziemlich hoch treiben kann. Die zusätzliche Belastung durch die Messungen führt lediglich zu einem langsameren Abklingen von Belastungsspitzen, so daß sich höchstens zu pessimistische Schätzungen, etwa für Antwortzeiten, ergeben.

3.2.2 Untersuchungen und Methoden

Die Messungen werden mittels eines als SGS (Statistics Gathering System) bezeichneten Komplexes von Programmen durchgeführt. Die dabei benutzten (analytischen) Techniken sind unterschiedlich; in vielen Fällen beschränkt man sich von vornherein darauf, Mittelwerte zu bestimmen. Das SGS soll vorallem Informationen über den zeitlichen Ablauf des Kontrollprogramms (Betriebssystem) und der Anwendungsprogramme, Angaben über die CPU-Belastung und Antwortzeiten für bestimmte "tasks" liefern. (In RTOS setzt sich jeder "job" aus mehreren "tasks" zusammen, das sind benannte Programmoduln, vor denen eine Warteschlange von Aufträgen gehalten wird; Antwortzeit ist also Wartezeit + Servicezeit). Auf Grund früherer Erfahrungen wurde das SGS in 6 Gruppen eingeteilt, die unabhängig voneinander eingesetzt werden können, um zu große Störungen des Systemverhaltens zu vermeiden. Diese Gruppen liefern:

a) Ausführungszeit und Häufigkeit des Aufrufs von Diensten und Routinen des Kontrollprogramms (RTOS/36o statistics);

b) Ausführungszeit und Häufigkeit des Aufrufs zuladbarer Programme, dazu Hinweise auf die von diesen Programmen aufgerufenen Dienste des Kontrollprogramms (load module statistics);

c) Prozentuale Anteile des Kontrollprogramms und der Anwendungsprogramme an der CPU-Zeit, Wartezeiten auf E/A, Leerzeit (gross CPU utilization statistics);

d) Häufigkeit des Aufrufs, Antwortzeiten und Betriebsmittelverbrauch für verschiedene tasks (independent-task statistics);

./. 18

e) Häufigkeiten, mit denen E/A-Geräte benutzt werden (I/O device statistics);

f) Ablauf der einzelnen Schritte beim Aufruf eines Dienstes des Kontrollprogramms durch ein Programm des Anwendungssystems (logic traces).

3.2.3 Bemerkungen zur Implementierung

Beim Einsatz des SGS sind drei Phasen zu unterscheiden: Initialisierung (der entsprechenden Teile) des SGS, Durchführung der Messungen und Transport der Daten zum Hintergrundspeicher. Bei der Initialisierung werden die benötigten Programme des SGS in den Hauptspeicher geladen. Die Durchführung der Messungen ist an den Unterbrechungsmechanismus gekoppelt. Bei einer Unterbrechung wird das "program status word" (PSW) gewechselt. Das neue PSW weist zunächst auf eine Adresse im SGS, und erst vom SGS aus wird die normale Unterbrechungsbehandlung angesprungen. Um auch Vorgänge registrieren zu können, die nicht mit einer Unterbrechung verbunden sind (etwa Ansprung einiger spezieller Unterprogramme) bedient man sich des folgenden Kunstgriffs: Bei der Initialisierung des SGS wird an den entsprechenden Stellen ein unzulässiger Befehlscode eingesetzt, wodurch eine Unterbrechung und damit ein Sprung in das SGS erzwungen wird. Im SGS wird dann (mittels einer Tabelle) die Ausführung des richtigen Befehls veranlaßt. Um auch das Verlassen eines Unterprogramms registrieren zu können, kann man das für die Rücksprunginformation verwendete Register (general register 14) bereits beim Ansprung des Unterprogramms so umbesetzen, daß der Rücksprung zunächst ins SGS erfolgt. Für den periodischen Transport der Daten aus den Puffern auf Magnetband sorgt ein normales Programm (task), dessen Priorität jeweils geeignet angepaßt wird. Sammeln von Meßwerten und Leeren der Puffer wechseln ab, wobei mit dem Leeren der Puffer jeweils eine Nullstellung der Zählgrößen usw.

./. 19

verbunden ist. Da bereits weitgehend Mittelwerte registriert werden, entsteht kein allzu großer Aufwand bei der Auswertung der auf Band gespeicherten Daten.

3.3 Multics (Multiplexed Information and Computing System)

3.3.1 Bemerkungen zum System

Multics läuft auf einer GE 645. Da Multics als Forschungsprojekt eine Reihe neuer Ideen und neuer Kombinationen alter Ideen enthält, mußte bei der Planung des Systems eine größere Anzahl von Entscheidungen über Strategien, Algorithmen, Parameter usw. getroffen werden, für deren Richtigkeit und Zweckmäßigkeit man nicht garantieren konnte. Von Anfang an wurde daher Wert auf Messungen gelegt, insbesondere wegen der Probleme im Zusammenhang mit Multiprogramming und mit dem Konzept des virtuellen Speichers. Von den Hardware-Voraussetzungen, die die Durchführung von Messungen erleichtern, ist vor allem zu erwähnen: Die GE 645 besitzt eine Uhr (ein 52-Bit-Register, das durch einen Quarzoszillator kontrolliert, jede Mikrosekunde um 1 erhöht wird); dieses Uhrregister kann praktisch wie eine Speicherbank angesehen und mit einem Maschinenbefehl wie eine doppeltgenaue Festkommazahl gelesen werden. Es gibt daher auch bei mehreren Prozessoren keine Probleme, wenn Anfang und Ende eines Vorgangs von verschiedenen Prozessoren registriert werden. Daneben besitzt jeder Prozessor der GE 645 einen Zähler für Speicherzugriffe. Außerdem existiert ein E/A-Kanal, der es gestattet, von einem anderen Rechner aus jeweils einen Ausschnitt aus dem Kernspeicher der GE 645 anzusehen.

./. 2o

Anwendungen der analytischen Methode

In Multics wird kein generelles "event recording" betrieben, weil angenommen wurde, daß das Verhalten des Systems für ein solches Vorgehen zu komplex sei. Statt dessen besitzt Multics einige "Pakete" von Meßprogrammen für verschiedene Untersuchungen. Diese Programme sind ständiger Bestandteil des Systems und jederzeit aufrufbar. Ein solcher Programmkomplex gestattet es, zu ermitteln, wie oft bestimmte Systemdienste aufgerufen werden und welcher Anteil der CPU-Zeit auf den betreffenden Dienst insgesamt entfällt. Probleme für die Zeitmessung ergeben sich aus dem Multiprogramming und aus der Tatsache, daß Systemdienste andere Systemdienste, einschließlich sich selbst, aufrufen können. Beispiele für Systemdienste sind der "missing-page handler" und der "missing-segment handler". (Untersuchungen in dieser Richtung führten zur Abschaffung der "Kleinseiten" in Multics!) Ein weiteres Meßprogramm verursacht periodisch (alle 1o Millisek.) eine Unterbrechung und ermittelt, welches Segment (also welches Benutzerprogramm oder welcher analog organisierte Systemteil) gerade ausgeführt wurde. Schließlich besteht die Möglichkeit, die oben erwähnten Hilfsmittel zur Untersuchung bestimmter (Benutzer-) Programme einzusetzen, indem man die entsprechenden Messungen nur dann ausführt, wenn das betreffende Programm an der Regie ist. Der bereits erwähnte spezielle E/A-Kanal wird vom "Graphical Display Monitor", einem Programmkomplex für eine PDP-8/338 benutzt. Damit kann man sich über einen längeren Zeitraum und ohne nennenswerte Störung des zentralen Rechners (insbesondere der Speicheransteuerung) einen Bereich aus dem Kernspeicher, etwa eine Datenbasis, auf einem Sichtgerät ansehen. Der Inhalt dieses Bereiches darf sich jedoch nur relativ langsam ändern. Zwei weitere Hilfsmittel dienen dazu, den Programmierer zu effektiverer Arbeit zu erziehen. Das erste besteht darin, daß der "command language interpreter" nach der Entschlüsselung eines Kommandos zunächst 3 Zahlen ausgibt: die Tageszeit, den Verbrauch an CPU-Zeit und die Anzahl der page defaults seit der letzten Ausgabe.

./. 21

Dadurch kann ein Programmierer feststellen, welchen Service sein Programm erhält und wie sich Änderungen in seinem Programm, im System oder in der Belastung durch die anderen Benutzer auswirken. Außerdem wird ein Benutzer bei Vorhandensein mehrerer Programme für den gleichen Zweck dazu gebracht, dasjenige zu verwenden, das weniger Betriebsmittel braucht und daher billiger ist. Das zweite Hilfsmittel besteht darin, daß für einen vorgegebenen Prozeß die Seitennummern der jeweils letzten 256 page defaults festgehalten werden können. Durch beide Hilfsmittel wird ein Programmierer dazu angehalten, seinen virtuellen Speicher nicht wie einen tatsächlich vorhandenen Kernspeicher entsprechender Größe zu behandeln. Es sei noch erwähnt, daß in Multics, vor allem im scheduler, eine ziemliche Menge von Infommationen (Regelinformation) gesammelt wird, die nicht nach außen weitergegeben wird. (Wegen der Steuerung des Konsolverkehrs vgl. Anhang).

3.3.3 Anwendungen der Stimulus-Methode (Benchmark-Technik)

Das einfachste Beispiel dafür ist ein Programm, das lediglich eine Schleife mit Uhrabfrage durchläuft: Wenn dieses Programm nicht unterbrochen wird, so ist die Zeitdifferenz zwischen zwei Uhrabfragen durch die Ausführungszeit der Befehle (bis auf Ungenauigkeiten beim Ablesen der Uhr) bestimmt. Läuft dieses Programm jedoch im normalen Betrieb, so ergeben sich auch Zeitdifferenzen, die deutlich höher liegen, weil der Rechnerkern zwischen zwei Uhrabfragen zur Behandlung einer Unterbrechung oder für einen anderen Prozeß abgezogen wurde. Auf diese Weise kommt man zu Aussagen über die Unterbrechungsbehandlung bzw. die Rechnerkernvergabe. Weiterhin existieren zwei Möglichkeiten, "typische" Benutzer zu simulieren. Das erste Projekt ist ein Programm für einen PDP-8-Rechner, der über Telefonleitungen mit der GE-645 verbunden ist und 1-12 typische Fortran-Benutzer simuliert. Da durch die hohe Belastung der Leitungen eine Grenze für die Anzahl der auf diese Weise simulierten Benutzer besteht, wurde

./.22

ein "interner Benutzer-Simulator" implementiert. Dieser Simulator gibt zwar nicht exakt die tatsächlichen Verhältnisse wieder, insbesondere arbeitet er mit files auf den Hintergrundspeichern statt mit Konsolen, aber er gestattet es, eine große Zahl von Benutzern mit vertretbarem Aufwand zu simulieren.

3.4 ADEPT

3.4.1 Bemerkungen zum System

ADEPT ist ein Timesharing-System, das im Zusammenhang mit dem Projekt ADP (Advanced Development Prototype) von der SDC (System Development Corporation) entwickelt wurde. Es läuft zur Zeit auf einigem IBM 36o/5o, soll jedoch größeren Modellen angepaßt werden können. Da dieses System auf die Verarbeitung geschützter Informationen zugeschnitten ist, war eine hohe Zuverlässigkeit, insbesondere einwandfrei arbeitende Zugriffskontrollen, notwendig.

3.4.2 Anwendungen der analytischen Methode

Die erste Anwendung besteht im wesentlichen darin, daß Eingriffe (mit Zeitangabe und zusätzlicher Information) registriert werden. Wegen der Häufigkeit des Auftretens dieser Ereignisse soll das Registrieren möglichst wenig Zeit kosten. Zu diesem Zweck wird kein Unterprogrammaufruf benutzt, sondern Code an den entsprechenden Stellen eingefügt; außerdem wird zunächst ein Puffer in Kachel 0 (adressierbar ohne Zuhilfenahme eines Basisregisters) verwendet. Der Inhalt dieses Puffers gelangt über mehrere Stufen auf den Hintergrundspeicher, wo er bis zur Vorarbeitung bleibt (Kachel 0 → "normale" Kachel → Trommel → Platte). Eine zweite Gruppe von Messungen befaßt sich mit dem Zugriff auf geschützte files. Es werden alle Ereignisse registriert, die im Zusammenhang mit einem solchen Zugriff stehen (oder stehen könnten).

./. 23

Es wird ein Unterprogramm verwendet, das die Information in einen Puffer im Kernspeicher absetzt, dessen Inhalt bei Bedarf auf die Platte transportiert wird. Die Auswertung besteht in der Aufstellung einer zeitlich geordneten Liste der Ereignisse.

3.4.3 Anwendungen der Stimulus-Methode (Benchmark-Technik)

Für ADEPT wurden 7 Benchmark-Programme geschrieben. Diese Programme stellen nicht nur verschiedene Betriebsmittelanforderungen, die typisch für bestimmte Benutzerklassen sind, sondern liefern auch selbst Meßergebnisse über ihren Ablauf. Ein solches Programm wird von einer Konsole aus gestartet. Dabei sind Startzeit und Zeit für den Abbruch des Programmlaufs anzugeben. Nach den Meßwerten, die während des Laufs geliefert werden, erscheint eine Zusammenfassung der Messungen mit Angaben etwa über den Anteil der CPU-Zeit an der Laufzeit des Programms oder über die Zahl der ausgegebenen Zeilen pro Minute.

Die Verwendung des exponentiellen Mittelwertes bei der Aufbereitung von Meßwerten.

1. Vorbemerkung

Bei der Untersuchung des Systemverhaltens treten in vielen Fällen Reihen von Meßwerten auf, die mit mehr oder minder starken zufälligen Schwankungen behaftet sind. Diese Meßwerte können zu bestimmten Zeitpunkten gewonnen werden, die entweder im gleichen Zeitabstand liegen oder mit dem Auftreten bestimmter Ereignisse verbunden sind. Um diese Meßwerte zur Regelung des Systems benutzen zu können, muß man den Einfluß dieser Schwankungen vermindern. Am geeignetsten für die Regelung ist es, wenn man aus den Meßwerten in der Vergangenheit einen Schätzwert für die zu messende Größe für den nächsten Zeitpunkt bestimmt. Es gibt eine Reihe von Möglichkeiten, solche Schätzwerte zu ermitteln.

2. Mathematische Theorie

Als Schätzwert kann man das arithmetische Mittel aus den jeweils letzten M (z.B. M = 6) Meßwerten benutzen ("gleitendes Mittel"). Dazu muß man jedoch diese M Werte speichern.

Dieser Nachteil wird vermieden, wenn man als Schätzwert den sog. exponentiellen Mittelwert benutzt.

Nehmen wir an, daß wir eine Größe X in den Zeitpunkten $t+k\,\Delta t$ $(k=0, \ldots,n)$ messen und die entsprechenden Meßwerte $x_k (k=0, \ldots,n)$ gewinnen. Dann wird der Schätzwert $\hat{x}_{n+1}$ für x_{n+1} nach folgender Formel bestimmt:

$$\hat{x}_{n+1}=S_n(x)=ax_n+(1-a)S_{n-1}(x);\quad o<a<1$$

$$S_o(x)=x_o$$

./. 2

Der Koeffizient a gibt an, wie stark die Vergangenheit berücksichtigt werden soll. Gewöhnlich wird a zwischen 0,01 und 0,3 gewählt; dann werden im wesentlichen die 6 bis 200 letzten Meßwerte berücksichtigt. Das Gewicht der Meßwerte in der Vergangenheit vermindert sich mit dem Zeitabstand wie eine geometrische Reihe mit dem Quotienten (1-a). Das ist leicht zu sehen, wenn obige Formel in die folgende Form umgeschrieben wird:

$$S_n(x) = a \sum_{k=o}^{n} (1-a)^k x_{n-k} + (1-a)^n x_o$$

Der Vorteil des exponentiellen Mittelwertes ist, daß seine Bestimmung einfach ist und daß man dazu nur 2 Werte, also wenig Speicherplatz benötigt.

Der Name "exponentieller Mittelwert" rührt daher, daß bei dem Analogon zu obiger Formel im kontinuierlichen Fall die Exponentialfunktion auftritt.

3. Verwendung des exponentiellen Mittelwertes in der Regelung des Konsolverkehrs (Multics, CTSS).

Beim Konsolbetrieb bemüht man sich, Antwortzeiten zu erreichen, die nicht zu lang sind.

In Multics tut man zu diesem Zweck folgendes: Man registriert periodisch die Anzahl N der Konsoljobs in der CPU-Warteschlange mit der höchsten Priorität und bestimmt eine Größe I nach der Formel:

I: = I*m + N; $0 < m < 1$, Anfangswert I: = 0, die einfach zu berechnen ist (eine Addition und eine Multiplikation, die bei geeigneter Wahl von m durch shiften erfolgen kann).

../. 3

Aus dem Wert I kann man den exponentiellen Mittelwert $\bar{N} = I/(1-m)$ bestimmen. Wie man leicht sieht, gilt:

$$\bar{N} := \bar{N}*m + (1-m) * N \quad ; 0 \leq m < 1.$$

In Multics wird der Wert I benutzt, um den Umfang der "eligible set" zu kontrollieren, d.h. der Menge der Programme, zwischen denen die Rechnerkernvergabe zu diesem Zeitpunkt auswählen kann.

Multipliziert man den Mittelwert der aktiven Konsoljobs mit der mittleren "Servicezeit" in der entsprechenden Warteschlange, so erhält man einen Schätzwert für die Antwortzeit. In CTSS wird dieser Schätzwert für die Antwortzeit zur dynamischen Kontrolle der Zahl der Benutzer, die ein Wechselgespräch eröffnen dürfen, verwendet.

4. Weitere Anwendungen

In manchen Betriebssystemen hat man vor der CPU Warteschlangen mit verschiedenen Prioritäten. Man kann nun versuchen, die Prioritäten der einzelnen Programme so festzusetzen, daß die Ausnutzung der CPU insgesamt möglichst gut ist. Dazu ist es sinnvoll, E/A-intensiven Programmen höhere und rechenintensiven Programmen niedrige Priorität zu geben. Da sich das Verhalten eines Programms während des Laufes ändern kann, scheint es zweckmäßig, auch seine Priorität zu ändern.

Zu diesem Zweck mißt man für die einzelnen Programme jeweils während eines bestimmten Zeitintervalles die CPU-Zeit und die Kanalbelegungszeit. Aus diesen Werten bestimmt man jeweils den exponentiellen Mittelwert. Das Verhältnis dieser Mittelwerte benutzt man, um die Prioritäten - evt. innerhalb vorgegebener Grenzen - neu festzusetzen.

./. 4

Diese Art der Prioritätsfestsetzung ist auf den Rechenanlagen B 5500 und CDC 6600 implementiert worden; in beiden Fällen ist CPU-Auslastung merklich vergrößert worden (20-30 %).

Außer der CPU-Zeit und der Kanalbelegungszeit kann man auch entsprechende Maße für die Speicherausnutzung festlegen. Alle Maße können sowohl für die einzelnen Programme als auch für das gesamte System bestimmt werden. Man könnte die exponentiellen Mittelwerte dieser Maße geeignet normieren und zur Steuerung des job - mix (so weit möglich) verwenden, um eine optimale Ausnutzung der Betriebsmittel zu erreichen (vgl. 8).

Literaturhinweise

[1] Pinkerton, T.B., Program Behavior and Control in Virtual Storage Computer Systems
University of Michigan, CONCOMP Report 4, April 1968

[2] Pinkerton, T.B., Performance Monitoring in a Time-Sharing System
CACM, November 1969

[3] Stanley, W.I. and Hertel, H.F., Statistics gathering and simulation for the Apollo real-time operating system
IBM Systems Journal, Volume 7, Number 2, 1968

ACM, Second Symposium on Operating Systems Principles, Princeton University, October 1969:

[4] Arden, B.W. and Boettner, D., Measurement and Performance of a Multiprogramming System

[5] Bryan, G.E. and Shemer, I.E., The UTS Time Sharing System: Performance Analysis and Instrumentation

[6] Karush, A.D., Two Approaches for Measuring the Performance of Time Sharing Systems

[7] Saltzer, I.H. and Gintell, I.W., The Instrumentation of Multics

[8] Wulf, W.A., Performance Monitors for Multiprogramming Systems