Interview Bill Weinberg, Montavista

'Een besturingssysteem niet realtime'

Omdat Linux primair nooit bedoeld is voor realtime toepassingen is volgens Bill Weinberg van Montavista Software benchmarking echt heel belangrijk. Daar komt bij dat volgens onderzoek van Montavista blijkt dat maar tien procent van de embedded applicaties realtime is. En als een embedded systeem geen realtime beperkingen kent is er ook geen rtos nodig. Standaard Linux voldoet dan uitstekend, zo betoogt Weinberg.

HANS VAN THIEL

PT Embedded Systems: U hebt meer dan vijftien jaar ervaring in embedded en realtime systemen. Hard Hat Linux is bedoeld voor realtime embedded applicaties, wordt geleverd door MontaVista op basis van een abonnement, maar is volledig open source en valt net als alle andere Linux-varianten onder de GPL[1]. Wat belet uw klanten om zelf deze producten en diensten te gaan verkopen als ze de kennis eenmaal in huis hebben?

Bill Weinberg: 'Dat zou inderdaad kunnen. Maar niet onder onze naam want Hard Hat en MontaVista zijn beschermde merken. Wij verkopen jaarabonnementen op onze software, met updates en support. Klanten kunnen hun abonnementen vernieuwen maar ook opzeggen. Het is hetzelfde model dat ook Microsoft nu hanteert, maar onze broncode is openbaar en andere partijen die er op door willen ontwikkelen zijn gebonden aan dezelfde voorwaarden als wij. Het grote voordeel van open source, voor klanten, is dat ze niet voor eeuwig aan dezelfde leverancier vast zitten.

Montavista heeft 140 medewerkers en kan alle Linux-veranderingen op de voet volgen. We kunnen evalueren en testen op geschiktheid voor specifieke toepassingen. We geven ook trainingen, bijvoorbeeld in het schrijven van *device drivers*, maar onze kern blijft toch het leveren van softwareproducten voor embedded ontwikkeling. Daarbij ondersteunen we op dit moment een vijftal processorfamilies en meer dan vijfenzeventig borden.'

PT Embedded Systems: U bent eerder in uw carrière, bij Lynx Real Time Systems, verantwoordelijk geweest voor embed-

is zelf

Foto: Jan Bogaerts

ded web-technieken en Java. MontaVista ondersteunt ook de Visual Age Micro Edition van IBM, een IDE voor Java. Het is bekend dat IBM momenteel veel met Linux doet, maar Java is toch niet echt realtime?

Bill Weinberg: 'Java was oorspronkelijk bedoeld voor embedded systemen maar heeft zich intussen gediversificeerd van zakelijke toepassing tot zwaar embedded. IBM/OTI doet veel aan optimalisatie van Java-code. Wat wij toevoegen is, om een voorbeeld te geven, een Java AWT[2] implementatie met Hard Hat Graphics. Een andere toepassing zit in opzetkastjes (set top boxes), waarbij de originele code in C is geschreven en de lokale besturing in Java. De term die wij gebruiken is *intelligent programmable middleware*. Maar Java wordt zeker ook voor hard realtime ingezet. Een bedrijf dat boorapparatuur maakt voor de oliewinning (Varco, zie www.varco.com) regelt de machinebesturing in Java onder Hard Hat Linux. Dat is toch wel een schoolvoorbeeld van een realtime toepassing!'

PT Embedded Systems: Volgens een onderzoek van MontaVista is maar een klein gedeelte van alle embedded systemen realtime.

Bill Weinberg: 'Als je met mensen praat op embedded symposia en beurzen krijg je de indruk dat bijna alle toepassingen realtime zijn. In werkelijkheid is dat maar zo'n tien procent en daarvan is maar de helft hard realtime. Met hard realtime bedoel ik dat de prestatie abrupt daalt na een bepaalde deadline. Als de taak niet binnen een zekere tijdgrens wordt voltooid heeft het geen zin meer. Onder soft realtime vallen alle taken waarvan de prestatie na een bepaalde tijd statistisch afneemt. Video- en audiostromen zijn soft realtime.
Als een embedded systeem geen realtime beperkingen kent is er ook geen rtos nodig. Standaard Linux voldoet dan uitstekend. Daar tegenover staat weer dat met de opkomst van multimedia zelfs een gewone pc realtime wordt.'

PT Embedded Systems: U benadrukt dat embedded ontwikkelaars altijd een goede analyse moeten maken van de tijdsbegrenzingen voor applicaties.

Bill Weinberg: 'Benchmarking is echt heel belangrijk. Er is een duidelijke historische evolutie geweest in embedded ontwikkeling, van assembler naar C en C++ en vervolgens naar een rtos. Elke stap heeft nieuwe code geïnjecteerd, maar de eisen aan een besturingssysteem komen voort uit de realtime voorwaarden van de taken. Unix - en Linux is een variant van Unix - is nooit bedoeld geweest voor realtime toepassingen. Het is een procesgebaseerd[3] besturingssysteem met aanvankelijk alleen ondersteuning voor *threads*[4] in *libraries*. Elk proces had dus een eigen thread *scheduler*[5], vergelijkbaar met de *green threads* in Java. Nu heeft Linux 2.4 weliswaar een *multi-threaded kernel*, maar die kerntaken zijn niet preemptable[6].
Begrijp me goed, het gaat bij embedded systemen niet direct om de besturing van de OS-kern. In een embedded systeem zijn *kernel threads* gewoon taken die door het OS worden opgeroepen ten behoeve van de applicatie, eigenlijk applicatie-threads in *kernel space*. Alle kritieke tijdspaden zijn dus gebonden aan toepassingen, niet primair aan de OS-kern.'

PT Embedded Systems: Maar er is toch ook een realtime standaard voor Unix-systemen, RT POSIX?

Bill Weinberg: 'Het *portable operating system interface* definieert alleen een realtime API maar niet de implementatie daarvan en evenmin de prestaties. Het is een IEEE-standaard die duidelijk door een commissie is ontworpen en hier en daar ook een verwarrende semantiek heeft. Zo beïnvloeden de *signals*[7] uit deel 1b van de POSIX-standaard de threads uit deel 1c. Linux stemt niet echt overeen met POSIX maar is gebaseerd op Unix SVR4[8]. De *inter process communication* (IPC) van Linux implementeert drie of vier van de System V aanroepen, semaforen[9], wachtrijen. De scheduler in Linux is ook hetzelfde. Dat betekent dat elke taak uiteindelijk aan de beurt komt (*fairness*) en dat taken die lang wachten geleidelijk een steeds hogere prioriteit opbouwen (*goodness*). Dat principe werkt uitstekend voor een algemeen multi-programming OS maar juist niet voor een rtos.'

PT Embedded Systems: Maar Linux had toch al aparte categorieën voor realtime taken?

Bill Weinberg: 'Jawel, maar die eigenschap wordt niet goed geïmplementeerd in standaard Linux, althans niet goed genoeg. Een taak met een hoge prioriteit, in SCHED_FIFO of SCHED_RR[10] hoeft dan misschien niet te wijken voor een ander, maar de scheduler gaat nog steeds bij een interrupt alle taken langs, ongeacht de categorie. Het is niet zozeer dat Linux het verkeerd doet, maar dat het langer duurt! Het eerste wat we in Hard Hat Linux hebben gedaan is dat *scheduling*-beleid correct implementeren. Nu wordt gegarandeerd de taak met de hoogste prioriteit uitgevoerd. Er is geen promotie van prioriteiten en geen fairness.
Terwijl we hieraan werkten hebben we nog een bug ontdekt in standaard Linux. De *yield*[11] functie gaf weliswaar het systeem vrij voor een andere taak maar vervolgens kon, in bepaalde gevallen, de eerste taak weer gewoon verder gaan. Natuurlijk hebben we de bug fix doorgegeven aan de Linux ontwikkelgemeenschap. Maar met het toevoegen en implementeren van realtime prioriteiten is nog maar de helft van het probleem opgelost.' ›

PT Embedded Systems: De standaard Linux kernel is niet *preemptable*.
Bill Weinberg: 'Een systeemaanroep met een device driver kan meer dan 100 ms vergen en al die tijd andere taken blokkeren. Een realtime event kan vast komen te zitten achter file service operaties. Dat soort dingen zijn niet acceptabel en er zijn verschillende manieren om dat op te lossen. Nu ontdekte ons engineering-team met Kevin Morgan en Nigel Gamble dat er eigenlijk al een Linux-variant bestond met een multi-processing kernel en preemptability. Dat is Linux voor SMP[12]. Het heeft faciliteiten voor thread-migratie over verschillende processoren en preemptability. SMP Linux heeft eigenlijk de architectuur van een rtos. Je ziet het veel in bedrijfs- en servertoepassingen. Een bijkomend voordeel is dat SMP Linux tot de Linux hoofdstroom behoort. Het is geen speciale afsplitsing, maar levend Linux dat volop in ontwikkeling zal blijven. Wat we met Hard Hat Linux eigenlijk hebben gedaan is SMP Linux aanpassen voor, en overbrengen naar, een systeem met één processor.'

PT Embedded Systems: Linux distributeur Red Hat[13] heeft onlangs gekozen voor RTLinux.
Bill Weinberg: 'Een andere manier om realtime faciliteiten te verkrijgen is om Linux te draaien als een gebruikersproces van een ander OS dat wel hard realtime gedrag vertoont. RT Linux van Finite State machine Labs (www.fsmlabs.com) is zo'n oplossing, maar het is geen Linux en het is gepatenteerd, al is het een vrij bijzonder patent.
Het RTAI-project (real time application interface, www.aero.polimi.it/projects/rtai) heeft een soortgelijke benadering als RTLinux. Het splitsen van de kernel is echter volgens ons niet de beste oplossing, al is het maar omdat je van het Linux-principe afwijkt. Victor Yodaiken van Red Hat is een goede vriend van me, maar op dit punt ben ik het niet met hem eens.'

PT Embedded Systems: U ziet het virtueel geheugengebruik[14] van Linux als een groot voordeel. Maar geeft dat geen extra overhead en is daar bovendien geen aparte hardware-ondersteuning voor nodig?
Bill Weinberg: 'Met virtual memory krijg je vanzelf een strikte scheiding in geheugengebruik tussen verschillende processen. Een van de grootste problemen in embedded toepassingen, die vaak weken, maanden of jaren dezelfde taken uitvoeren, is het per ongeluk overschrijven van geheugen. Dergelijke fouten treden op onder zeldzame condities, hebben onvoorspelbare gevolgen en zijn buitengewoon moeilijk op te sporen. Met virtueel geheugen krijg je gewoon een *page fault* melding als een taak probeert illegaal geheugen aan te roepen. Die bescherming is echt zeer waardevol. *Swapping*[15] kan je als ontwerper zelf beperken en hoeft dus niet tot overhead te leiden. Voor de ondersteuning in hardware is alleen een MMU (memory manager unit) vereist.'

PT Embedded Systems: Het komt dus neer op het goed ontwerpen van gebruikerstaken. De ontwikkelaar moet zelf rekening houden met alle tijdsaspecten.
Bill Weinberg: 'Een besturingssysteem is zelf niet realtime. Een rtos is een besturingssysteem dat realtime taken niet in de weg loopt. Er zijn een paar manieren waarop een ontwerper aan tijdvoorwaarden tegemoet kan komen. De eerste is met snelle processoren of het opvangen van realtime vereisten met speciale hardware. Vergeet niet dat de buitenwereld van machines en apparaten de laatste twintig jaar niet zoveel sneller is geworden, maar de elektronische hardware wel. In veel gevallen zal standaard Linux meer dan voldoende zijn om aan alle realtime eisen te kunnen voldoen. Vooral de 2.4 kernel is buitengewoon goed, in elk geval voor de x86-familie, waarop we bij MontaVista benchmarks hebben uitgevoerd. Als dat nog niet voldoende is kun je er aan denken om Linux selectief aan te passen. Misschien kun je de uit te voeren taak in een device driver implementeren. Een gebruiker kan dit met Linux zelf doen. Probeer dat eens met een bedrijfsgebonden rtos!
Tot slot kan je dan nog kiezen voor een preemptable kern en een scheduler met harde, vast instelbare prioriteiten. Met Linux heb je ontzettend veel mogelijkheden. Je kan je eigen kernel samenstellen en als je dat wilt zou je zelfs je eigen scheduler kunnen schrijven. Ik wil niet beweren dat het iets is voor newbies, maar het kan!' ◘

1 General Public License, de licentie overeenkomst van de Free Software Foundation die open en vrije distributie van onder meer Linux reguleert.
2 AWT (Abstract Window Template) is de Java grafische gebruikersinterface.
3 Een proces is het executieverloop van een programma. In een *multi-programming*-omgeving kunnen meerdere processen (schijnbaar) tegelijkertijd draaien door afwisselend een stukje van de een en dan weer van de ander af te werken. Processen, ook copieën van hetzelfde programma, zijn volledig gescheiden maar kunnen communiceren via IPC (Inter Process Communication) mechanismen.
4 Een thread is het executieverloop binnen een en hetzelfde programma (proces). Threads hebben, in tegenstelling tot processen, direct toegang tot alle hulpbronnen van hun programma (proces). Threads hebben daarom minder overhead dan processen en worden ook wel *light weight processes* genoemd.
5 De scheduler (dienstregelaar) is de module van het OS die de afwisseling van processen dan wel threads bestuurt.
6 Als een hulpbron, proces of thread (tijdelijk) kan worden verwijderd voor ander gebruik is die bron preemptable. Een taak die non-preemptable is moet dus doorlopen tot hij helemaal is afgewerkt en kan andere taken (te) lang ophouden.
7 Een *signal* in Unix 'signaleert' een asynchrone gebeurtenis en fungeert daarmee als een software-interrupt.
8 System Five Release Four
9 Een semafoor is een twee- of meerwaardige vlag waarmee de scheduling van taken onderling is af te stemmen.
10 SCHED_FIFO handelt de threads af op first in, first out basis, SCHED_RR doet dat ook maar geeft elke taak een bepaald tijdsinterval waarna een volgende aan de beurt komt (*round robin*). De normale categorie is SCHED_OTHER.
11 Programmeurs kunnen de 'dienstregeling' beïnvloeden door taken zichzelf te laten blokkeren, slapen, controle te laten opgeven, enzovoort. Deze *cooperative programming style* vergt een zeer goede afstemming.
12 Bij Symmetric Multi Processing (SMP) zijn meerdere (gelijkwaardige) processoren verbonden met een gemeenschappelijk geheugen.
13 Niet te verwarren met Hard Hat. Bill Weinberg vertelde dat er een briefwisseling is geweest met Linux distributeur Red Hat over de naam Hard Hat, maar dat het tot nu toe daarbij is gebleven (een *hard hat* is een bouwvakkershelm).
14 Een logische (virtuele) geheugenruimte wordt in kleine blokken (*pages of segments*) afgebeeld op het echte geheugen. Dit gebeurt automatisch of semi-automatisch.
15 *Swapping* is het omwisselen van *pages* tussen werkgeheugen en secundair geheugen (bijvoorbeeld harde schijf).