Interview mit dem Scheduling-Spezialisten Con Kolivas

Der Desktop profitiert

Obgleich eigentlich Arzt, zählt der Australier Con Kolivas (links im Bild) zu den Performance-Spezialisten unter den Kernelprogrammierern. Für Linux 2.4 gab er eine eigene Patchsammlung heraus, schrieb einen Benchmark und zeichnet für wichtige Scheduling-Algorithmen im Kernel 2.6 verantwortlich. Timo Hönig

Con Kolivas (33) ist Herausgeber von die Kernelgeschwindigkeit optimierenden Patches und Autor des Benchmarks Contest. Zuletzt hat er wichtige Teile am Scheduler vom Kernel 2.6 neu programmiert. Con ist eigentlich Facharzt für Anästhesie und hat keine formale EDV-Ausbildung. Er lebt mit seiner Frau Despina und dem 2-jährigen Sohn Lucas in Melbourne, Australien.

Linux-Magazin: Con, was verbindet dich mit dem Linux-Kernel?
Kolivas: So richtig los ging es mit meinem Patchset für den Kernel 2.4.18. Ich wollte ursprünglich nur vorhandene Performance-Patches zusammenführen, um das Verhalten von Linux-Desktops zu verbessern. Daraus entstand der allgemeine Kernel »-ck« mit besserem Antwortverhalten und Interaktivität. Um den Erfolg der Patches zu belegen, hatte ich noch den Benchmark Contest (Achtung: ein Wortspiel) entwickelt. Mein jüngster Beitrag sind die Interaktivitäts-Patches für den Standardkernel 2.6.

Linux-Magazin: Das Ergebnis von Ingo Molnars zweijähriger Entwicklungsarbeit am O(1)-Scheduler vom Kernel 2.6 sind dramatische Verbesserungen. Was genau hast du dazu beigetragen?
Kolivas: Am Ende des 2.5-Zyklus mehrten sich auf der Kernel-Mailingliste die Beschwerden über die System-Interaktivität bei Linux-Desktops. Ein Problem wurde immer wieder genannt: die holprige Soundwiedergabe. Wie es aussah, hatten Scheduler-Änderungen zur Verbesserung der X11-Performance bei Last das Soundproblem sogar verschärft.
Ich fand es peinlich, mit dem sonst so tollen Kernel MP3-Dateien nicht ohne Aussetzer auf einem P4 mit 2,5 GHz abspielen zu können! Da ich viel über die Interna des Schedulers wusste, war ich ziemlich sicher, die Ursache zu kennen, und habe gleich ein paar Variablen optimiert. Da niemand mitmachen wollte, habe ich allein weitergeforscht.
Ich habe dann drei Monate an den bis dahin wenig effektiven Algorithmen des Interactivity Estimator experimentiert. Ich habe mich für Ingos Basisdesign entschieden und eigene Algorithmen entwickelt. Als meine Patches stable waren, hat sie Andrew Morton in den offiziellen Kernel 2.6.0-test6 aufgenommen.

Was tun, wenn die CPU glüht

Linux-Magazin: Kann es nach Erscheinen von 2.6.0 nötig werden, den Scheduler zu verändern?
Kolivas: Wahrscheinlich nicht. Ich habe sehr viel Arbeit in Problemlösungen investiert. Ein Beispiel für meine Herangehensweise: Ich musste entscheiden, was bei CPU-Überlast zu tun ist, also in Situationen, wenn die Systemlast absurde 30 erreicht, wo 0 bis 1 normal sind. Mit meinem Design steigt die Latenz CPU-gebundener Tasks proportional zur Last, bei interaktiven Tasks jedoch nicht. Sinnvoll ist es, interaktive Tasks vorrangig zu behandeln, da etwa X11 bedienbar bleibt. Die Situation verschlechtert sich erst mit einsetzendem Swapping. Auf meiner Maschine mit 1 GByte RAM bleibt XFree bis zu einer Kernellast von 112 nutzbar, dann swappt sie.
Insgesamt beeinträchtigt mein Patch den Durchsatz nicht, CPU-intensive Tasks laufen sogar etwas schneller. Schlecht ist, dass die Latenz der CPU-intensiven Tasks stark steigt, sobald interaktive Prozesse ins Spiel kommen. Zum Glück brauchen CPU-Bund-Tasks aber meist keine niedrige Latenz. Letztlich bedeutet Interaktivität kontrollierte Unfairness – die sich mit der Zeit ausgleicht.
In Spitzen kann die Ungerechtigkeit aber überhand nehmen. Beispielsweise bedeutet ein Nice-Wert nicht nur „Diese Task wird nie von einer niedriger priorisierten Task unterbrochen“ oder „Diese Task erhält *X* Prozent CPU-Zeit“. Die eigentliche Ursache ist die Unterscheidung des Schedulers zwischen I/O- und CPU-intensiv – und mein Code verschärft sie noch. In der Praxis scheint sich das aber nicht bemerkbar zu machen.

Linux-Magazin: Welche Änderungen in Kernel 2.6 sind die wichtigsten?
Kolivas: Die 2.5-Entwickler haben statt an raffinierten Neuerungen am Beheben der Engpässe im Linux-Design gearbeitet, die der Skalierbarkeit und der Infrastruktur im Wege standen. Angenehmer Nebeneffekt: Man kann äußerlich den Kernel nahezu wie einen 2.4er benutzen. Beim Aufrüsten müssen die Anwender nur wenig tun: Devfs, IDE-SCSI, bei den Modutils und so weiter.
Zwischenzeitlich waren die Arbeiten der Desktop-Performance abträglich – das ist behoben. Ich glaube, dass dieser Kernel ein wichtiges und nötiges Fundament für Entwicklungsarbeiten späterer Kernelgenerationen schafft.

Linux-Magazin: Wie viele Kernel-Builds hast du im letzten Jahr gestartet?
Kolivas: Schätzungsweise allein 2000 während der Entwicklung der Interaktivitäts-Patches. Insgesamt waren es wohl 2500 – fast so viele wie Reboots beim Installieren von Windows (lacht).
Linux-Magazin: Etwa sieben Kernel am Tag – ein netter Durchschnitt. Danke für das Gespräch! ■