IMPLEMENTIERUNG EINES APPLIKATIONSORIENTIERTEN BENCHMARKS FÜR ECHTZEIT-UNIX-BETRIEBSSYSTEME*

F. Golatowski, H. Bösel, A. Paukert
Universität Rostock
Fachbereich Elektrotechnik
Institut für Technische Informatik

1 KURZFASSUNG

Im Beitrag wird eine Implementierung des Hartstone-Benchmarks in „C“ beschrieben. Dieser applikationsorientierte Benchmark ermöglicht die Ermittlung der maximale Auslastungsrate, bis zu der noch alle Endtermine durch das unterlagerte Echtzeitsystem bei unterschiedlichen synthetischen Lasten eingehalten werden. Es werden ausgewählte Ergebnisse für die Echtzeitbetriebssysteme SORIX und LYNX vorgestellt.
Die vorgestellte Implementierung erlaubt eine wahlfreie Definition von Prozeßsätzen und auf der Basis synthetischer Lasten die simulative Ausführung von Applikationen. Diese Simulation ermöglicht das Aufzeichnen eingehaltener oder verpaßter Endtermine und gestattet somit über einen längeren Zeitraum eine Beobachtung des einzusetzenden Echtzeitbetriebssystems.

2 EIN APPLIKATIONSORIENTIERTER BENCHMARK

Einen Ansatz, um das Gesamtverhalten eines Echtzeitsystems bzw. Echtzeitbetriebssystems unter Berücksichtigung verschiedener Lastkomponenten zu analysieren, stellt der Hartstone-Benchmark [2] dar. Dieser Benchmark wurde an der Carnegie Mellon Universität entwickelt. Der Hartstone-Benchmark zielt auf die Definition von Anforderungen im Echtzeitumfeld. Das zugrundeliegende Modell betrachtet ein Echtzeitsystem als einen Satz von periodischen, aperiodischen und Synchronisations-Tasks.
Der Benchmark besteht aus fünf unterschiedlichen Testserien, zu denen mehrere Experimente gehören. Ausgangspunkt einer Testserie ist ein Basisprozeßsatz (s.Tabelle 3), der charakterisiert ist durch die Anzahl der Prozesse, die Priorität der Prozesse, deren Ankunftsrate, deren Arbeitslast, deren Ausführungszeit und deren Endterminen (Deadlines).
Die Prozesse des Benchmarks können sowohl harte als auch weiche Endtermine besitzen. Bei einem harten Endtermin muß der Prozeß bis zu dessen Endtermin beendet sein, während ein Prozeß mit einem weichem Endtermin so schnell wie möglich beendet sein soll.
Im Hartstone-Modell haben periodische Prozesse harte Endtermine, die durch die Periode der Prozesse bestimmt sind. Aperiodische Prozesse können harte oder weiche Endtermine besitzen. So werden in der AH-Testserie die Endtermine harter aperiodischer (sporadischer) Prozesse durch die minimale Ereignisrate und die Endtermine weicher aperiodische Prozesse durch die durchschnittliche Ereignisrate repräsentiert.
Einen Überblick über die einzelnen Testserien und deren Prozesse gibt Tabelle 1 [2].
Die Testserien sind durch eine zunehmende Komplexität charakterisiert und bestehen aus mehreren Experimenten. In diesen wird jeweils ein bestimmter Parameter soweit erhöht, bis das zu testende System die gesetzten Endtermine überschreitet und damit den harten Echtzeitanforderungen nicht mehr genügt. Als Parameter dienen entweder die Endtermine der Prozesse, die von den Prozessen zu verrichtenden Arbeitslasten oder die Anzahl der beteiligten Prozesse. Die maximale Auslastungsrate U_{max}, bis zu der alle zeitlichen Forderungen eingehalten werden, kann als ein Leistungsmaß des unterlagerten Echtzeitsystems betrachtet werden.
Die Testserien des Benchmarks sind nützlich zur Evaluierung des Gesamtsystems.

*In: Rzehak, H. (Hrsg.): "Echtzeit 95" Kongreßvorträge, Karlsruhe, 20.-22.6.1995, S.63-70

PH-Serie 1-4	• 5 unabhängige periodische Prozesses mit harmonischen Frequenzen und harten Endterminen
PN-Serie 1-4	• 5 unabhängige periodische Prozesse mit nichtharmonische Frequenzen und harten Endterminen.
AH-Serie 1-6	• 5 unabhängige periodische Prozesse mit harmonischen Frequenzen • 1 unabhängiger aperiodischer Prozeß mit weichem Endtermin • 1 unabhängiger aperiodischer Prozeß mit weichem Endtermin
SH-Serie 1-5	• 5 unabhängige periodische Prozesse mit harmonischen Frequenzen • 1 Synchronisationsprozeß, mit der sich jeder periodische Prozeß jeweils einmal je Periode synchronisieren muß
SA-Serie 1-4	• 5 unabhängige periodische Prozesse mit harmonischen Frequenzen • 1 unabhängiger sporadischer Prozeß • 1 unabhängiger aperiodischer Prozeß mit weichem Endtermin • Synchronisationsprozeß der SH- Serie

Tabelle 1: Die Testserien des Hartstone Uniprocessor Benchmarks

3 IMPLEMENTIERUNG DES HARTSTONE UNIPROCESSOR BENCHMARKS

An unserem Institut wurde eine „C"-Implementierung des Hartstone-Benchmarks realisiert. Ziel dieser Implementierung ist die Untersuchung kommerzieller Echtzeit-UNIX-Betriebssysteme und die Entwicklung eines Werkzeuges zur Ausführung synthetischer Applikationen. (Die Beispiel-Implementierung in ADA des Software Engineering Institute zielt auf die Evaluierung von Targetplattformen [1]). Aus den in [2] definierten Testbedingungen wurden die speziellen Anforderungen an Echtzeit-UNIX-Systeme abgeleitet.
Im weiteren wird auf die Testserien PH und PN und auf gewonnne Ergebnisse, die mit dem Echtzeit-UNIX-Betriebssystemen SORIX und LYNX ermittelt wurden, eingegangen. Entsprechend dem Grundkonzept der einzelnen Experimente wird eine analysierbare synthetische Arbeitslast in jedem Prozeß eines Experimentes ausgeführt.

3.1 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) [3]. 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):

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

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

3.2 Definition der Testreihen PH und PN

Bei der Durchführung der Messungen wird beim ersten Test stets von einem Basis-Prozeßsatz ausgegangen (s. Tabelle 3). Mit jedem weiteren Test wird dann einer der Parameter solange erhöht, bis die Summe der nicht erreichten Endtermine aller Prozesse einen Grenzwert überschreitet. Dabei werden folgende Werte gemessen und am Ende der jeweiligen Testreihe zusammengefaßt:

- Anzahl der eingehaltenen Endtermine. Ein Endtermin gilt als eingehalten, wenn der Prozeß die ihm zugeteilte periodische Arbeit U_{KWIPP} *vor* Ablauf seiner Ausführungsperiode T abgearbeitet hat (Endtermin = Ausführungsperiode).
- Anzahl der überschrittenen Endtermine
  Wurde die Arbeit erst *nach* Ablauf der Ausführungsperiode beendet, dann wird der entsprechende Endtermin als überschritten angesehen.
- Anzahl der übersprungenen Endtermine
  Verfehlt ein Prozeß einen Endtermin, so wird nicht versucht, in der bis zum nächsten Ereignis verbleibenden Zeit der aktuellen Periode noch die zugehörige Arbeit zu verrichten. Statt dessen wird dieser Endtermin übersprungen, so daß die nächste Prozeßaktivierung am Beginn der folgenden Ausführungsperiode stattfindet. Wurde ein Endtermin um die Zeit mehrerer Ausführungsperioden überschritten, so werden die dazugehörigen Endtermine ebenfalls übersprungen.
- Kumulierte Zeit, um die die Endtermine überschritten wurden
  Für jeden Prozeß wird die Zeit gemessen und aufsummiert, um die die Abarbeitung der Arbeitslast den zugehörigen Endtermin überschritten hat.

Die Summe der eingehaltenen, verfehlten und übersprungenen Endtermine eines periodischen Prozesses ergibt dabei die Anzahl der für ihn bestimmten Ereignisse.
Der Basis-Prozeßsatz besteht aus 5 periodischen Prozessen. Damit lassen sich zum einen Rückschlüsse auf das Systemverhalten bei einer größeren Zahl von Prozessen ziehen. Andererseits bleibt ein derartiger Satz noch gut überschaubar in Bezug auf die Definition von Frequenzen, Arbeitslasten und zu erwartenden Prozessorauslastungen.
Aus dem so definierten Prozeßsatz lassen sich durch Wahl verschiedener Parameter die folgenden vier Testreihen ableiten [2]:

- PH-1: Bei jedem Schritt wird die Frequenz des 5. Prozesses um den Betrag der Frequenz des 4. Prozesses erhöht. Die anderen Kenngrößen des Basissatzes bleiben konstant. Der Test wird beendet, wenn eine bestimmte Zahl von Endterminen verfehlt bzw. überschritten wurde. 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.
- PH-2: In dieser Testreihe werden die Frequenzen aller 5 Prozesse des Basissatzes schrittweise um jeweils 10 Prozent des Ausgangswertes erhöht. Damit sollen Schlußfolgerungen über das Systemverhalten gezogen werden, wenn bei größer werdender Arbeitslast gleichzeitig der für das Scheduling notwendige Overhead wächst.
- PH-3: Durch schrittweise Erhöhung der Arbeitslasten aller Prozesse um einen konstanten Betrag kann mit dieser Testreihe überprüft werden, inwiefern das System in der Lage ist, eine wachsende Arbeitslast zu bewältigen. Im Unterschied zu den beiden vorangegangenen Testreihen bleibt dabei der Scheduling-Overhead auf relativ niedrigem Niveau konstant.
- PH-4: Ausgehend vom Basissatz der Tabelle 3 wird in dieser Testreihe mit jedem Schritt ein zusätzlicher Prozeß vom Typ des 3. Prozesses hinzugefügt (gleiche Frequenz und Arbeitslast). Daraus lassen sich Schlußfolgerungen ziehen, wie gut das zu testende System in der Lage ist, effektiv mit einer relativ großen Zahl von Prozessen zu arbeiten. Voraussetzung dafür ist vor allem ein möglichst geringer Overhead des Schedulers.

Die Definitionen der Testreihen PN-1 bis PN-4 unterscheiden sich von denen der Reihen PH-1 bis PH-4 nur dahingehend, daß bei diesen die Frequenzen der Prozesse des Basissatzes nicht in einem ganzzahligen Verhältnis zueinander stehen (nichtharmonische Prozesse). Durch eine entsprechende Anpassung der periodischen Arbeitslasten wird auch hier die gleiche anfängliche Arbeit je Sekunde für alle 5 Prozesse erreicht. Da sich eine ungleichmäßige zeitliche Verteilung der Arbeitslastrate ergibt, ist das Scheduling solcher Prozesse durch das System schwieriger durchzuführen als bei Verwendung harmonischer Frequenzen. Es ist

demnach zu erwarten, daß der Prozessorauslastungsgrad, ab dem zum ersten Mal Endtermine nicht eingehalten werden, bei den PN-Testreihen geringer sein wird. [4]

3.3 Anforderungen an die Implementierung

Folgende Anforderungen und Ziele wurden bei der Realisierung des Programms Benchmarks gestellt:

- Freie Wahl der Eigenschaften des Basis-Prozeßsatzes und der schrittweise zu verändernden Parameter durch Verwendung einer Testbeschreibungsdatei,
- Durchführbarkeit der Messungen der Testreihen PH und PN mit periodischen Prozessen (vgl. Abschnitt 3.2),
- Erweiterbarkeit auf andere Prozeßtypen, zum Beispiel aperiodische und sporadische Prozesse nach[2],
- Abbruch der Tests nach einstellbarer Anzahl verfehlter bzw. übersprungener Endtermine (vgl. Abschnitt 3.2),
- Ausgabe aller Zwischenschritte bis zum Abbruch einer Testreihe (abschaltbar durch Kommandozeilenoption) und einer Zusammenfassung,
- Gleichzeitige Ausgabe der Ergebnisse auf dem Bildschirm und in eine optionale Datei
- möglichst einfache Portierbarkeit auf andere PC-UNIX-Versionen.

Basis aller durchzuführenden Tests ist eine Testbeschreibungsdatei, in der alle Parameter des gewünschten Prozeßsatzes festgelegt werden. Damit sind auch frei definierbare Tests möglich, die unabhängig von den Testserien des Hartstone-Benchmarks sind.
Der prinzipielle Aufbau einer solchen Datei soll an folgendem Beispiel demonstriert werden:

```
#Kommentarzeilen beginnen mit Doppelkreuz.

Experiment
        Dies ist der Beschreibungstext des ersten Experiments.
Test
#         Typ      Klasse    Prio      Start     Dauer     Freq      KWIPP
          P        RT        20        5         10        1.00      512
          P        RT        21        5         10        2.00      256
          P        RT        22        5         10        4.00      128
          P        RT        23        5         10        8.00      64
          P        RT        24        5         10        16.00     32

Test
          P        RT        20        5         10        1.05      512
          P        RT        21        5         10        2.10      256
          P        RT        22        5         10        4.20      128
          P        RT        23        5         10        8.40      64
          P        RT        24        5         10        16.80     32
                                    • • •
Experiment
        Dies ist der Beschreibungstext des zweiten Experiments.
Test
          P        RT        0         5         10        1.00      512
          P        RT        1         5         10        2.00      256
          P        RT        2         5         10        4.00      128
Test
          P        RT        0         5         10        1.00      516
          P        RT        1         5         10        2.00      260
          P        RT        2         5         10        4.00      132
                                    • • •
```

Abbildung1: Prinzipieller Aufbau einer Testbeschreibungsdatei

Wie das Beispiel erkennen läßt, kann eine Testbeschreibungsdatei auch mehrere Testreihen enthalten, welche jeweils durch das Schlüsselwort „Experiment“ am Zeilenanfang eingeleitet werden. Diesem kann zur Beschreibung der Testreihe ein mehrzeiliger Text folgen, welcher in der späteren Ausgabe des Programms erscheint. Darauf folgen die Parameter der einzelnen Prozeßsätze. Sie beginnen jeweils mit dem Schlüsselwort „Test“ am Zeilenanfang und enthalten für jeden einzelnen Prozeß eine gesonderte Zeile mit den im folgenden beschriebenen Feldern.

Die Definition des Basissatzes in der Testbeschreibungsdatei enthält für jeden Prozeß folgende Parameter:

- Prozeßtyp („P“- Periodischer Prozeß, „A“- Aperiodischer Prozeß, „S“- Synchronisationsprozeß),
- Prozeßklasse (in der Regel „RT“-Echtzeitprozeß, aber auch „TS“ für Timesharing),
- Priorität,
- Startzeit und Dauer der Ausführung,
- Ereignisfrequenz,
- Arbeitslast in KWIPP (Kilo-Whetstone-Instructions per Period).

Durch diese Art der Parameterübergabe ergibt sich ein hoher Komfort bei der Arbeit mit dem Programm, der sich besonders bei der Ermittlung geeigneter Anfangsbedingungen bemerkbar macht.
Die zur schrittweisen Erhöhung der Arbeitslast zu verändernden Parameter sind ebenfalls in der Testbeschreibungsdatei enthalten. Um deren Aufbau möglichst einfach zu gestalten und den Programmieraufwand zu verringern, wird der gesamte Prozeßsatz für jeden Schritt vollständig definiert und vom Meßprogramm neu eingelesen. Der Anwender kann deshalb durch einfaches Editieren einer Datei völlig neue Testreihen definieren. Außerdem ist eine Erweiterung des Programms auf andere Prozeßtypen relativ leicht durchzuführen (zum Beispiel aperiodische Prozesse).
Der Nachteil dieses Verfahrens liegt in den zum Teil recht groß werdenden Testbeschreibungsdateien. Um den Aufwand des Editierens trotzdem möglichst gering zu halten, wurde ein spezieller Editor zur Generierung dieser Dateien erstellt.
Einige wichtige Programmeigenschaften lassen sich über Optionen und Argumente in der Kommandozeile steuern (Tabelle 2). Die Angabe der zu verwendenden Testbeschreibungsdatei ist dabei obligatorisch.
Aufruf des Benchmarkprogramms:

```
hartstone [-d deadlines] [-v] [-o outfile] testfile
```

Argument	Beschreibung
`-d deadlines`	Anzahl der verfehlten bzw. übersprungenen Endtermine, bei der die Testreihe abgebrochen wird (optional, Standard: 50 Endtermine)
`-v`	Ausgabe aller Zwischenschritte bis zum Abbruch der Testreihe
`-o outfile`	Name der zusätzlichen Ausgabedatei (optional)
`testfile`	Name der Testbeschreibungsdatei (obligatorisch)

Tabelle 2: Kommandozeilenargumente und Optionen des Meßprogramms hartstone

3.4 Meßergebnisse

Tabelle 3 zeigt für die gewählte Rechnerkonfiguration (s. Tabelle 4) den ermittelten Basisprozeßsatz. Es wurden nicht der in [2] vorgeschlagene Basisprozeßsatz gewählt, da das untersuchte Rechensystem wesentlich leistungsstärker ist.

	Startzeit <s>	Dauer <s>	Priorität SORIX/ LYNX	Frequenz <Hz>	Arbeitslast je Periode <KWIPP>	Arbeitslast je Sekunde <KWIPS>
1	5	10	0/20	1.00	512	512
2	5	10	1/21	2.00	256	512
3	5	10	2/22	4.00	128	512
4	5	10	3/23	8.00	64	512
5	5	10	4/24	16.00	32	512
gesamt:	5	10		31.00		2560

Tabelle 3: Definition des Basis-Prozeßsatzes für die PH-Testreihe

Damit die Dauer eines Experimentes gering bleibt, geht dieser Basisprozeßsatz etwa von 50% der Last aus, die das Referenzsystem in der Lage ist zu bearbeiten. In den einzelnen Experimenten erfolgt, ausgehend von diesem Basisprozeßsatz die kontinuierliche Erhöhung der Last (s. 3.2).

Prozessortyp:	i80486
CPU- Frequenz	33 Mhz
Cache	2^{nd} Level/ 256 kBytes
Wait States	1 Wait State
Memory Size	16 MByte

Tabelle 4: Getestetes Referenzsystem: Hardware

Betriebssystem	LYNX V 2.2.1	SORIX 386/486 V 03.00.07
C-Compiler	gcc 1.4.2	cc
Compiler-Optionen	-O (-O -m -X)	O
Implementiertes Prozeßmodell	Prozeß (alternativ Threads)	Prozeß

Tabelle 5: Untersuchte Betriebssysteme

Zur Bestimmung der Anzahl von KWIPS (Kilo-Whetstone-Instructions per Second), die einer Prozessorauslastung von 100 Prozent entsprechen, wird die ohne Unterbrechung erreichbare Arbeitslastrate eines einzelnen Echtzeitprozesses mit höchster Priorität gemessen. Mit diesem Bezugswert läßt sich der Basisprozeßsatz, der von einer 50%-igen Auslastung U ausgeht, und die von einem Prozeß im späteren Test geforderte Prozessorauslastung ermitteln. Das folgende Beispiel soll dies verdeutlichen.
Ein einzelner Echtzeitprozeß mit höchster Priorität hat bei der Messung ohne Unterbrechung eine Arbeitslastrate von 5600 KWIPS erreicht. Dies entspricht einer Prozessorauslastung von 100 Prozent. Ein periodischer Prozeß mit einer Ereignisfrequenz von 10 Hertz und einer je Periode auszuführenden Arbeit von 100 KWIPP bewirkt damit eine Prozessorauslastung von:

$$U_{\%} = \frac{10\ \text{Hz} * 100\ \text{KWIPP}}{5600\ \text{KWIPS}} * 100\,\% = \frac{1000\ \text{KWIPS}}{5600\ \text{KWIPS}} * 100\,\% = 17.86\,\% \qquad (2)$$

Als Auswahl sind die Ergebnisse der Experimente PH-1 und PN-1 für die Echtzeitbetriebssysteme SORIX und LYNX graphisch dargestellt. (s.Abb. 2) und für die Experimente der Testserien PH und PN die maximalen Auslastungsraten angegeben (s. Tabelle 6 und Tabelle 7) bei denen die Systeme noch alle zu verarbeitenden Endtermine eingehalten haben. Die Diagramme zeigen die prozentualen Anteile der verfehlten und übersprungenen Endtermine (nicht eingehaltene) an der Gesamtzahl der gesetzten Endtermine jedes beteiligten Prozesses. Auf der Abszisse ist dabei die resultierende Prozessorauslastung abgetragen. Prozeß 1 hat stets die niedrigste und Prozeß 5 die höchste Priorität. Die Ergebnisse der PH-1 Experimente zeigen, daß der Prozeß mit der niedrigsten Frequenz ab einer Gesamtauslastung von 87,5% (SORIX) bzw. 97,1% (LYNX) anfängt, seine Endtermine zu verfehlen und zu überspringen. Das liegt daran, daß dieser auch die niedrigste Priorität besitzt und damit bei knapper werdenden Ressourcen (CPU-Zeit) als erster von den höherpriorisierten Prozessen verdrängt wird. Steigt die die Gesamtauslastung weiter, so verfehlen auch die anderen Prozesse in der Reihenfolge ihrer Prioritäten zunehmend ihre Endtermine.

	LYNX	SORIX
PH-1	97,1 %	87,5 %
PH-2	97,2 %	89,9 %
PH-3	98,8 %	92,9 %
PH-4	94,2 %	82,9 %

Tabelle 6: Maximale Auslastung bei der PH-Serie

	LYNX	SORIX
PN-1	90,3 %	86,8 %
PN-2	86,1 %	83,7 %
PN-3	88,4 %	88,1 %
PN-4	82,7 %	83,0 %

Tabelle 7: Maximale Auslastung bei der PN-Serie

Folgendes Verhalten ist für beide Systeme repräsentativ:

- PH: Zuerst gehen Endtermine des Prozesses mit der niedrigsten Priorität verloren. Danach folgen die Prozesse entsprechend ihrer Priorität. Besonders in den PH-Serien ist zu beobachten: Erst nachdem die Endtermine des jeweils niederprioren Prozesses nicht mehr eingehalten und dieser Prozeß auch nicht mehr bearbeitet wird, verpassen die anderen höherpriorisierten Prozesse ihre Endtermine.
- PH: Die Ergebnisse zeigen, daß nach Erreichen der maximalen Auslastung U_{max} maximal zwei Testschritte erforderlich sind, bis die Prozesse alle zugehörigen Endtermine verpaßt haben.
- PN: Bedingt durch die nichtharmonischen Frequenzen der Prozesse ist ein anderes Verhalten zu erkennen. Die maximale Auslastungsrate ist geringer als in den Experimenten der PH-Serie. Hier können zwei Prozesse ihre Endtermine gleichzeitig verpassen.
- Es ist möglich, daß Prozesse scheinbar wieder ihre Endtermine einhalten, obwohl die theoretische Last weit über 100% erreicht ist. Das ist dem zugrundeliegenden Meßverfahren geschuldet, wobei sich das System „erholen“ kann, wenn, nachdem ein Endtermin verpaßt wurden, der nächste übersprungen wurde.
- Von besonderem Interesse sind Meßreihen in denen scheinbar unerwartete Ergebnisse ermittelt wurden. Diese geben Auskunft über das Verhalten der unterlagerten Betriebssoftware und weisen auf Besonderheiten hin.
- Die gewonnenen Ergebnisse sind vom gewählten Basisprozeßsatz abhäng.

4 ZUSAMMENFASSUNG:

Mit der vorgestellten Implementierung des Hartstone-Benchmarks ist es möglich, das Verhalten eines Echtzeitsystems unter steigenden Lastbedingungen zu testen. Die ermittelten Ergebnisse geben einen Gesamtüberblick über die Leistungsfähigkeit der Komponenten des Echtzeitsystems, wie Compiler, Taskwechselzeiten, Overhead der unterlagerten Betriebssoftware, insbesondere bei steigender Last. Durch die universelle Handhabung ist es möglich, eine definierte synthetische Last auf einem Echtzeitsystem ausführen zu lassen.
Darüber hinaus erlaubt die Implementierung die Nachbildung von Applikationen und deren simulativen Ablauf. Beim Starten des Benchmarkprogramms wird die in Abbildung1 beschriebene Testbeschreibungsdatei eingelesen. Anhand der eingelesenen Parameter werden die entsprechenden Prozesse kreiert und gestartet.
Damit steht dem Anwender ein Werkzeug zur Verfügung, mit dem eine konkrete Applikation nachgebildet werden kann. Dazu ist es erforderlich, die in einer solchen Applikation auszuführenden Last in KWIPS auszudrücken und die Prioritäten von Prozessen entsprechend der Applikation zu verteilen. Diese Simulation gestattet so Aussagen über das Einhalten von Echtzeitbedingungen, das Aufzeichnen eingehaltener oder verpaßter Endtermine und ermöglicht somit über einen längeren Zeitraum eine Beobachtung des einzusetzenden Echtzeitbetriebssystems.

Literatur:

[1] Donohoe,P.; Shapiro, R.; Weidermann, N.: Hartstone Benchmark User´s Guide. Software Engineering Institute, Carnegie Mellon University, Technical Report, Mai 1990
[2] 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
[3] Wichmann, B.A.: Validation Code for the Whetstone Benchmark. Technical Report DTIC 107/88, National Physical Laboratory, Teddington.Middlesex, UK, 1988
[4] Liu, C.L.; Layland, J.W.: Scheduling Algorithms for Hard Real-Time Environments. Journal of the ACM (20) 1, S.46-61, 1973