Changes for the Better

Experience with realtime performance of the current Linux technologies

Shinichi Ochiai
Mitsubishi Electric Corporation
Information Technology R&D Center

Agenda

1. Background
2. Realtime Technologies for Linux
3. Evaluation Targets
4. Evaluation
   - 4.1. 2.4/2.6 Preemptible Kernel Effect
   - 4.2. Efficiency of New Realtime Patches
   - 4.3. Porting RT Patches to SH4
   - 4.4. Case of Interrupt Response
   - 4.5. Comparison with Hybrid Approach
5. Conclusion

1. Background

Embedded Linux moves into realtime.

- Since 2.4/2.6 kernel, realtime area is going to be one of the target scope of Linux.
- Mainline kernel supports priority based and preemptible scheduling.
- Number of realtime extension patches are also available on Linux.
- To use Linux as an operating system for dedicated system, predictable response time is important factor for system design.

1. Background (*cont.*)

But realtime capability of Linux in practical area is still unclear.

- Number of new technologies are inserted each new version of kernel.
- Target processor and I/O are varied in embedded area. And also application requirements are varied
- Few reference benchmark are available.

Enlightening realtime performance of Embedded Linux from external behavior

2. Realtime Technologies for Linux

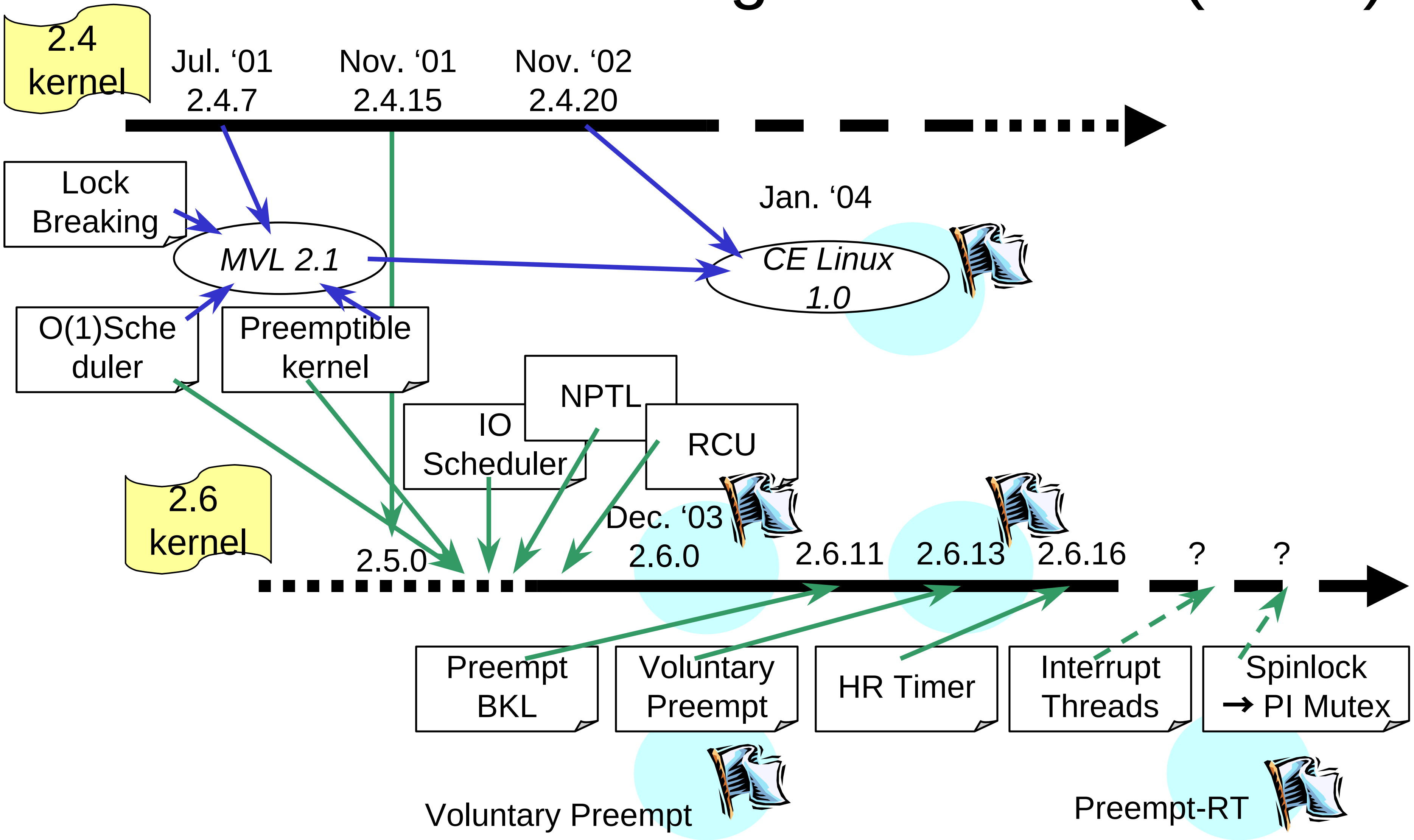
	2.4	CE Linux 1.0 Jan, '04	2.6.0 Dec, '03	2.6.8 (Voluntary-preempt patch) Aug, '04	2.6.14 Oct, '05	2.6.14 (Preempt-RT patch) Oct, '05	2.6.16 Jan, '06
O(1) Scheduler	X	O	O	O	O	O	O
Preemptible Kernel	X	O	O	O	O	O	O
Lock Breaking	X	O	---	---	---	---	---
Voluntary Kernel Preemption	X	X	X	O	O	O	O
Preempt Big Kernel Lock	X	X	X	X	O	O	O
Interrupt Threads	X	X	X	O	X	O	X
Replace spinlock with PI Mutex	X	X	X	X	X	O	X
Read-Copy Update	X	X	O	O	O	O	O
IO Scheduler	X	X	O	O	O	O	O
High Resolution Timer	X	X	X	X	X	O	O
NPTL	X	X	O	O	O	O	O

2. Realtime Technologies for Linux (*cont.*)

Key Features for Realtime

- O(1) scheduler
  enable priority based scheduling with fixed overhead
- Preemptible Kernel
  enable preemption except spinlock region
- Voluntary Preemption
  insert additional preemption points
- Threaded IRQ (Interrupt Thread)
  handle IRQ routines in thread
- PI Mutex (preempt-RT)
  replace spinlock with priority inheritance mutex

2. Realtime Technologies for Linux (*cont.*)



3. Evaluation Targets

Target Applications

- Video recorder with Network interface
- Smart Phone
- Distributed Controller

Requirements

- *Device support: networks, Storage/Flash Storage, VGA, special IO*
- *Dynamically invoked multi-applications*
- *Realtime response time:*

  10s microseconds – 100s microseconds

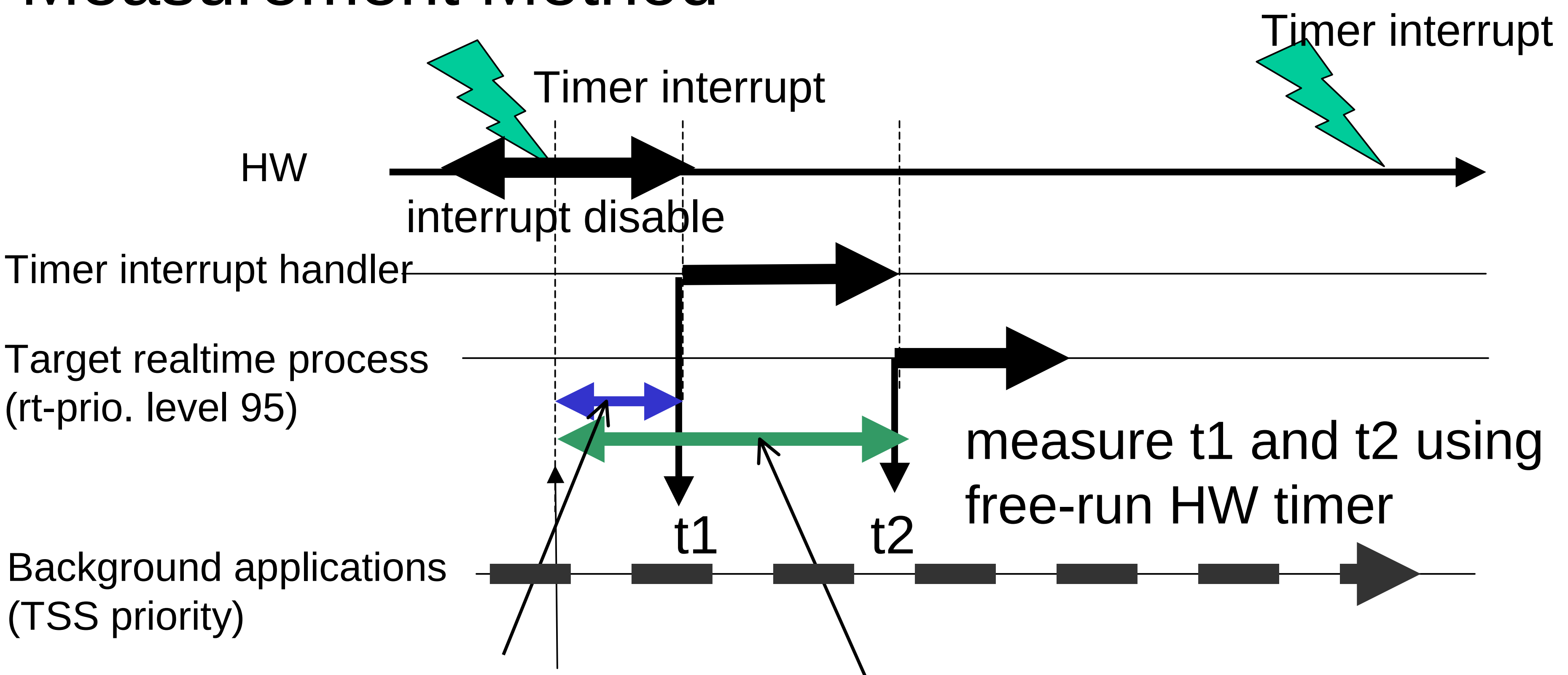
3. Evaluation Targets (*cont.*)

Target Boards

	CPU	Memory, Bus	IO
Target 1	SH4 (Renesas) 240MHz	64MB, 120MHz	serial, CF, LAN
Target 2	Eden (VIA) 600MHz	256MB, 400MHz	
Target 3	MPC7410 (freescale) 500MHz	512MB, 100MHz	

3. Evaluation Targets (*cont.*)

Measurement Method



Interrupt latency:
t1 – timer register setting

Process preemption latency:
t2 – timer register setting

3. Evaluation Targets (cont.)

Application Scenarios

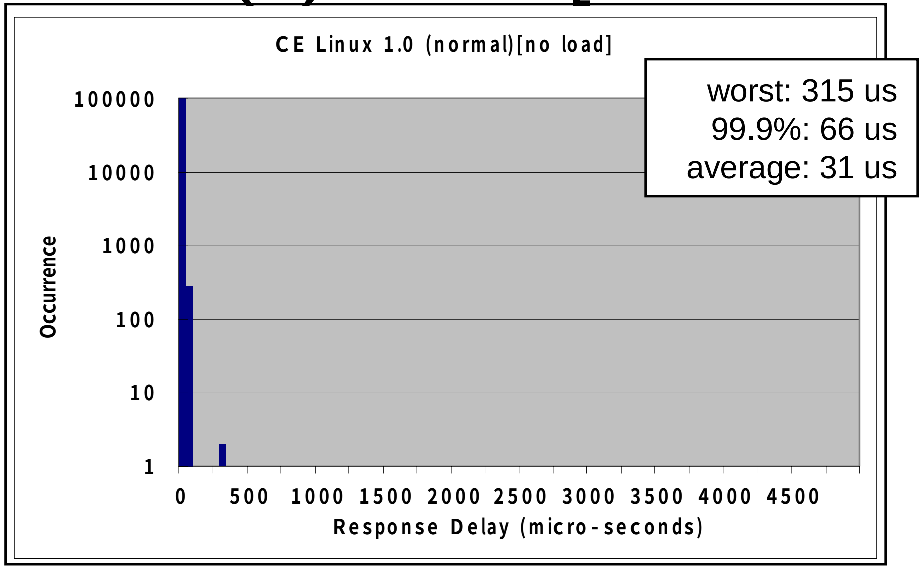
1. Stable State:
   stable ready state
2. Dynamic Process:
   process/thread creation and termination
3. Dynamic Memory:
   shared/device memory mapping control
4. CF PIO:
   access to CF storage device and other special devices

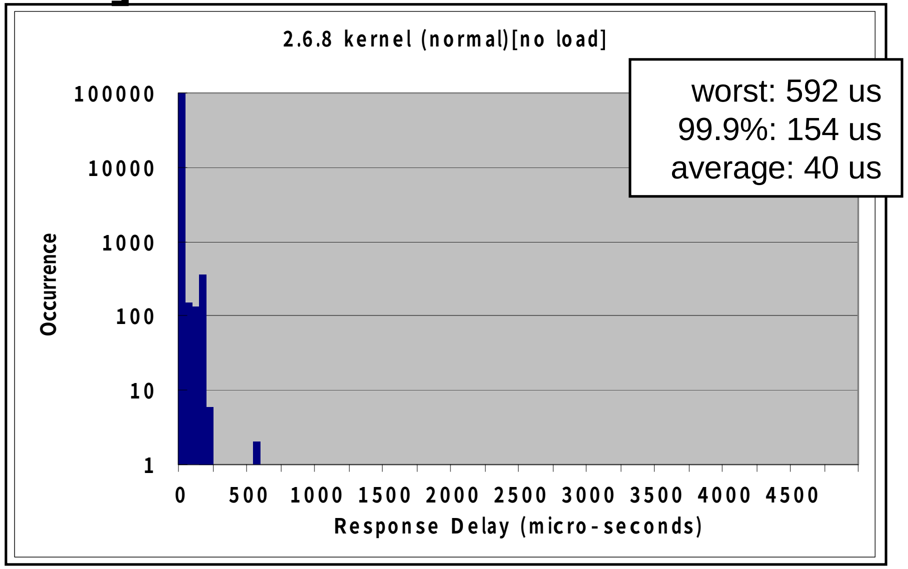
4. Evaluation

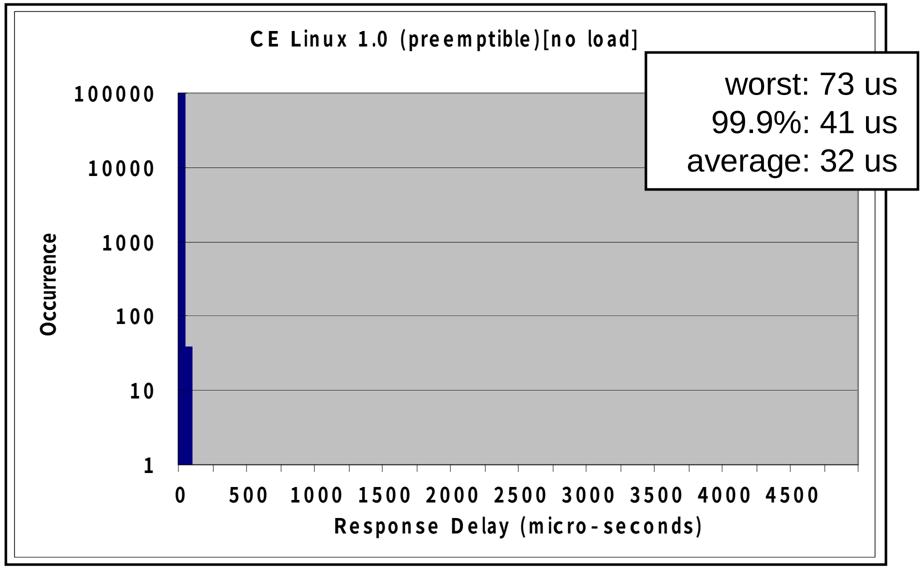
4.1. 2.4/2.6 Preemptible Kernel Effect

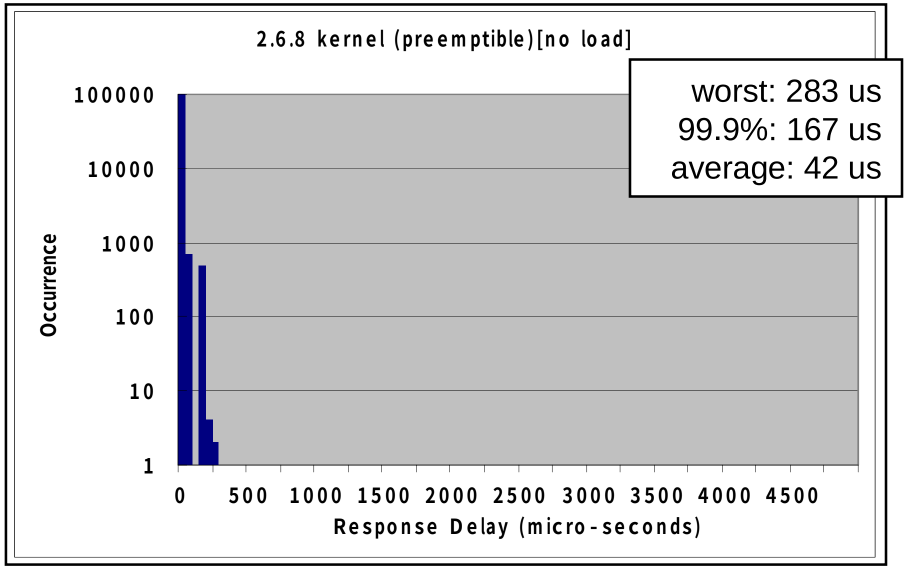
- RT performance improvement between 2.4 (CE Linux 1.0) kernel and 2.6.8 kernel
- RT performance improvement between non-preemptible kernel and preemptible kernel
- Target: SH4/240MHz

(1) SH4: [Stable State] 2.4/2.6Kernel

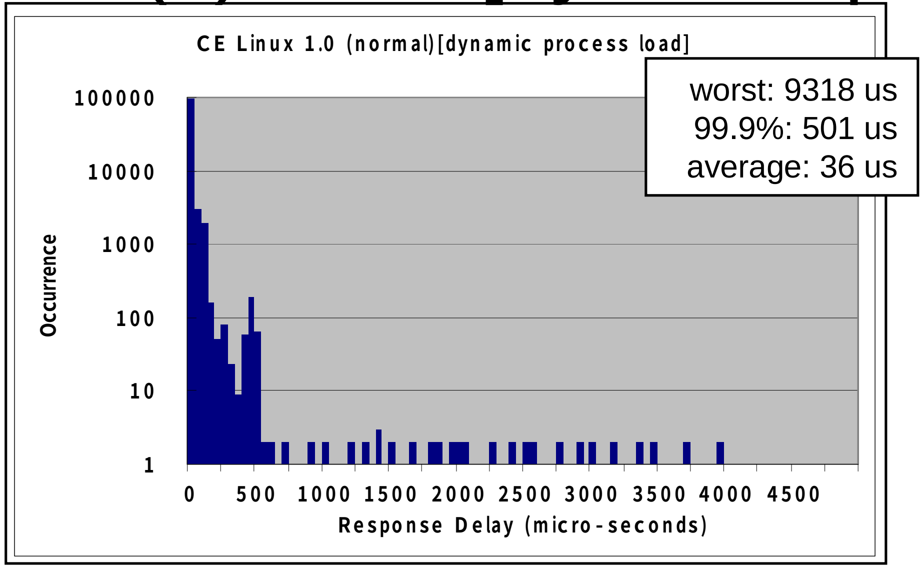


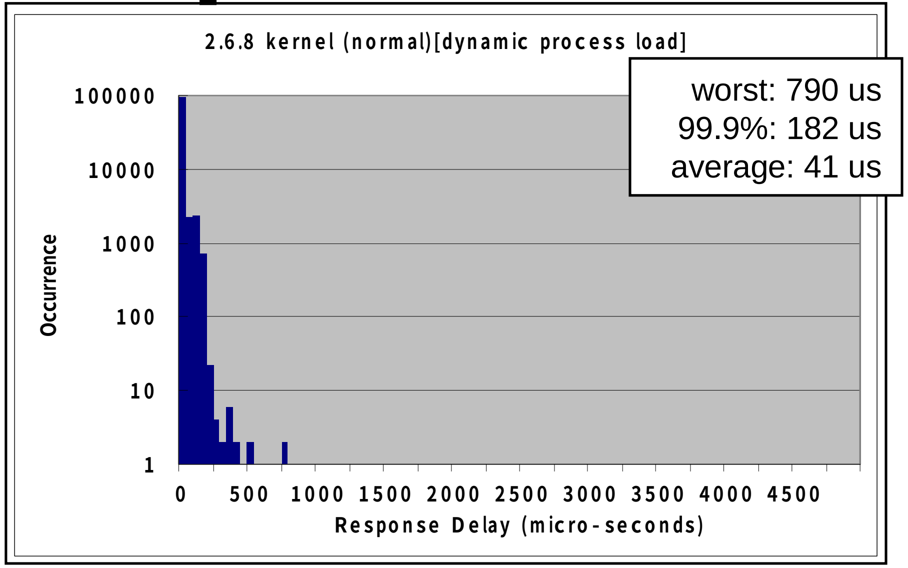


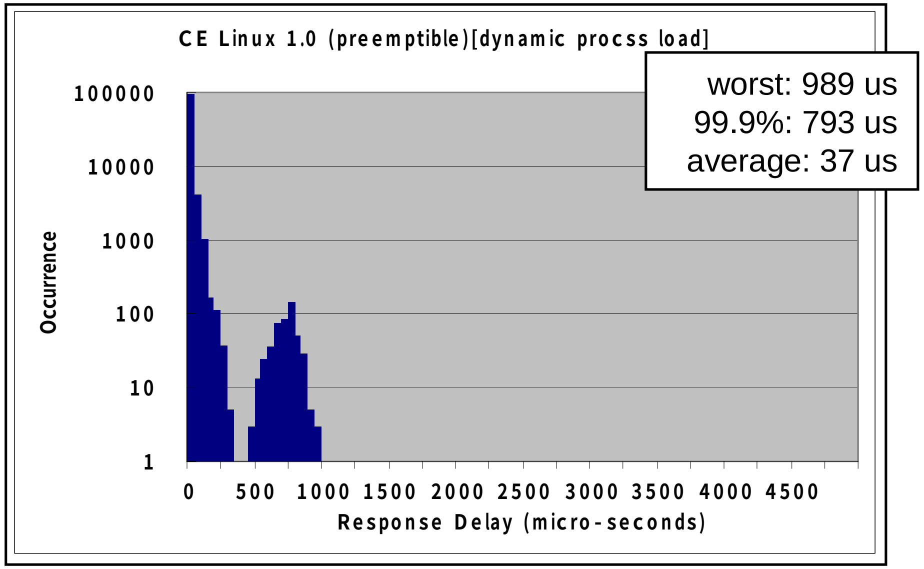


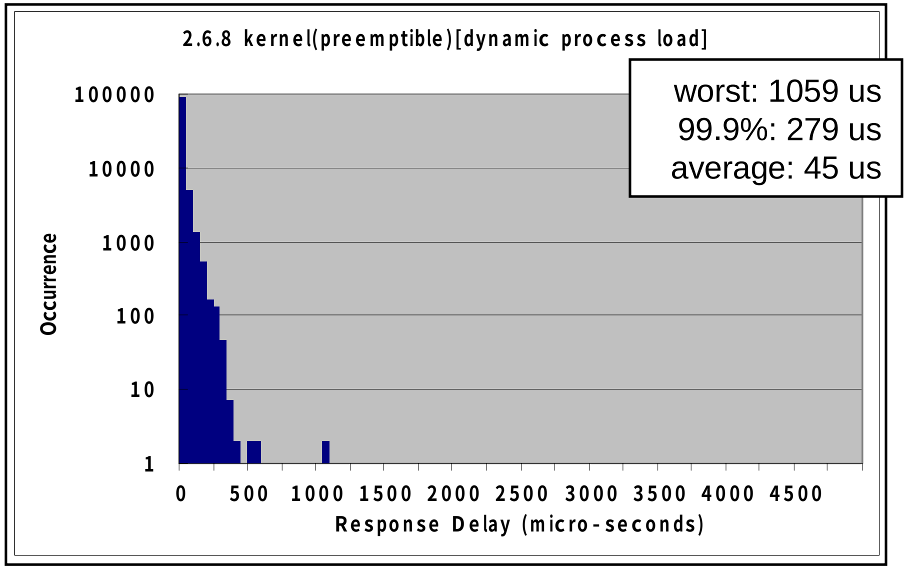


(2) SH4: [dynamic process] 2.4/2.6Kernel

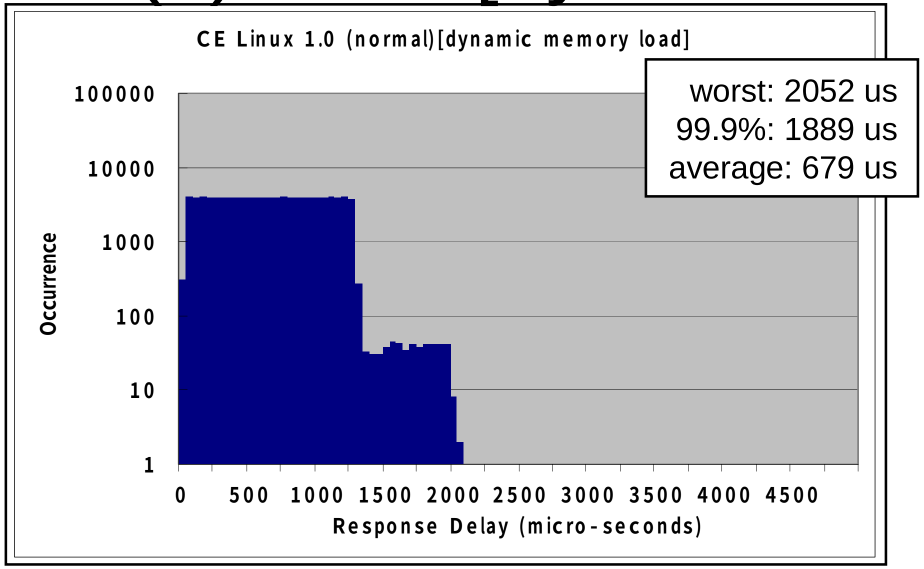


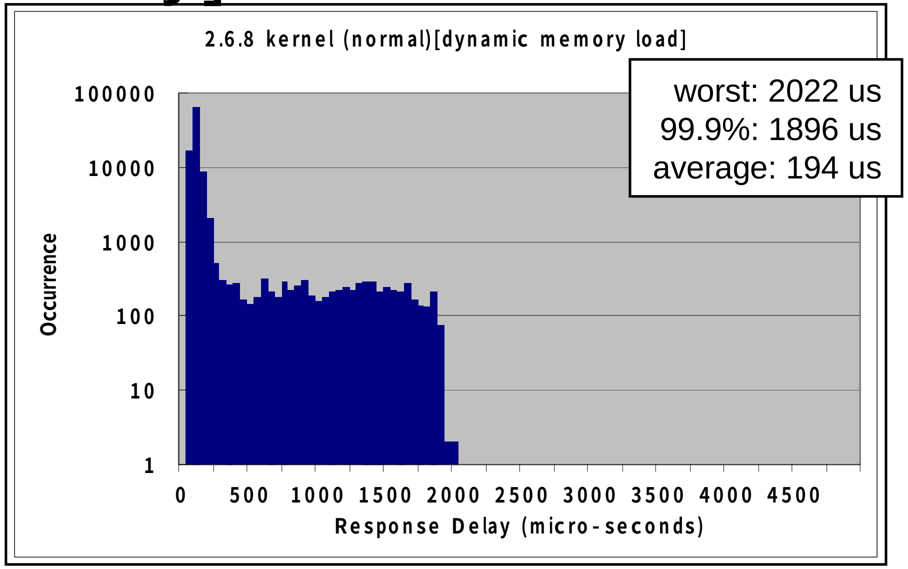


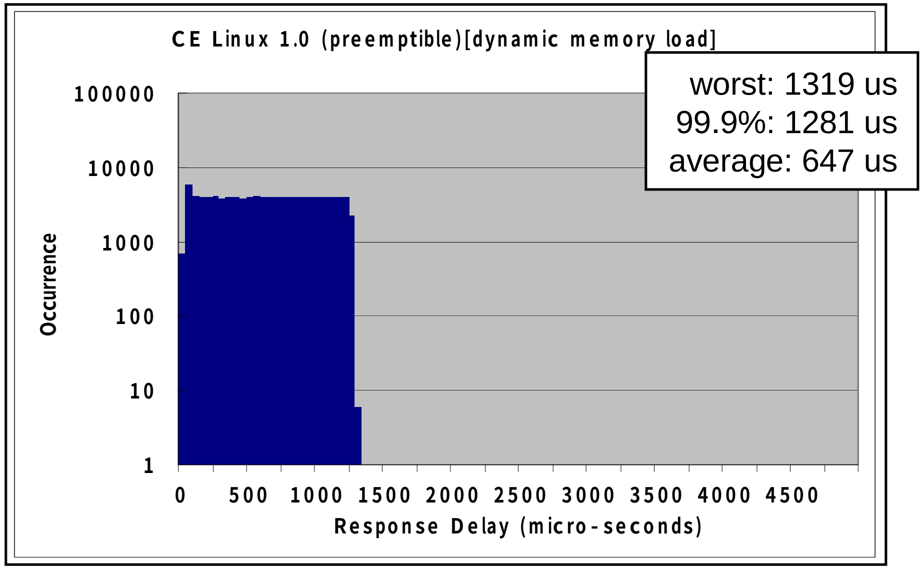


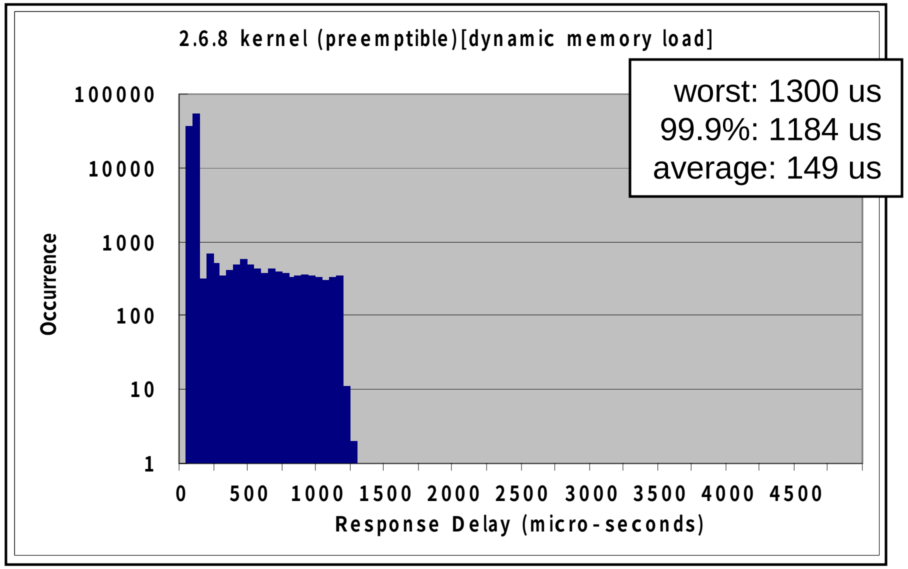


(3) SH4: [dynamic memory] 2.4/2.6Kernel

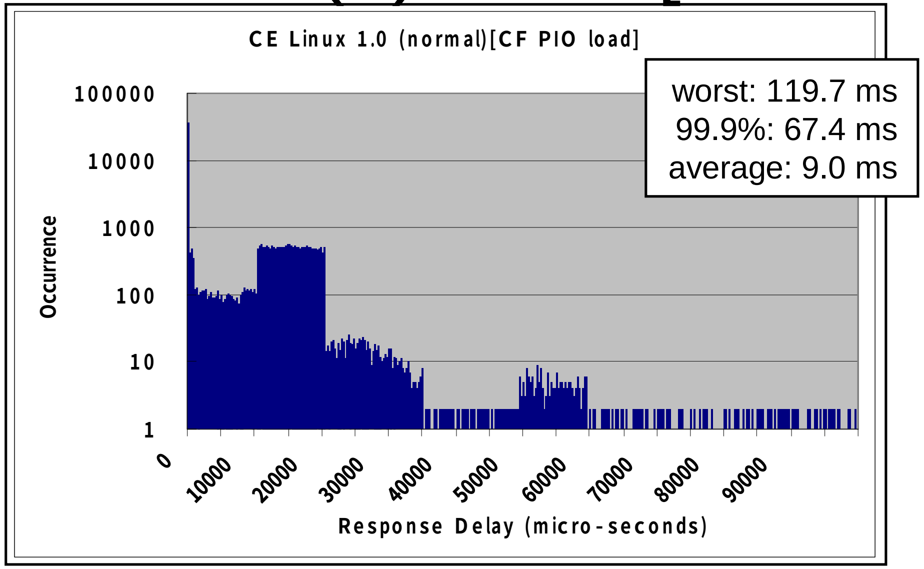


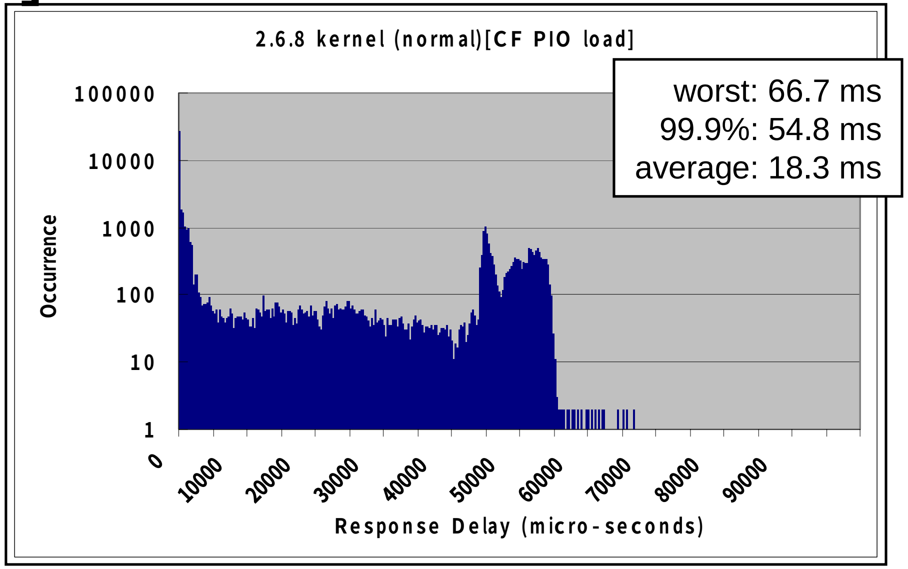


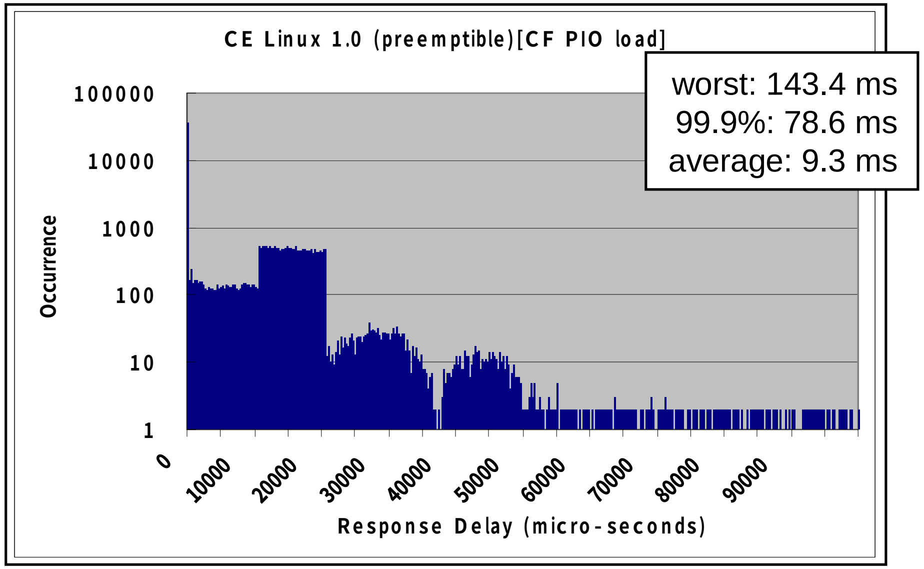


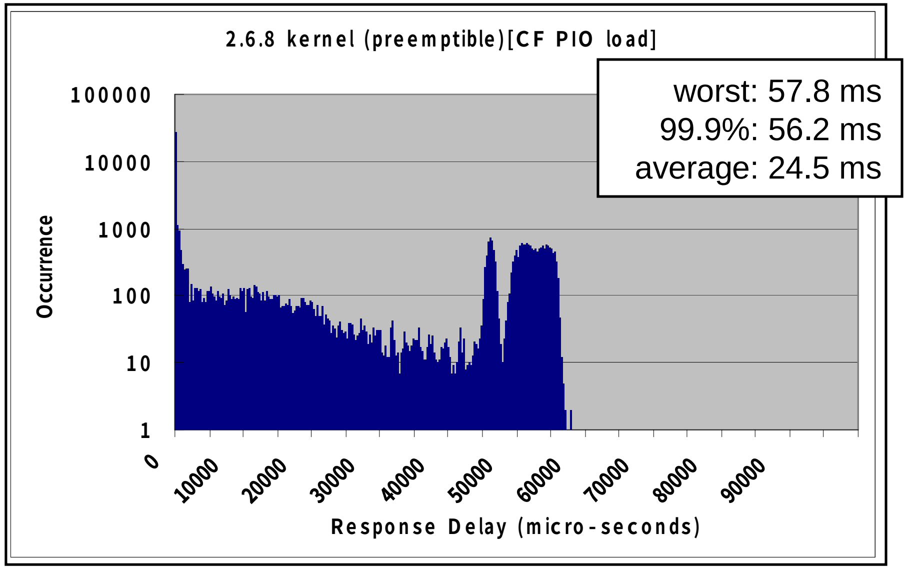


(4) SH4: [CF PIO] 2.4/2.6Kernel









2.4/2.6 Preemptible Kernel Effect (*cont.*)

Summary

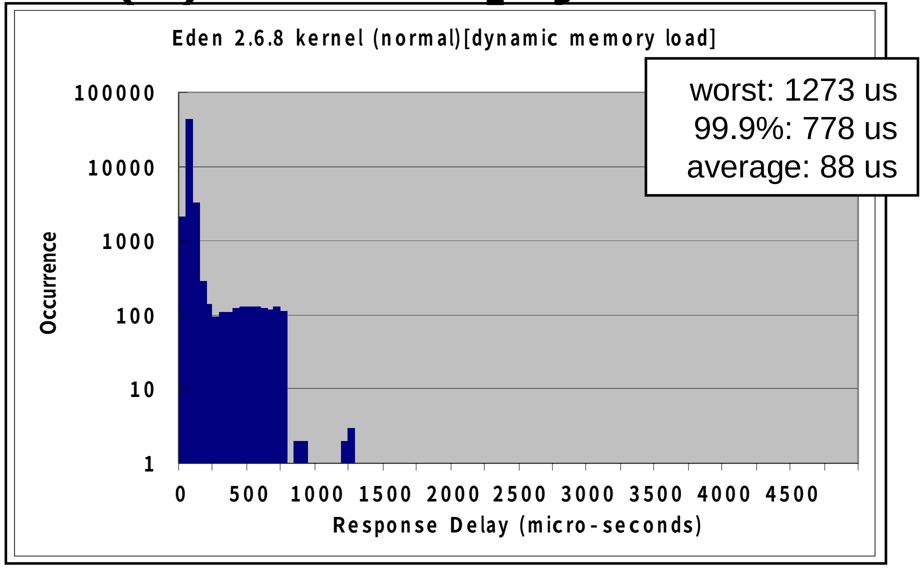
- RT behavior differs on background application type. (memory management, file system and CF access make worse RT performance.)
- Preemptible kernel fairly improved RT performance on 2.4 version.
- 2.6 kernel makes good RT performance even non-preemptible. (RT performance difference by preemptible is less than 2.4.)
- Looks like unusable for RT while CF access has run. (driver implementation issue?)

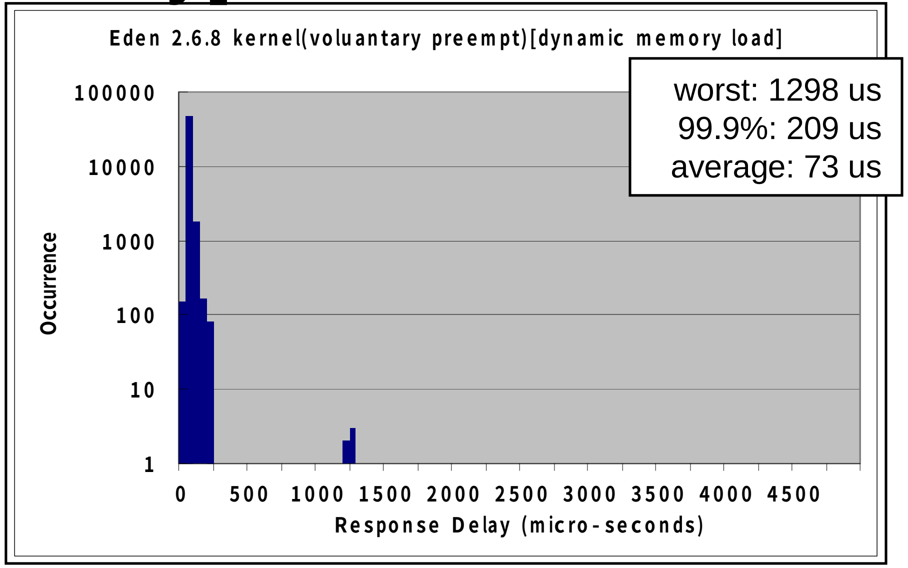
4. Evaluation (*cont.*)

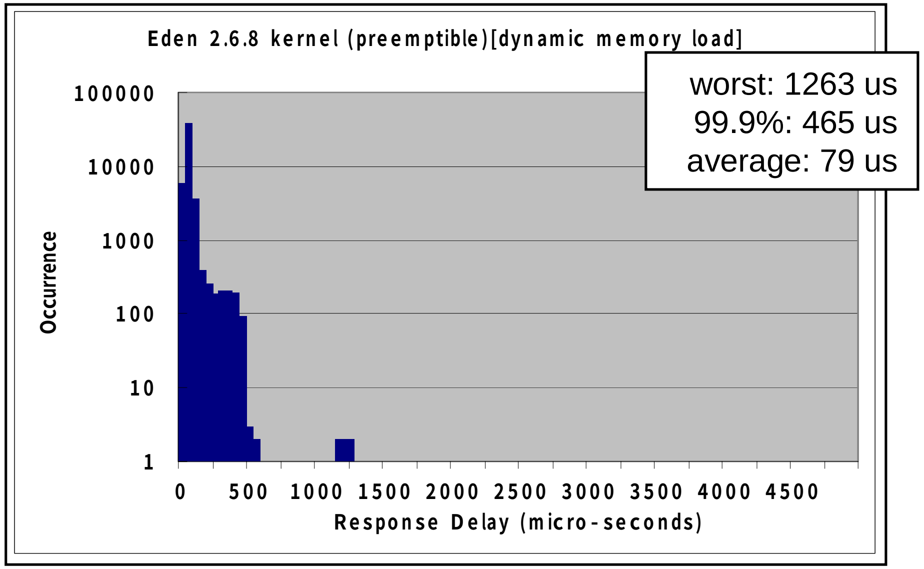
4.2. Efficiency of New Realtime Patches

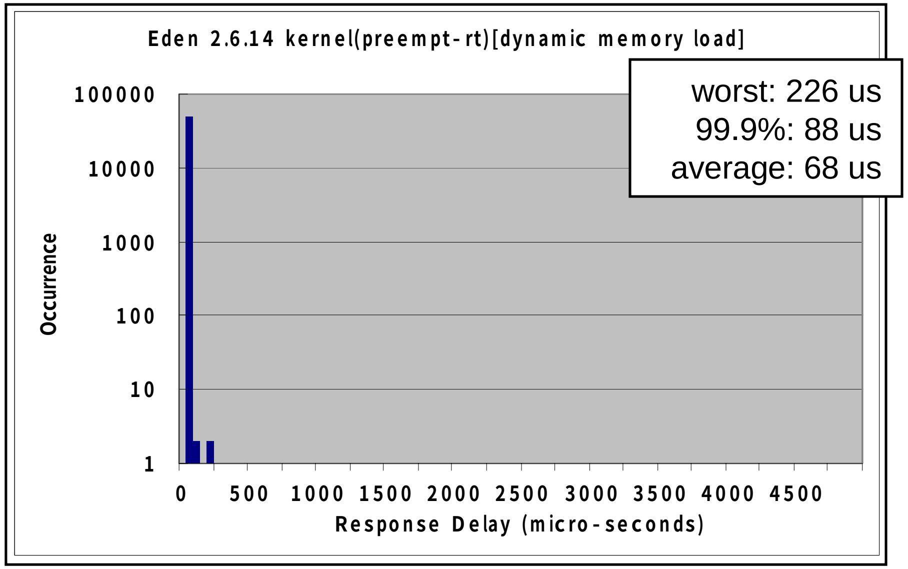
- RT performance improvement between standard 2.6 kernel, voluntary preempt kernel, and preempt-RT kernel
- Reference performance of reference architecture (IA32 compatible).
- Target: VIA Eden/600MHz

(3) Eden: [dynamic memory] 2.6Kernel+RT

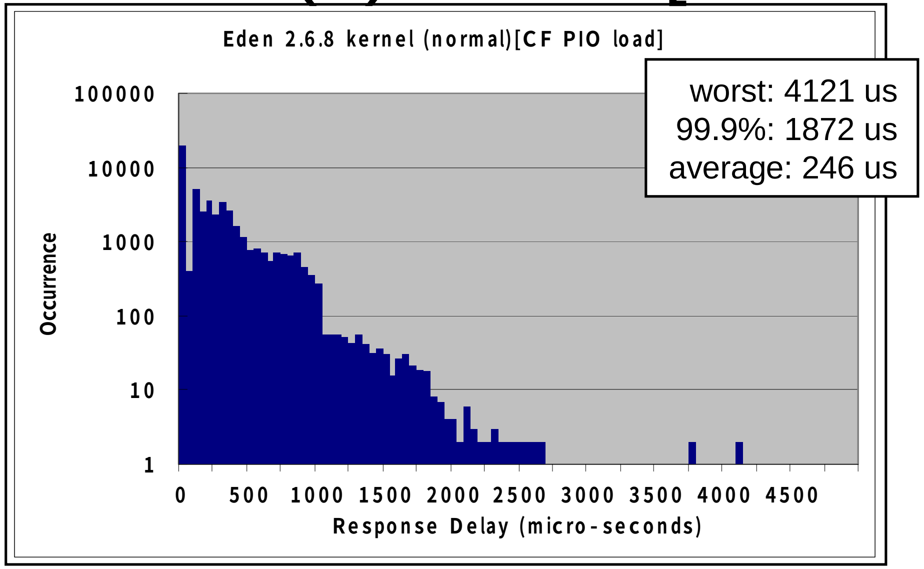


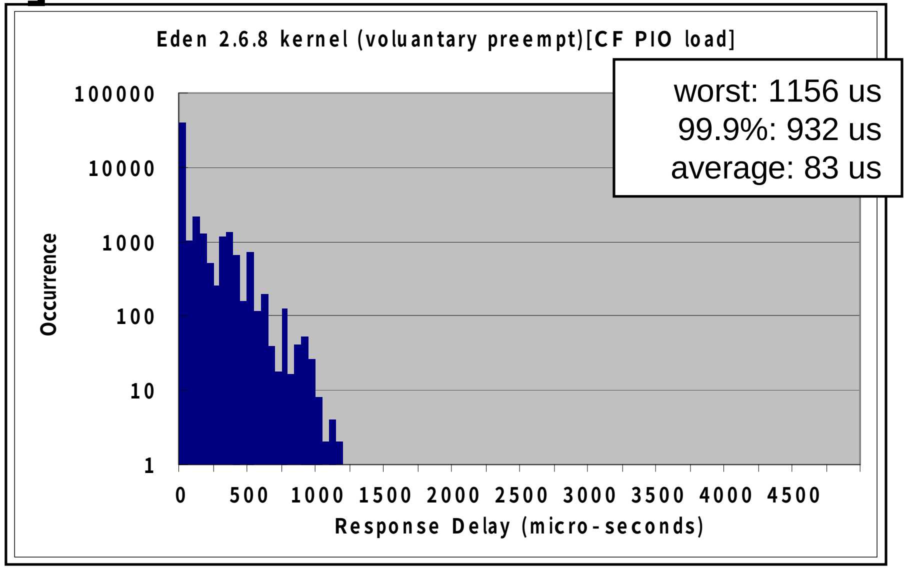


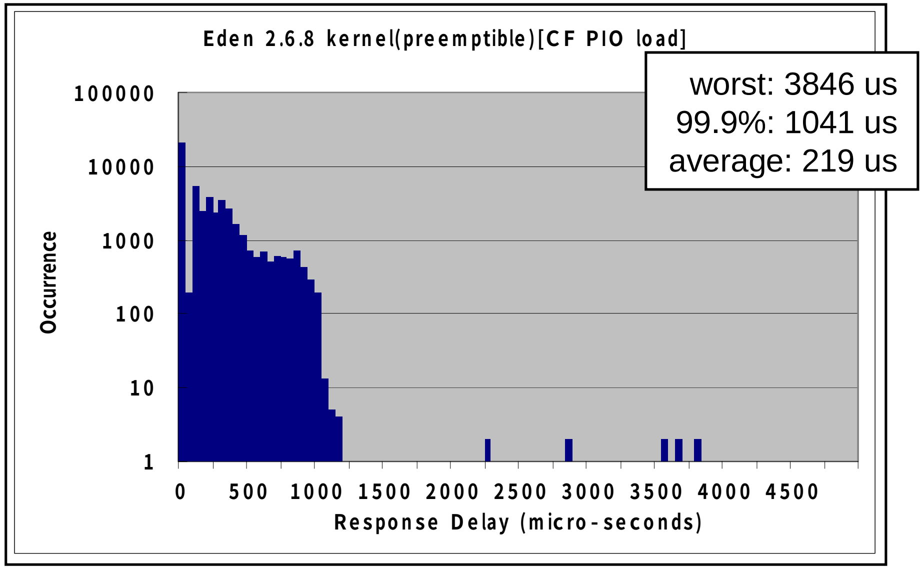


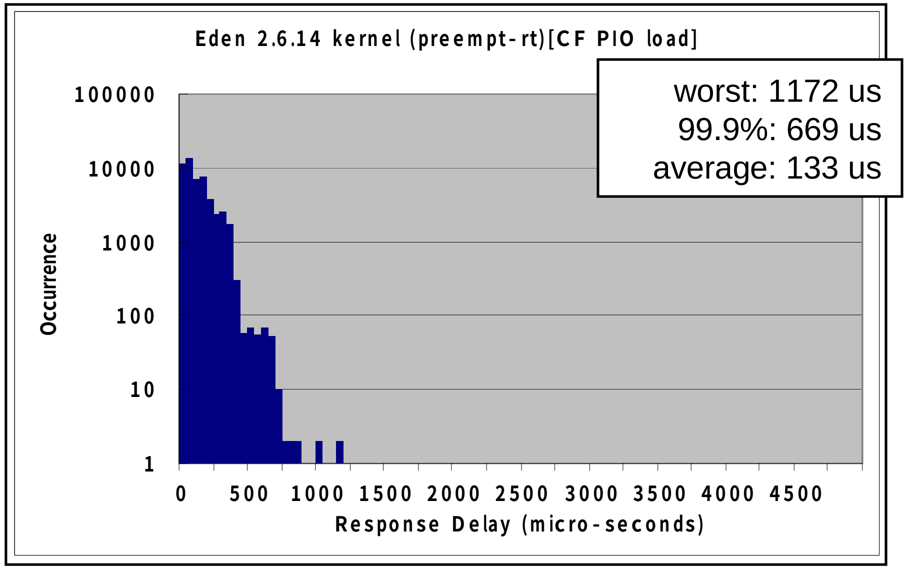


(4) Eden: [CF PIO] 2.6Kernel+RT









Efficiency of New Realtime Patches (*cont.*)

Summary

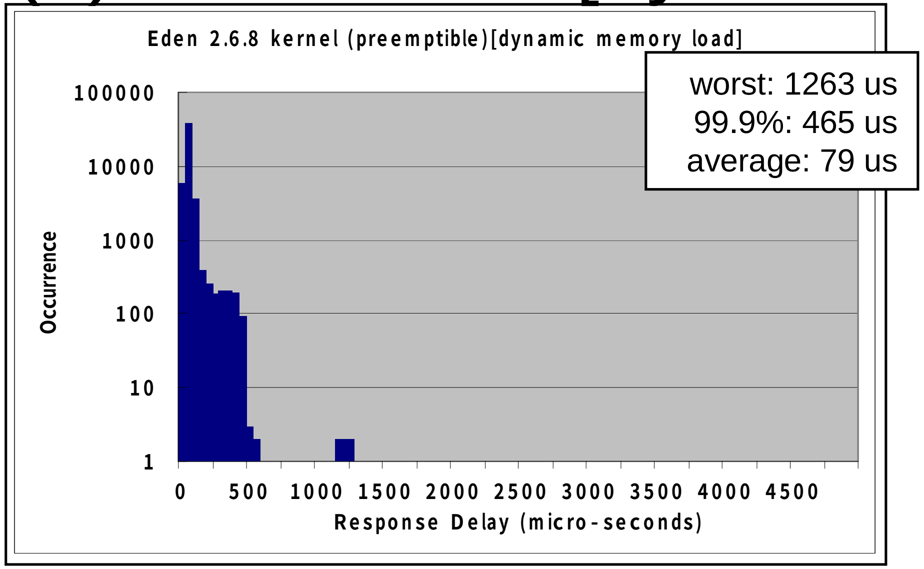
- Each RT technology insertion improves 2.6 RT performance. Especially preempt-RT patch is effective remarkably.
- CF access makes RT performance worse, but it is permissible as comparison with SH4 results.

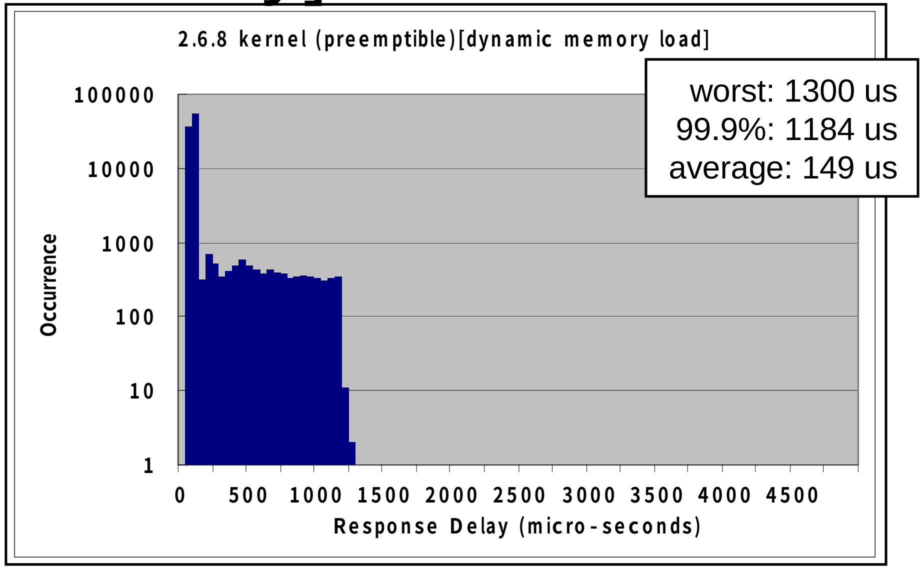
4. Evaluation (*cont.*)

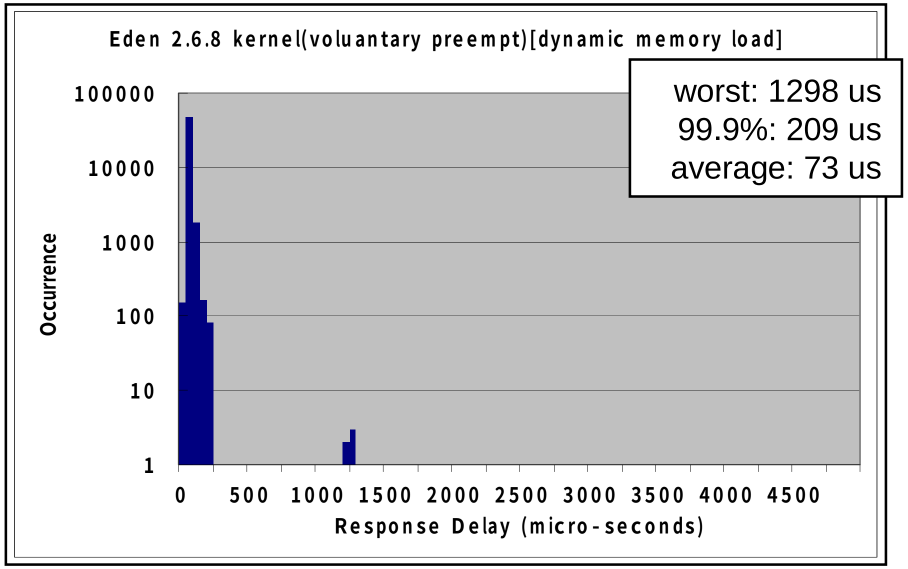
4.3. Porting RT patches to SH4

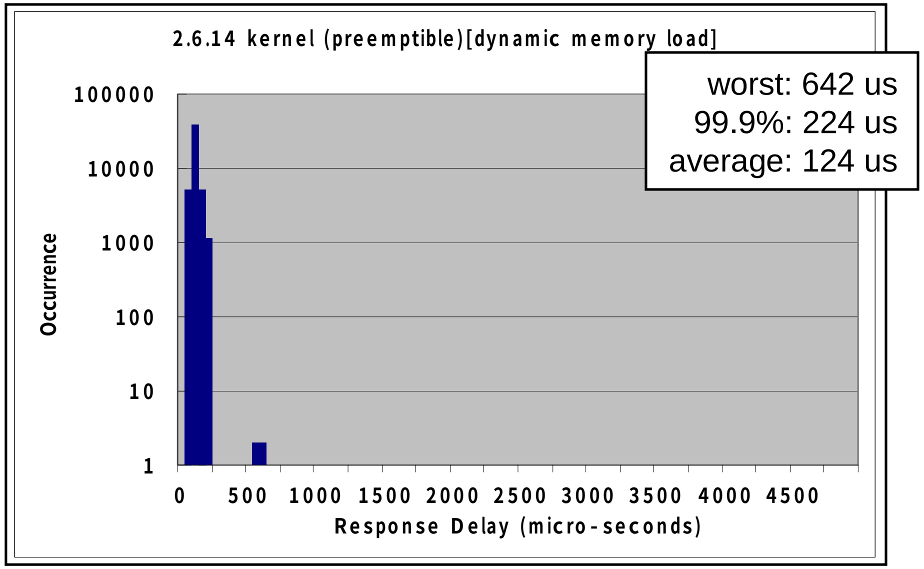
- Preempt BKL and voluntary preemption are merged to standard kernel at 2.6.14. They can be enabled on SH.
- Porting preempt-RT patch (threaded IRQ and PI mutex) to SH is still on going. It is not included in this evaluation.
- Target: SH4 240MHz

(3) Eden/SH4: [dynamic memory] 2.6Kernel+RT

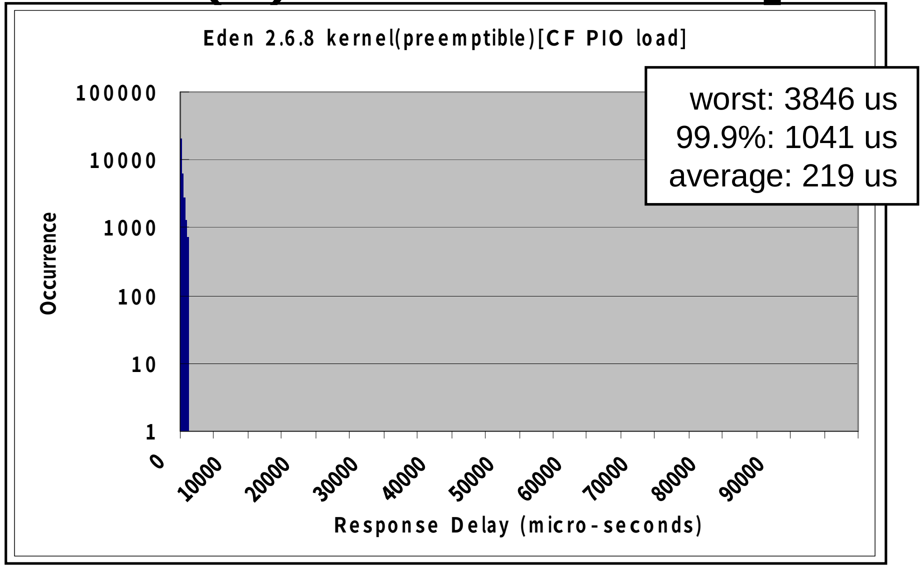


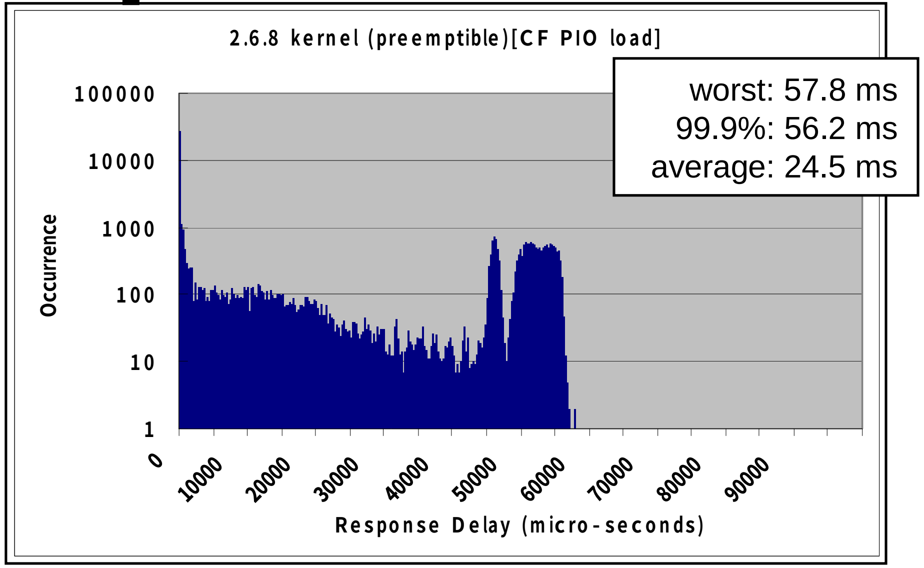


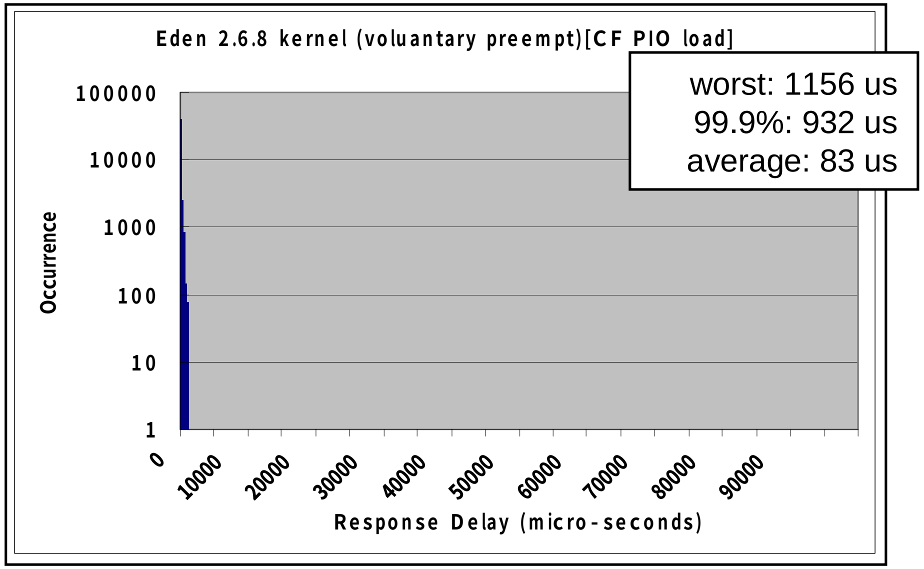


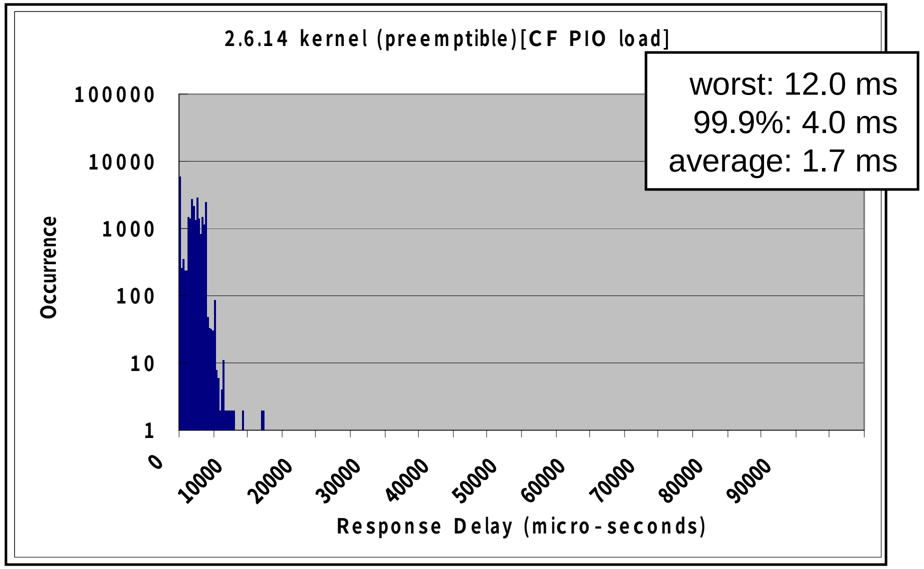


(4) Eden/SH4: [CF PIO] 2.6Kernel+RT









Porting RT patches to SH4 (*cont.*)

Summary

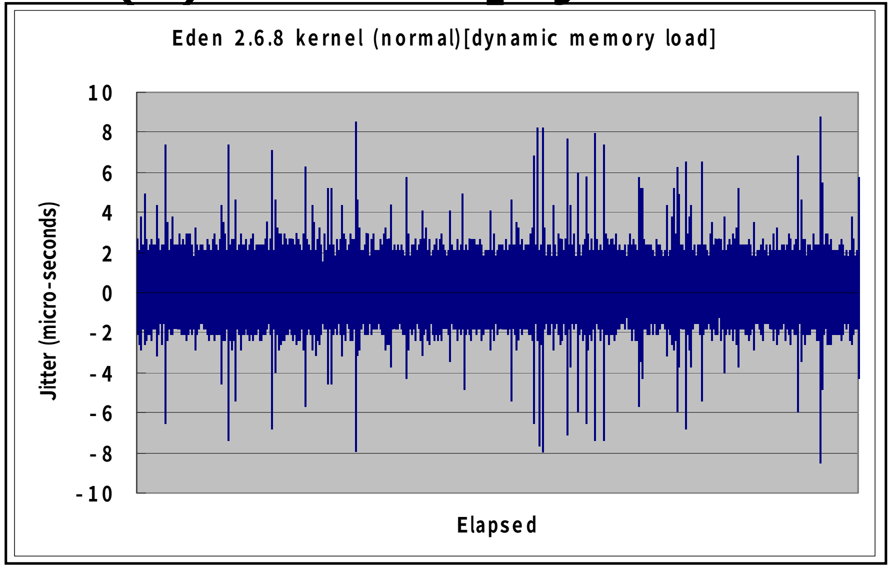
- Relatively same effects were gained both Eden and SH4 on dynamic memory load. (no architecture dependency.)
- 2.6.14 kernel has better RT performance since 2.6.8 kernel on SH4. RT performance while CF access is also improved but not enough.
- Expects preempt-RT patch will improve performance much more either on SH4.

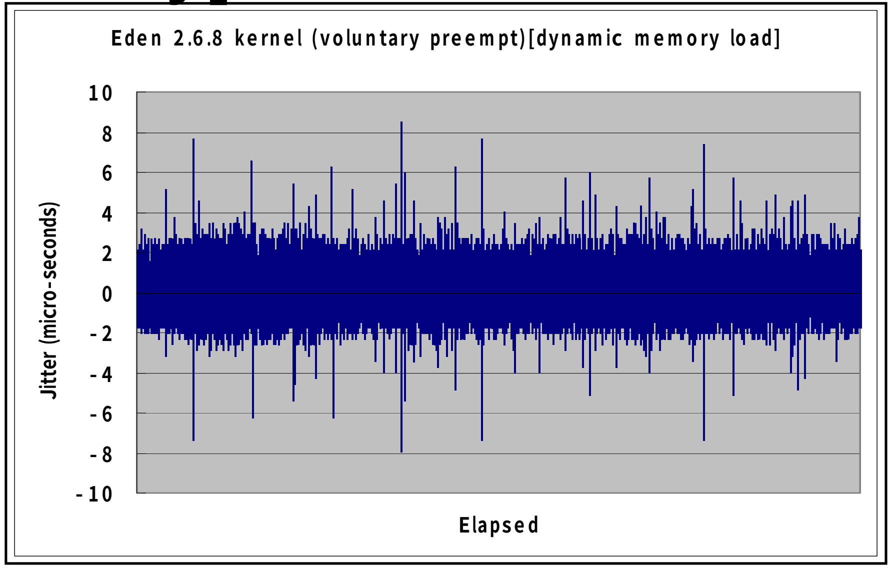
4. Evaluation (*cont.*)

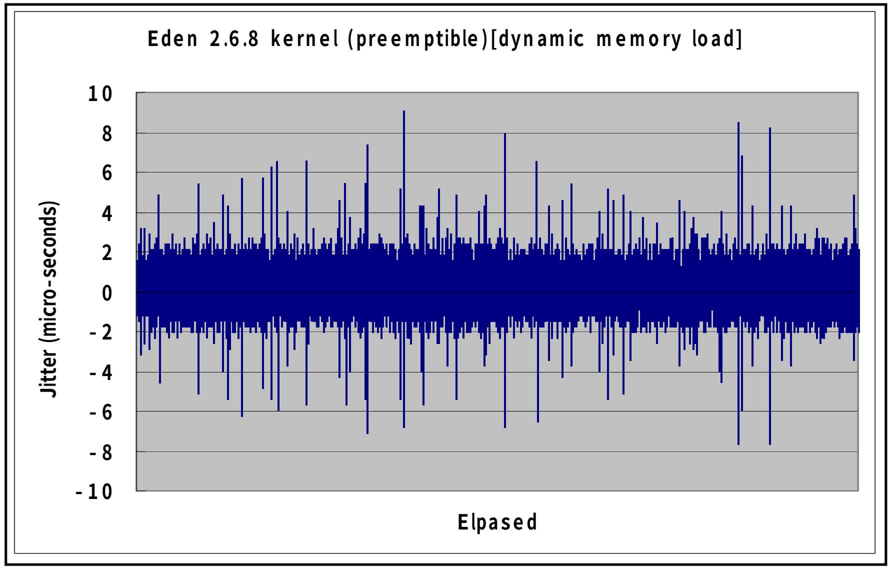
4.4. Case of Interrupt Response

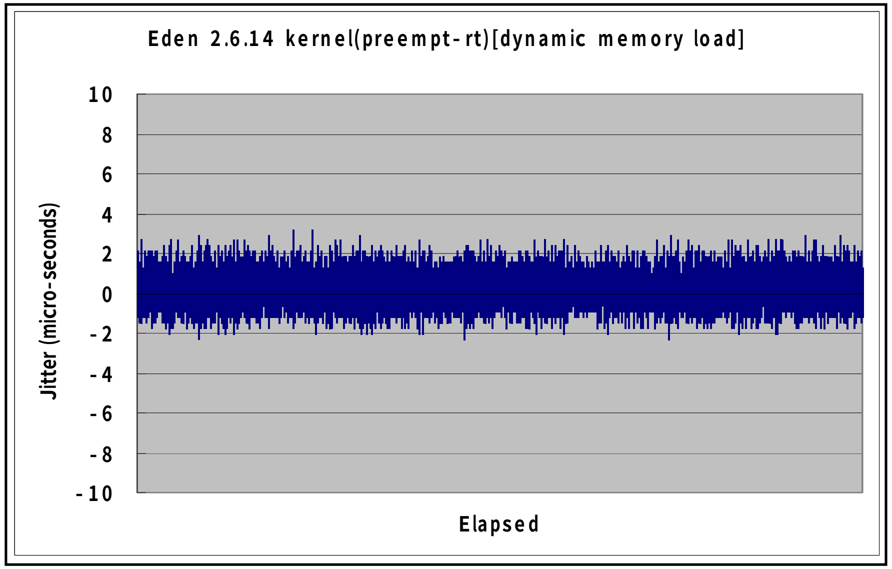
- Measurement of kernel level latency
- Interrupt response time improvement by each RT patch
- Architecture dependency: difference between Eden and SH4
- Target: Eden 600MHz/SH4 240MHz

(3) Eden: [dynamic memory] 2.6Kernel+RT

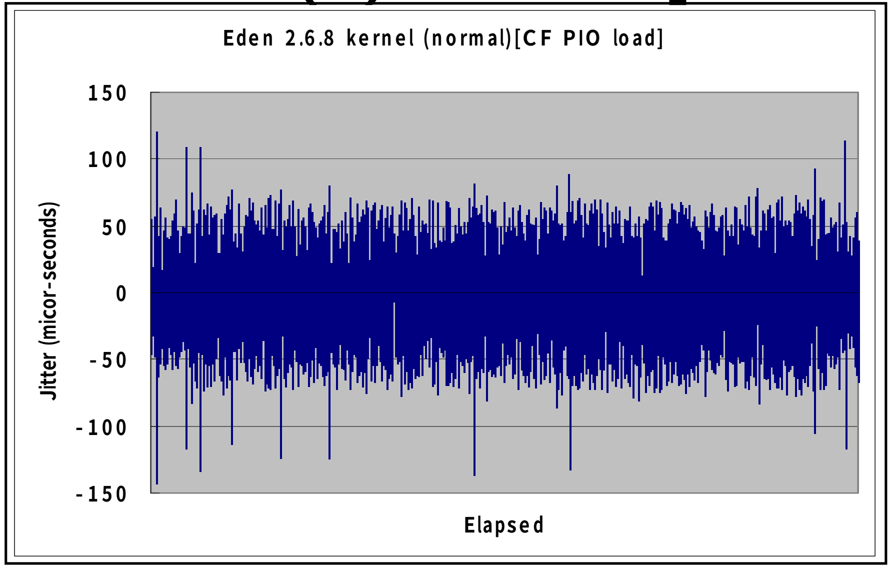


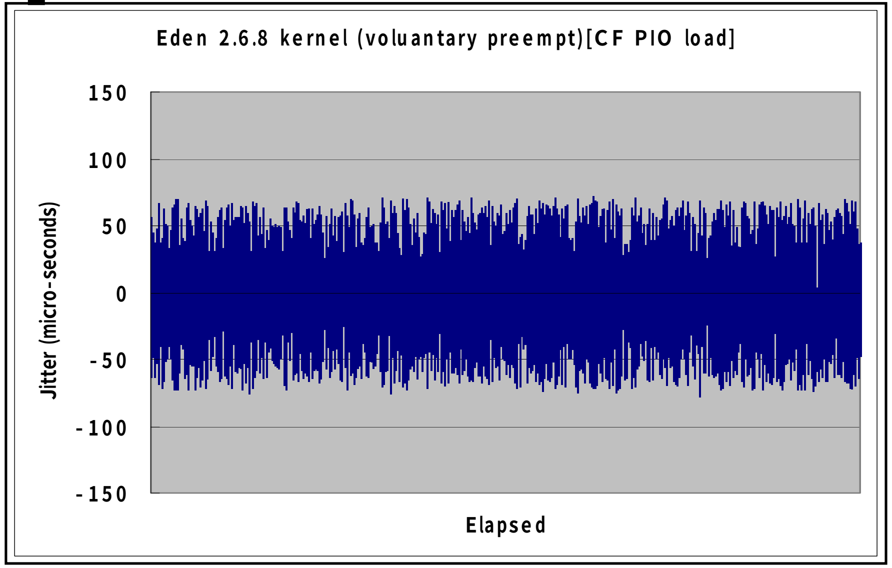


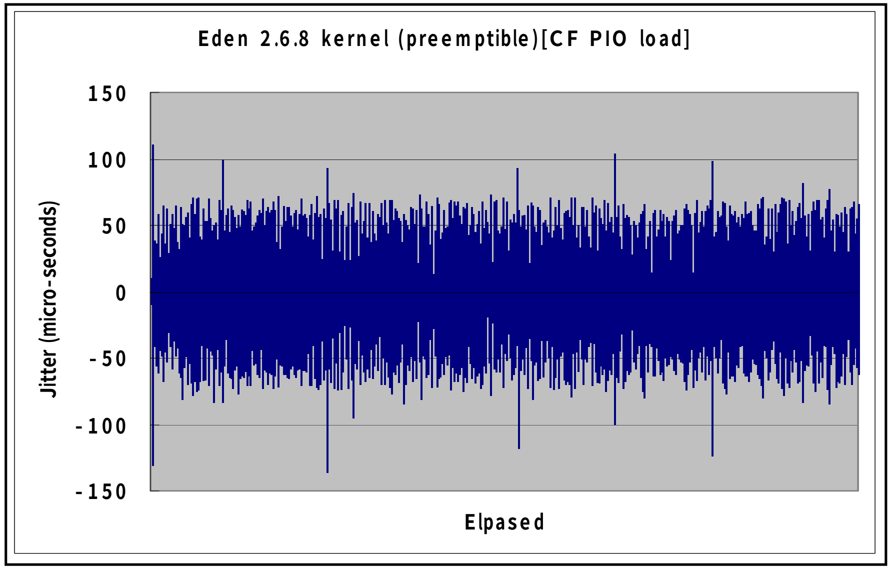


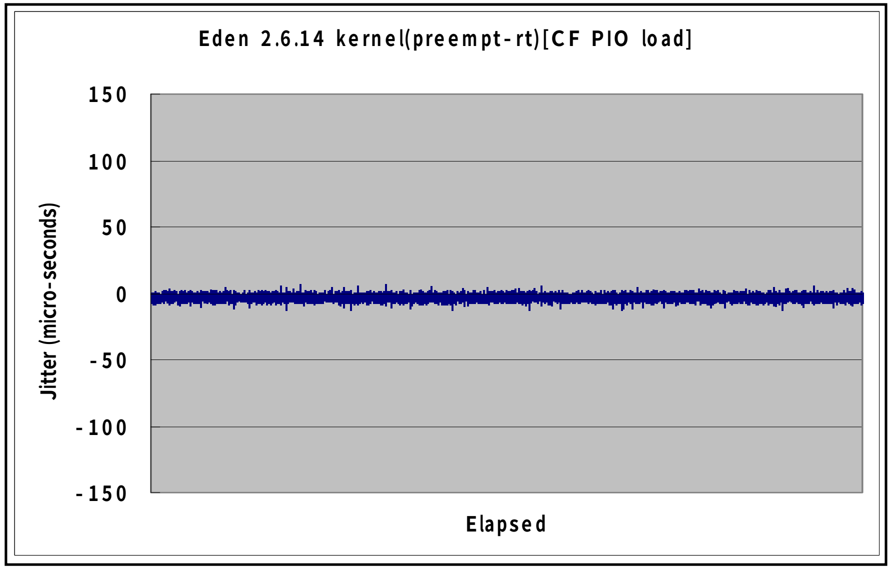


(4) Eden: [CF PIO] 2.6Kernel+RT

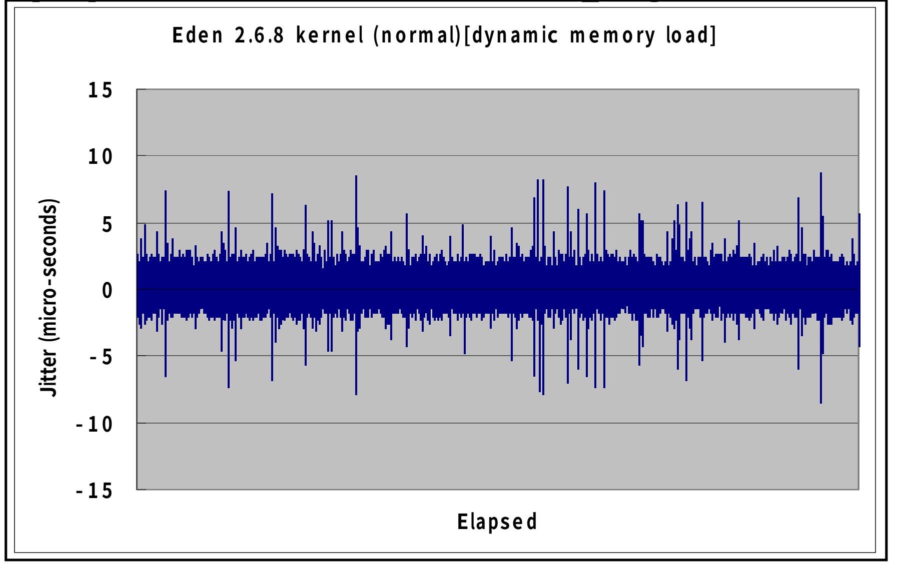


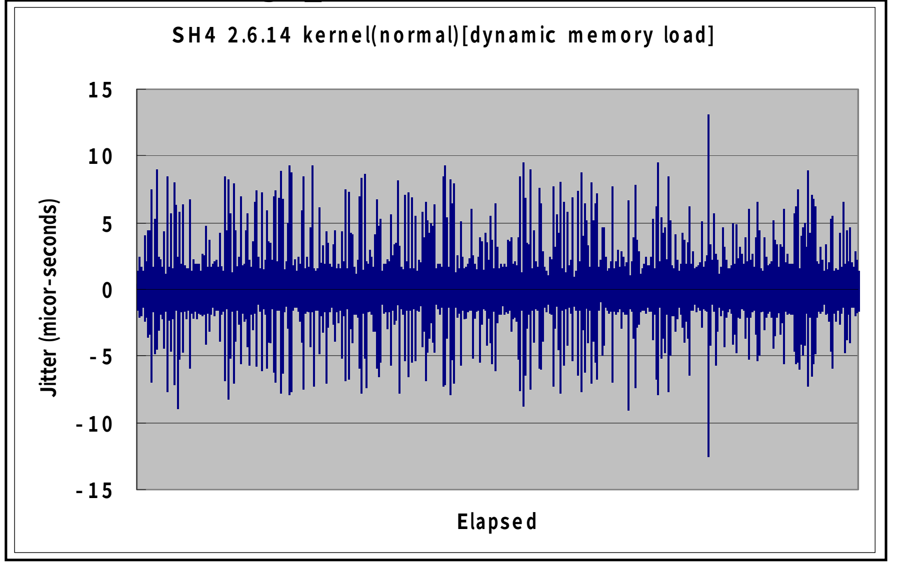


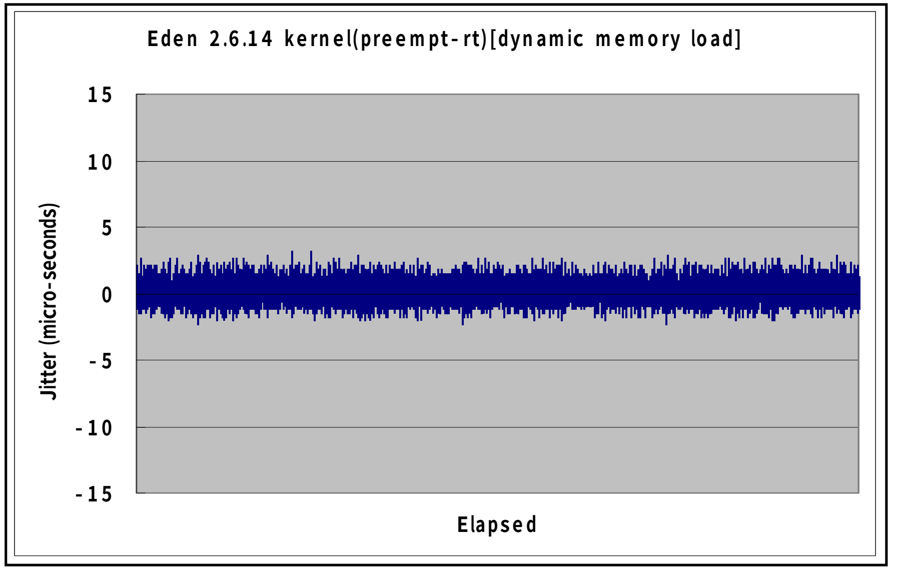


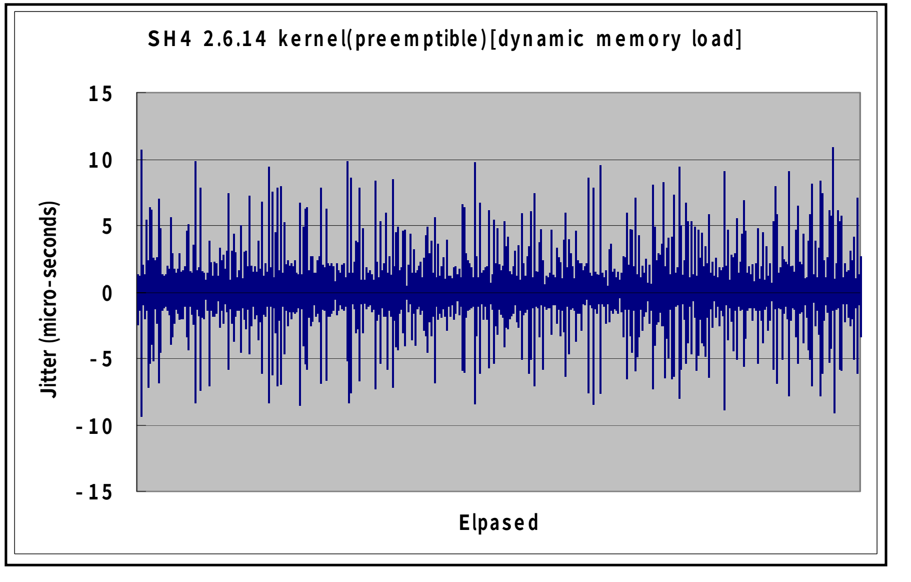


(3) Eden/SH4: [dynamic memory] 2.6Kernel+RT

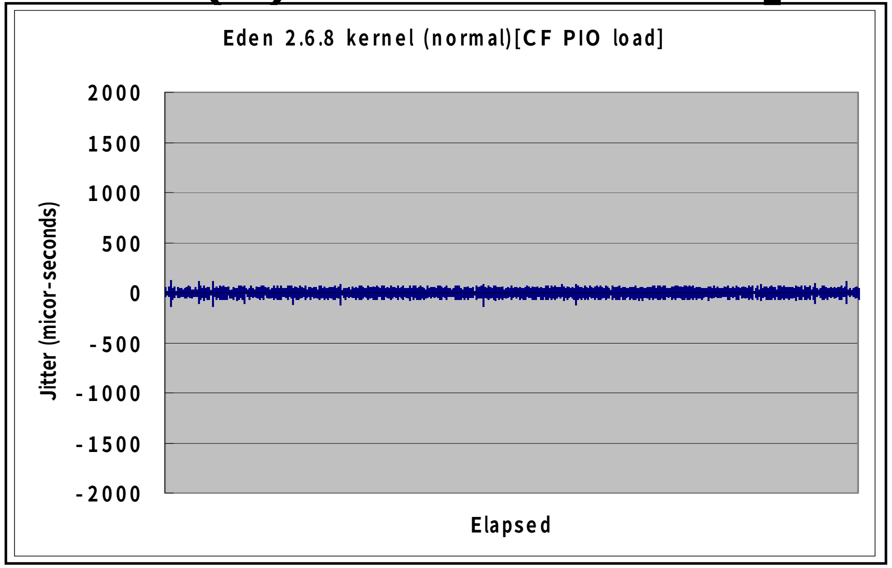


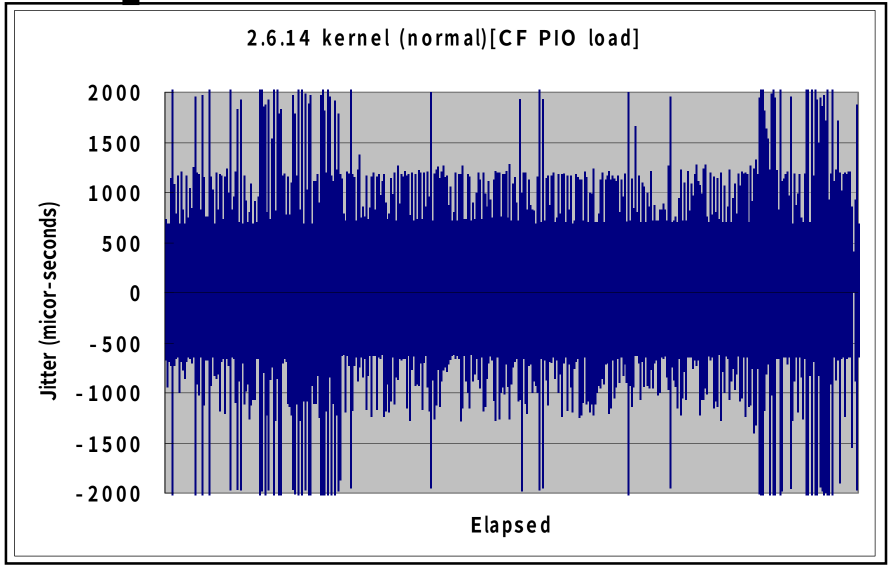


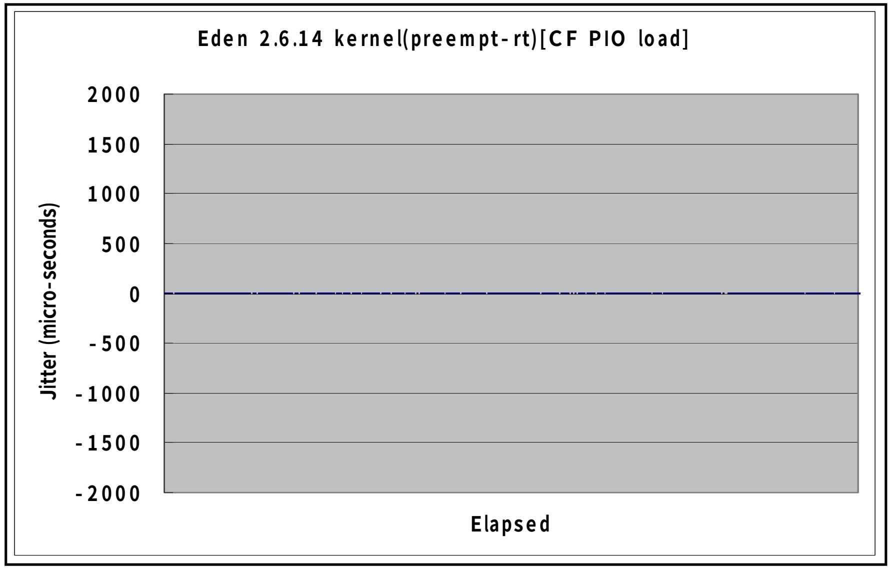


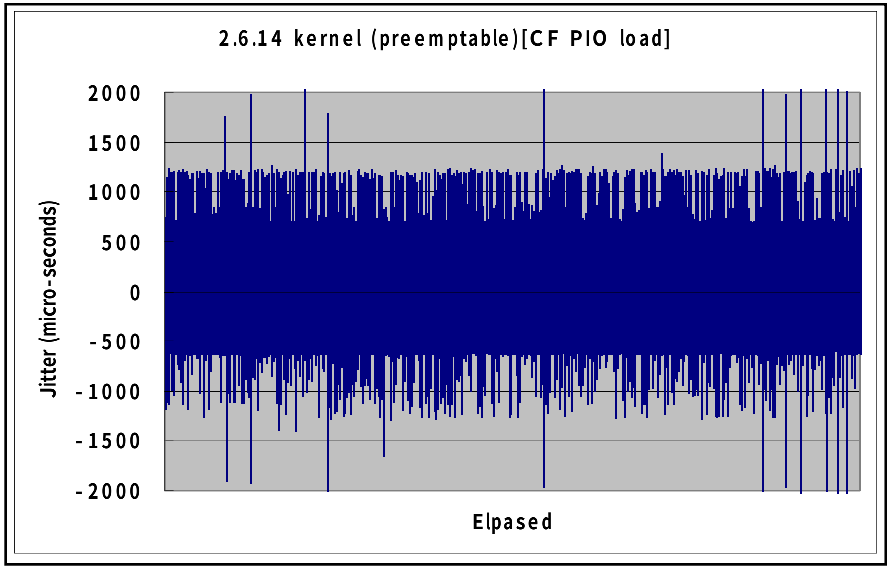


(4) Eden/SH4: [CF PIO] 2.6Kernel+RT









Case of Interrupt Response (*cont.*)

Summary

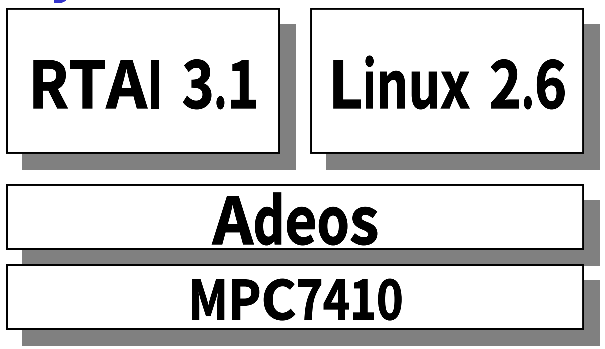
- Preempt-RT patch can realize 10 micro-seconds response time inner Linux kernel. It is effective to long-term running interrupt handler.
- Bad RT performance is also shown in interrupt response on SH4. (to be solved.)
- Expects threaded IRQ and PI mutex will be a key for performance improvement.

4. Evaluation (*cont.*)

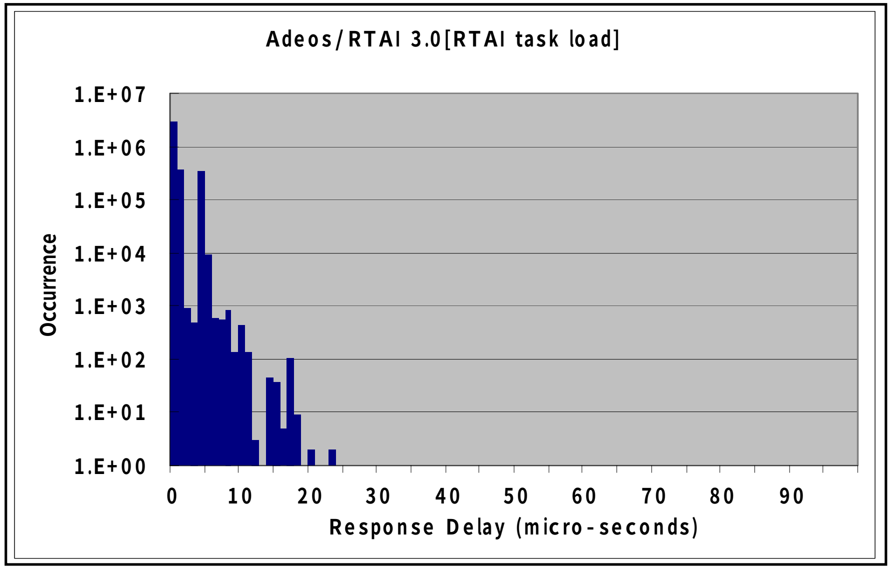
4.5. Comparison with Hybrid Approach

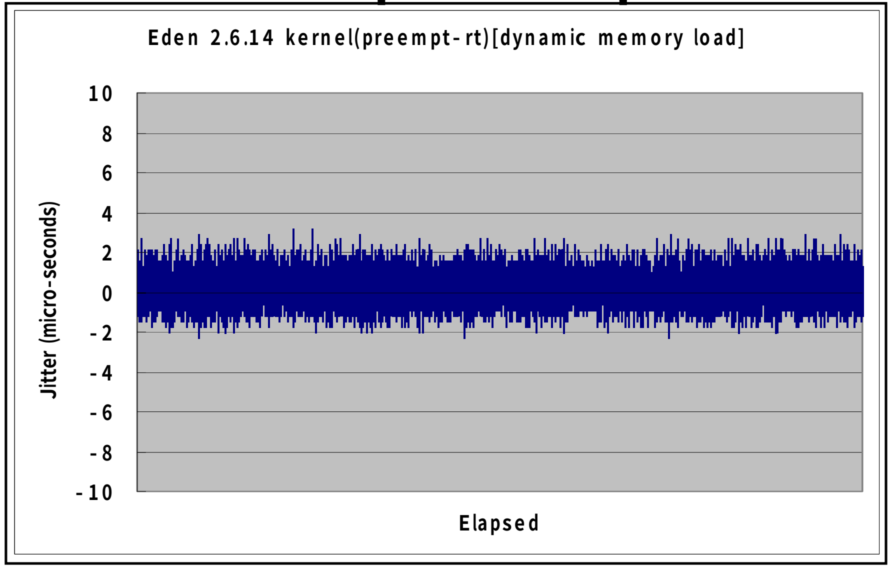
- Hybrid operating system will enable stable RT performance by its design. Is performance of preempt-RT Linux comparable with the hybrid Linux?
- Target: MPC7410 500MHz/ Eden 600MHz

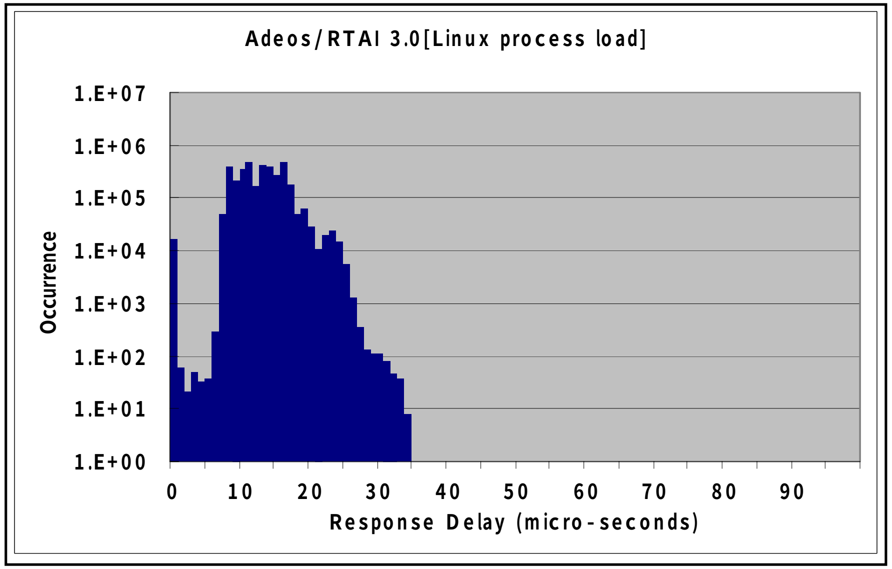
Reference of hybrid Linux

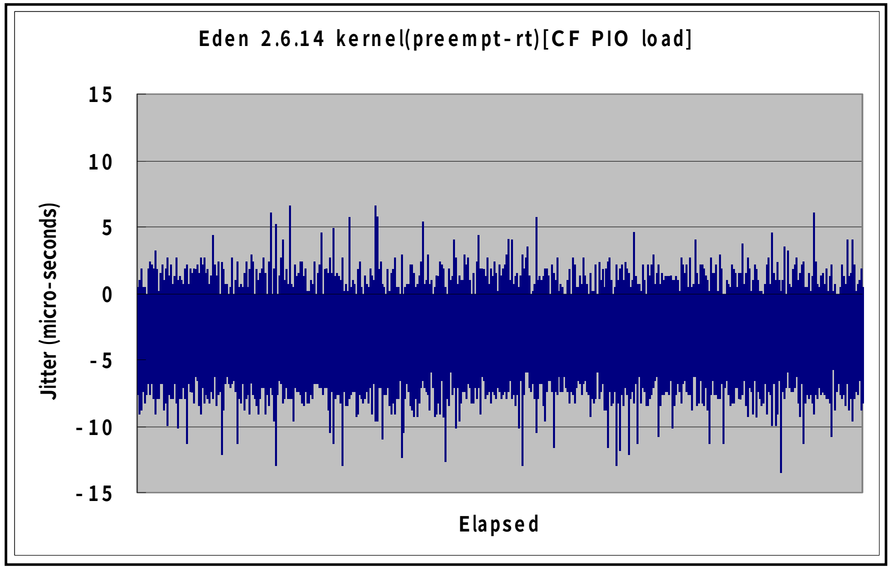


PPC7410: Adeos/RTAI vs. Eden preempt-RT









Comparison with Hybrid Approach (*cont.*)

Summary

- RT performance of hybrid Linux is stable on various application loads on RTAI or Linux.
- Preempt-RT patch can realize comparable RT performance with the hybrid Linux. Interrupt response time is same level as realtime operating system.

Issues

- Priority mapping of current kernel threads:
Which service should be prioritize? How to avoid priority inversion?
- Driver design guideline for realtime:
Need the reference design for good realtime performance: interrupt level control, nested interrupt handling, interrupt handler, tasklet, kernel thread.

5. Conclusion

- Embedded Linux will be ready for realtime soon. We will accelerate porting architecture dependent code of those RT patches.
- We understand we have solutions to satisfy our requirements.
  We can choose standard Linux kernel or it's extensions to adapt required realtime performance.