Echtzeit im Echtheits-Test

Das OSADL-Testzentrum misst die Echtzeit-Performance verschiedenster Prozessoren unter Linux – 24 Stunden live und öffentlich einsehbar

Für die „Stiftung Warentest" sind Embedded-Prozessoren wohl etwas zu speziell. Das Team des Open Source Automation Development Lab kennt sich damit besser aus. In einem Testlabor laufen 24 Prozessoren verschiedenster Leistungsklassen und Architekturen. Neben dem Beweis der Echtzeit-Fähigkeit von Linux entsteht als Nebenprodukt ein architekturübergreifender Prozessor-Benchmark. Die Ergebnisse sind mindestens so spannend wie die Veröffentlichungen der „Warentester".

Von Dr. Carsten Emde

Beim Kernel-Summit in Ottawa im Jahre 2006 wurde beschlossen, komplett deterministische Echtzeit-Fähigkeit in den Hauptentwicklerzweig des Linux-Kernels (Mainline-Linux) aufzunehmen. Seitdem wurde eine große Anzahl an Echtzeit-Komponenten für Mainline-Linux bereitgestellt und von Linus Torvalds in den Kernel aufgenommen:

- Mutexe mit Priority-Inheritance,
- hochauflösende Timer,
- preemptives RCU (Read-Copy Update),
- Interrupt-Threads,
- Spinlock-Kategorisierung nach Unterbrechbarkeit.

Hiermit und im Zusammenspiel mit weiteren Komponenten lassen sich Linux-Kernel mit Echtzeit-Fähigkeit für die Architekturen x86, ARM, PowerPC, MIPS und 68k herstellen. Ziel der Entwicklung ist es, dass mit diesen Kerneln ausgerüstete Systeme unter allen Umständen mit einer hinreichenden statistischen Wahrscheinlichkeit in einer vorher definierten Maximalzeit auf externe Ereignisse reagieren. Damit sollen diese eine wesentliche Voraussetzung für den Einsatz in der Automatisierungsindustrie erfüllen und allgemein für Bedingungen geeignet sein, bei denen harte Echtzeit gefordert wird. Aber ist dies wirklich so? Wer überprüft die Echtzeit-Fähigkeit eigentlich, und wer sorgt dafür, dass eventuell nötige Korrekturen vorgenommen werden?

Open Source garantiert hohe Code-Qualität

Es ist wohl unbestritten, dass die hohe Code-Qualität des Linux-Kernels in besonderem Maße dem Open-Source-Entwicklungsverfahren zu verdanken ist. Und dieses ist unter anderem eine Folge der im Linux-Kernel verwendeten Lizenz, der GNU General Public License (GNU GPL). Diese Lizenz fordert, dass bei Weitergabe des Linux-Kernels dessen Quell-Code offengelegt werden muss – und damit auch alle möglicherweise vorgenommenen Verbesserungen und Erweiterungen in den Hauptentwicklerzweig zurückfließen können. Eine anderer Grund für die hohe Code-Qualität liegt im mehrstufigen, weltweit öffentlichen Begutachtungsverfahren, den eine Code-Änderung oder ein neuer Code durchlaufen müssen, bevor schließlich die Aufnahme in einen Release-Kandidaten erfolgt. Bis zum Release dauert es dann noch einmal etwa zehn Wochen mit etwa sechs bis acht weiteren Release-Kandidaten. In dieser Zeit wird der neue Kernel kontinuierlich getestet und korrigiert. Danach liegt ein stabiler und für Produktionszwecke verwendbarer Kernel vor. Allerdings gilt dies nicht notwendigerweise auch für die Echtzeit-Fähigkeit.

Im Gegensatz zu den relevanten Testkriterien des nicht echtzeitfähigen Standardkernels stellt die Echtzeit-Fähigkeit eine besondere Herausforderung an die Testumgebung dar. Während nämlich beim Standardkernel statische Tests im Vordergrund stehen – also die kurzfristige Überprüfung der grundsätzlichen Eigenschaften von Kernel und Treibern –, müssen bei der Überprüfung des Determinismus über einen längeren Zeitraum möglichst viele unterschiedliche Bedingungen im Zusammenspiel der einzelnen Kernel-Komponenten geschaffen und dabei die Echtzeit-Parameter mit möglichst hoher Messfre-

```
$ head -18 timerandwakeup/CPU0
#Minimum latency: 0 microseconds
#Average latency: 2 microseconds
#Maximum latency: 10 microseconds
#Total samples: 864868
#There are 0 samples lower than 0 microseconds.
#There are 0 samples greater or
#equal than 10240 microseconds.
#usecs              samples
      0                 1152
      1               144953
      2                39397
      3               678260
      4                  858
      5                  197
      6                   36
      7                    6
      8                    4
      9                    4
     10                    1
```

Histogramm-Datei mit Messungen über die Echtzeit-Performance des Linux-Kernels. Nach einigen statistischen Kennwerten am Anfang ist zu jeder Zahl der Mikrosekunden die Zahl der „Treffer" aufgeführt. Die gesamte Liste ergibt die Latenzverteilung.

quenz bestimmt werden. Dies kann bedeuten, dass mehrere Milliarden Einzelmessungen erforderlich sind, so dass eine Messdauer von mehreren Tagen oder sogar Wochen resultiert. Derartig intensive Tests können von den Entwicklern des Standard-Kernels naturgemäß nicht in voller Bandbreite geleistet werden; dies muss vielmehr von der relativ kleinen Gruppe von Anwendern, die an der Echtzeit-Fähigkeit des Kernels interessiert sind, selbst organisiert werden. Aus diesem Grunde hat sich eine Reihe von Firmen, die an der Verfügbarkeit eines echtzeitfähigen Linux-Kernels interessiert sind, in einer Organisation zusammengeschlossen: Sie sorgt gemeinsam dafür, dass die Echtzeit-Fähigkeit des Linux-Kernels gemessen, kommuniziert und garantiert wird. Bei dieser Organisation handelt es sich um das Open Source Automation Development Lab (OSADL), und das Echtzeit-Testzentrum heißt „QA Farm Realtime". In dieser Farm wird eine große Anzahl verschiedener Systeme unter einer Vielzahl von System- und Lastbedingungen kontinuierlich auf ihre Echtzeit-Fähigkeit getestet. Viele relevante Messergebnisse werden kontinuierlich im Internet für jedermann verfügbar gemacht. Nur spezielle Messergebnisse, die z.B. für Zertifizierungen spezieller Gerätekombinationen verwendet werden können, sind exklusiv den jeweiligen OSADL-Mitgliedsfirmen zugänglich.

Kernel-Messung mit Bordmitteln

Der echtzeitfähige Linux-Kernel bietet eine große Anzahl eingebauter Diagnose-Systeme, von denen einige zur Messung der Echtzeit-Eigenschaften verwendet werden können. Dabei handelt es sich im Speziellen um

- Wakeup-Latenz,
- Timer-Offset und
- Summe von Timer-Offset und Wakeup-Latenz.

Wenn ein Prozess lauffähig geworden ist und in die Warteschlange eingefügt wird, z.B. weil ein Interrupt eingetroffen ist oder weil ein anderer, höher priorisierter Prozess keine Rechenzeit mehr beansprucht, wird die aktuelle Uhrzeit bestimmt und gespeichert. Wenn der lauffähig gewordene Prozess ein wenig später erfolgreich aktiviert wurde und tatsächlich Rechenzeit erhält, wird die aktuelle Uhrzeit wiederum bestimmt und die Dauer zwischen der aktuellen und der gespeicherten Uhrzeit berechnet. Unter der Bedingung, dass der beobachtete Prozess die höchste Priorität des Systems hat und während des gesamten Aufweckvorgangs kein anderer Prozess mit dieser oder einer höheren Priorität Rechenzeit benötigt, kann die berechnete Zeitdauer als zu diesem Zeitpunkt wirksame Wakeup-Latenz des Systems betrachtet werden. Die berechnete Zeitdifferenz wird sodann in einem Kernel-internen Histogramm gespeichert, das nach einer genügend langen Messdauer ausgewertet werden kann. Bei dieser Auswertung wird der höchste jemals gemessene Wert bestimmt; als maximale Wakeup-Latenz des Systems („worst-case wakeup latency") stellt dieser Wert zwar nur eine eingeschränkte Kenngröße eines Echtzeit-Systems dar, wird aber häufig verwendet und deswegen zu Vergleichszwecken auch hier angegeben. Die Granularität und damit die Messgenauigkeit aller Kernel-Histogramme beträgt 1 µs, der Messbereich 0 bis 10,24 ms. Der letzten Histogrammzelle kommt dabei eine besondere Bedeutung zu, denn hier werden alle Werte von 10,24 ms und höher eingetragen, so dass diese Zelle als Überlauf-Indikator fungiert. Nur wenn kein Überlauf eingetreten ist, darf das Histogramm tatsächlich für Analysezwecke verwendet werden.

In einigen Fällen wird ein Prozess nicht deswegen lauffähig, weil ein anderer, höher priorisierter Prozess keine Rechenzeit mehr beansprucht, sondern weil ein Interrupt eingetroffen ist – z.B. ausgelöst von einem Kommunikationscontroller im Falle von empfangenen Daten oder von einem Timer, wenn ein Alarm aufgesetzt wurde. In diesem Fall wird die Latenz nicht nur von der Wakeup-Latenz, sondern auch von einer eventuellen Verspätung des Timer-Interrupts und dessen Verarbeitung („Timer-Offset") bestimmt. Daher verfügt der Echtzeit-Linux-Kernel über ein weiteres internes Histogramm. In dieses Histogramm wird jede Verspätung eingetragen, die sich aus der Differenz zwischen der aktuellen und der geplanten Uhrzeit eines Timer-Interrupts ergibt.

Architektur	Hersteller	Prozessorbezeichnung	Taktfrequenz	Bitbreite
x86	AMD	LX800	500 MHz	32 bit
		K6 3D	333 MHz	32 bit
		Athlon XP 2000+	1667 MHz	32 bit
		Athlon 64 2800+	1800 Mhz	64 bit
		Phenom II X6	3200 MHz	64 bit
	Intel	Pentium	133 MHz	32 bit
		Pentium II Klamath	233 MHz	32 bit
		Atom N270	1600 MHz	32 bit
		Atom D510	1667 MHz	64 bit
		Atom Z530 (Standard-Kernel als Referenz)	1600 MHz	32 bit
		Celeron M	1500 MHz	32 bit
		Pentium M dual-core	2300 MHz	32 bit
		Xeon	2800 MHz	32 bit
		Core 2 Duo	2400 MHz	64 bit
		Core 2 Quad	2400 MHz	32 bit
		i7 Nehalem 975	3333 MHz	32 bit
		i7 Gulftown X980	3333 MHz	64 bit
	VIA	C3 Samuel 2	533 MHz	32 bit
		C3 Nehemiah	533 MHz	32 bit
		C7	1000 MHz	32 bit
ARM	Marvell	SheevaPlug	1200 MHz	32 bit
	Texas Instr.	AM3517	600 MHz	32 bit
PowerPC	Freescale	MPC 5121	400 MHz	32 bit
MIPS	ICT	Loongson 2F	800 MHz	64 bit

Prozessoren im OSADL Echtzeit-Testzentrum. Bei Taktfrequenz, Bitbreite und Prozessor-Topologie wurde auf eine möglichst große Vielfalt geachtet. Die Größe des Systemspeichers reicht von 64 Mbyte bis 12 Gbyte, Dateisysteme wurden von ext2 bis ext4 aufgesetzt.

Schließlich ist im Echtzeit-Linux-Kernel noch ein drittes Histogramm vorgesehen, in welchem die Summe einer eventuellen Timer-Verspätung und der Wakeup-Latenz eingetragen wird. Diese Summe reflektiert die aktuell wirksame Gesamt-Latenz des Systems. Als Ergebnis darf aber wiederum nicht ein einmalig gemessener Wert verwendet werden. Es muss vielmehr über einen sehr langen Zeitraum bei möglichst vielen unterschiedlichen Zuständen des Systems gemessen werden, damit der höchste jemals gemessene Wert als einzig relevante Kenngröße eines Echtzeit-Systems, nämlich als maximale Gesamt-Latenz („worst-case latency"), ermittelt und bei der System-Integration verwendet werden kann. Bei schnellen Prozessoren mit für Echtzeit gut geeigneten Systemkomponenten kann die maximale Gesamt-Latenz nur wenige Mikrosekunden

betragen; in anderen Fällen liegt sie bei bis zu 400 Mikrosekunden.

Kernel für Echtzeit-Messaufgaben konfigurieren

Damit der Echtzeit-Linux-Kernel die genannten Histogramme enthält, muss dieser bei der Herstellung entsprechend konfiguriert werden. Dafür werden zwei Konfigurationsparameter verwendet (Kernel-Version 2.6.33.7.2-rt30):

```
CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_MISSED_TIMER_OFFSETS_
    HIST=y
```

Es kann durchaus empfohlen werden, den Kernel routinemäßig so zu konfigurieren; denn durch die Konfiguration allein werden die Histogramme nur bereitgestellt, aber noch nicht eingeschaltet. Wenn die Histogramme dann tatsächlich verwendet werden sollen, müssen sie zur Laufzeit aktiviert werden. Dies erfolgt durch Schreiben eines Wertes ungleich 0 in die jeweiligen virtuellen Dateien im „enable"-Verzeichnis des Debug-Filesystems debugfs:

```
cd /sys/kernel/debug/
    tracing/latency_hist
echo 1 >enable/wakeup
echo 1 >enable/missed_timer_
    offsets
echo 1 >enable/timerandwakeup
```

Aber auch nach dem Einschalten der Histogramme kommt es zu keinem relevanten Performance-Verlust, so dass die Messung auch unter Produktionsbedingungen durchgeführt werden kann. Dadurch geringfügig verlängert gemessene Latenzen würden ein eher konservatives Ergebnis liefern und wären daher sicherheitsmäßig unbedenklich.

Im jeweils gleichnamigen Unterverzeichnis findet sich pro CPU (d.h. pro Core und Hyperthread) eine Datei mit dem Namen „CPU", gefolgt von der Nummer der CPU. Darin ist jeweils eine Zeile pro Mikrosekunde Latenz enthalten, welche die Anzahl an Messwerten der jeweiligen Latenzkategorie enthält. Am Anfang der Datei werden statistische Kennwerte der Verteilung ausgegeben wie z.B. im **Listing** auf S. 54.

Damit eine gewisse zeitliche Zuordnung einer Latenz zu einem bestimmten Zeitraum ermöglicht wird, ist eine virtuelle Reset-Datei vorgesehen. Durch Schreiben eines Wertes ungleich 0 in die entsprechende Datei werden alle Zellen des Histogramms auf Null zurückgesetzt. Der vorher ausgelesene maximale Latenzwert bezieht sich entsprechend auf den Zeitraum seit dem letzten Reset.

Applikations-Messung mit „Cyclictest"

Um die im Kernel eingebauten Histogramme empirisch überprüfen zu können, sollte eine zweite unabhängige Messmethode verfügbar sein. Diese sollte möglichst gut die Bedingungen simulieren, die auch in einem unter Produktionsbedingungen verwendeten Echtzeit-System herrschen. Im OSADL Echtzeit-Testzentrum wird das ursprünglich von Thomas Gleixner geschriebene und von der Linux-Echtzeit-Community weiterentwickelte Tool „Cyclictest" verwendet. Es ist in den RT-Testprogrammen „RT-Tests" enthalten, deren Maintainer Clark Williams (Red Hat) ist. Die jeweils aktuelle Version ist als Git-Tree im Internet verfügbar [1]. Nach dem Programmstart von Cyclictest wird eine bestimmte, durch Kommandozeilen-Optionen vorgegebene Anzahl an Mess-Threads gestartet, die zyklisch angehalten und mit Hilfe von Alarmen wieder gestartet werden. Nach jedem Start wird die aktuelle Uhrzeit mit der erwarteten Uhrzeit verglichen, die Differenz dem Hauptprozess verfügbar gemacht und von diesem angezeigt bzw. – je nach Kommandozeilen-Option – in einem Histogramm akkumuliert. Auch dieses Histogramm weist eine Granularität von 1 µs auf, und auch hier ist wieder nur der höchste jemals gemessene Wert relevant; aus dem Kurvenverlauf des Histogramms lassen

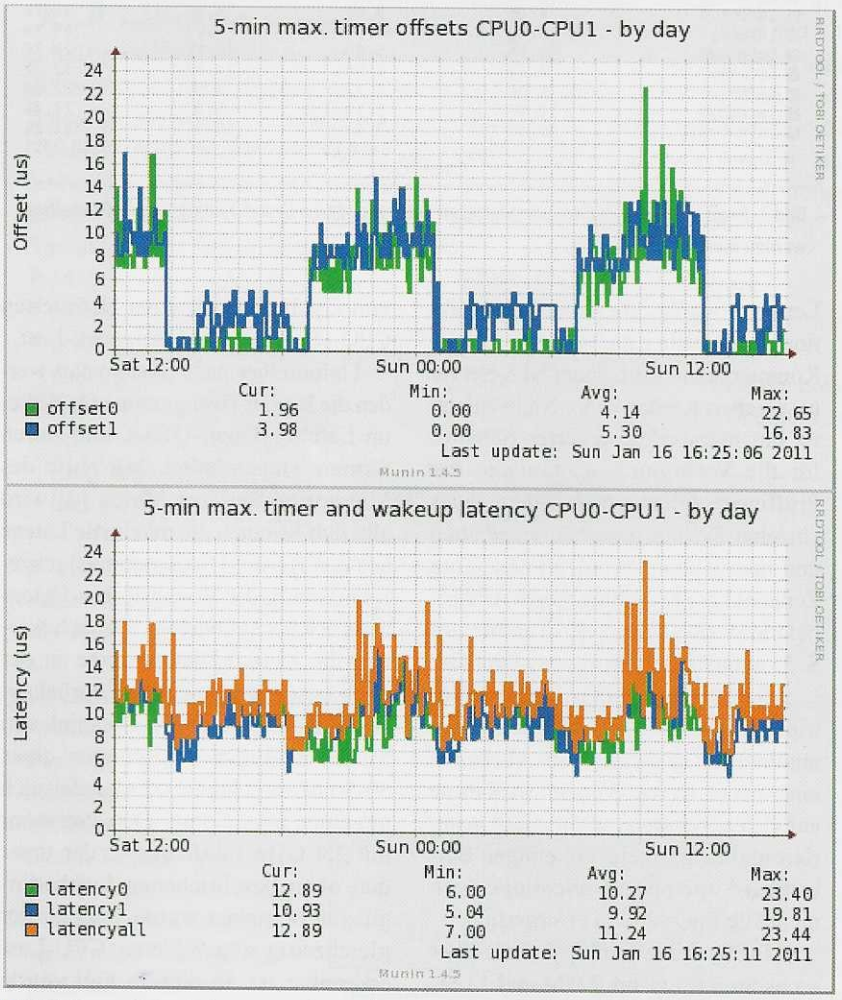

Bild 1. Langzeit-Registrierung von Systemdaten und Echtzeit-Parametern mit Hilfe von Munin. Die im 5-Minuten-Takt registrierten Werte zeigen eine normale Reaktion der Latenzen auf die verschiedenen Lastbedingungen.

sich aber auch wichtige Rückschlüsse auf die Stabilität der Echtzeit-Fähigkeit und auf die Ursache möglicher verlängerter Latenzen ableiten. Von der sehr großen Anzahl an Kommandozeilen-Optionen des Programms Cyclictest ist für die Routine-Messung nur eine kleine Anzahl bedeutsam. Für Single-Core- bzw. Multi-Core-Systeme werden folgende Optionen verwandt:

- Single-Core-Systeme: Mit der Kommandozeile `cyclictest -l100000000 -m -a0 -t1 -n -p99 -i200 -h200 -q` werden eine Gesamtzahl von 100 Millionen Messzyklen, eine Priorität von 99, ein Mess-Intervall von 200 µs und 200 Histogrammzellen mit einer Granularität von 1 µs gewählt. Die Option „-n" führt zur notwendigen Verwendung des hochauflösenden Timers, und die Option „-m" sorgt dafür, dass der vom Prozess verwendete Speicher niemals ausgelagert werden kann. Die gewählten 100 Millionen Messzyklen von jeweils 200 µs führen zu einer Gesamtdauer der Messung von 5 Stunden und 33 Minuten.
- Multi-Core-Systeme: Mit der Option „-S" der Kommandozeile `cyclictest -l100000000 -m -Sp99 -d0 -i200 -h200 -q` wird zusätzlich symmetrisches Multiprocessing (SMP) konfiguriert. Dies führt dazu, dass genauso viele Threads gestartet werden, wie Cores vorhanden sind, und alle Threads mit der gleichen Priorität von in diesem Fall 99 laufen. Mit der Option „-d0" wird dafür gesorgt, dass alle Threads komplett zeitgleich ablaufen, was eine besonders große Herausforderung an die Echtzeit-Fähigkeit darstellt.

Das OSADL Echtzeit-Testzentrum

Im OSADL Echtzeit-Testzentrum sind die Prüflinge auf einzeln entnehmbaren Testtabletts in Racks von jeweils acht Systemen angeordnet. Diese Testracks wurden vom Linux-Kernel-Entwickler Thomas Gleixner konzipiert und sind speziell für Embedded-Systeme und für hardwarenahe Entwicklungsarbeiten vorgesehen. Auf den Testtabletts befinden sich je eine standardisierte Netzwerk- und Seriell-Verbindung. Diese sind rackseitig mit einem portmirrorfähigen Netzwerk-Switch und einem seriellen Netzwerk-Adapter verbunden.

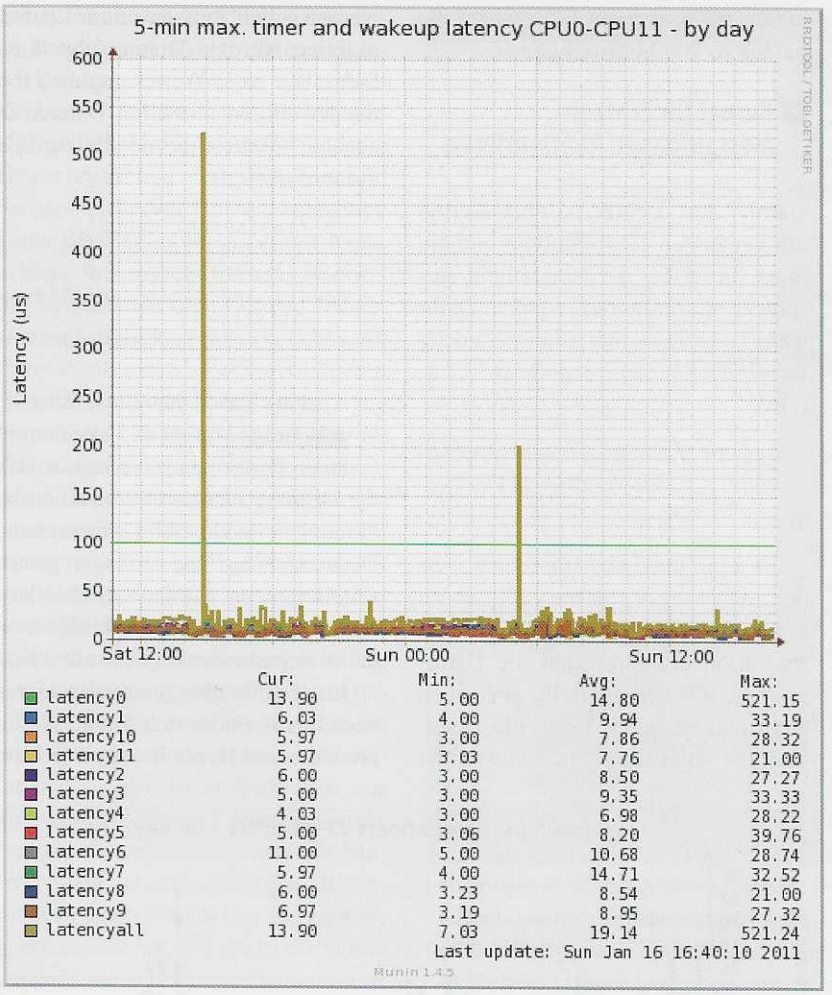

Bild 2. Registrierung von zwei unerwarteten Latenzen, deren Ursache analysiert und beseitigt werden muss.

Letzterer verfügt über eine Log-Funktion, mit welcher die gesamte serielle Kommunikation auf einem NFS-Server gespeichert werden kann. Nicht zuletzt ist ein netzwerkgesteuerter Schalter für die Versorgungsspannungen der Prüflinge vorhanden, mit dem jedes einzelne System gestartet, angehalten und neu gebootet werden kann. Zum Zeitpunkt der Erstellung dieses Artikels befanden sich die in der **Tabelle** auf S. 55 aufgelisteten Prozessoren im Test.

Alle Systeme laufen im Dauerbetrieb, sämtliche Messabläufe werden automatisch gesteuert. Bei Vorliegen einer neuen Kernel-Version wird diese auf vielen Systemen automatisch generiert und eingespielt. Bei einigen Embedded-Systemen ist allerdings noch manuelle Intervention erforderlich.

Alle Systeme laufen jeweils über sechs Stunden (1 bis 7 Uhr und 13 bis 19 Uhr) unter Leerlaufbedingungen und in der übrigen Zeit unter Last durch Cyclictest bzw. in Last-Kombination von Cyclictest mit einer definierten CPU-, Speicher- und Netzwerk-Last.

Unmittelbar nach Systemstart werden die Kernel-Histogramme für Wakeup-Latenz, Timer-Offset und deren Summe eingeschaltet. Mit Hilfe des Monitoring-Systems Munin [2] wird alle fünf Minuten die maximale Latenz aus den Kernel-Histogrammen ausgelesen und in der Round-Robin-Datenbank RRD [3] abgelegt. Danach werden die Kernel-Histogramme in der oben beschriebenen Weise zurückgesetzt. **Bild 1** zeigt als Beispiel von Munin erzeugte Abbildungen eines 30-Stunden-Zeitraums. Es handelt sich um einen Intel-Core-2-Duo-Prozessor mit 2,4 GHz Taktfrequenz, der unter den oben beschriebenen Lastbedingungen betrieben wurde, was an der gleichzeitig abgebildeten CPU-Last erkennbar ist. In diesem Fall wurde während der Leerlaufphase noch ein vierstündiger Test einer Echtzeit-Ethernetverbindung dazwischengeschaltet,

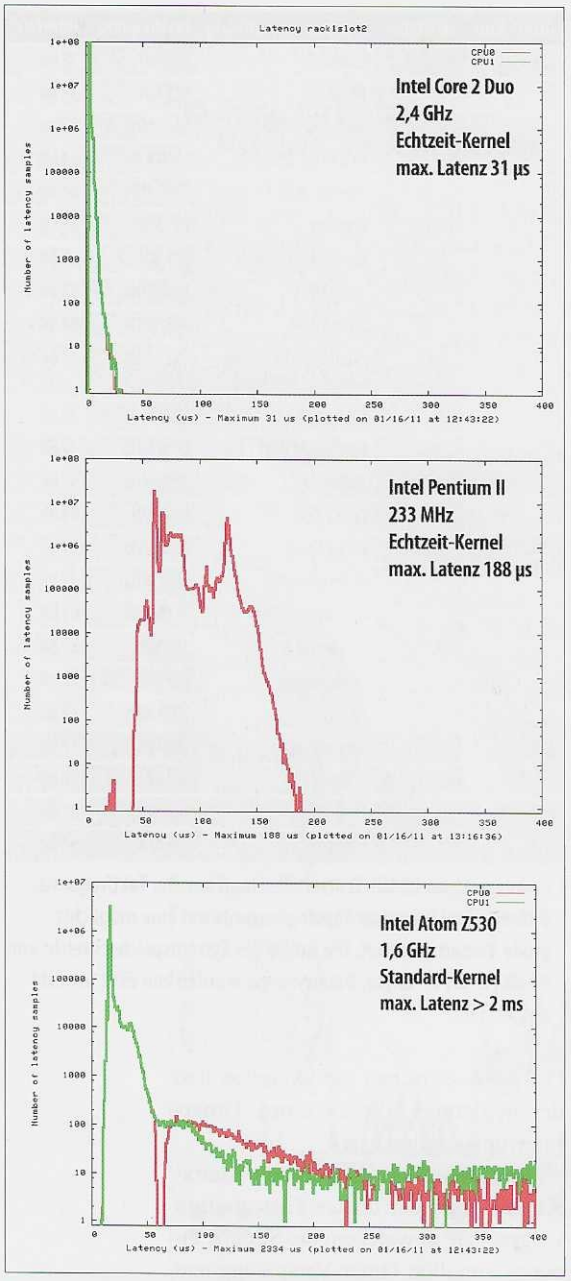

Bild 3. Mit dem Programm Cyclictest gewonnene Latenz-Plots. Zur besseren Vergleichbarkeit wurde in allen drei Plots die gleiche Skalierung gewählt. Grüne und rote Linien stehen für die verschiedenen Cores – oben und unten jeweils ein Dual-Core-Prozessor.

der zu einer erhöhten Interrupt-Frequenz, nicht aber zu einer sichtbaren CPU-Belastung führt. Sollte einmal eine unerwartete Latenz auftreten, deren Ursache gesucht und beseitigt werden muss, ergibt sich z.B. der in **Bild 2** gezeigte Kurvenverlauf. Deutlich erkennt man die gegen 15 und 5 Uhr aufgetretenen Latenzen der CPU Nummer 0 von 520 bzw. 200 µs.

Am Ende des Cyclictest-Programms werden die Histogramme ausgelesen und graphisch dargestellt. Dabei wird – wie allgemein empfohlen – die Anzahl an Messwerten pro Latenz-Kategorie in der y-Achse logarithmisch dargestellt, während die Latenzwerte auf der x-Achse linear skaliert werden. **Bild 3** zeigt auf diese Weise gewonnene Latenz-Plots von zwei mit Echtzeit-Kerneln und einem mit Standard-Kernel ausgerüsteten Systemen.

Um auf besondere Vorkommnisse im Testzentrum rasch reagieren und alle Ereignisse langfristig archivieren zu können, wurde das Munin-System mit Nagios-Monitoring [4] verbunden. Zum einen lässt sich Nagios dahingehend konfigurieren, dass je nach Art des Ereignisses E-Mail, SMS, Telefax oder ähnliches an eingetragene Kontaktpersonen versandt werden. Zum anderen verfügt Nagios auch über sehr weitgehende Archivier- und Suchfunktionen, mit denen einzelne Ereignisse im Zeitverlauf analysiert und charakterisiert werden können. OSADL-Mitgliedsfirmen können diese Archivdaten für ihr internes Qualitäts-Management, für MTBF-Bestimmungen und für Zertifizierungen nutzen.

Ergebnisse online verfügbar

Auf der OSADL-Website [5] können Arbeitsweise und Ergebnisse des Testzentrums von jedermann verfolgt werden. Es sind sowohl die kontinuierlichen Latenzregistrierungen als auch die Latenzplots der Testsysteme verfügbar. Die Daten werden unmittelbar nach Vorliegen auf den OSADL-Internetserver übertragen; bei den Latenzregistrierungen beträgt das Update-Intervall 5 Minuten, bei den Latenzplots 12 Stunden.

Damit die Ergebnisse des OSADL-Testzentrums an anderer Stelle reproduziert werden können, werden alle relevanten Daten der Testsysteme ebenfalls kontinuierlich generiert und auf der genannten OSADL-Website alle 12 Stunden aktualisiert. Im einzelnen handelt es sich dabei um:

- Name und Hersteller des Mainboards,
- Version und Hersteller des BIOS,
- Distribution,
- Kernel-Version, Kernel-Parameter, Kernel-Konfiguration,
- Kernel-Parameter,
- Cyclictest-Kommandozeile,
- Charakteristika, Cache und Taktfrequenzen der CPU,
- Auflösung der Timer,
- Größe und Kenndaten des Systemspeichers, DIMM-Analyse,
- installierte Peripheriesysteme am PCI-Bus.

Ausgereifte Kernel-Version

Während der Stabilisierungsphase der kürzlich freigegebenen „Latest Stable"-Echtzeit-Version des Linux-Kernels 2.6.33 wurde erstmals auf die Daten des OSADL Echtzeit-Testzentrums zurückgegriffen. Diese Kernel-Version wurde erst dann als „Latest Stable" freigegeben, nachdem sämtliche in den Testsystemen aufgetretenen Fehler beseitigt waren und alle Systeme über mehrere Wochen fehlerfrei und unter Einhaltung der vorgegebenen Echtzeit-Bedingungen lauffähig waren. Dies hat zwar zu einem verzögerten Release geführt. Auf der anderen Seite liegt dadurch nun ein echtzeitfähiger Linux-Kernel vor, dessen Eigenschaften und Produktionsreife in weiten Bereichen garantiert werden kann. *jk*

Dr. Carsten Emde
blickt auf eine mehr als 25-jährige Tätigkeit als Software-Entwickler, System-Integrator und Software-Consultant für industrielle Computersysteme zurück. Seit Gründung des OSADL im Jahr 2005 ist er dessen Geschäftsführer.
Carsten.Emde@osadl.org

Referenzen

[1] RT-Tests: http://git.kernel.org/?p=linux/kernel/git/clrkwllms/rt-tests.git
[2] Munin: http://munin-monitoring.org/
[3] Round-Robin-Datenbank RRD: http://www.mrtg.org/rrdtool/
[4] Nagios Monitoring: http://nagios.org/
[5] OSADL Echtzeit-Testzentrum (QA-Farm): http://osadl.org/QA/