Frank Golatowski

Performance Metrics für Echtzeitbetriebssysteme*

Die Angabe von Leistungsparametern ist für Echtzeit-Betriebssysteme von entscheidender Bedeutung, da diese Systeme häufig in sicherheitskritischen Umgebungen eingesetzt werden und folglich das Echtzeitsystem über ein deterministisches Gesamtverhalten verfügen muß. Bei der Vielzahl der angebotenen Systeme hat sich bis heute keine Standardmethode zur Leistungsevaluierung durchgesetzt. Im Vortrag werden Performance-Meßmethoden und Kennzahlen vorgestellt und es wird herausgestellt inwieweit diese Methoden eine Aussagefähigkeit über die Leistung von Echtzeit-Betriebssystemen ermöglichen. Am Beispiel von drei Echtzeitbetriebssystemen (Lynx, RMOS, SORIX) werden Ergebnisse zu diesen Meßmethoden vorgestellt.
Echtzeitbetriebssysteme zeichnen sich durch besondere Eigenschaften aus, die sie von der Klasse universeller und Standardbetriebssysteme unterscheidet. Diese betreffen insbesondere das Interruptsystem, die Zeitsteuerung, das Scheduling, das Exceptionhandling und die Fehlertoleranz.
Eine mögliche Methode Aussagen über die Leistungsfähigkeit eines Echtzeitbetriebssystems zu gewinnen, ist das Ausführen von Benchmarks.

Klassifikation von Echtzeit-Benchmarks

Benchmarks, die zur Leistungsbewertung von Echtzeitbetriebssystemen eingesetzt werden, lassen sich in drei Kategorien einordnen:

1. feinkörnige Benchmarks [1] , [2]

2. applikationsorientierte Benchmarks [3] , [4]

3. simulationsbasierende Benchmarks [5]

Darüber hinaus sind auch Kombinationen dieser Testmethoden möglich.

Feinkörnige Benchmarks

Bei der Leistungsbewertung auf der Basis *feinkörniger* Test- und Analysemethoden werden Absolutzeiten ermittelt.
Der bekannteste feinkörnige Benchmark im Echtzeitbereich ist der Rhealstone. Im Rhealstone-Benchmark werden sechs Zeiten bestimmt, die die Echtzeiteigenschaften eines Betriebssystems charakterisieren (s. Abbildung 1). Der entscheidende Nutzen den dieser Benchmark bringt, ist die Definition dieser Zeiten. Aus den ermittelten Werten kann die eigentliche „Rhealstone Performance Number“ durch Addition der einzelnen Werten ermittelt werden. Der so gewonnene Wert hat aber nur theoretische Bedeutung, da die Aussagekraft der einzelnen Parameter größer ist.

Im Institut für Technische Informatik wurde dieser Benchmark für die Betriebssysteme SORIX, Lynx-OS und RMOS implementiert. Ausgangspunkt unserer Untersuchungen ist eine identische Hardware-Plattform für unterschiedliche Echtzeitbetriebssysteme. Deshalb wurden diese Betriebssysteme auf der gleichen Hardware-Plattform installiert. Es wurde großer Wert auf Portabilität gelegt, und es sollte keine zusätzliche Hardware Verwendung finden. [6]

* Gekürzte Fassung In: Lampe, B. (Hrsg): 8. Symposium Maritime Elektronik, Arbeitskreis Maritime Energie- u.Steuerungstechnik, Kongreßvorträge, Rostock, 19.-21.4.1995, S.51-55

- Prozessor: INTEL 80486, Taktfrequenz: 33 MHz, 2nd Level Cache: 256 KBytes, SiS-Chipsatz, SCSI
- Hauptspeichergröße: 16 MBytes

Rhealstone Metric

1. Task switching time t_{TS}
2. Preemption Time t_P
3. Interrupt latency time t_{IL}
4. Semaphore shuffling time t_{SS}: Zeit, vom Freigeben eines Semaphors durch einen Prozeß bis zum Neubelegendurch einen anderen.
5. Deadlock breaking time t_{DB}:Zeit, die benötigt wird, um einen Deadlock zu lösen
6. Intertask Message time t_{IM} :Zeitdauer für die Übertragung eines Message

Rhealstone-Wert

$R = f_1 + f_2 + f_3 + f_4 + f_5 + f_6$

Gewichteter Rhealstone-Wert

$R_W = c_1f_1 + c_2f_2 + c_3f_3 + c_4f_4 + c_5f_5 + c_6f_6$

Abbildung 1: Die Rhealstone-Charakteristik

	Interrupt Latency Time	*Task Switch Time*	*Preemption Time*	*Intertask Message Latency Time*	*Semaphor Shuffling Time*	*Deadlock Breaking Time*
RMOS3-PC1	10,1	4,0	4,1	5,8	4,4	6,2
RMOS3 für Windows	31,1	23,0	23,4	32,0	23,0	29,0
Lynx	17,6	27,0	105,0	337,0	113,5	n.e.
Unix SV Rel. 4 (SORIX)	n.e	137,0	480,0	755,6	450,0	n.e

Tabelle 1: Ergebnisse des Rhealstone- Benchmarks für vier Echtzeitbetriebssysteme (Werte in µs)

Es ist ersichtlich, daß das proprietäre Echtzeitbetriebssystem RMOS3-PC1 die besten Zeiten erreicht. Die dort ermittelten Werte stehen in einem ausgewogenen Verhätnis zueinander. Wenn RMOS3 in einer MS-Windows-Umgebung läuft, verschlechtern sich die einzelnen Werte um das 3- bis 6-fache. Mit dem System Lynx steht ein leistungsfähiges Echtzeit-Unix-Betriebssystem zur Verfügung, das allerdings bezüglich der gemessenen Zeiten nur für die Interruptlatenz und die Taskwechselzeit in den Bereich der RMOS3 für Windows-Zeiten kommt.
Die Angabe der ermittelten Werte ermöglicht nur eine eingeschränkte Aussagefähigkeit über die Leistungsfähigkeit in sicherheitskritischen Systemen, da mit diesem Benchmark nur die Bestimmung durchschnittlicher Werte möglich ist. Darüber hinaus können auf der Basis dieser Werte keine Aussagen zum Verhalten des Systems unter Lastbedingungen gemacht werden.
Zusammenfassend ergeben sich folgende Nachteile beim Einsatz feinkörniger Benchmarks:

1. Bei universellen Benchmarks werden in der Regel durchschnittliche Werte ermittelt. Das ist dem Fakt geschuldet, daß die implementierten Zeitmeßfunktionen eines Betriebssystems nicht hinreichend hochauflösend sind. Von besonderem Interesse sind aber Worst-Case-Werte. Ermittlung von Worst-Case-Werten mittels Software gestaltet sich jedoch aus diesem Grund als schwierig.

Eine genaue Methode zur Bestimmung relevanter Zeiten ist mit Hardware-Unterstützung möglich: Es werden Analyser, Hardware-bzw. Hybrid-Monitore verwendet. Der Einsatz ist aber sehr aufwendig und wenig portabel.

2. Selbst bei Ermittlung von Worst-Case-Zeiten mit Hardware-Unterstützung ist die Frage des Einflusses der Last auf diese Zeiten nicht geklärt. Die Angabe bestimmter Zeitwerte ermöglicht zwar eine grobe Einschätzung der Leistungsfähigkeit eines Betriebssystems, das Verhalten unter bestimmten Lastsituationen ist jedoch nicht geklärt.

Applikationsorientierte Bewertung

Ein interessanter Ansatz, um das Gesamtverhalten eines Echtzeitsystems bzw. Echtzeitbetriebssystems zu analysieren auch unter Berücksichtigung verschiedener Lastkomponenten, stellt der Hartstone-Benchmark [4] dar. Dieser Benchmark wurde an der Carnegie Mellon Universität entwickelt. Der Hartstone-Benchmark zielt auf die *Definition* von Anforderungen im Echtzeitumfeld. Das HUB- (Hartstone Uniprocessor Benchmark) Modell betrachtet ein Real-Time-System als einen Satz von periodischen, aperiodischen (sporadischen) und Synchronisations-(Server-) Tasks.
Der Benchmark besteht aus fünf unterschiedlichen Testserien zu denen mehrere Experimente gehören. Ausgangspunkt einer Testserie ist ein Basistasksatz (s. Tabelle 2), der charakterisiert ist durch die Anzahl der Tasks, die Priorität der Tasks, deren Ankunftsrate, deren Arbeitslast, deren Ausführungszeit und deren Deadlines.
Die verschiedenen Experimente, die aus dem Rate- Monotonic und Earliest Deadline First Algorithmus abgeleitet sind, verwenden ein Sammlung von Tasks, die aus

(1) nur periodischen Tasks mit harmonischen Perioden,
(2) nur periodischen Tasks mit nichtharmonischen Perioden,
(3) sowohl periodischen als auch aperiodischen Tasks,
(4) periodischen harmonische Tasks und einer Synchronisationstask, mit der die anderen Tasks sich zu bestimmten Intervallen synchronisieren, und
(5) einer Kombination von periodischen, aperiodischen und Synchronisationstasks bestehen.

Die Experimente in diesem Benchmark werden so ausgeführt, daß bestimmte Prozeßkenngrößen schrittweise erhöht werden, während die anderen konstant gehalten werden. Der Abbruch einer Testreihe erfolgt, wenn eine bestimmte Anzahl an Endterminen (Deadlines) nicht mehr eingehalten werden kann bzw. wenn die Dauer einer Testzeit überschritten wird. Die Benchmarks sind nützlich zur Evaluierung des Gesamtsystems.
An unserem Institut wurde eine C-Implementierung des Hartstone-Benchmarks realisiert. Ziel dieser Implementation ist die Untersuchung kommerzieller Echtzeit-UNIX-Betriebssysteme. (Die Beispiel-Implementierung in ADA des Software Engineering Institute zielt auf die Evaluierung von Targetplattformen [3]). Im weiteren sollen Ergebnisse dargestellt werden, die mit dem Echtzeit-UNIX-Betriebssystem SORIX ermittelt wurden. Entsprechend dem Grundkonzept der einzelnen Experimente wird eine analysierbare synthetische Arbeitslast ausgeführt.

Eigenschaften der zugrundeliegenden Arbeitslast

Jeder periodische Prozeß muß als Reaktion auf ein Ereignis eine bestimmte Arbeit verrichten. Dafür wurde eine synthetische Arbeitslast gewählt. Es handelt sich dabei um eine Variante des bekannten Whetstone-Benchmarks, bei der auf einige Berechnungsmodule verzichtet wird (Small-Whetstone). Durch die damit verbundene geringere Anzahl von Operationen kann die Arbeitslast eines Prozesses feiner eingestellt werden. Um eine möglichst allgemeingültige Last zu erhalten, sind die Operationen aus den folgenden repräsentativen Teilgebieten zusammengestellt worden:

- Ganzzahlige Berechnungen (Addition, Subtraktion, Multiplikation und Division),
- Gleitkommaberechnungen (Kosinus-, Logarithmus-, Exponential- und Wurzelfunktionen),
- Funktionsaufrufe,
- Feldzugriffe,
- Zeigeroperationen.

Der Small- Whetstone ist dadurch charakterisiert, daß in jedem Durchlauf etwa 1000 Operationen ausgeführt werden, wofür im weiteren die Abkürzung 1 KWI verwendet wird (**K**ilo-**W**hetstone-**I**nstructions). Die von einem periodischen Prozeß je Periode zu verrichtende Arbeit U_{KWIPP} wird dementsprechend in KWIPP ausgedrückt (**K**ilo-**W**hetstone-**I**nstructions **p**er **P**eriod), die je Sekunde zu verrichtende Arbeit U_{KWIPS} in KWIPS (**K**ilo-**W**hetstone-**I**nstructions **p**er **S**econd):

$$U_{KWIPS} = f * U_{KWIPP} = \frac{1}{T} * U_{KWIPP}$$

Dabei ist T die Ausführungsperiode des Prozesses und f die Frequenz, mit der die Ereignisse eintreffen.

Beschreibung des Experimentes PH-1

Am Beispiel des Experimentes PH-1 der PH- Serie des Benchmarks soll auf die Vorgehensweise bei der Leistungsevaluierung von Echtzeit-Unix-Systemen eingegangen werden. Die PH- Testserie besteht aus einem Basistasksatz von 5 periodischen Tasks, deren Taskfrequenzen ein ganzzahliges Vielfaches der Frequenz jeder niederfrequenten Task sind- die Frequenzen stehen in einem harmonischen Verhältnis zueinander. Die Task sind unabhängig, d.h. es erfolgt keine Synchronisation und keine Kommunikation zwischen ihnen. Tasks sind konkurrierend und bewerben sich um Prozessorzeit. Jede Task ist gekennzeichent durch deren Frequenz (Periode), Bearbeitungszeit, Deadline und Arbeitslast. Jede Task führt bei Ihrer Aktivierung jeweils einmal die Basislast von einem KWIPP aus. Ausgehend vom Basistasksatz (s. Tabelle 2) wird bei jedem Schritt die Frequenz des 5. Prozesses um den Betrag der Frequenz des 4. Prozesses erhöht. Die anderen Kenngrößen des Basissatzes bleiben konstant. Mit der Erhöhung seiner Frequenz führt diese Task im Laufe des Experimentes eine immer höhere Arbeitslast aus. Der Test wird beendet, wenn eine bestimmte Zahl von Endterminen verfehlt bzw. überschritten wurde. Dabei gilt ein Endtermin (Deadline) als verfehlt, wenn die abzuarbeitende Routine nicht bis zur erneuten Aktivierung des Prozesses abgearbeitet wird, d.h. hier gilt die Annahme Endtermin = Periode. Da die Frequenz des 5. Prozesses dabei relativ groß wird, kann man mit dieser Testreihe Informationen darüber gewinnen, wie gut das System in der Lage ist, schnell zwischen verschiedenen Prozessen umzuschalten. Aus den Ergebnissen dieser Reihe lassen sich deshalb unter anderem auch Aussagen über die Auflösung der zugrundeliegenden Zeitbasis gewinnen.

Prozeß-Nummer	Startzeit <s>	Dauer <s>	Priorität	Frequenz <Hz>	Arbeitslast je Periode <KWIPP>	Arbeitslast je Sekunde <KWIPS>
1	5	10	0	1.00	512	512
2	5	10	1	2.00	256	512
3	5	10	2	4.00	128	512
4	5	10	3	8.00	64	512
5	5	10	4	16.00	32	512
gesamt:	5	10		31.00		2560

Tabelle 2: Definition des Basis-Prozeßsatzes für die PH-Testreihen

Meßergebnisse

Die Ergebnisse des zuvor beschriebenen Experimentes PH-1 wurden in zwei Diagrammen zusammengefaßt.

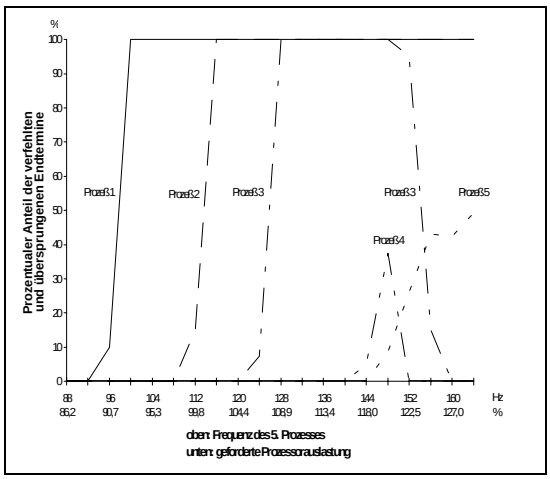

Abbildung 2: PH-1: Abhängigkeit der verfehlten und übersprungenen Endtermine von der geforderten Prozessorauslastung

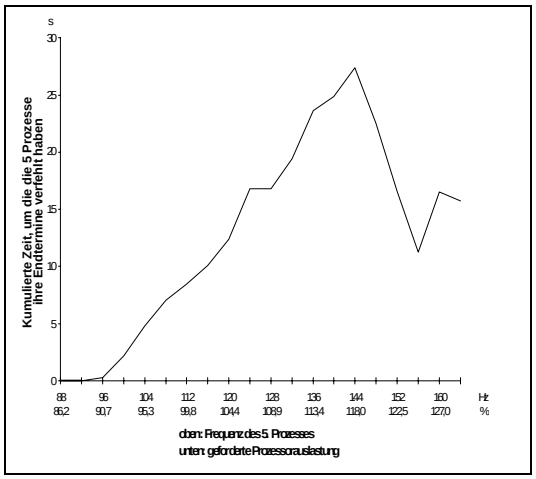

Abbildung 3: PH-1: Abhängigkeit der kumulierten Zeitüberschreitung der 5 Prozesse von der geforderten Prozessorauslastung

Das erste Diagramm (Abbildung 2) zeigt die prozentualen Anteile der verfehlten und übersprungenen Endtermine an der Gesamtzahl der gesetzten Endtermine jedes beteiligten Prozesses. Auf der Abszisse sind dabei sowohl der jeweilige Testparameter (oben) als auch die daraus resultierende Prozessorauslastung (unten) abgetragen. Prozeß 1 hat stets die niedrigste und Prozeß 5 die höchste Priorität.
Im zweiten Diagramm (Abbildung 3) folgt dann die verbrauchte Zeit, um die die Endtermine überschritten wurden. Dazu wurden die Zeiten aller Prozesse addiert und wieder in Abhängigkeit vom Testparameter bzw. der entsprechenden Prozessorauslastung dargestellt.
In beiden Diagrammen entsprechen die ersten beiden Teilstriche der Abszisse jeweils den beiden letzten erfolgreichen Tests, bei denen noch keine Endtermine verfehlt

wurden. Der zweite Teilstrich gibt demzufolge stets die maximal erreichbare Prozessorauslastung an.

Literatur:

[1] Kar, R.P.; Porter Kent: Rhealstone -A Real-Time Benchmarking proposal. In: Dr. Dobb´s Journal, S.14-24, Febr. 1989

[2] Kar, R.P.: Implementing the Rhealstone Real-Time Benchmark. In: Dr. Dobb´s Journal, S.46, April 1990

[3] Donohoe,P.; Shapiro, R.; Weidermann, N.: Hartstone Benchmark User´s Guide. Software Engineering Institute, Carnegie Mellon University, Technical Report, Mai 1990

[4] Weidermann; N.H., Kamenoff, N.I.: Hartstone uniprocessor benchmark: Definitions and Experiments for Real-Time Systems. In: Real-Time Systems Journal, 4 (4), S 353-383, Kluwer Academic Publishers, Dez. 1992

[5] Panzieri, F.; Donatiello, L., Poretti, L.: Scheduling Real-Time Tasks: A Performance Study. Laboratory for Computer Science, University of Bologna, Technical Report UBLCS-93-10, 1993

[6] Golatowski, F.: Entwicklung und Untersuchung geeigneter Meßmethoden zur Charakterisierung von Echtzeitbetriebssystemen. 8. Berliner Automatisierungskolloquium, Berlin, 1993

[7] Wichmann, B.A.: Validation Code for the Whetstone Benchmark. Technical Report DTIC 107/88, National Physical Laboratory, Teddington.Middlesex, UK, 1988