Elektronik

Offizieller Medienpartner
embeddedworld2012
Exhibition&Conference
Nürnberg

elektroniknet.de
November 2011

embedded

Fachmedium für die Entwicklung von Embedded-Systemen

4K – Vierfache HDTV-Auflösung auf einem Chip:

FPGA verarbeitet Video in Kinoqualität

>> S. 16

Sonderdruck Linux: Embedded für alle

MEINUNG

„Thunderbolt wird ein Nischendasein führen"
Subu Sankaran, FTDI >> Seite 11

Apple-A5: Motor für das iPhone 4S

>> Seite 51

Multi- und Many-Cores im Griff

>> Seite 26

Linux: Embedded für alle

Immer mehr Embedded-Features werden in den Mainline-Linux-Kernel integriert

Linux ist heute erste Wahl geworden, wenn es um die Entscheidung für ein Betriebssystem in einem leistungsfähigen Embedded-System geht. Wie kann es sein, dass eine Open-Source-Software gerade in einem in jeder Hinsicht kritischen Bereich wie bei Embedded-Systemen so erfolgreich ist? Dieser Artikel beschreibt die Hintergründe und erläutert die einzelnen Aspekte der ungewöhnlichen Erfolgsgeschichte von Linux.

Von Heinz Egger, Casten Emde und Thomas Gleixner

Als Linus Torvalds im August zu Linux schrieb „Wird nichts Großes“ und im Oktober 1991 den Quellcode für Linux 0.02 verfügbar machte, ahnte keiner, dass bereits im März 1992 mit der Version 0.95 ein für viele Anwendungen brauchbares Betriebssystem zur Verfügung stehen würde. Und natürlich konnte auch keiner vorhersehen, dass knapp 20 Jahre später aus den weniger als 200.000 Zeilen Quellcode der Version 0.95 mehr als 13 Millionen Zeilen in Version 3.0 würden. Heute unterstützt Linux praktisch alle relevanten Prozessor-Architekturen und Hardware-Geräte und damit mehr als jedes andere Betriebssystem.

Seit Februar 1992 wird Linux unter der GNU General Public License (GPL) lizenziert. Die mit dieser Lizenz verbundene garantierte Freiheit und uneingeschränkte Verwendbarkeit, die im Gegensatz zur Lizenzierung des damals alternativen Minix stand, waren sicher wichtige Gründe, warum Linux weltweit so viele Mitarbeiter begeistern konnte und eine so rasche Entwicklung nahm. Die Wahl der GNU GPL hat Linux Torvalds 1997 mit den Worten kommentiert: „Linux unter die GPL zu stellen, war eindeutig das Beste, was ich jemals getan habe.“

Bis auf den Desktop-PC hat Linux in praktisch allen Bereichen der Computerbranche eine dominierende Position eingenommen. Den wohl höchsten Anteil hat Linux bei Supercomputern mit über 90 % erreicht; aber auch in anderen Bereichen wird Linux in nennenswertem Maße eingesetzt. Der Marktanteil von Linux in Embedded-Systemen kann nicht genau angegeben werden, weil verlässliche Daten dazu nicht öffentlich verfügbar sind. Allgemein wird aber davon ausgegangen, dass Linux bei zur Zeit neu entwickelten Embedded-Systemen eine Führungsrolle eingenommen hat.

Hürden: Portierbarkeit, Echtzeit-Fähigkeit, Zertifizierung

Die oben genannten frühen Versionen von Linux waren ausschließlich auf Systemen mit Intels 32-bit-Architektur lauffähig, und Linus Torvalds selbst hielt anfänglich eine Portierung auf andere Architekturen für praktisch ausgeschlossen. Aber bereits ab 1994 begannen Aktivitäten, Linux auch auf andere Architekturen zu portieren. Dies betraf etwa gleichzeitig PowerPC- und ARM-Prozessoren, die heute weitgehend uneingeschränkt in allen ihren Subarchitekturen und Facetten von Linux unterstützt werden. Damit war eine wesentliche Voraussetzung für den Einsatz von Linux in Embedded-Systemen erfüllt. Die wohl spektakulärste Portierung führte zu ersten offiziellen Kernel-Patches im Dezember 1999 und bereits wenige Monate später zu einer lauffähigen Distribution: Linux auf IBM-Mainframes. Offensichtlich läuft heute ein bedeutender Anteil dieser Systeme unter Linux. Diese Portierung erhöhte natürlich das Vertrauen in das Linux-Betriebssystem und dessen Portierbarkeit, und es folgten viele weitere Architektur-Ports.

Als eine zweite wichtige Voraussetzung, die ein Betriebssystem für Embedded-Systeme erfüllen muss, ist die Echtzeit-Fähigkeit zu nennen. Zwar gilt diese Anforderung nur für eine Minderheit der heute eingesetzten Embedded-Systeme. Aber es wird häufig diskutiert, dass zukünftige Anwendungen und weiterentwickelte Versionen möglicherweise ein deterministisches Antwortverhalten benötigen, und deshalb entscheidet man sich vorsichtshalber in vielen Fällen gleich für ein Echtzeit-Betriebssystem – speziell auch unter Berücksichtigung der langen Produktlebensdauer von Embedded-Systemen. Dies stellte nun eine besondere Herausforderung für Linux dar; denn bis vor einigen Jahren ging man davon aus, dass ein Echtzeit-Betriebssystem von der ersten Zeile an mit besonderer Berücksichtigung dieser Eigenschaft geschrieben werden muss.

Dies war bei Linux natürlich nicht der Fall, und die nachträgliche Ausrüstung eines Betriebssystem-Kernels mit Echtzeit-Fähigkeit wurde als unmöglich angesehen. Auf der anderen Seite hat aber ein General-Purpose-Betriebssystem mit Echtzeit-Fähigkeit ein extrem großes Potential wie z.B. die folgenden Anwendungen:

- Audio-Recording und Audio-Playback auf Standard-PC-Hardware,
- Video-Recording und Video-Playback auf Standard-PC-Hardware,
- Transaktions-Server mit genauen Zeitstempeln, z.B. Highspeed-Trading,
- Automatisierungs-Industrie.

Der rasche Task- und Kontextwechsel bei Echtzeit-Fähigkeit stellt hohe Ansprüche an die korrekte Serialisierung von Zugriffen auf exklusive Ressourcen des Betriebssystems und der Hardware. Dadurch können Fehler schneller und zuverlässiger aufgedeckt werden, so dass eine allgemeine Qualitätsverbesserung des Kernels eintritt, die auch für den Betrieb ohne Echtzeit-Fähigkeit gilt. Um Linux

Echtzeit-Eigenschaften zu geben, wurden etwa ab 1999 gleichzeitig zwei sehr verschiedene Wege gegangen (**Bild 1**): Single-Kernel und Dual-Kernel.

Echtzeit: erweitert oder eingepflanzt

Mit „Single-Kernel“ wird die ursprünglich als unmöglich angesehene Vorgehensweise bezeichnet, Linux nachträglich mit Echtzeit-Eigenschaften auszurüsten. „Dual-Kernel“ bedeutet, dass ein kleiner Echtzeit-Kernel (so genannter Nano-Kernel) Linux vorangestellt wird, der die echtzeitpflichtigen Aufgaben wahrnimmt, während der Linux-Kernel nur dann Rechenzeit erhält, wenn der Nano-Kernel in der Warteschleife läuft. Typische „Dual-Kernel“-Systeme sind z.B. RTAI, RTLinux, RTCore und Xenomai. Die Linux-Erweiterungen, mit denen ein „Single-Kernel“-Echtzeit-System erzeugt werden kann, werden „PREEMPT_RT-Patch“ genannt. Dieser Name leitet sich von der früheren Form der Konfigurations-Variablen CONFIG_PREEMPT_RT ab, mit der die Echtzeit-Fähigkeit bei der Herstellung des Linux-Kernels eingeschaltet wurde.

Industrietaugliche „Dual-Kernel“-Systeme waren relativ kurzfristig bereits ab 2002 verfügbar und haben sicher einen großen Anteil daran, dass Linux bereits zu diesem Zeitpunkt als sehr attraktiv für Embedded-Systeme angesehen wurde. Der Grund für die relativ rasche Verfügbarkeit lag darin, dass das Funktionsprinzip schon lange bekannt war und auf anderen Systemen wie z.B. DOS und Windows bereits eingesetzt wurde. Nachteil dieses Verfahrens ist aber, dass Linux für diesen Zweck angepasst werden muss und dass diese Anpassungen niemals in den offiziellen Mainline-Linux-Kernel aufgenommen würden.

Die Entwicklung des „Single-Kernel“-Echtzeit-Linux gestaltete sich etwas schwieriger. In einem ersten Schritt mussten sämtliche potentiellen Latenzquellen lokalisiert und Umgehungsmechanismen dafür gefunden werden. Eines der größten Probleme war dabei die so genannte Prioritäts-Inversion, die auftritt, wenn ein Prozess mit hoher Priorität Zugriff auf eine nur einmalig im System vorhandene Ressource benötigt, die von einem Prozess mit niedrigerer Priorität belegt und dadurch gesperrt wird. In einem solchen Fall wird dem höher priorisierten Prozess die niedrigere Priorität aufgezwungen, wodurch eine wesentliche Eigenschaft eines prioritäts-gesteuerten Echtzeit-Systems verlorengeht.

Eine der Methoden, um Prioritäts-Inversion zu verhinden, stellt Prioritäts-Vererbung („Priority Inheritance“, abgekürzt „PI“) dar. Bei diesem Verfahren erhält der blockierende Prozess vorübergehend die Priorität des höher priorisierten Prozesses, so dass das kritische Code-Segment mit seiner eigenen Priorität durchlaufen wird. Das eigentliche Problem bestand aber darin, dass Linus Torvalds die Aufnahme von PI in den Mainline-LinuxKernel zunächst abgelehnt hat. Denn es war bereits damals unumstritten, dass ohne eine effiziente Maßnahme zur Verhinderung von Prioritäts-Inversion niemals ein befriedigendes Echtzeit-Verhalten erreicht werden kann.

Erst auf dem – deshalb legendären – Kernel-Summit in Ottawa im Jahre

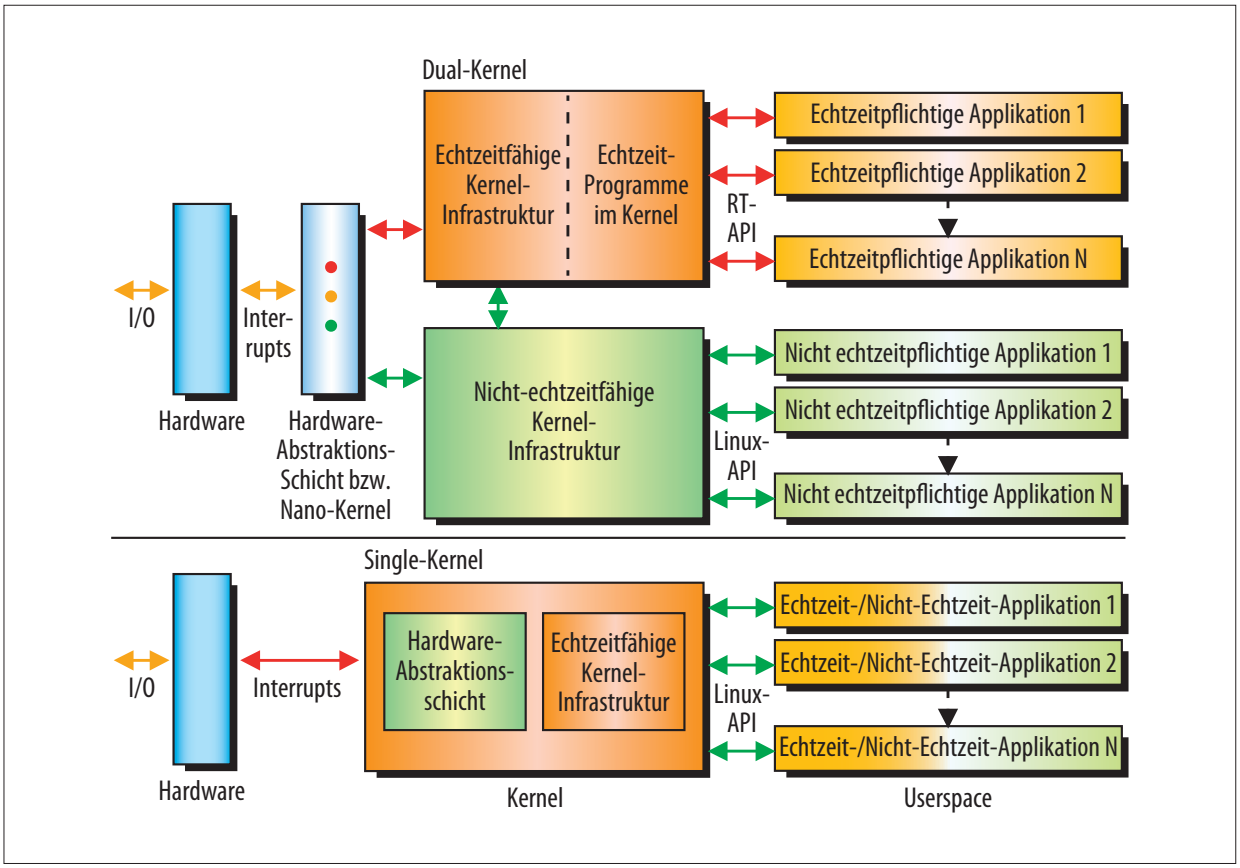

Bild 1. Vergleich der beiden Zugangsweisen, Echtzeit-Fähigkeit des Linux-Kernels herzustellen. (Quelle: OSADL)

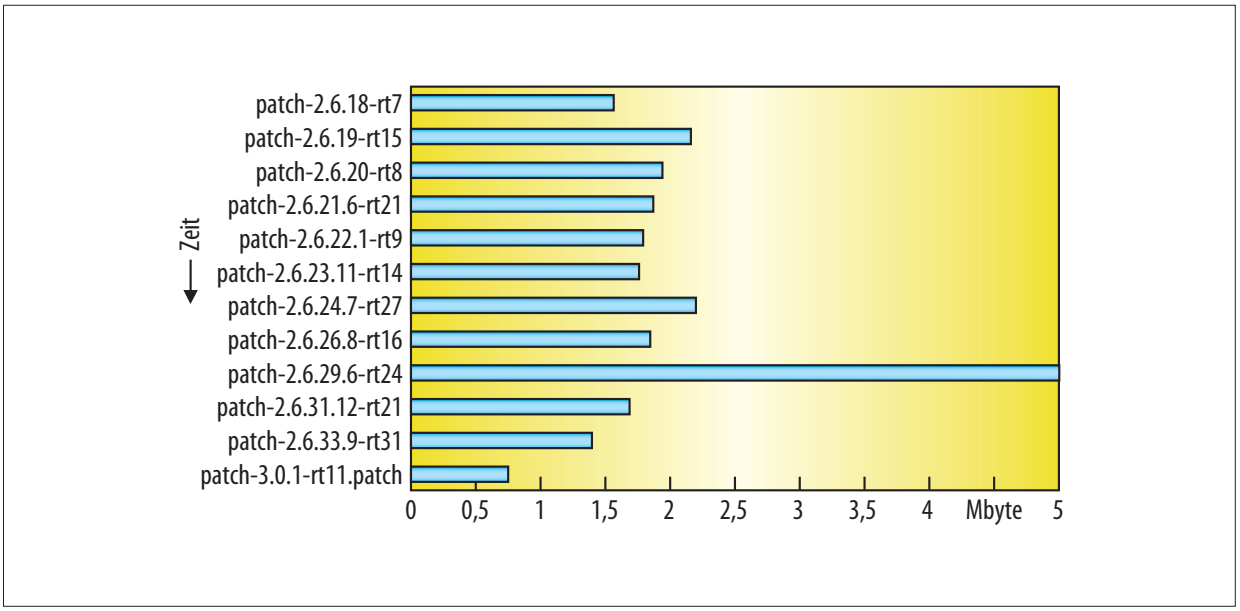

Bild 2. Größenreduktion der so genannten PREEMPT_RT-Patches, mit denen aus Linux ein echtzeitfähiges Betriebssystem wird, von 2006 bis heute.

(Quelle: OSADL)

2006 hat Linus Torvalds der Aufnahme von PI-Mutexen zugestimmt und damit den Weg frei gemacht für den kontinuierlichen Einbau der Echtzeit-Komponenten in den Linux-Kernel. Diese Komponenten sind jeweils in einem Patch zusammengefasst; an der Größe der Patches (**Bild 2**) lässt sich deren kontinuierliche Übernahme in den Mainline-Kernel erkennen: Von anfänglich 1,56 Mbyte konnte der Patch inzwischen auf 750 Kbyte reduziert werden. Deutlich ist aber auch sichtbar, dass zwischendurch neue Komponenten hinzugefügt werden mussten, von denen die meisten durch die Weiterentwicklung des Kernels bedingt waren. Der Ausreißer bei der Version 2.6.29.6-rt24 hängt damit zusammen, dass hier ausnahmsweise sehr viele Komponenten bereits enthalten waren, die für die folgende Mainline-Version vorgesehen waren. An dieser Abbildung ist nicht zu erkennen, dass weit über ein Drittel der verbliebenen 750 Kbyte bereits für die Übernahme nach Mainline vorbereitet ist, so dass in den nächsten Versionen eine weitere deutliche Größenreduktion des PREEMPT_RT-Patch zu erwarten ist, was natürlich die zukünftige Pflege und Integration wiederum sehr erleichtern wird.

Bei Performance-Messungen von Echtzeit-Systemen spielt die maximale Latenz des Systems eine wichtige Rolle. Diese Variable bezeichnet die Zeitdauer vom Eintreffen eines nicht vorhersagbaren, also asynchronen externen Ereignisses bis zu dessen Bearbeitung durch das System. Diese Bearbeitung kann bereits im Treiber (Interrupt-Latenz) oder erst in einem Userspace-Programm (Gesamt-Latenz) erfolgen. Die Interrupt-Latenz ist zwar immer kürzer als die Gesamt-Latenz, aber die Bearbeitung im Treiber ist programmtechnisch schwieriger und nicht portabel; außerdem gestaltet sich die Fehlersuche komplizierter als im Userspace. Als weiterer wichtiger Nachteil ist zu beachten, dass bei einem Fehler im Treiber das gesamte System in Mitleidenschaft gezogen werden kann, während bei der Programmierung im Userspace nur die jeweilige Applikation betroffen ist und das System stabil bleibt. In jedem Fall sollte bei der Evaluation verschiedener Echtzeit-Systeme in Bezug auf deren Latenz darauf geachtet werden, dass nur gleiche Variablen miteinander verglichen werden. Die Interrupt-Latenzen von „Single-Kernel“- bzw. „Dual-Kernel“-Systemen unterscheiden sich z.B. nicht wesentlich, und in beiden Fällen lassen sich auf modernen, dafür geeigneten Prozessoren Latenzwerte im einstelligen Mikrosekundenbereich erreichen.

Mit den genannten Methoden lässt sich also ein echtzeitfähiges Linux herstellen, das den Vergleich mit klassischen Echtzeit-Kerneln nicht zu scheuen braucht, was mit zahlreichen erfolgreichen Industrieprojekten bewiesen wurde. Damit wurde auch die zweite Hürde genommen.

Zertifizierung

Schließlich bleibt noch die Aufgabe, den Einsatz von Linux in sicherheitskritischen Umgebungen zu ermöglichen. Diese letzte der drei wichtigen Hürden ist zur Zeit weniger gut als die ersten beiden gelöst; denn der Linux-Kernel wurde nun einmal nicht nach einem zertifizierten Verfahren entwickelt, und es ist praktisch ausgeschlossen, dies nachzuholen. Allerdings bahnen sich auch hier Lösungsmöglichkeiten an. Denn bestimmte Zertifizierungsverfahren erlauben durchaus die Verwendung vorbestehender Software, fordern aber eindeutige Belege und Dokumentation, dass die Herstellung der Software nach einem vergleichbaren Verfahren erfolgte – insbesondere, was den Prozess des Code-Reviews und des Bug-Trackings anbelangt. Viele Konzepte bei der Entwicklung des Linux-Kernels sind in der Tat ebenbürtig, und nicht zuletzt zeigt die hohe Qualität des Quellcodes, dass diese Konzepte effizient sind. Es geht also unter anderem darum, diese Prozesse zu dokumentieren.

Dass es möglich ist, Linux in sicherheitskritischen Umgebungen zu verwenden und entsprechende Zertifizierungen zu erhalten, zeigen die weltweit zahlreichen Linux-Projekte, die bereits erfolgreich zertifiziert wur-

den. Dabei handelt es sich unter anderem um Steuerungen für Eisenbahnen und im Automotive-Bereich. Es gibt inzwischen sogar Stimmen, die den Einsatz von Linux in sicherheitskritischen Systemen wegen der hohen Qualität der Software und der ungleich großen Testbasis fordern und diese als überlegen gegenüber proprietären Systemen ansehen.

Wie funktioniert das Geschäftsmodell?

Das Geschäftsmodell fast aller Open-Source-Projekte basiert auf „Open Innovation“. Mit diesem Begriff wird die Zusammenarbeit von unabhängigen Unternehmen und Institutionen auf einem Gebiet bezeichnet, das für keinen Teilnehmer relevant in Bezug auf seine Marktposition ist. Bei den Teilnehmern kann es sich durchaus um Mitbewerber handeln. Die Wertschöpfung findet im Fall der Open-Source-Software nicht durch den Vertrieb der Software selbst statt; denn diese kann von jedermann ohne Einschränkungen genutzt werden. Vielmehr liegt die Wertschöpfung zum einen beim Hersteller, bei dem auf diese Weise ausgerüstete Systeme einen Marktvorteil bewirken. Zum anderen erfolgt eine Wertschöpfung bei Software-Dienstleistern, die Open-Source-Software um die primär nicht vorhandenen Dinge wie Einrichtung, Schulung, Programmpflege, Gewährleistung, Testumgebung, Zertifizierung usw. ergänzen. Gemeinsam sind aber alle Beteiligten an der Weiterentwicklung der Software selbst interessiert und tragen entsprechend dazu bei. Als Basistechnologie und „Open Source“-Projekt bietet sich Linux in idealer Weise für „Open Innovation“ an und ist vermutlich unter anderem deswegen so erfolgreich.

Wenn ein Unternehmen sich für den Einsatz von Linux in Embedded-Systemen entscheidet, kann dies auf praktisch allen Ebenen innerhalb der Entwicklungs-Hierarchie von vollständiger Eigenverantwortung bis zur Verwendung einer fertigen Distribution erfolgen. Das Unternehmen kann z.B. den Linux-Kernel aus dem Internet beziehen und diesen selbstständig an die verwendete Hardware anpassen und die anderen benötigten Komponenten hinzufügen. Wenn nötig, kann dafür jederzeit die Unterstützung durch einen dafür ausgewiesenen Linux-Dienstleister in Anspruch genommen werden. Im anderen Extremfall kann man eine auf bestimmte Hardware angepasste gebrauchsfertige Distribution für Embedded-Systeme verwenden, für die vom Hersteller in der Regel auch Support angeboten wird. Allerdings sollte man im letzteren Fall darauf achten, dass die Distribution kei-

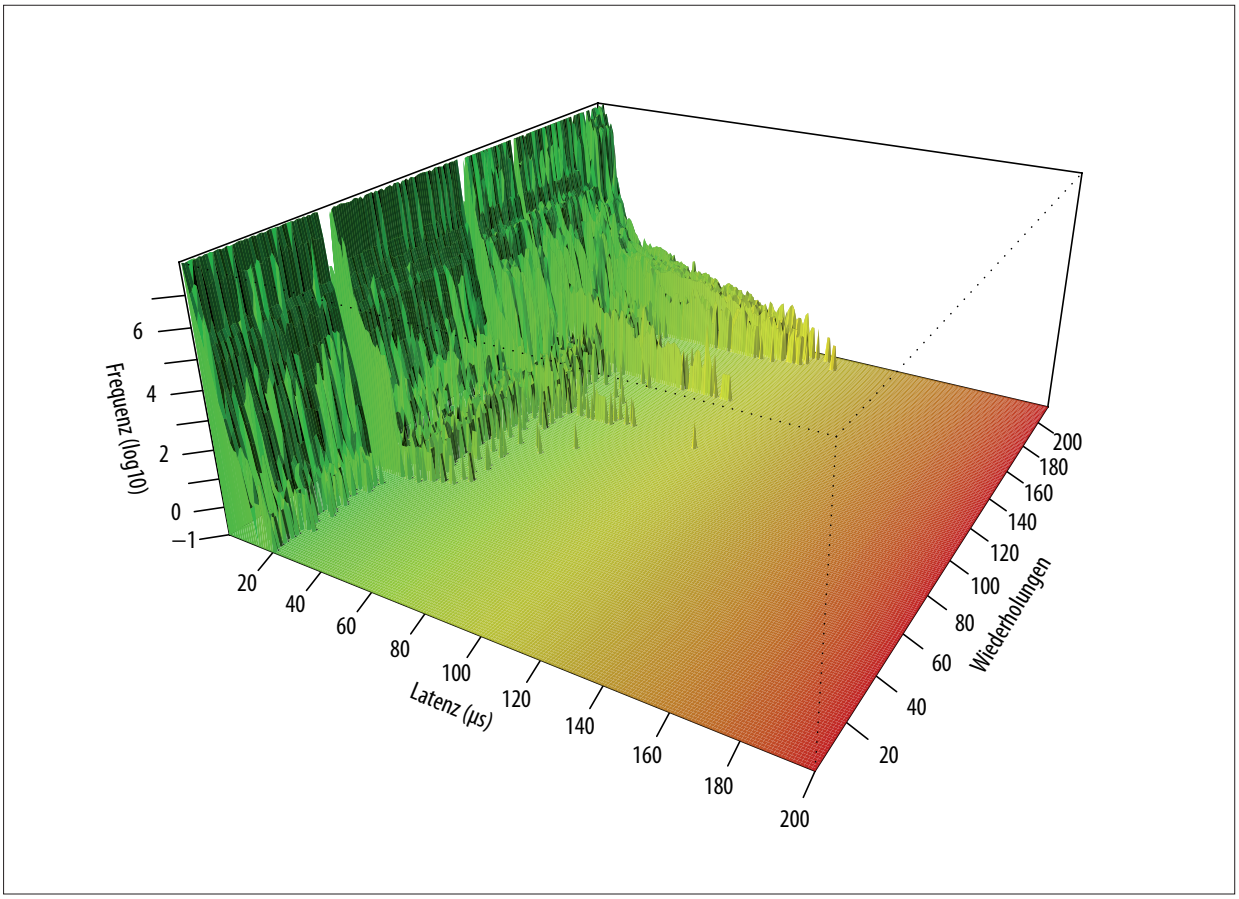

Bild 3. Langzeitmessung der maximalen Gesamt-Latenz des Systems am Beispiel des Intel-Sechs-Core-Prozessors i7-2600K, der mit 3,40 GHz getaktet ist. Abgebildet sind konsekutive Latenzhistogramme mit einer Auflösung von 1 µs, die jeweils über 5 Stunden und 33 Minuten mit einem Zyklusintervall von 200 µs gemessen wurden. Dabei wurden für jedes Histogramm 100 Millionen Messwerte pro CPU aufgezeichnet; die gesamte Abbildung basiert also auf über 20 Milliarden Einzelmessungen pro CPU. Die Zeitachse verläuft von hinten nach vorn. Deutlich erkennt man die zunehmende Optimierung von CPU-Einstellungen im BIOS, Kernel-Konfiguration und Laufzeit-Bedingungen, bis schließlich kein einziger Maximalwert eine Gesamt-Latenz von 20 µs überschreitet. (Quelle: OSADL)

ne versteckten Elemente enthält, mit denen der Wechsel auf eine andere Ebene erschwert wird, z.B. weil die Installation lizenzierte Tools enthält. Denn gerade diese Möglichkeit, die Kompetenzebene zu wechseln, stellt einen wichtigen Vorteil der Open-Source-Entwicklung dar. Und auch dafür bieten Linux-Dienstleister ihre Unterstützung an.

Software-Entwicklung und Qualitätskontrolle in Gemeinschaftsarbeit

Für bestimmte Branchen und Anwendungsgebiete besteht zudem die Möglichkeit, dass die beteiligten Unternehmen sich enger zusammenschließen und gemeinsam Projekte finanzieren; auch dies gehört zu den Aktivitäten von „Open Innovation“ und bietet sich entsprechend für Linux an. Einen derartigen Zusammenschluss stellt das Ende 2005 gegründete Open Source Automation Development Lab (OSADL) dar. Dieses als Genossenschaft eingetragene Unternehmen hat das Ziel, die Entwicklung von Open-Source-Software für den Maschinen- und Anlagenbau und für die Automatisierungsindustrie zu fördern und zu koordinieren. Mitglied kann weltweit jedes Unternehmen werden. Die gemeinsamen Aktivitäten betreffen Software-Entwicklung und -Zertifizierung sowie Vermittlung von Rechtsberatung und die Organisation von Schulungsveranstaltungen, Konferenzen und Workshops.

Seit etwa einem Jahr betreibt OSADL ein Testzentrum, in dem viele verschiedene Linux-Echtzeit-Systeme auf ihre Eignung untersucht werden. Unter anderem werden kontinuierliche Latenzmessungen durchgeführt, mit denen die Echtzeit-Fähigkeit der Systeme geprüft und optimiert wird. Dabei stehen die Hardware-Komponenten, eventuell vorhandene Firmware und auch der Linux-Kernel auf dem Prüfstand. Die meisten Messergebnisse des Testzentrums sind öffentlich online verfügbar (osadl.org/QA). **Bild 3** zeigt exemplarisch den Verlauf einer Optimierung bei einem modernen Mehrkern-Prozessor. In allen inzwischen über 40 sehr verschiedenen Systemen des Testzentrums zeigt der Linux-Kernel mit dem PREEMPT_RT-Patch hervorragende Echtzeit-Eigenschaften und bestätigt unter anderem auch hiermit seine ideale Eignung für Embedded-Systeme. *jk*

Dipl.-Ing. Heinz Egger
studierte Elektrotechnik an der TU München und blickt auf eine über 25-jährige Erfahrung mit industriell eingesetzten Embedded-Systemen zurück. Seit 2006 als geschäftsführender Gesellschafter der linutronix GmbH tätig mit Fokus auf Vertrieb/Marketing.
Heinz.Egger@linutronix.de

Dr. Carsten Emde
hat sich bereits seit den Anfängen der Mikrocontroller mit Embedded-Systemen beschäftigt und für Forschungszwecke genutzte echtzeitfähige Systeme, zunächst unter OS-9, entwickelt und programmiert. Inzwischen heißt sein bevorzugtes Betriebssystem Linux und bestimmt den größten Teil seiner beruflichen Tätigkeit als Software-Entwickler, System-Integrator und Trainer. Seit Gründung des Open Source Automation Development Lab (OSADL) eG ist er dessen Geschäftsführer.
C.Emde@osadl.org

Thomas Gleixner
blickt auf eine über 25-jährige Erfahrung mit industriell eingesetzten Embedded-Systemen zurück. Neben seiner Tätigkeit als geschäftsführender Gesellschafter der linutronix GmbH ist er einer der führenden Kernel-Entwickler (v.a. für Echtzeit-Linux), Mitglied im Technical Advisory Board der Linux Foundation und technischer Berater der Linaro.
tglx@linutronix.de