CE Linux Forum

Realtime BoF Session

RealTime Testing Best Practice of RealTime WG

YungJoon Jung

Contents

- Introduction
- Current RealTime Testing Methods
- Plan

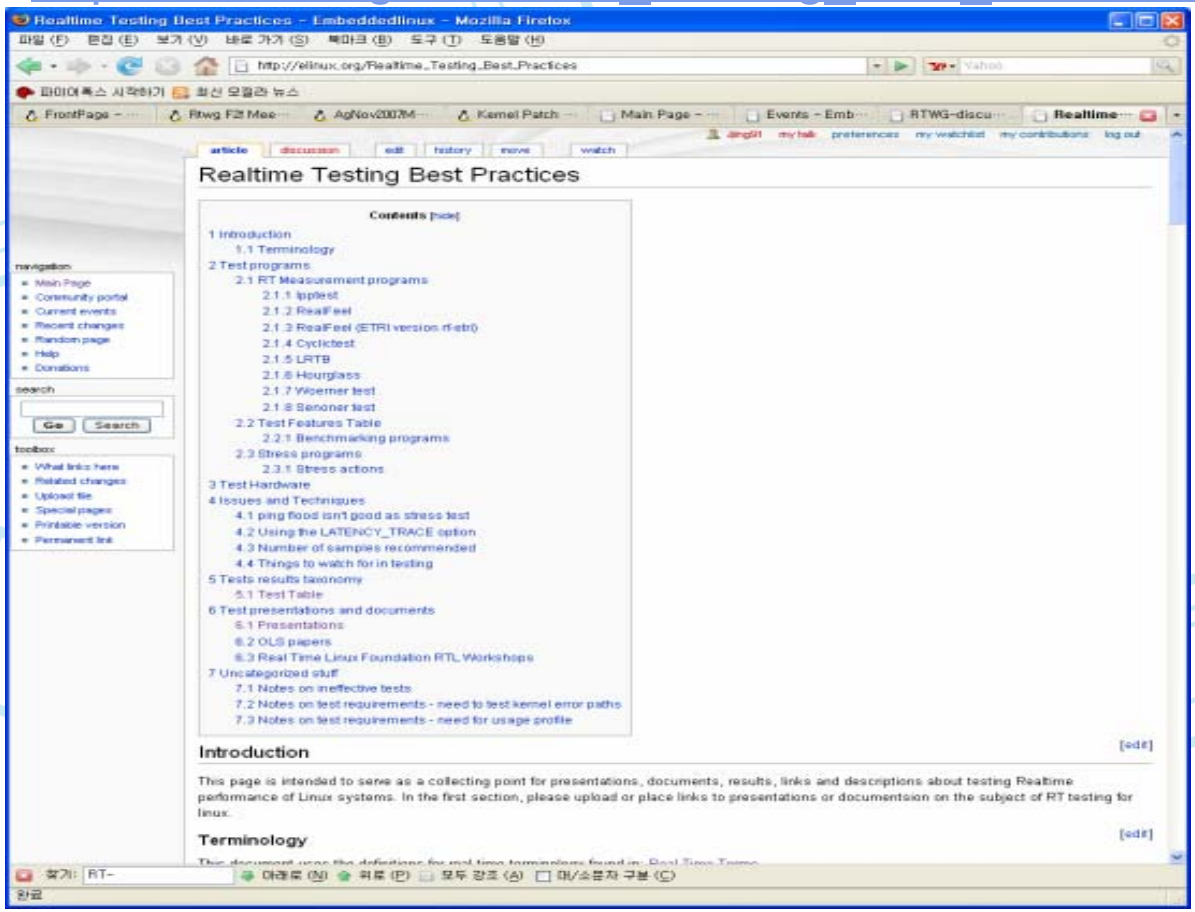
Introduction

Introduction

- CELF RTWG has been collecting realtime testing methods
- What has been collected?
  - Test presentations and documents
  - Test programs
  - Test results
  - Benchmark programs
  - Stress programs and actions
  - Issues and Techniques
  - Etc

Introduction

- You can see our this works at:
  - See http://elinux.org/Realtime_Testing_Best_Practices



Current RealTime Testing Methods

lpptest

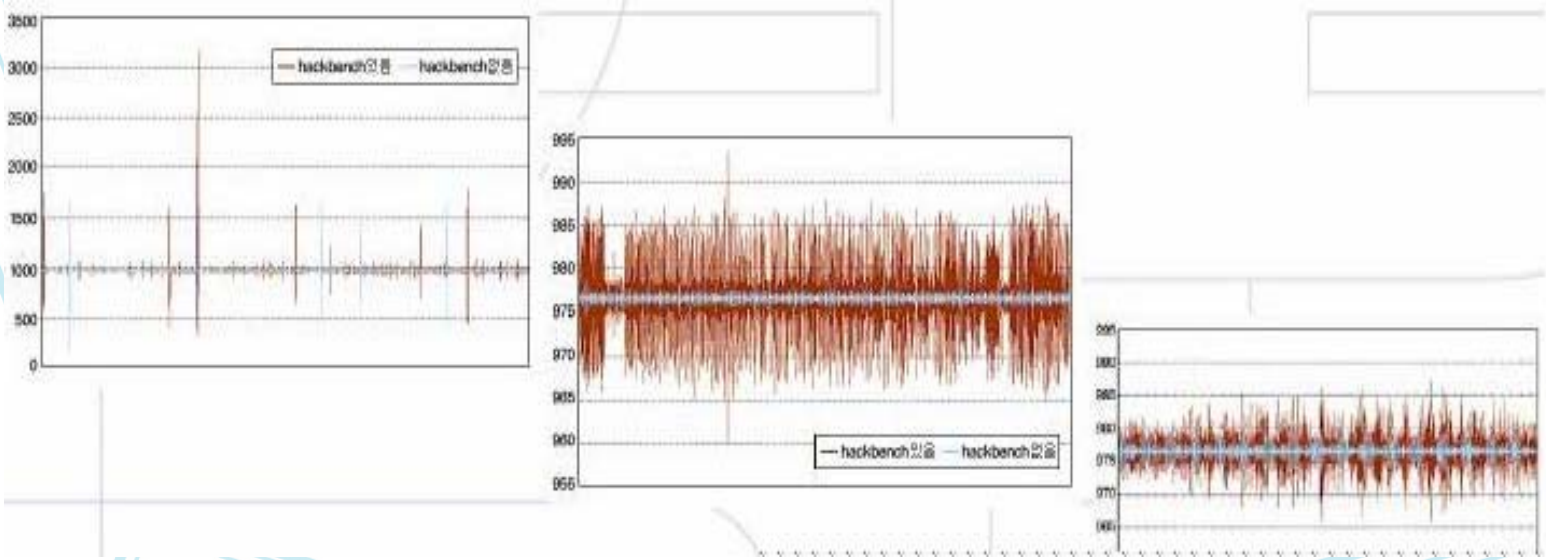
- Included in the RT-preempt patch
- Use the driver in the linux kernel, to toggle a bit on the parallel port, and watch for a response toggle back
- Rt-preempt patch applied
  - drivers/char/lpptest.c
  - scripts/testlpp.c
- Update of lpptest
  - See http://www.ussg.iu.edu/hypermail/linux/kernel/0702.2/0342.html
  - Removed dependency on TSC
- This requires a separate machine to send the signal on the parallel port and receive the response

RealFeel (1/2)

- Simple test program that how well a periodic interrupt is processed
- This program let periodic interrupt(/dev/rtc) have a fixed interval
- The program measures the time duration from interrupt to interrupt, and compares this to the expected value for the duration
- Depends on TSC in user space for timestamps
- See http://brain.mcmaster.ca/~hahn/realfeel.c

RealFeel (2/2)

- Result samples



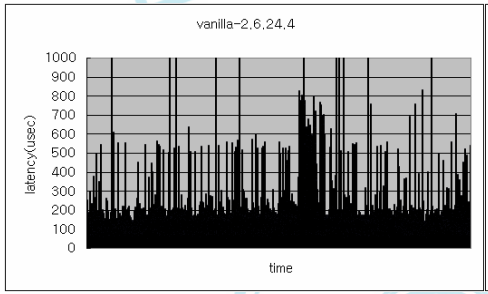
RealFeel-ETRI

- Measures latency from interrupt occur to application, interrupt should wakeup, invocation
- Extends realfeel in several ways
  - Supports interrupt handler patch file for /dev/rtc
    - Send timestamp information to application through rtc parameter when interrupt occurs
  - Adds a histogram feature to dump the results to a histogram
  - Locks the process pages in memory
  - Changes the scheduling priority to SCHED_FIFO, at highest priority
- Still, depends on TSC.
  - But, we have ported to mainstone board(Xscale)

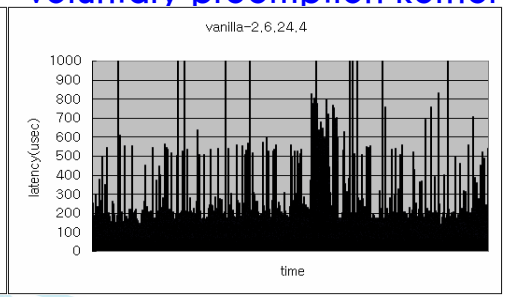
RealFeel-ETRI Results (1/2)

- Platform : Via EPIA(Nehemiah) 1GHz, 256Mbyte memory
- Kernel version : Linux 2.6.24.4
- Stress : ping (per 100 nano sec) from other machine, hackbench 20 (per 50sec)
- Test time : 10 hours

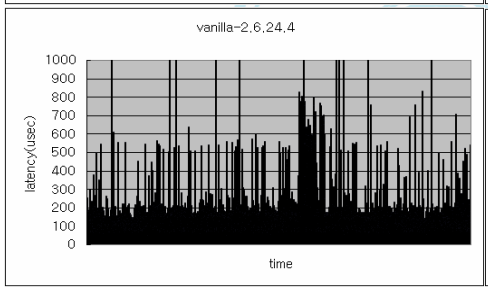
vanilla kernel



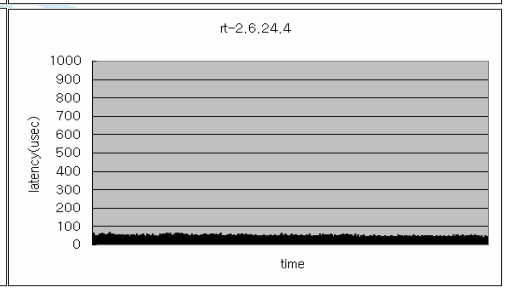
voluntary preemption kernel



preemptible kernel



real-time preemption kernel

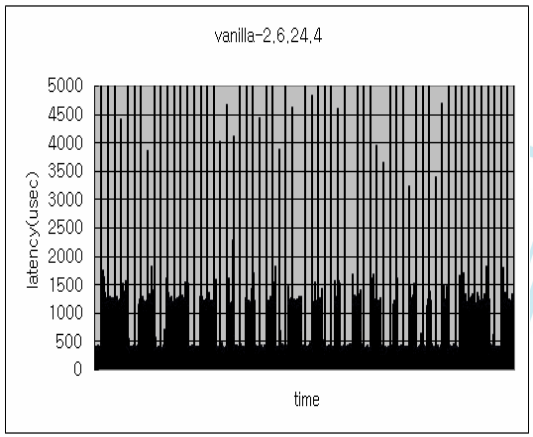


	Max latency time (usec)	Min latency time (usec)	Ave latency time (usec)
Vanilla	3888.57	3.96148	13.8279
Voluntary	3904.31	3.88947	11.2108
Preemptible	6792.74	4.75657	11.6637
Realtime-reempt	65.8249	9.36812	13.0913

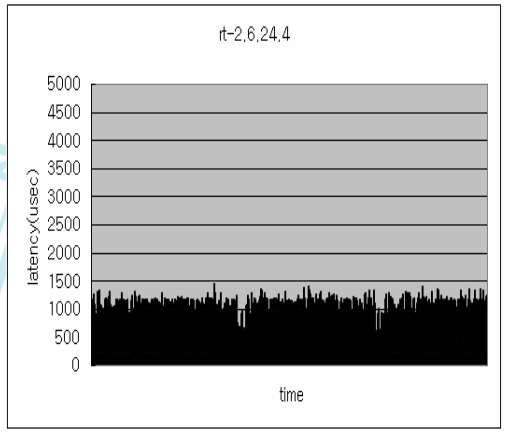
RealFeel-ETRI Results (2/2)

- Platform : Mainston Board(Xscale) PXA270 520MHz, 64Mbyte Memory
- Kernel version : Linux 2.6.24.4
- Stress : ping (per 100 nano sec) from other machine, hackbench 10 (per 50sec)
- Test time : 10 hours

vanilla kernel



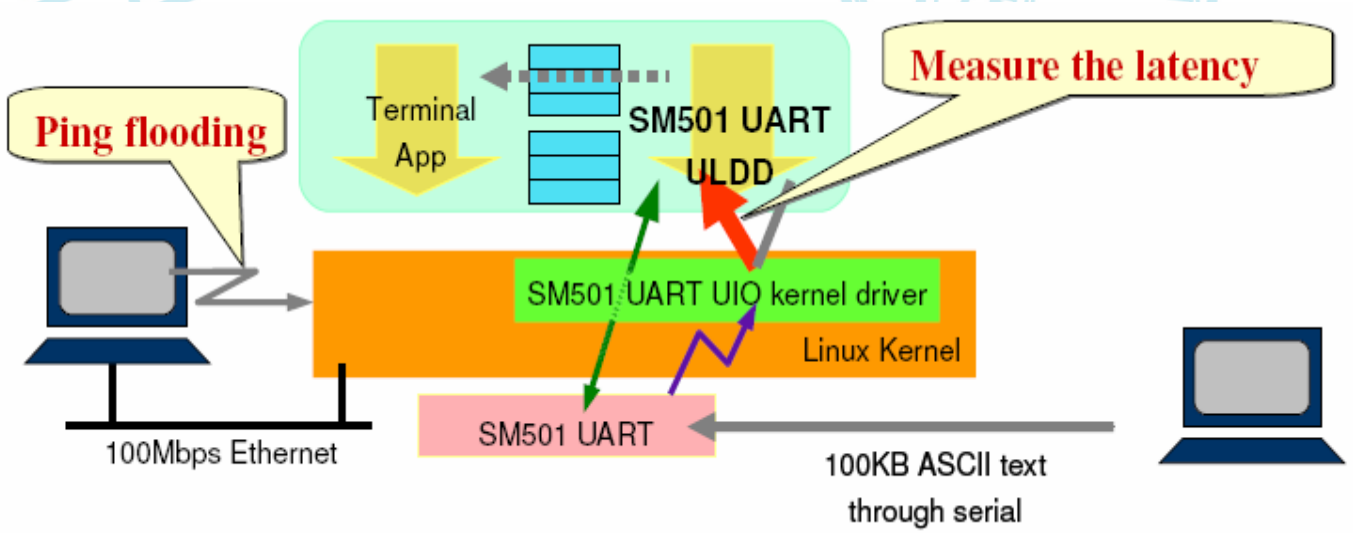
real-time preemption kernel



	Max latency time (usec)	Min latency time (usec)	Ave latency time (usec)
Vanilla	7195.3	1.46731	223.65
Realtime-reempt	1451.32	52.1442	118.653

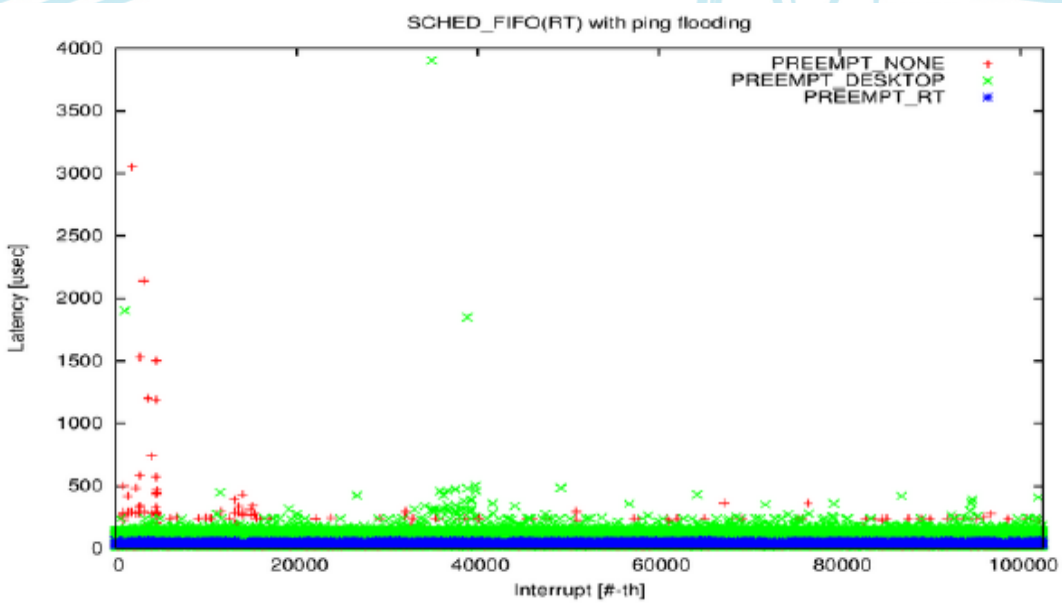
An Experiment with ULDD

- Implement a SM501 UART driver with UIO
- Measure the schedule latency (when kernel handler invokes user handler)



The Result of an experiment with ULDD

- The effectiveness of PREEMPT_RT can be particularly observed when the scheduler policy for ULDD is set to SCHED_FIFO under high system load



Cyclictest

- See http://rt.wiki.kernel.org/index.php/Cyclictest
- Wants to measure from periodic event to task wakeup time(measuring worst case response time of realtime task)
  - Periodic event can be software timer or nanosleep
  - Creates a number of tasks with different priority

Cyclictest Results

- Platform : Pentium III 400MHz based PC(tglx's reference machine)
- Kernel version : Linux 2.6.16
- Test case 1 : clock_nanosleep, interval 10000 microseconds, 10000 loops, 100% load
  - Commandline : cyclictest –t 1 –p 80 –n –i 10000 –l 10000

Kernel	Min(usec)	Max(usec)	Avg(usec)
2.6.16	55	4280	2198
2.6.16-hrt5	11	458	55
2.6.16-rt12	6	67	29

LRTB

- Bechmark framework for realtime responsiveness of linux
  - See http://www.opersys.com/lrtbf/
  - For this test, 3 computers are needed(target, host and logger)
    - Target : test system
    - Host : control system and collects the data
    - Logger : cause to interrupts on the target and record the time it takes for the target to respond
- Lastest results from opersysinc.
  - Kernel version: 2.6.12
  - Max_preempt_delay : 55us
  - Average_preempt_delay : 14us

Hourglass

- is a synthetic real-time application that can be used to learn how CPU scheduling in a general-purpose operating system works at microsecond and millisecond granularities
  - creates very detailed map of when each Hourglass thread has access to the CPU
  - supports multiple thread execution models; e.g. periodic and CPU-bound
  - acts as an abstraction layer for threading, timing, and CPU scheduling functionality on Unix- and Win32-based systems
- See http://www.cs.utah.edu/~regehr/hourglass/

Others

- Woerner test
  - received an interrupt on the serial port, and pushed data through several processes, before sending back out the serial port
  - requires an external machine for triggering the test and measuring the results.
  - See http://geek.vtnet.ca/embedded/LatencyTests/html/index.html
- Senoner test
  - This is a latency test that simulates and audio workload
  - See http://www.gardena.net/benno/linux/audio/

Test Program Summary

Feature / Test Program	lpptest	RealFeel-ETRI	Cyclictest	LRTB	Woerner test
platform specific(for target)?	No. But requires a separated machine	Yes – i386. But ported to Xscale.	Probably no. hrt should be ported	No. but requires parallel port on target	No. but requires serial port on target
How to generate interrupt?	Data on parallel port	Periodic timer programmed i386 /dev/rtc	Clock and Timer(clock_ nanosleep, POSIX interval	Data on parallel port	Data on serial port
What does test measure?	End-to-end response latency	From interrupt hander to task invocation	Interrupt and scheduling latency	End-to-end response latency	End-to-end response latency

Stress Method

- Usual Stress programs
  - Find, du, ping, several benchmark programs(i.e. hackbench, lmbench, unixbench)
  - Ingo Molnar's stress
    - Big file copy & hackbench & ping
- Stress actions
  - Things to spoil your RT performance test
    - have a bus-master device do a long DMA on the bus
    - get a page fault on your RT process (can be prevented with mlockall)
    - get multiple TLB flushes on your RT code path (how to cause this??)
    - get lots of instruction and data cache misses on your RT code path

Test results summary

Platforms Tested and in Use with CONFIG_PREEMPT_RT

- http://rt.wiki.kernel.org/index.php/CONFIG_PREEMPT_RT_Patch

Processor	Model #'s	Kernel version(s)	Contact(s)	Comment(s)
Opteron	IBM LS-20, LS-21, x3455	2.6.16-rt22, 2.6.20-rt8	Theodore Ts'o, Darren Hart	
Xeon	IBM 6850-P4U	2.6.20-rt8, 2.6.21-rc6-rt0	Dave Sperry	
P4	Compaq Evo N610	2.6.16-rt22	Dave Sperry	
P3	Celeron 1300	2.6.21.5-rt14	Oleksandr Natalenko	
Prescott	PentiumD 3 GHz	2.6.22.6-rt9	Dmitry Pisklov	
Intel	Pentium 4 2.4 GHz	2.6.23.9-rt12	Jaswinder Singh	
Intel	Pentium 4 2.8 GHz with HT	2.6.23.9-rt12	Jaswinder Singh	
Intel	FSC D2151S Celeron D 2.93 Ghz	2.6.20-rt8	Remy Bohmer	Avg. latency < 10us, worst case latency: < 30us (see Note 1)
Intel	FSC D2151S Core 2 Duo E6300	2.6.20-rt8	Remy Bohmer	Avg. latency < 10us, worst case latency: < 30us (see Note 1)
Intel	Asus P5B Core 2 Duo E6600 2.4GHz	2.6.23-rc4-rt1	Remy Bohmer	Avg. latency < 10us, worst case latency: < 30us (see Note 1)
Intel	IBM Thinkpad T43 Pentium M CPU 2 GHz	2.6.23-rt3	PhilK	Avg. latency < 14us, worst case latency: < 32us
ARM9 (v4t)	Atmel AT91RM9200-EK (See AT91-Notes)	2.6.23.12-rt14	Remy Bohmer	worst case latency: < 300us (see Note 1)
ARM9 (v4t)	FREE_ECB_AT91 (See AT91-Notes)	2.6.23.12-rt14	Carlos Camargo	
ARM9 (v5t)	Atmel AT91SAM9261-EK (See AT91-Notes)	2.6.23.1-rt5	Remy Bohmer	worst case latency: < 500us (see Note 1)
ARM9 (v5tej)	TI OMAP5912 OSK	2.6.23-rt1	OMAP ML	A small additional OMAP specific patch on top of -rt is needed. See ML.
ARM9 (v5tej)	TI DM6446 DaVinci	2.6.23-rt1	DaVinci ML	A small additional DaVinci specific patch on top of -rt is needed. See ML.
SH4	Renesas R2D/R2D+ SH7751R CPU	2.6.21-rt2 and later	linux-sh ML	

Test Summary of CELF RTWG

Company	Platform	Kernel version	Results with RT-preempt
Samsung	Omap 5912(ARM9), 192 MHZ	2.4.20 2.6.10	Both 30 ~35us worst case(without RT-preempt, interrupt latency)
IGEL	SH7551R(SH4), 240 MHZ	2.6.21	<50us worst case
ETRI	Via EPIA(x86), 1 GHZ	2.6.24.4	65 us worst case
ETRI	Mainstone(Xscale), 520 MHz	2.6.24.4	1451 us worst case
Mitsubishi	SH4, 240 MHZ	2.6.8	1300 us worst case
Mitsubishi	VIA Eden, 600 MHZ	2.6.8	226 us worst case
Toshiba	Cell BE, 3.2 GHz	2.6.12	less variability

Plan

- RTWG will try to test by using prior mentioned testing methods and collect testing results
- port necessary test program to various architectures
- write stress program and how to give a stress
- collect more related presentations and documents
- would like to write valuable rt testing methods metric and test results taxonomy