



Linux jako real-time systém

Red Hat Czech

Michal Schmidt

mschmidt@redhat.com

Listopad 2008



Část I

Úvod do real-time

Úvod do real-time

- 1 Real-time úloha
- 2 Soft vs. hard real-time
- 3 Real-time operační systémy (RTOS)



Section 1

Real-time úloha

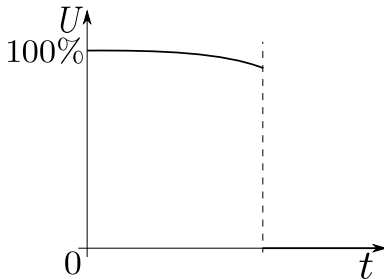
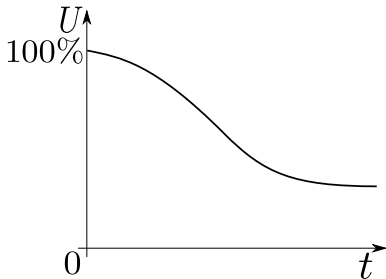
Kdy se jedná o real-time problém

- Na zpracování jsou kladena časová omezení
- Výpočet musí být dokončen před časovým limitem (**deadline**)
- Překročení deadline se považuje za chybu

Real-time **neznamená** dosáhnout co nejvyššího výkonu

Užitková funkce

Užitečnost výsledku v čase





Section 2

Soft vs. hard real-time

Rozdíl z hlediska případných následků nedodržení deadline

- Hard RT – při selhání může být ohroženy životy, zdraví, nebo mohou vzniknout velké škody
- Soft RT – nedodržení deadline není fatální, zhoršená kvalita produktu

(Ale jsou i jiné pohledy na rozdíl mezi hard a soft RT.)

Co není nutně měřítkem hard RT

Velikost časových konstant nehraje důležitou roli při rozlišení soft a hard RT. Není pravda, že hard RT úlohy mají vždy krátké deadline.

- Hard RT úlohy mohou být i relativně pomalé
 - Stroj se přehřeje, když se chlazení nezapne do 1 minuty
- Soft RT úloha naopak s krátkou deadline
 - Čtení příchozích dat ze síťové karty je vhodné stihnout ve zlomku sekundy

Příklady RT úloh

- Zpracování audio/video
- Měření a získávání dat
- Řízení (logické, PID, ...)
- Průmyslová automatizace
- Robotika
- Lékařské stroje
- Auta
- ...

Section 3

Real-time operační systémy (RTOS)

Úplně nejlepší RTOS

Nejlepší RTOS == obejít se úplně bez OS ;-)

- HW vyhrazený pouze pro danou úlohu
 - specializované obvody
 - mikrokontrolery
- Snadno zaručíme předvídatelnost
- Žádné další běžící procesy
- Velmi vhodné pro jednodušší úlohy
- Ale nemáme pohodlí a výhody OS

Operační systémy pro obecné účely

OS pro univerzální účely (jako Linux)

- Většinou optimalizované na vysoký celkový výkon
- Předvídatelnost je až druhořadá
- Mohou poskytovat real-time služby, ale bez záruk (best-effort)
 - funkce POSIX real-time (`SCHED_FIFO`, `SCHED_RR`, ...)
 - Pro mnohé soft RT úlohy stačí

Real-time operační systém

OS specializovaný na real-time úlohy

- Při správném použití dokáže poskytnout záruky, že dodříme deadline
- Předvídatelnost a přesné časování jsou ceněny více než vysoký výkon
- Zajímavé je znát chování v nejnepříznivějším stavu (**worst-case**)
- Používá real-time plánovač procesů (scheduler)
 - Preemptivní plánovač podle priorit
 - Nejbližší deadline nejdříve (EDF)
 - Rate-Monotonic Scheduler

Příklady RTOS

- Free software
 - Speciální: eCos, FreeRTOS, TRON, ...
 - založené na Linuxu: RTLinux, RTAI, Xenomai, **Linux + realtime-preempt patch**
- Proprietární
 - LynxOS, QNX, VxWorks, PikeOS, Windows CE, Win+RTX
 - ...

Měřítko RTOS

Důležité hodnoty jsou

- **interrupt latency** (čekací doba přerušení) – vyvoláno přerušení; za jak dlouho se spustí jeho obslužná rutina?
- **scheduling latency** (čekací doba plánování) – proces s nejvyšší prioritou je připraven k běhu; za jak dlouho dostane přidělen CPU?
- **jitter** (chvění) – nechtěná proměnlivost periody

RTLinux

RTLinux (V. Yodaiken, FSMLabs, nyní Wind River)

- První snaha dostat Linux do světa RT
- RTLinux Free, RTLinux Pro
- Malé RT jádro – Linux běží jako proces s nízkou prioritou
- Patentováno – kontroverzní
- Real-time programy jsou vlastně moduly pro RTLinux jádro, používají jeho API
- Ostatní procesy běží pod Linux jádrem.
- Komunikace RT↔ne-RT prostřednictvím FIFO nebo sdílené paměti

Xenomai

Má za cíl poskytovat API tradičních RTOS, pro snadné portování programů

- Pod Linuxem je nanokernel Adeos/I-pipe
- Různé 'skiny' poskytují různá API pro real-time procesy
 - POSIX, pSOS+, VxWorks, RTAI, ...
- Plánuje se i možnost běhu na Linuxu s realtime-preempt

Obojí jsou řešení s duálním kernelem

Vlastnosti řešení se dvěma kernely:

- Výborné nízké latence (desítky μs) pro RT procesy
- Speciální API pro RT programy, nejsou to normální procesy v Linuxu.

Chtěli bychom RT procesy jako normální linuxové programy v userspace.



Část II

RT-preempt

RT-preempt

4 Preemptivní jádro

5 RT-preempt patch



Section 4

Preemptivní jádro

Preemptivní jádro

- Linux odjakživa plánoval procesy v userspace **preemptivně**
- Až po Linux 2.4.x – **kooperativní** multitasking pro kód **v jádře**
- Práce na škálovatelnosti v SMP – jemnější zamykání v jádře
- Můžeme preemptnout proces v jádře, pokud zrovna není v kritické sekci!
 - Nelze když: drží spinlock, přistupuje k per-CPU proměnným,
...
- Linux 2.6 – **preemptivní jádro** (volba CONFIG_PREEMPT)

Vliv preemptivního jádra

- Maličko větší režie – spinlocky nyní zakazují preempci
- Průměrná plánovací latence se snížila
- Stále dlouhé latence v nejhorších případech

Odkud se berou latence

- Kritické sekce se spinlocky a rwlocky
- Kritické sekce při čtení v mechanismu RCU
- Kód, kde se výslovně zakazují přerušení nebo preempce
- Vykonávání obsluh přerušení a softIRQ
- Inverze priorit



Section 5

RT-preempt patch

RT-preempt patch

- Chce všechny ty zdroje latencí zrušit
- Dělá z Linuxu RTOS
- Normální userspace programy mohou běžet v realtime
- Žádný druhý kernel
- Vývoj: Ingo Molnar, Thomas Gleixner, Steven Rostedt, ...
 - jmenovaní pracují pro Red Hat, umějí spolupracovat s upstreamem, od začátku jasná snaha začleňovat tam postupně části patche

SoftIRQs

Nejsnáze se lze vypořádat s latencemi od softIRQ.

Co jsou SoftIRQs v normálním Linuxu

- Jsou to "spodní poloviny" zpracování přerušení (časovače, síť, TX/RX, ...)
- Běží s povolenými IRQ
- Ale nejsou to procesy – nelze jim sebrat CPU a přidělit ho procesu
- Ochrana proti přetížení od softIRQ
 - `ksoftirqd` – jaderné vlákno pro odložené zpracování softIRQ

Preemptivní SoftIRQs

- Už jsme se smířili s tím, že softIRQ mohou být odloženy na později do vlákna
- Můžeme je tam odložit **vždy**
- A když už, tak rovnou samostatné vlákno pro každý typ softIRQ

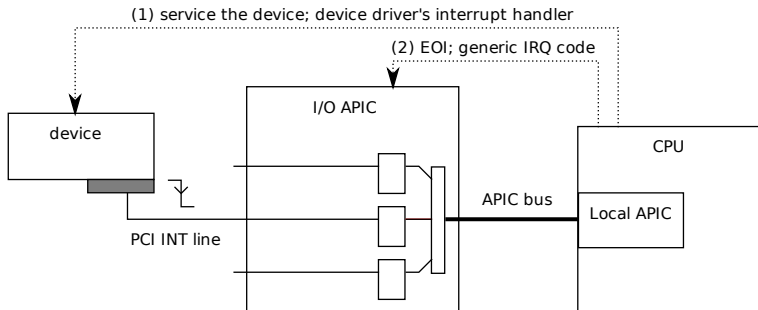
```
$ ps -eo pid,pri,rtprio,cmd
PID  PRI  RTPRIO  CMD
  4   90     50  [softirq-high/0]
  5   90     50  [softirq-timer/0]
  6   90     50  [softirq-net-tx/]
  7   90     50  [softirq-net-rx/]
  8   90     50  [softirq-block/0]
  9   90     50  [softirq-tasklet]
 10   90     50  [softirq-sched/0]
 11   90     50  [softirq-hrtimer]
 12   90     50  [softirq-rcu/0]
```

Obsluha IRQ ve vláknech

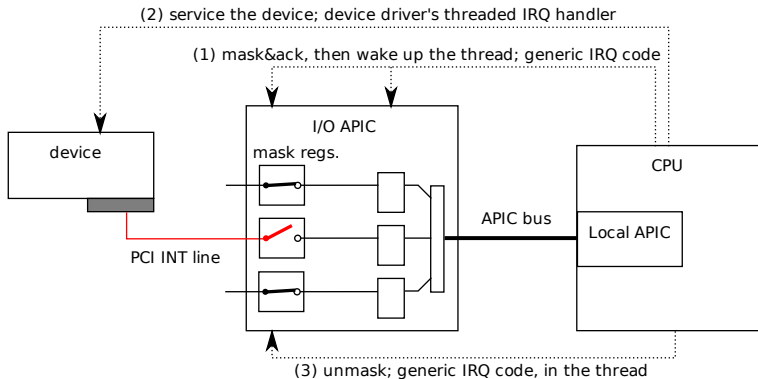
- V normálním Linuxu má obsluha každého IRQ přednost před všemi procesy
- To i tehdy, když proces má realtime prioritu
- Spouštějme obsluhy IRQ také ve vláknech (preemptivní)

PID	PRI	RTPRIO	CMD
304	90	50	[IRQ-8]
347	90	50	[IRQ-15]
381	90	50	[IRQ-12]
382	90	50	[IRQ-1]
393	90	50	[IRQ-4]
400	90	50	[IRQ-16]
401	90	50	[IRQ-18]
402	90	50	[IRQ-17]
413	90	50	[IRQ-19]

Klasická obsluha IRQ



Obsluha IRQ z vlákná



Spící spinlocky

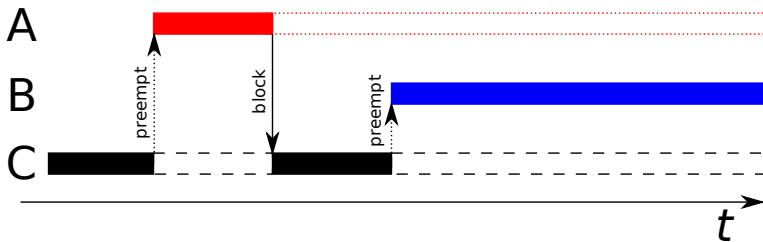
Potřebujeme se zbavit dalšího zdroje latencí – sekcí se spinlocky

- Jsou nutné pro přístup ke sdíleným zdrojům na SMP
- Většinou krátké kritické sekce, ale najdou se i dlouhé
- Potřebujeme, aby byly preemptivní
- Náhrada spinlocků spícími zámky – mutexy
 - Aby to fungovalo, IRQ musí být ve vláknech
 - Jen několik vybraných zůstává jako `raw_spinlock_t`

Inverze priorit

- ...je když proces s vysokou prioritou musí čekat na proces s prioritou nižší
 - Nevyhnutelné, pokud oba přistupují ke sdílenému prostředku – musí počkat, až druhý proces uvolní zámek
- Špatná je ale **neohraničená** inverze priorit

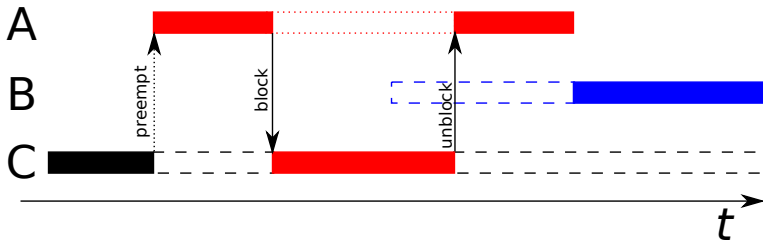
Neohraničená inverze priorit



Dědičnost priorit

- Řešení prioritní inverze
- Když prioritní proces čeká na uvolnění zdroje držného nižším procesem, priorita nižšího procesu je dočasně povýšena na úroveň toho čekajícího
- Tento mechanismus využívají rt-mutexy

Dědičnost priorit



Časovače s vysokým rozlišením

- RT aplikace chtějí jemné časování
- clock sources (zdroje hodin)
 - `gettimeofday()`
- clock events (události od hodin)
 - `nanosleep()`, POSIXové časovače
- Padlo omezení na rozlišení periodického tiků (HZ)
- API používá nanosekundy – skutečná přesnost a rozlišení je dáno schopnostmi HW

Další lahůdky

- Validátor zámků (lockdep)
 - Našel mnoho chyb v zamykání
 - Dokáže detekovat těžko vyvolatelná uváznutí (deadlock), aniž by k nim muselo opravdu dojít
- Sledovač latencí (latency tracer)
 - Také s jeho pomocí bylo nalezeno mnoho chyb
 - Vyvinul se z něj dnešní ftrace (CONFIG_FTRACE)

Výkon

- Latence – desítky μs dosažitelné
- Hrubý výkon o něco snížen, ale ve většině benchmarků jen málo
 - Objeví-li se výraznější propady, řeší se

Začleňování do upstreamu

- Vývoj RT-preempt má jasný cíl dostat se do upstream Linuxu
- Mnohé části už tam jsou
 - obecné semaforey vytlačeny vhodnějšími mutexy
 - jednotná infrastruktura pro IRQ: genirq
 - časovače s vysokým rozlišením
 - robustní futexy a futexy s děděním priorit
 - validátor zámků lockdep
 - preemprivní RCU
 - ftrace
- Nepřímé příznivé dopady
 - Byly odhaleny četné chyby souběhu, opraveny další chyby
- Výhody z existence RT patche plynou všem

Co ještě zbývá začlenit

Stále ještě zbývají dvě nejdůležitější části

- IRQ a softIRQ ve vláknech
 - První patche pro IRQ vlákna již byly zaslány
 - Zatím pouze pro vybrané drivery
 - Umožní zbavit se ošklivých taskletů
 - Možná v 2.6.29
- náhrada spinlocků



Část III

Red Hat Enterprise MRG



Section 6

Messaging + Realtime + Grid

Nasazení realtime-preempt Linuxu

RT patch není jen hračka.

Red Hat Enterprise MRG

- **M**essaging – výkonná implementace otevřeného standardu AMQP (Qpid)
- **R**ealtime – kernel pro předvídatelnou odezvu, nástroje pro vyladění a monitorování
- **G**rid – distribuované výpočty (Condor)
- Vrstvený produkt nad Red Hat Enterprise Linux 5
- Zaručena 100 % binární kompatibilita aplikací
- Podporovaná i real-time Java



Část IV

Závěr

Zdroje

- Rostedt, S., Hart, D. V.: Internals of the RT Patch. *In Proceedings of The Linux Symposium*, Ottawa, 2007, <https://ols2006.108.redhat.com/2007/Reprints/rostedt-Reprint.pdf>
- McKenney, P. E., Attempted summary of "RT patch acceptance" thread, take 2, <http://lwn.net/Articles/143323/>
- McKenney, P. E., A realtime preemption overview, <http://lwn.net/Articles/146861/>
- Edge, J.: Moving interrupts to threads, <http://lwn.net/Articles/302043/>
- <http://rt.wiki.kernel.org>
- <http://www.kernel.org/pub/linux/kernel/projects/rt/>
- <http://www.redhat.com/mrg/>