Kernel Performance Tests

aka Latency Test Episode III

Takashi Iwai <tiwai@suse.de>

SUSE Labs
SUSE Linux Products GmbH, Nuremberg, Germany

SUSE Labs Conference, September 21, 2005

Outline

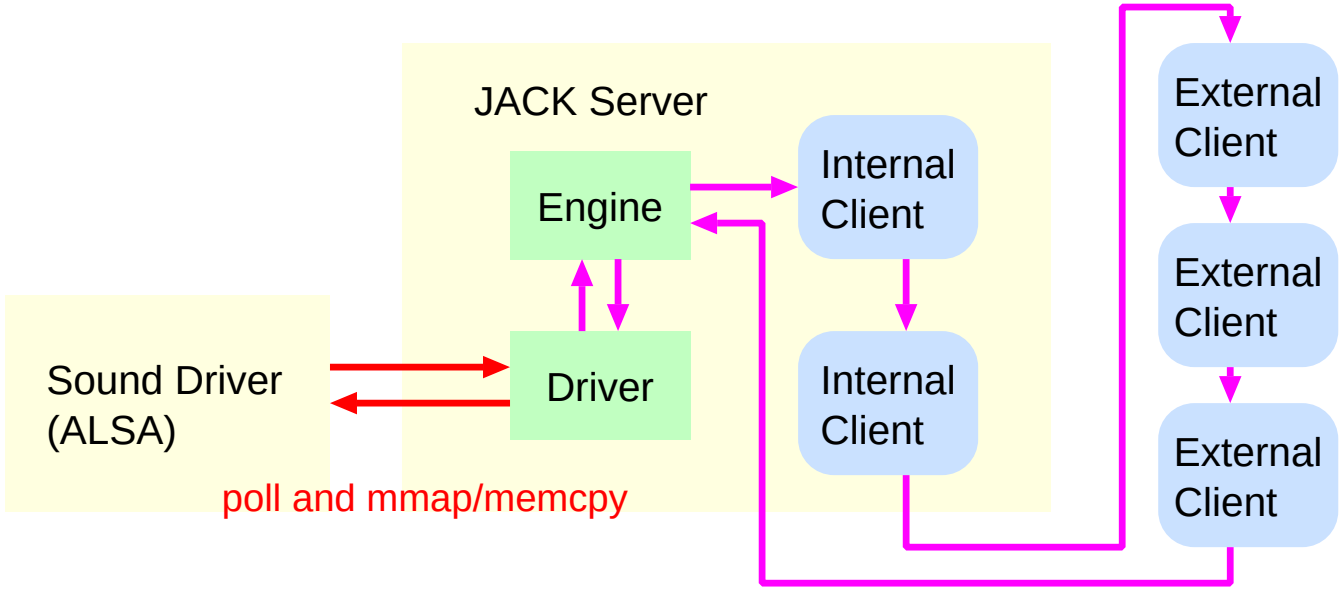
- RT on Linux
- RT-Latency Tests
- Throughput Benchmarks
- Interactivity Tests

What Is Real-Time OS?

- Real-Time (RT) operations
  - Can start a task exactly at the programmed time
  - Can process in the given limited time when an event occurs
  - Hard-RT - Assures in 100%
  - Soft-RT - Mostly performs in RT
- Demands: Embedded devices and Desktop
- RT-critical devices
  - Cellphone, RT-Controlling system
- RT-critical applications
  - RT-effect, editing: Audio, Video

Example: RT-Audio System with JACK

- JACK: Jack Audio Connection Kit

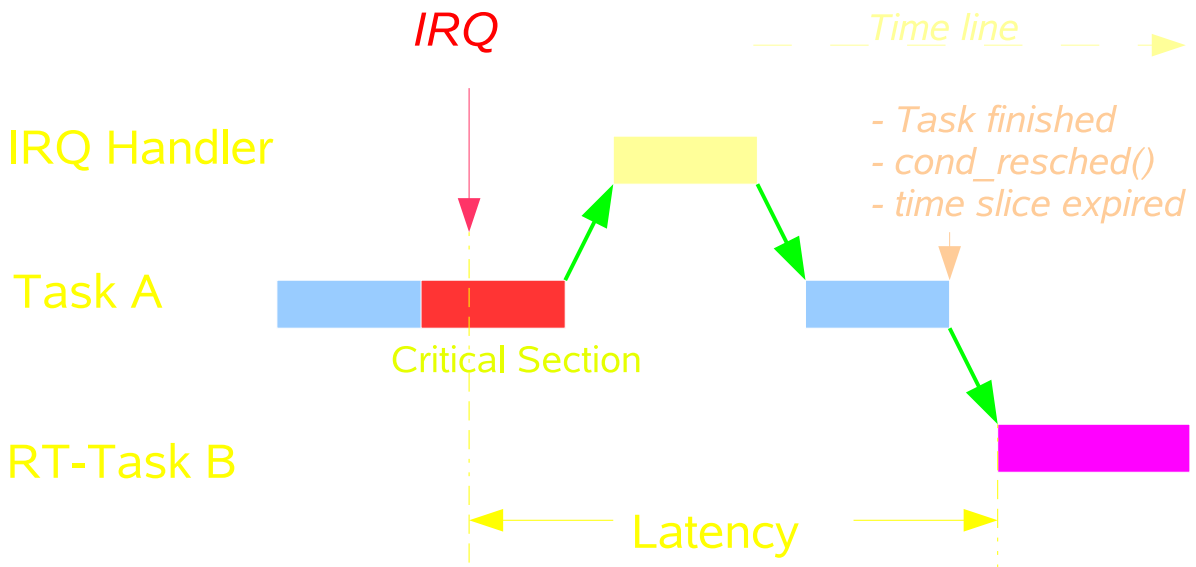


- Real-time audio system
  - External client: independent processes
  - Woken up at each time the audio is processed
- Typically, 1-2 ms latency is required

RT on Linux

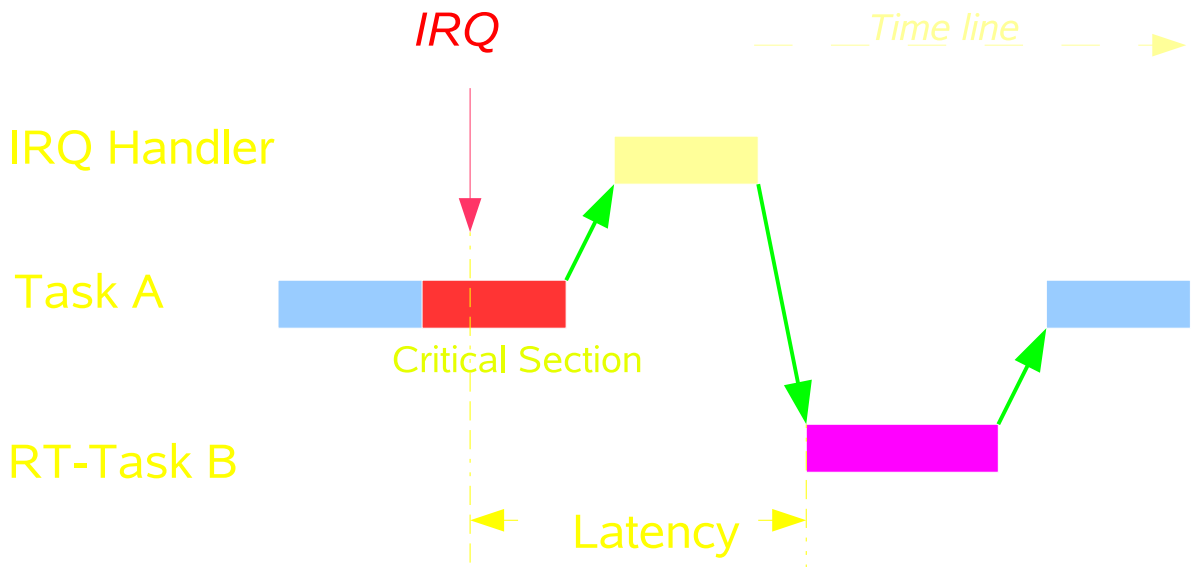
- Linux is not RT-OS
  - ... but can be Soft- or Semi-Hard-RT
- POSIX Real-Time API
  - SCHED_FIFO, SCHED_RR classes
  - /usr/include/sched.h
    - sched_setscheduler(), sched_yield(), ...
  - pthread version for threads
    - pthread_attr_setschedpolicy(), ...
- RT-process can block the system easily
  - Available only for privilege users
  - Permission control via rlimit (since 2.6.12)
    - Flexible control via PAM

Normal Linux Kernel



- Non-preemptive task switch
  - Task is finished
  - Manual switch by cond_resched()
    - LL-Patch, Voluntary Preemptive Patch
  - Force to switch by scheduler: time slice expired
- Critical section prevents switching

Preemptive Linux Kernel



- Task can be switched after irq handler
- Still can’t work during critical section
  - Leads to big latencies
- Overhead at each spinlock
- Scalability problems, e.g. RCU

A Hard Way To Hard RT

- RT-Patch
  - Fix RT problems in Linux kernel itself
    - Threadize IRQs (hard and soft-irq threads)
    - Preemptive spinlocks using mutex
    - Priority inheritance mutex
    - Preemptive RCU
  - Other similar patches
    - MontaVista patch
    - Patches on CELF (irq threads, prio irq)
    - R2-Linux, ART-Linux (2.4 kernels)
- Pro
  - API is kept
    - The same binaries can run on RT-kernel
- Con
  - Complexity, more overheads

Another Hard Way To Hard RT

- Micro-Kernel
  - RT-Linux
  - RTAI / Adeos (i-pipe) / Fusion
  - Based on a micro/nano kernel
  - Run Linux kernel as a task (domain)
  - Run RT-tasks parallel
- Pro
  - Simpler implementation (from Linux kernel POV)
- Con
  - Own RT-APIs (similar with POSIX)

- Hybrid Methods
  - Linux kernel on another RT-kernel
    - Linux on ITRON

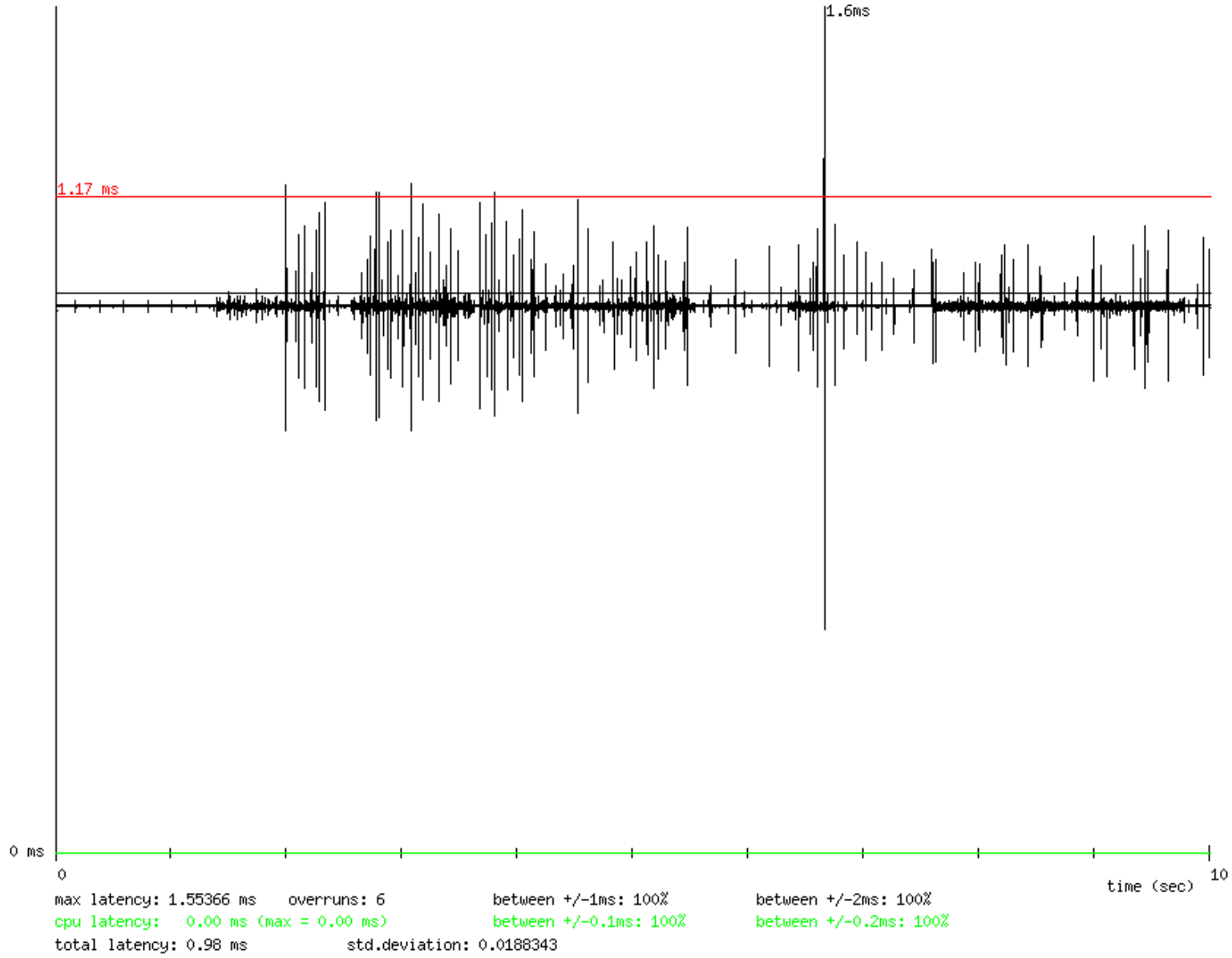
Latency Test Suite

- Latency test suite
  - Originally developed by Benno Senoner
  - User-space tool
  - Measure periodical interrupt response time
    - Using a sound driver for interrupt source
    - Use RTC as an irq source
  - Under different loads
    - Disk I/O, Procfs, X, Processes
  - Records stack traces over IRQ deadline
    - Using a dedicated kernel module
- Let’s check our latest and greatest kernel

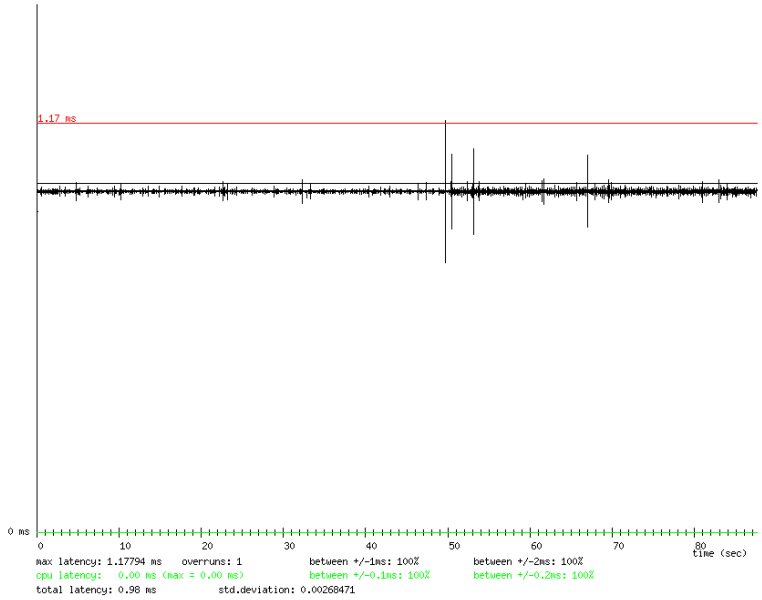
Condition

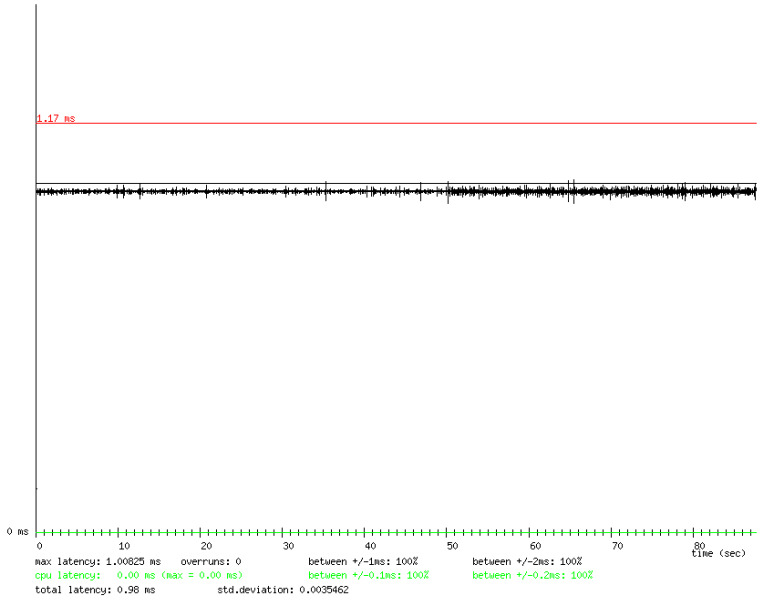
- CPU: Pentium 4 / 2.25GHz
- Memory: 512MB, 2GB
- Kernel: UP & SMP/HT 2.6.13 kernel (SL10.0)
  - NOPE = No Preemption
  - VP = Voluntary Preemption
  - PE = Preemptive Kernel
  - RT = RT-Kernel
- FS: Reiserfs
- Background loads:
  - X11 - X11perf
  - Procfs - top on 0.1 sec interval
  - Disk - read / write / copy bulk (2GB, 4GB)
  - File create/delete (2 x 2 x 10000 files)

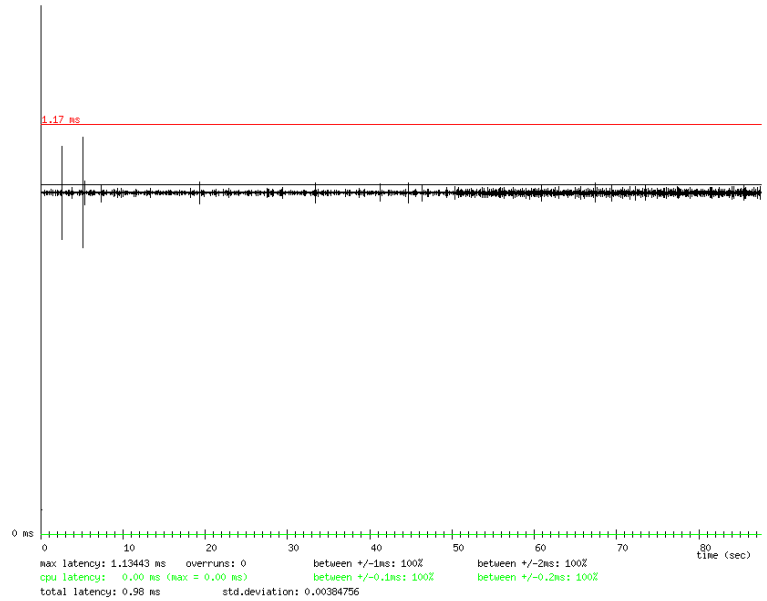
Example

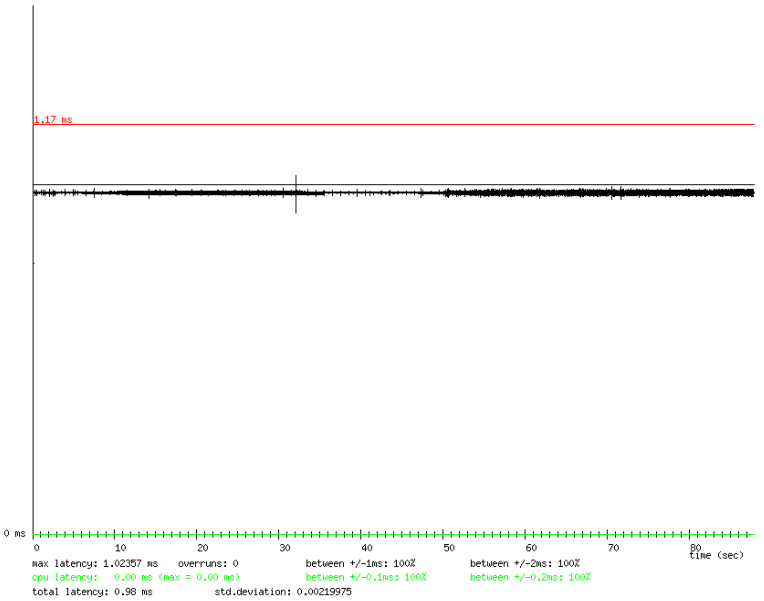

1.6ms
1.17 ms
0 ms
0
10
time (sec)
max latency: 1.55366 ms
overruns: 6
between +/-1ms: 100%
between +/-2ms: 100%
cpu latency: 0.00 ms (max = 0.00 ms)
between +/-0.1ms: 100%
between +/-0.2ms: 100%
total latency: 0.98 ms
std.deviation: 0.0188343

UP-kernel (X11 Loads)

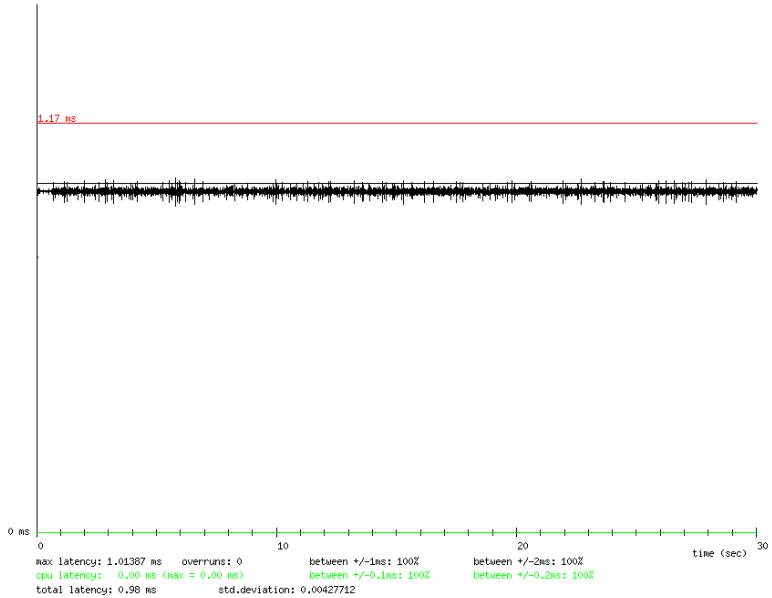


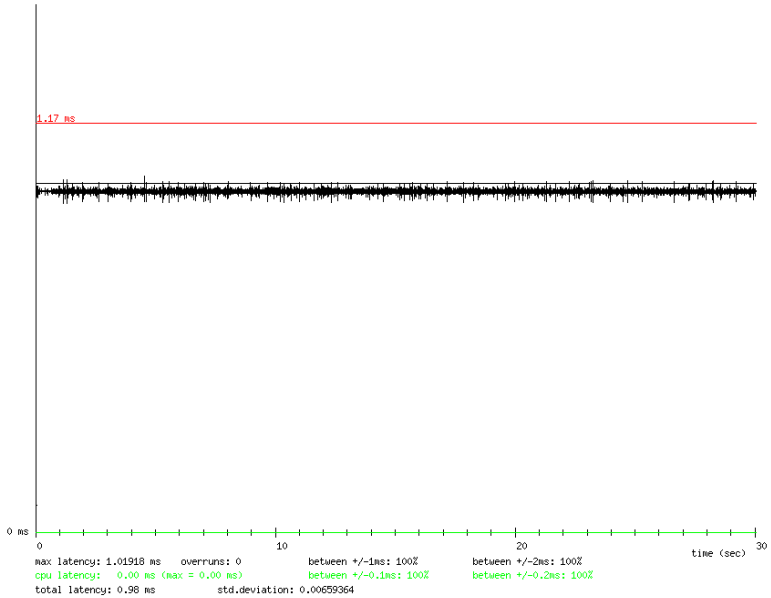


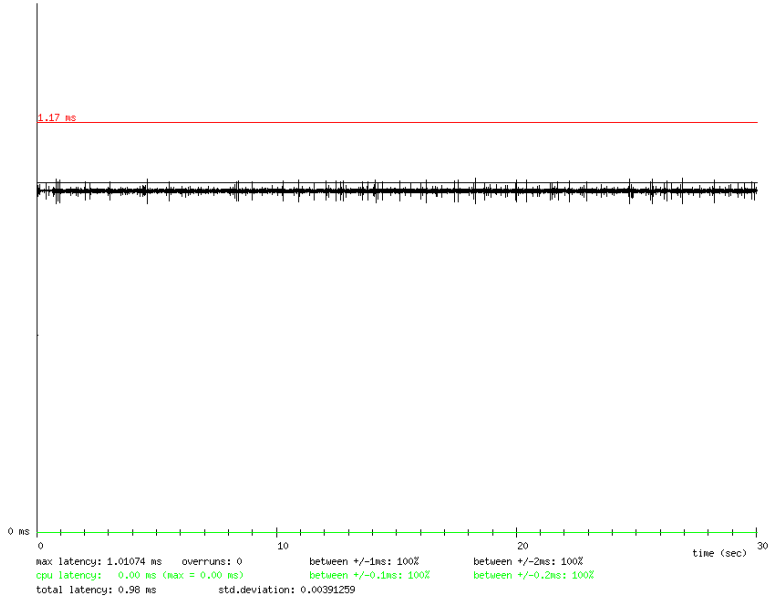


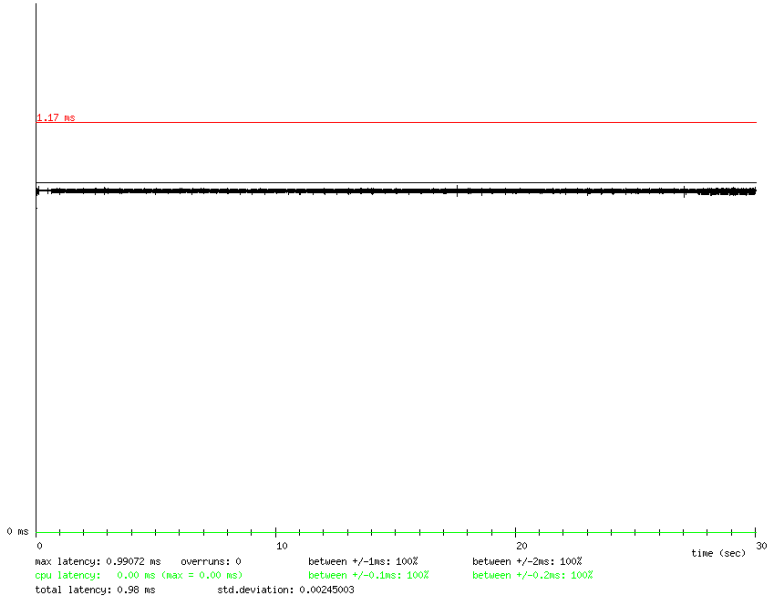


UP-kernel (Procfs)

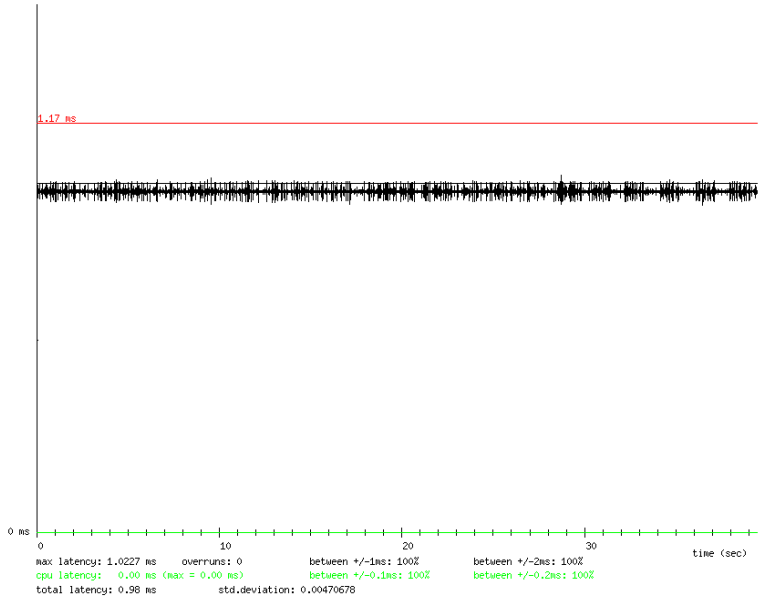


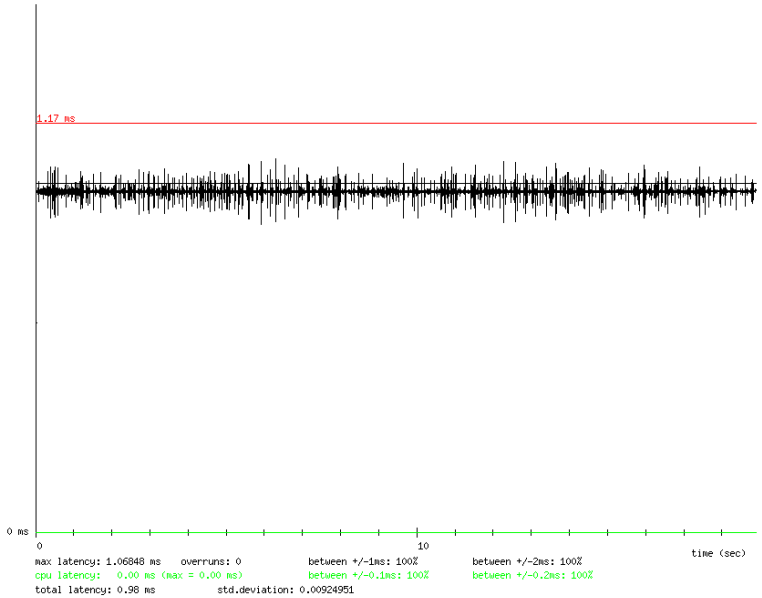


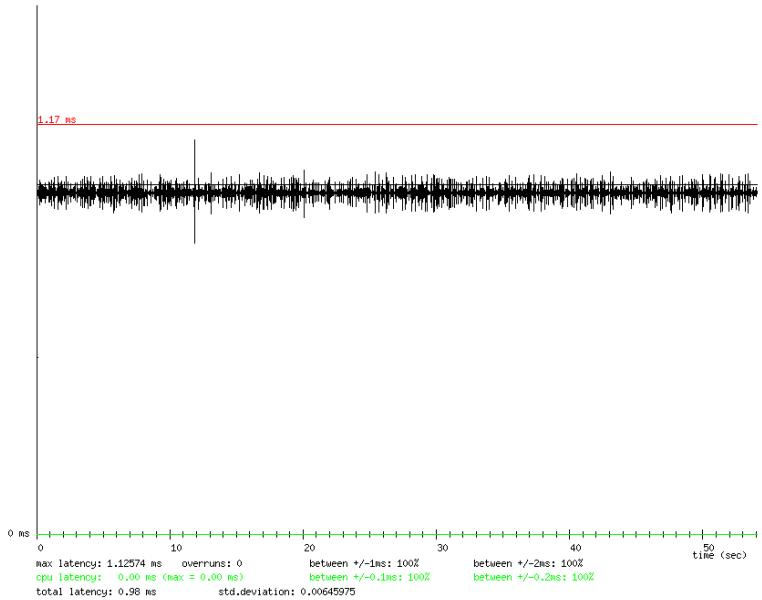


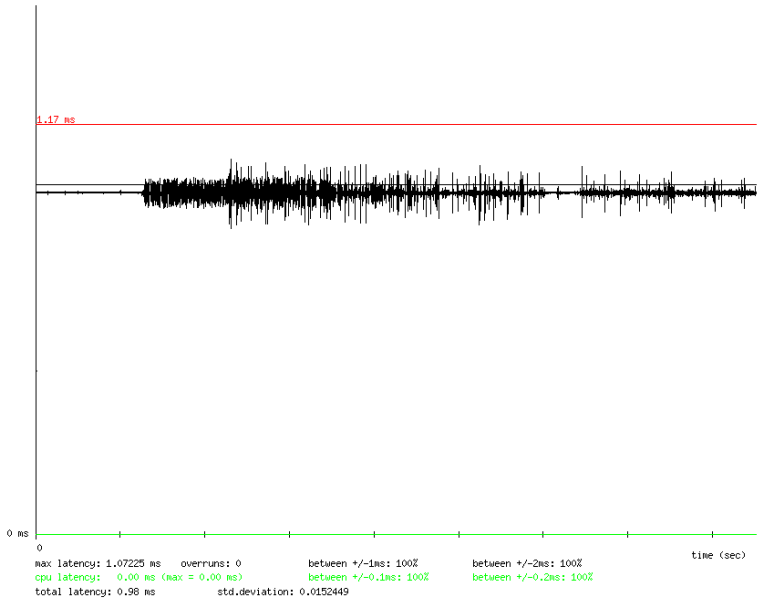


UP-kernel (NOPE / Disk)

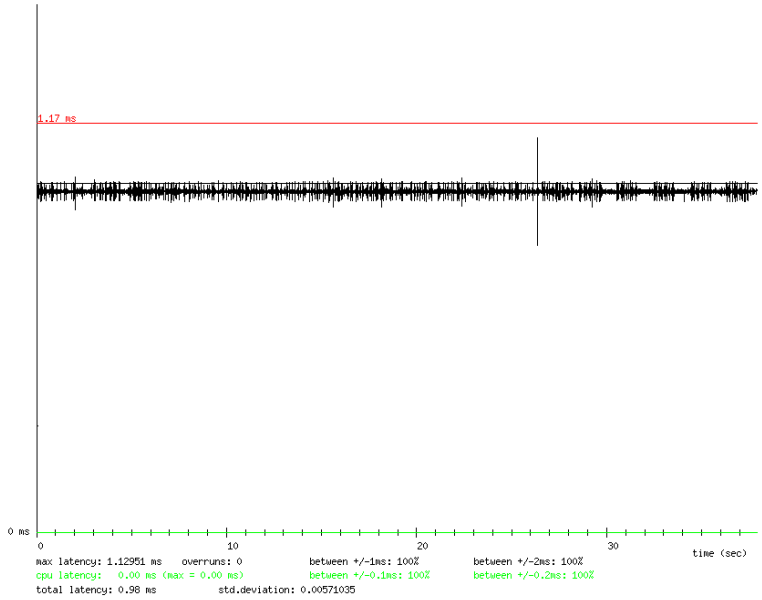


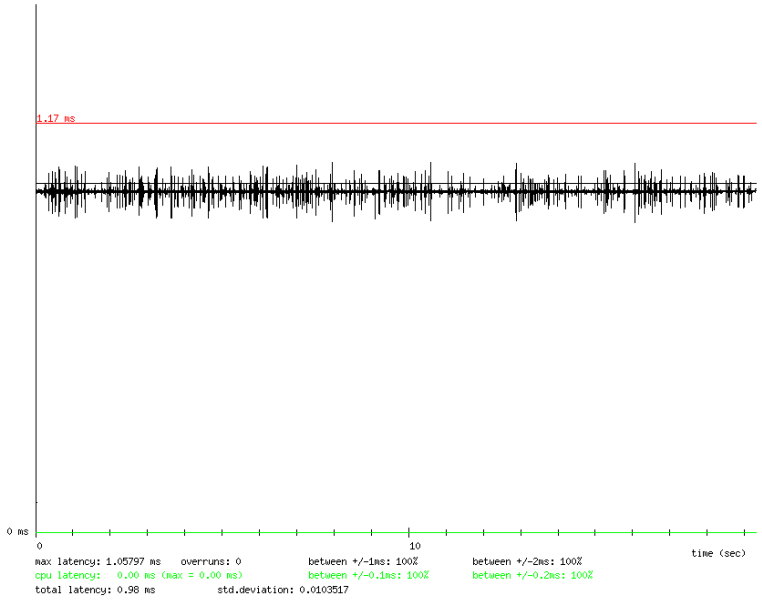


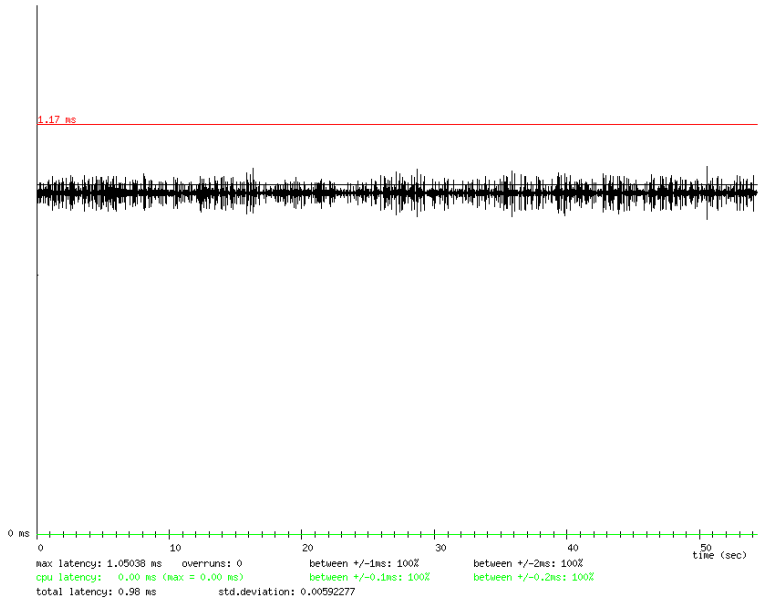


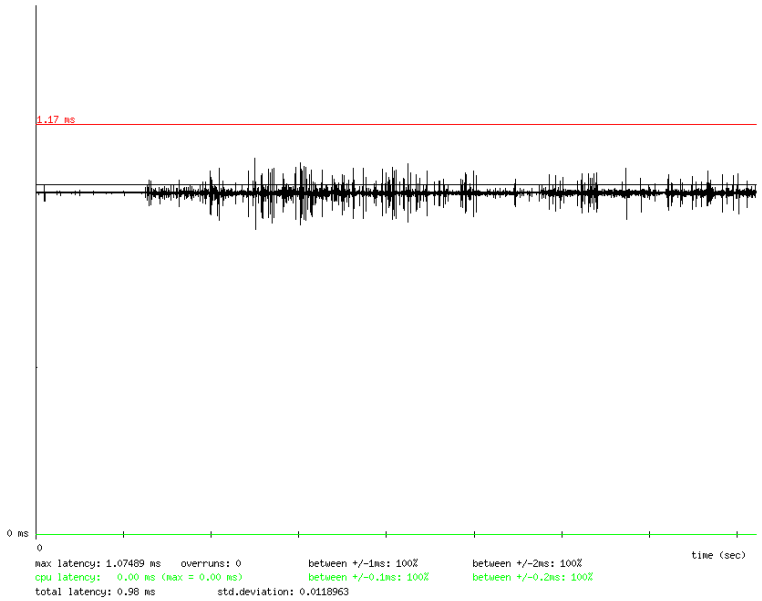


UP-kernel (VP / Disk)

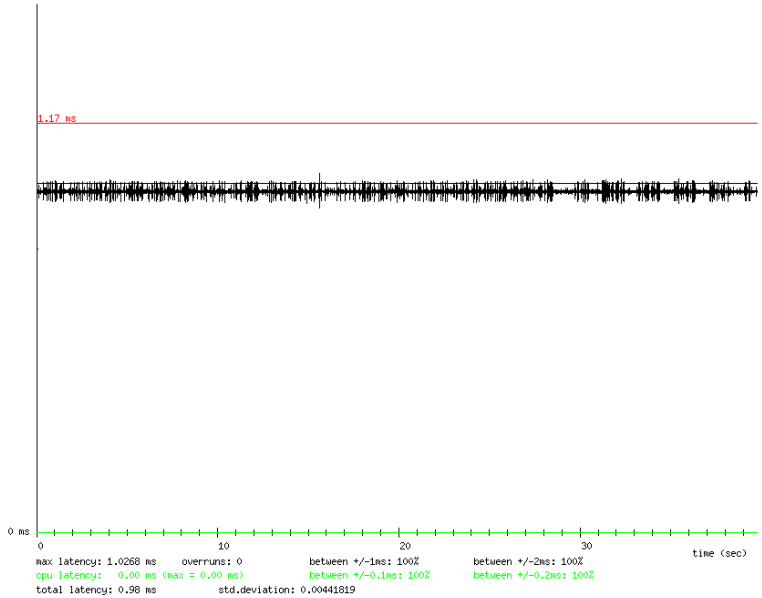


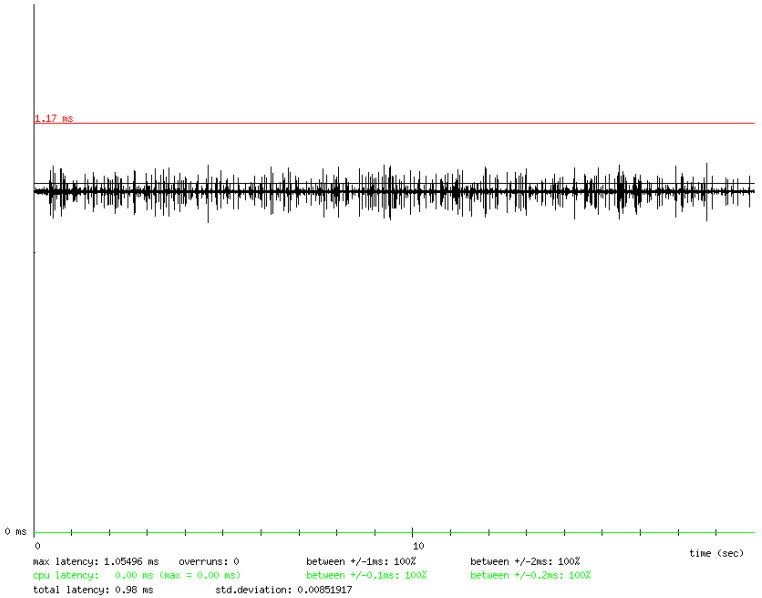


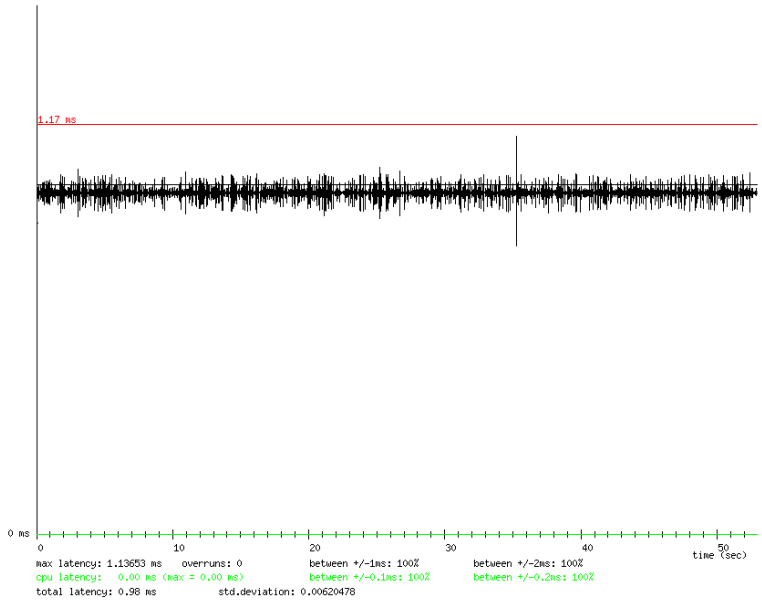


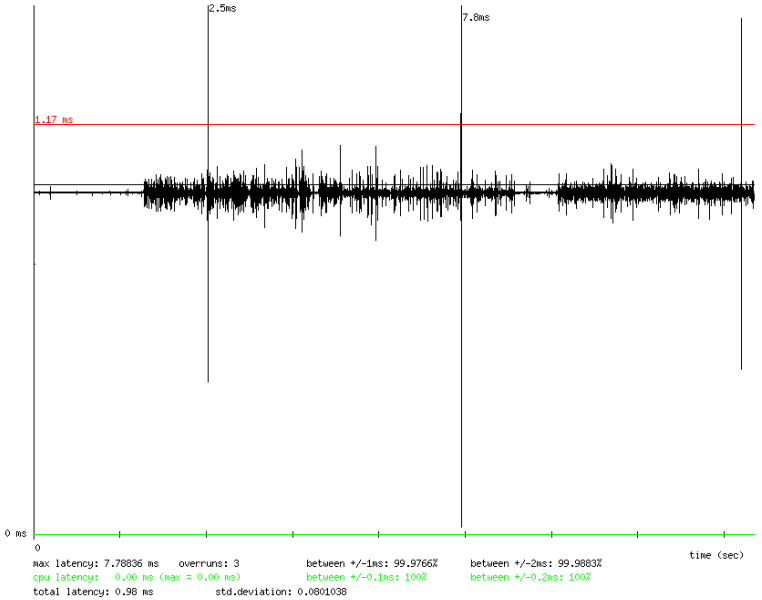


UP-kernel (PE / Disk)

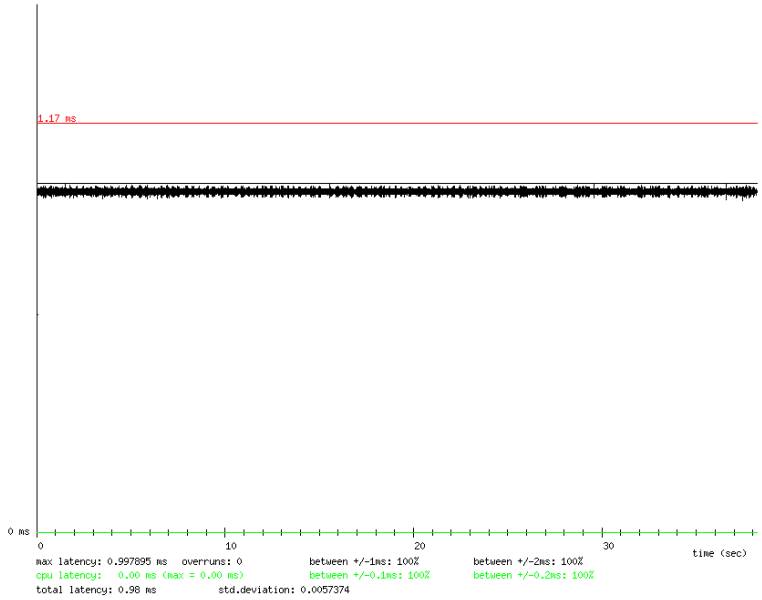


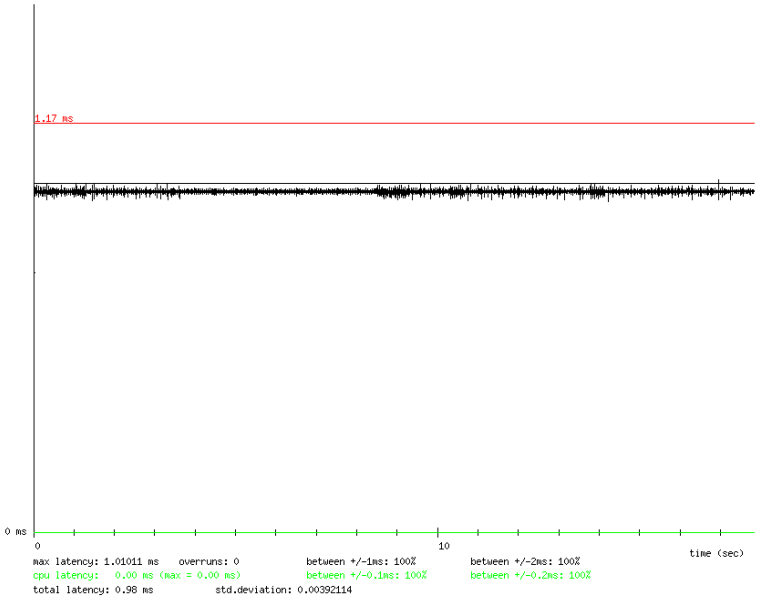


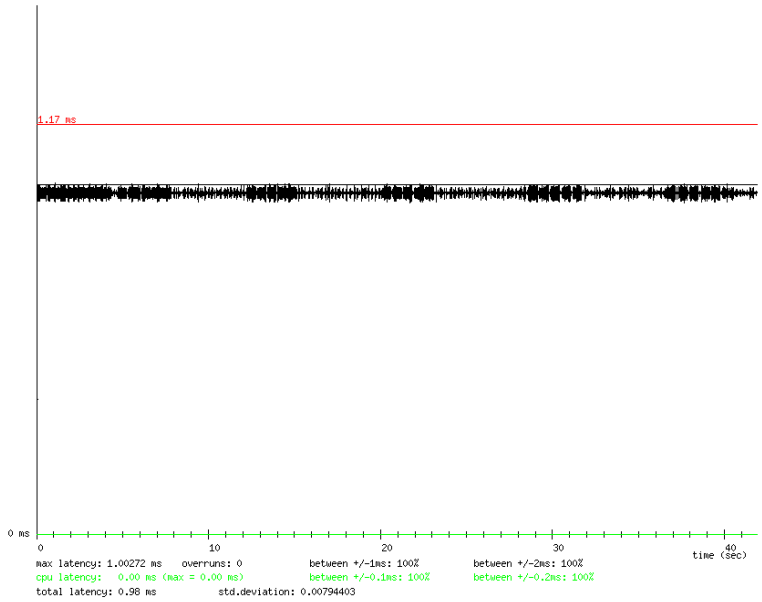


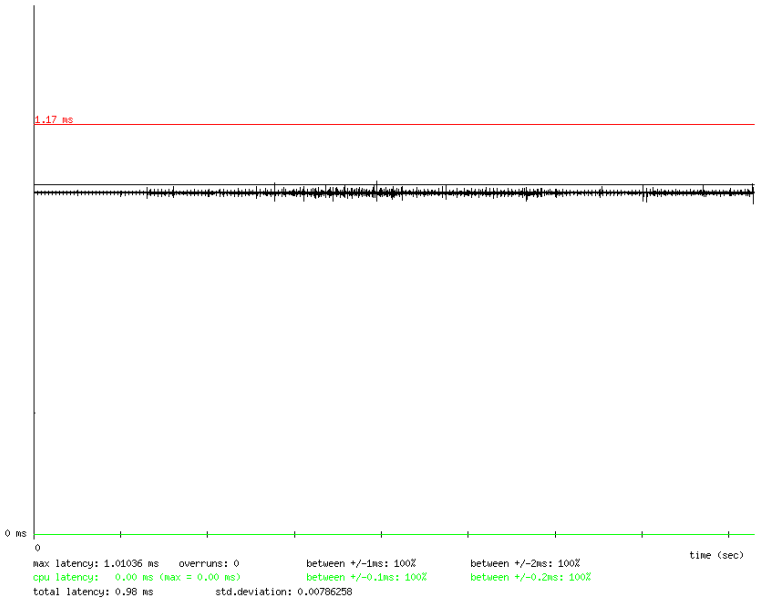


UP-kernel (RT / Disk)

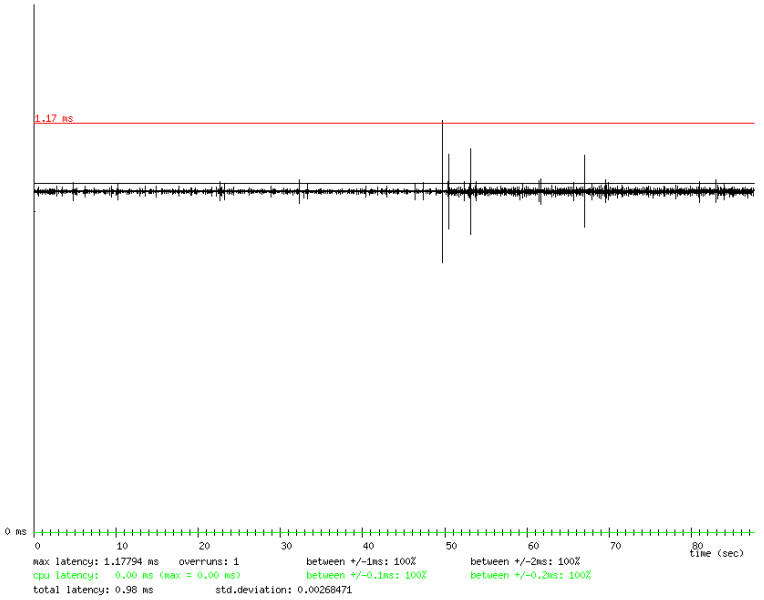


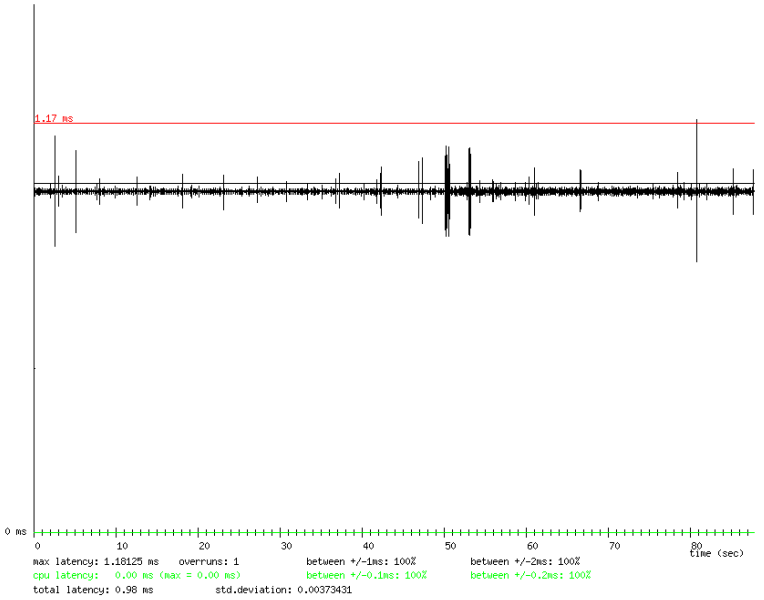


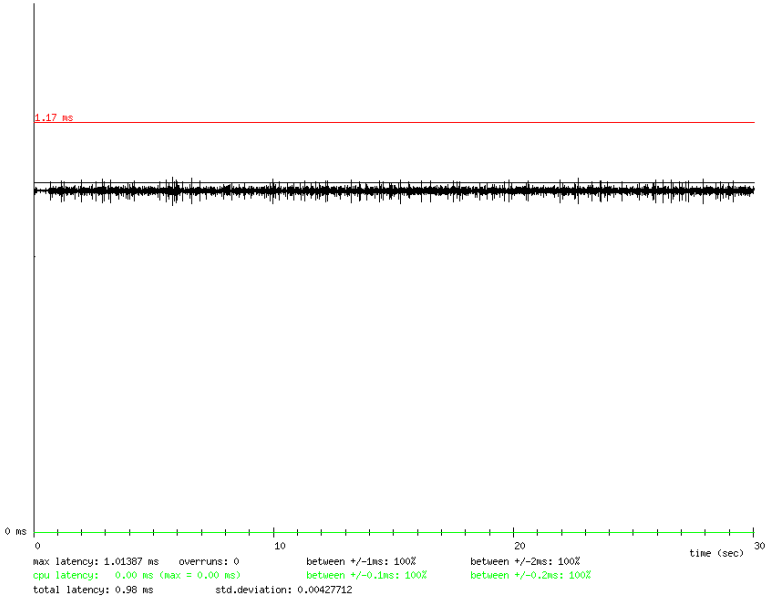


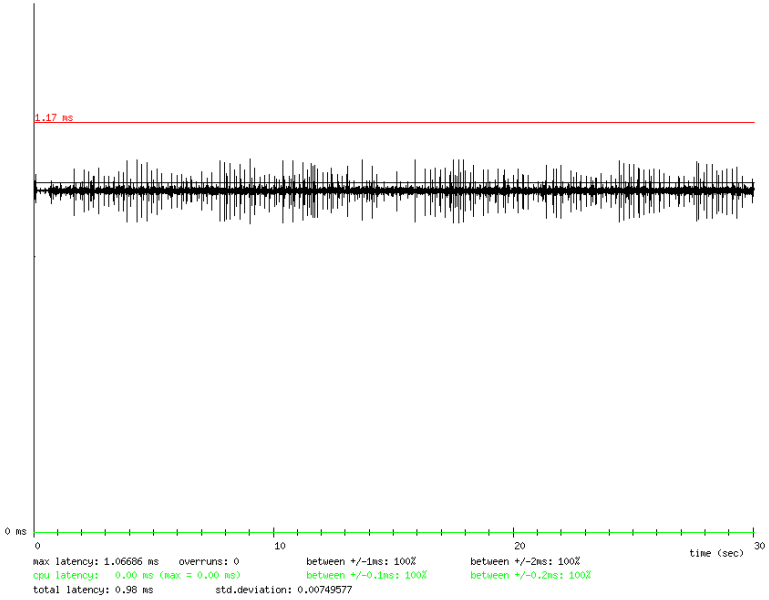


Influence of HIGHMEM (X11, Procfs)

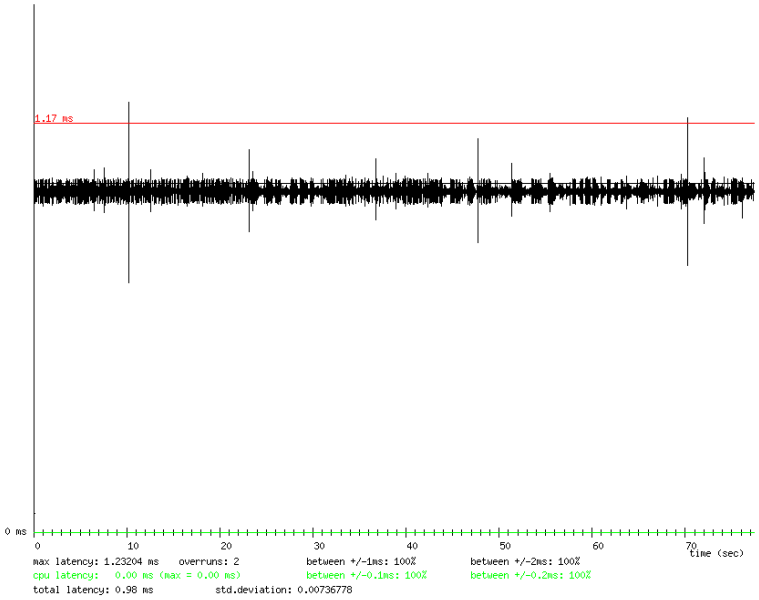


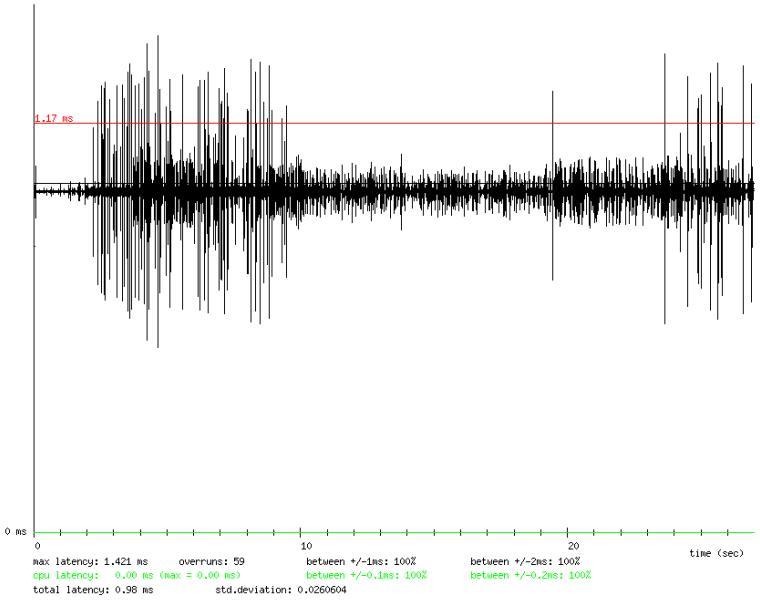


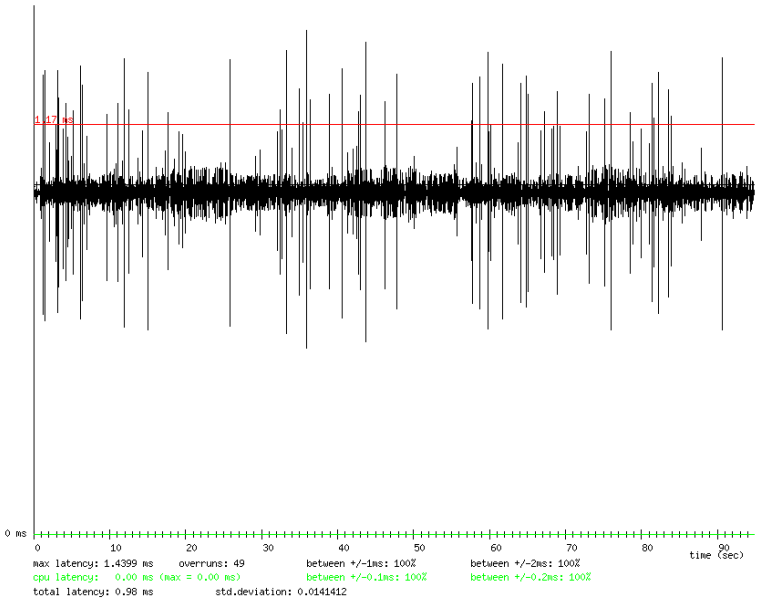


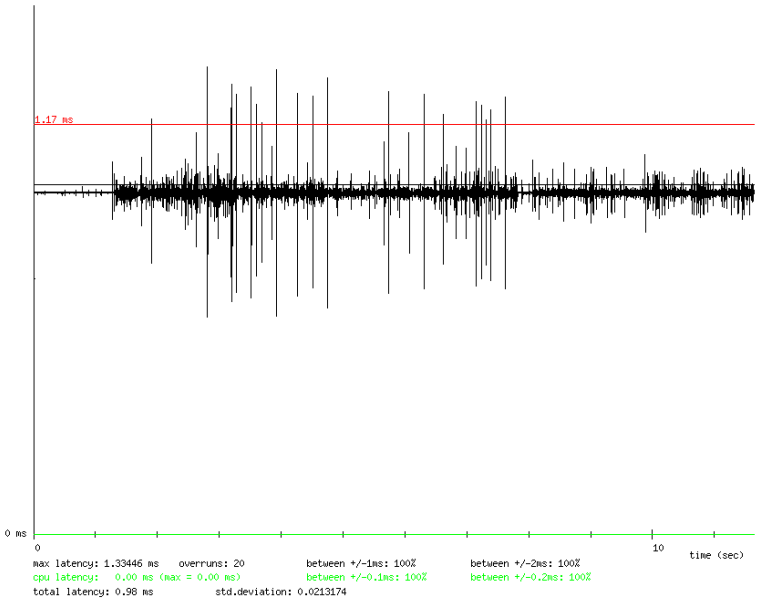


Influence of HIGHMEM (DISK)

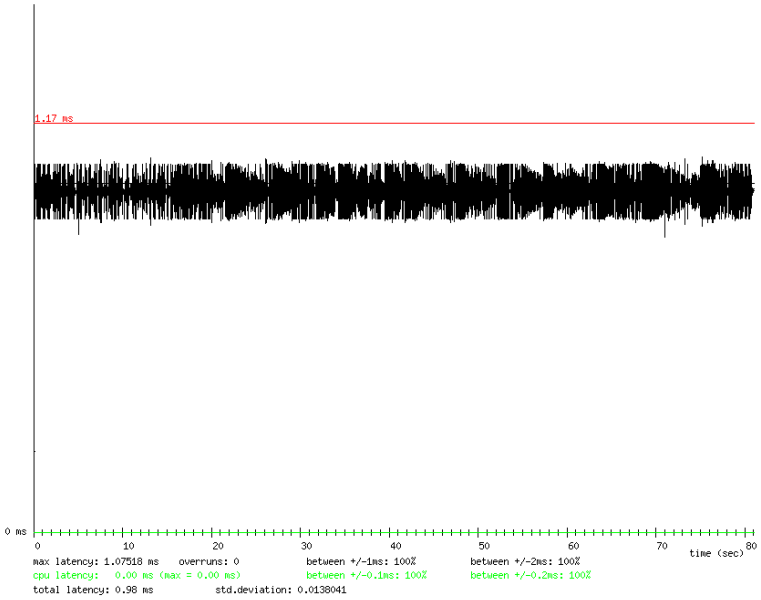


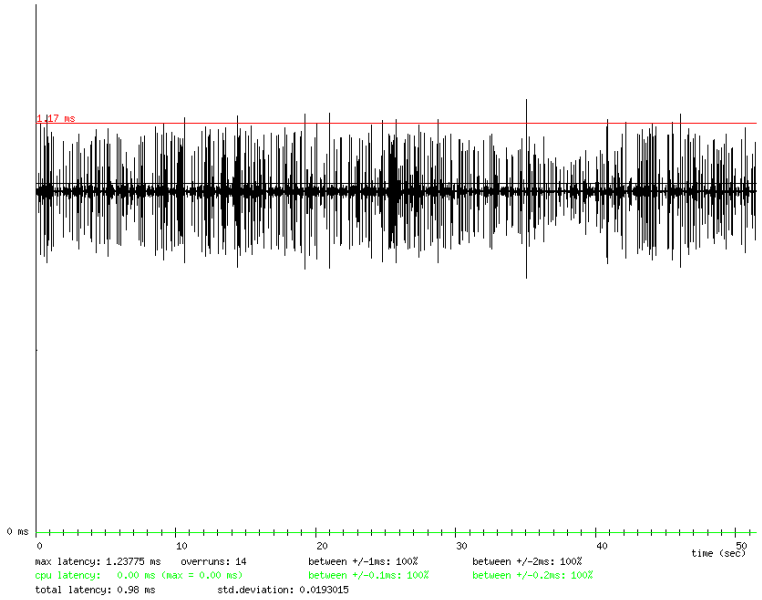


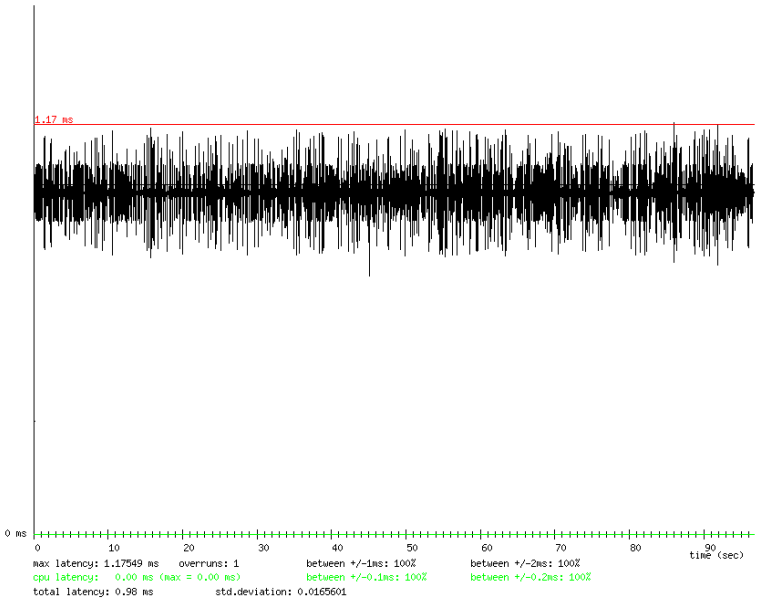


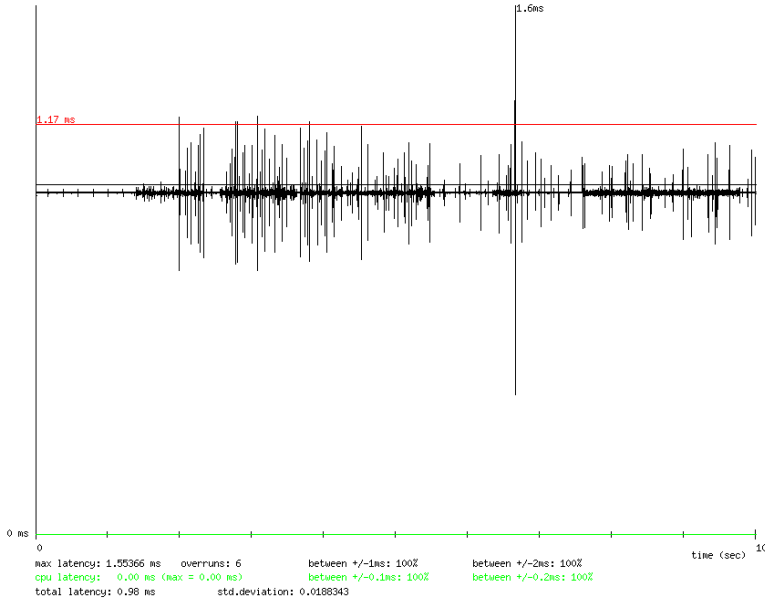


SATA DISK Device (NOPE / DISK)





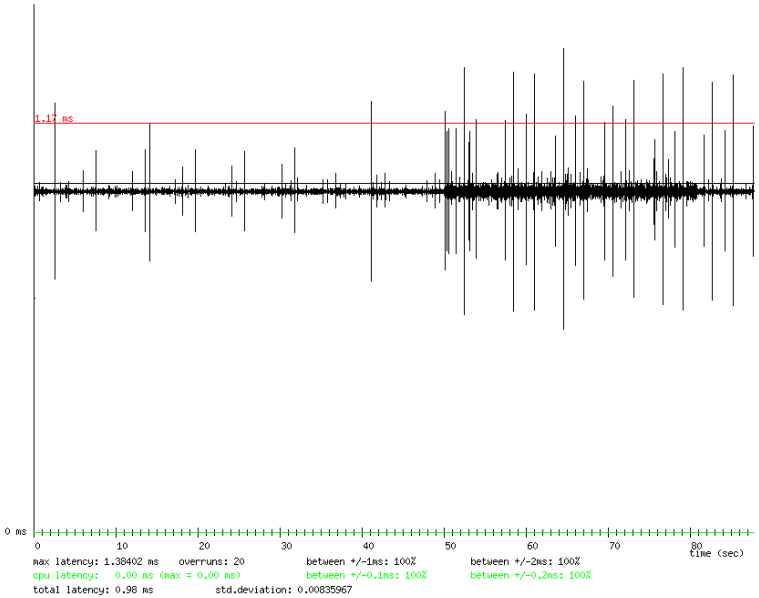


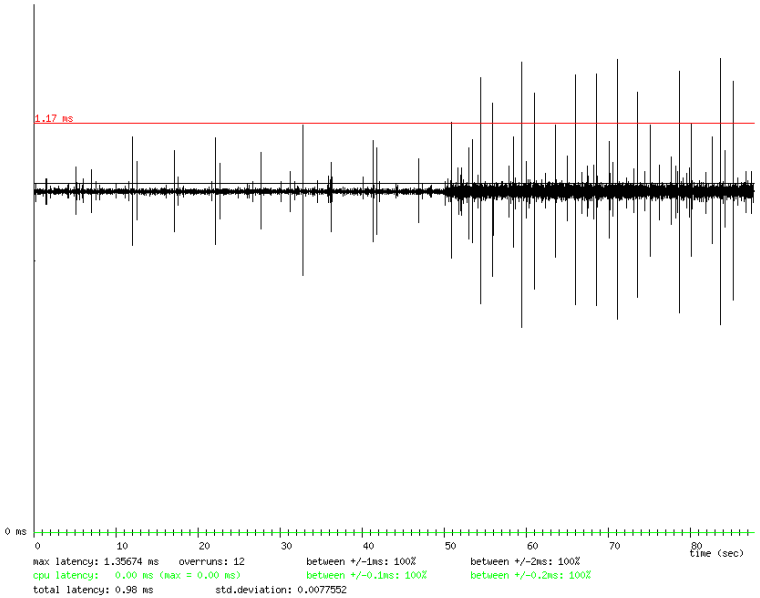


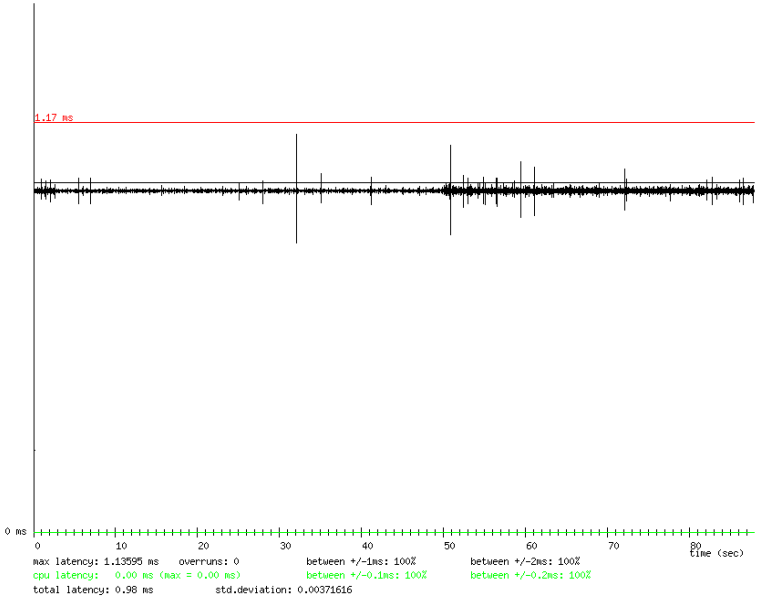
Remarks on Tests with UP Kernels

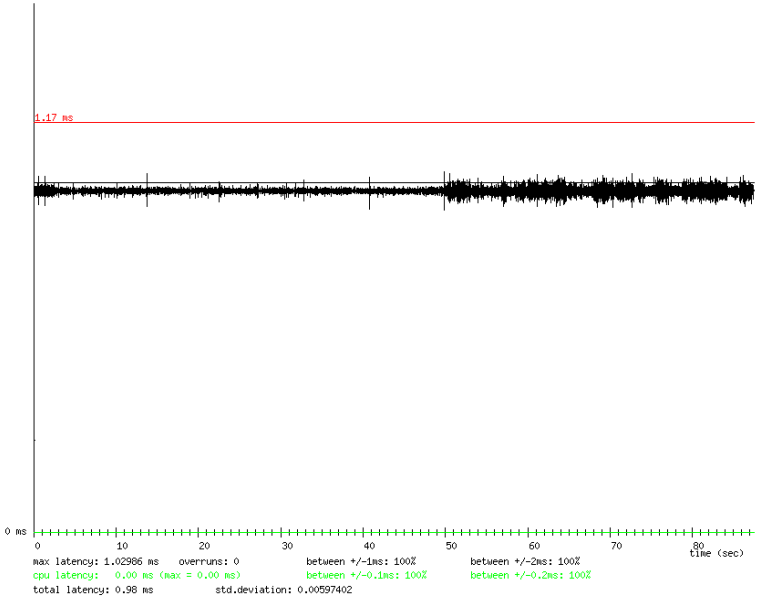
- All UP-kernels show good latency
  - Usually < 1ms, even on NOPE kernel
- VP-kernel improves some corner cases
- No big difference between VP and PE kernels
  - Critical section must be solved
- RT-kernel shows ~100us latency
- HIGHMEM (2GB) leads to higher latency
- SATA disk tends to result in higher latency
  - Both still acceptable (< 1ms)

SMP/HT-Kernel (X11 Loads)

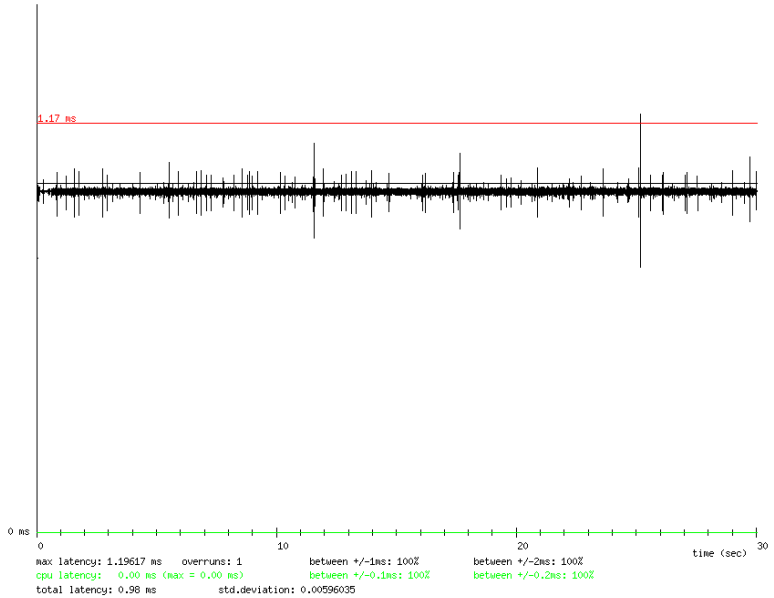


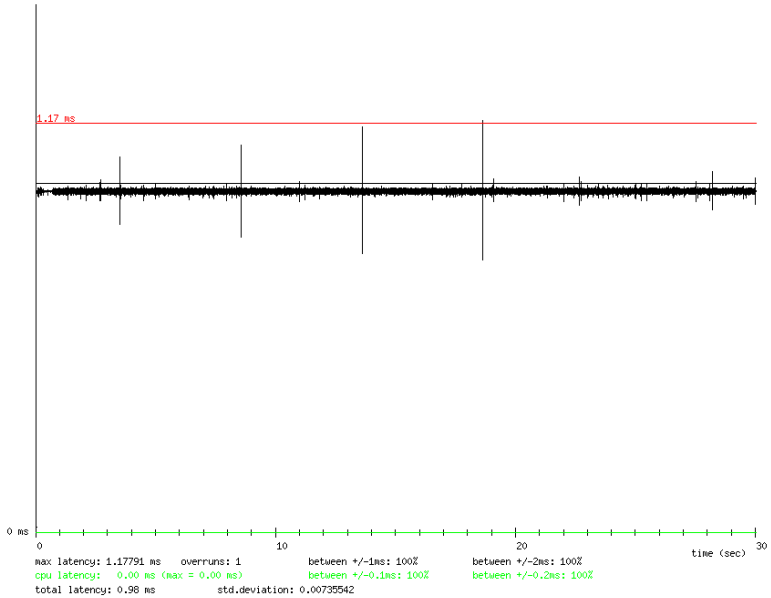


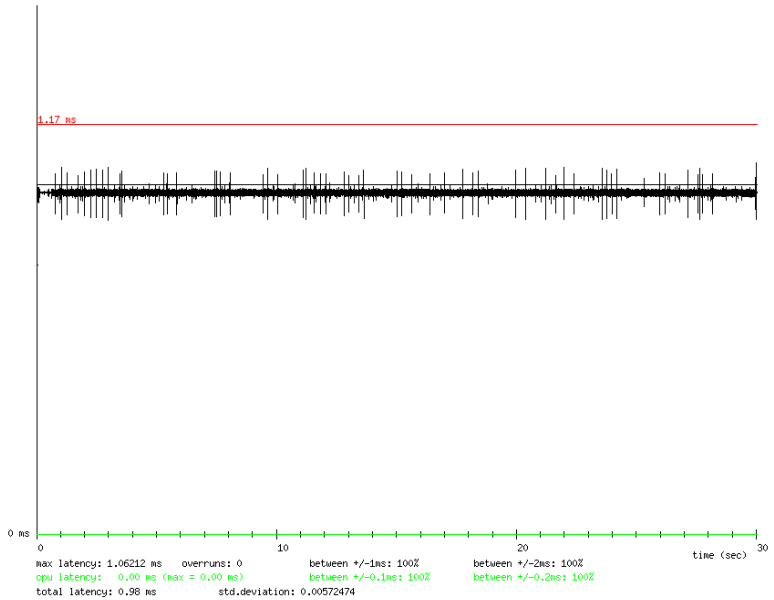


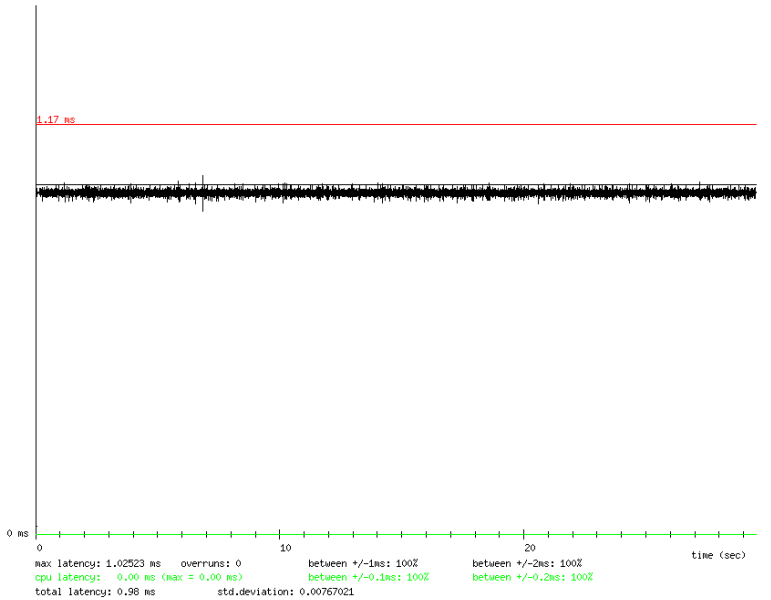


SMP/HT-Kernel (Procfs)

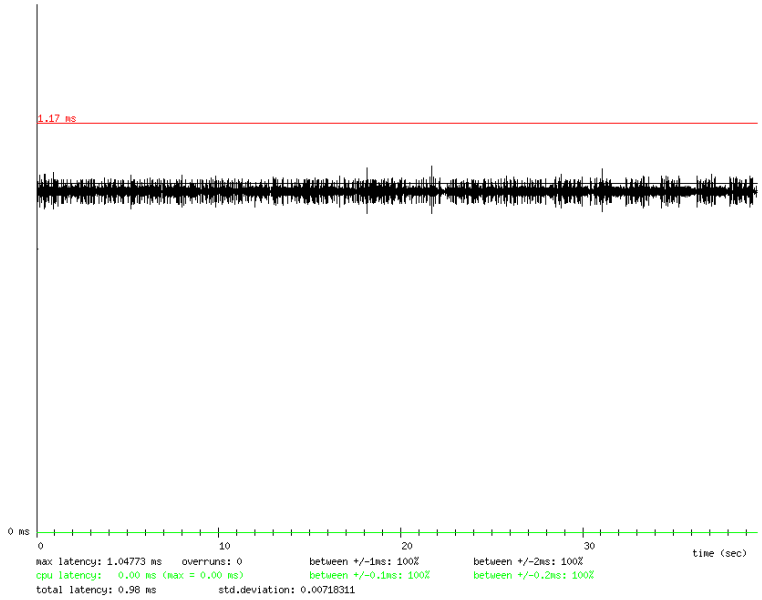


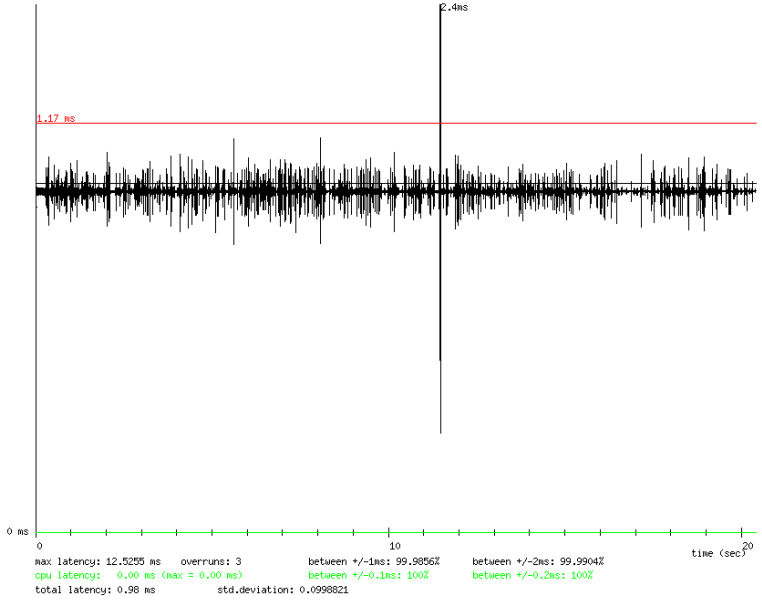


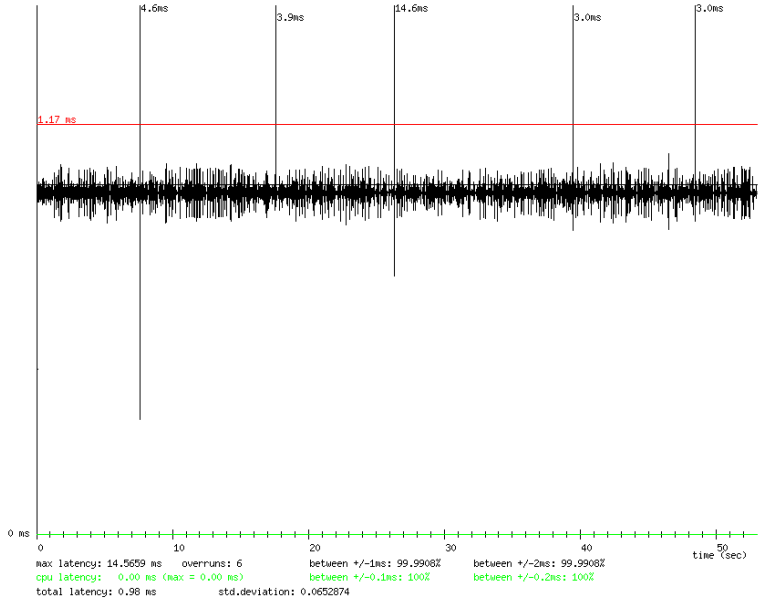


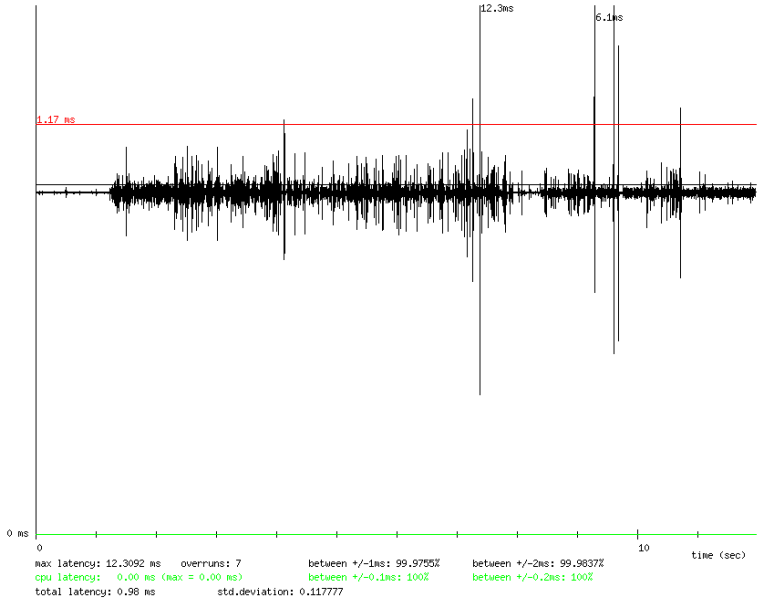


SMP/HT-Kernel (NOPE / Disk)

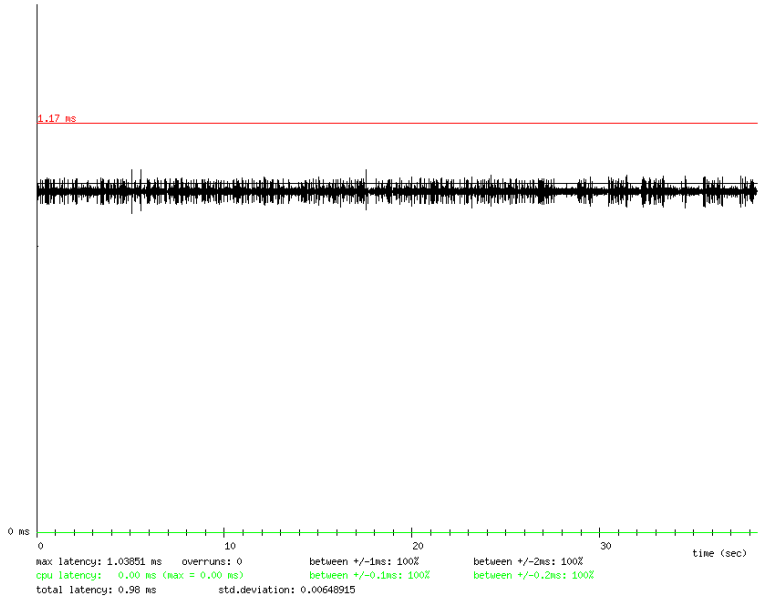


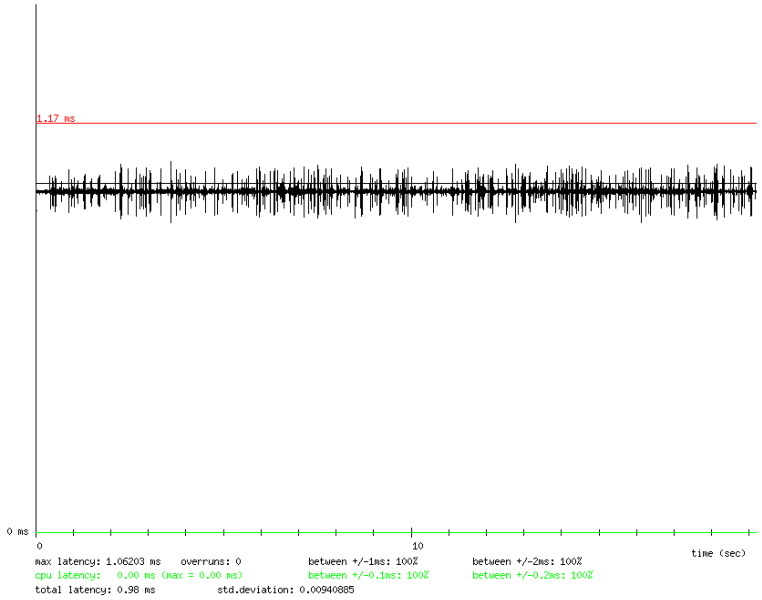


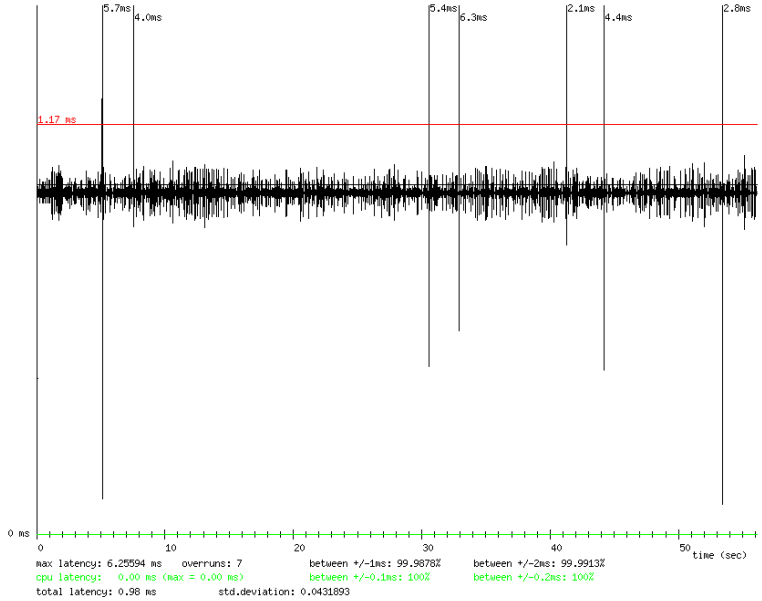


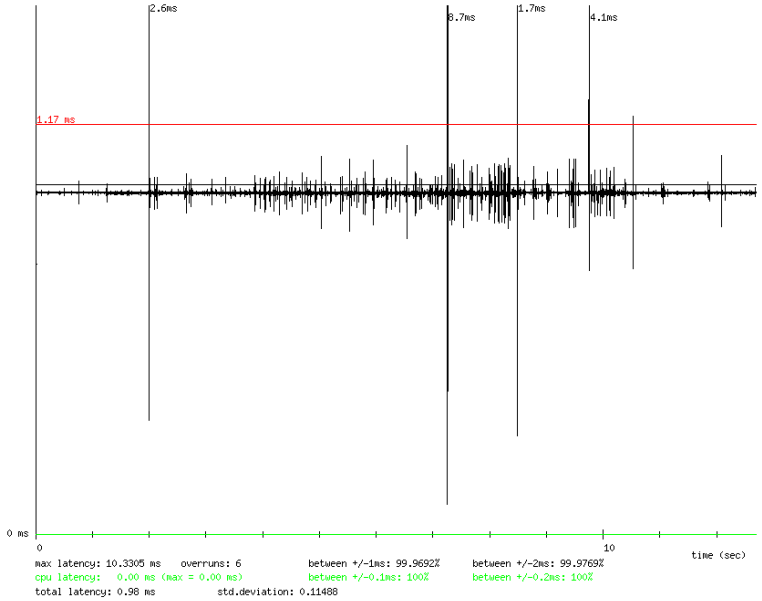


SMP/HT-Kernel (VP / Disk)

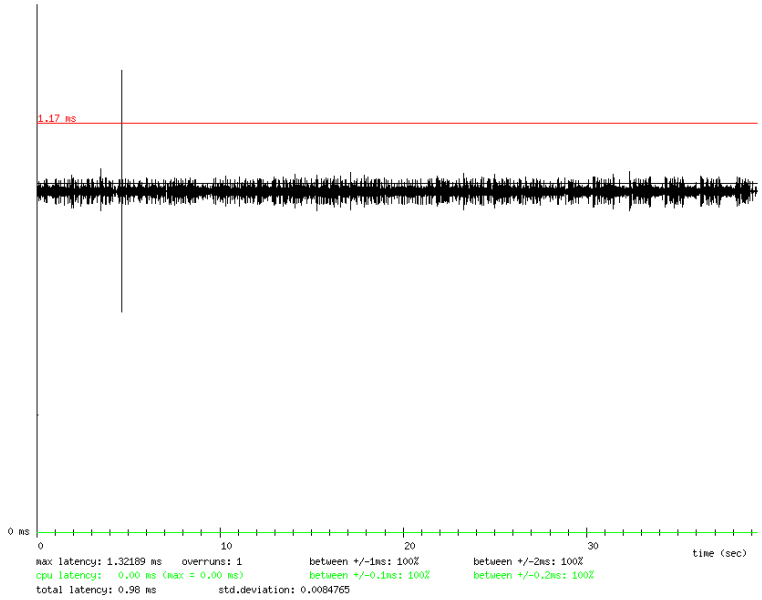


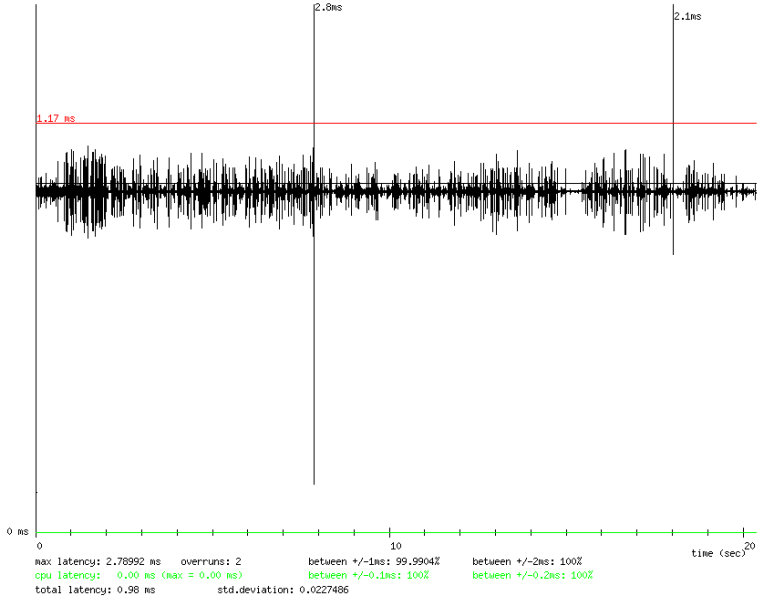


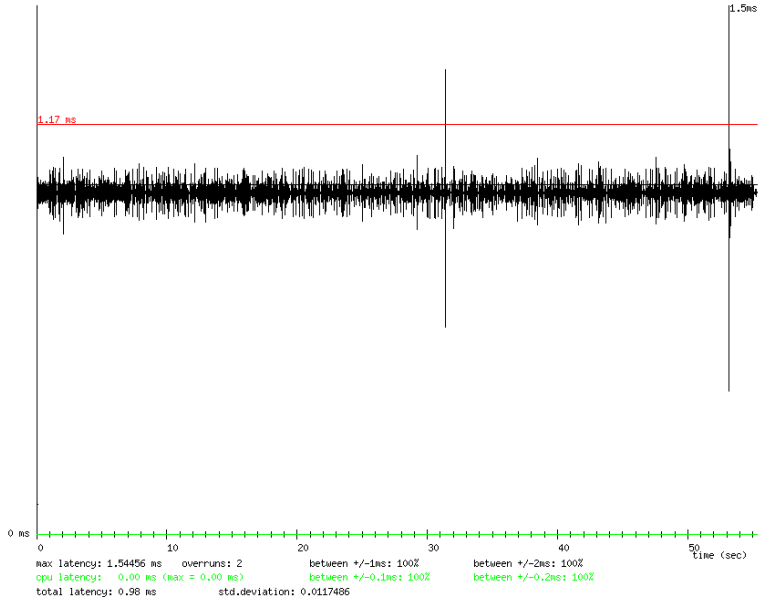


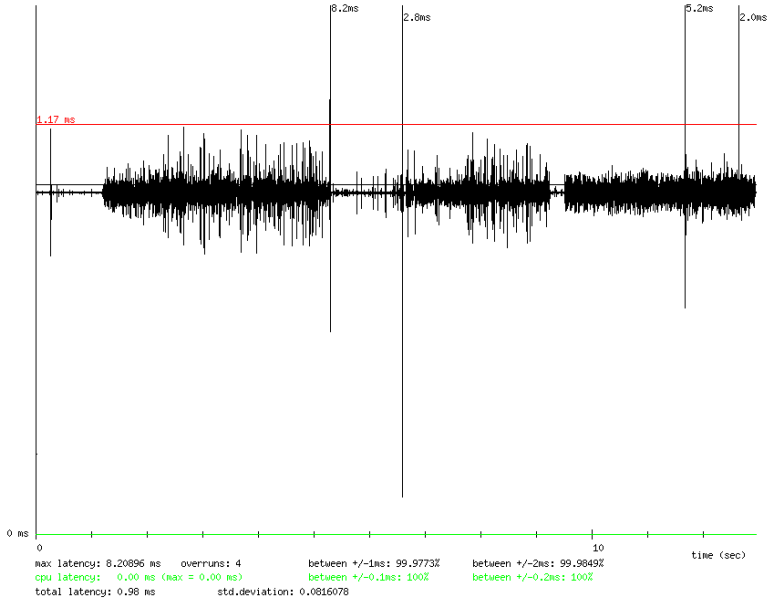


SMP/HT-Kernel (PE / Disk)

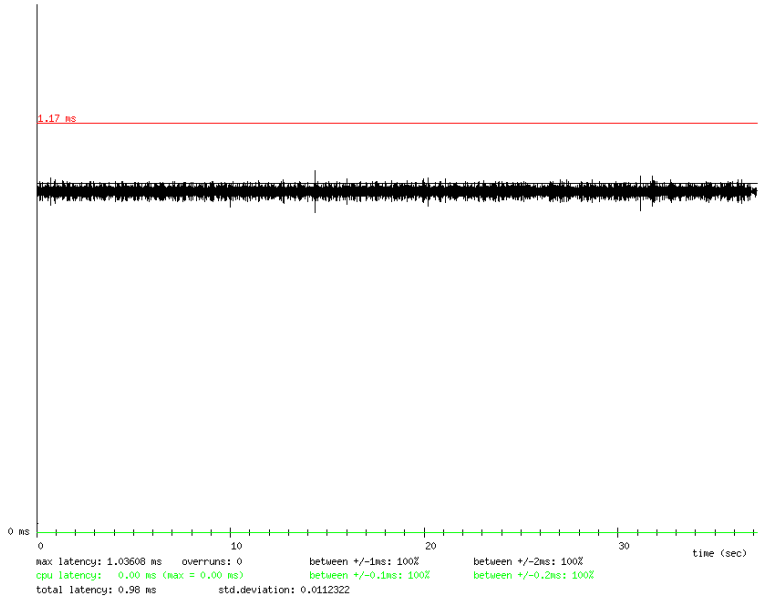


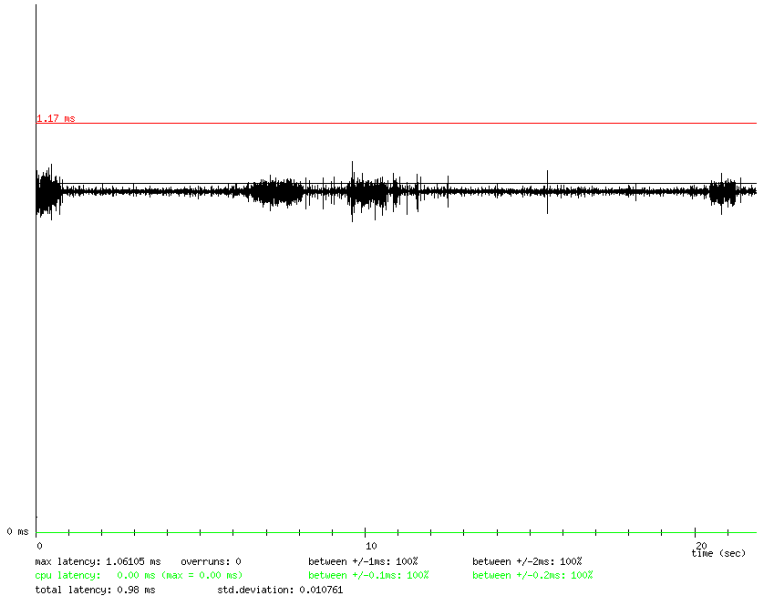


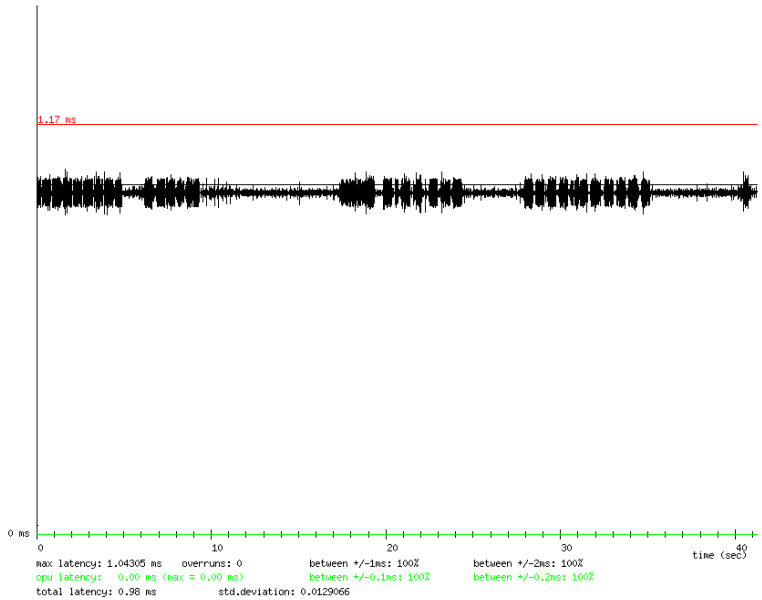


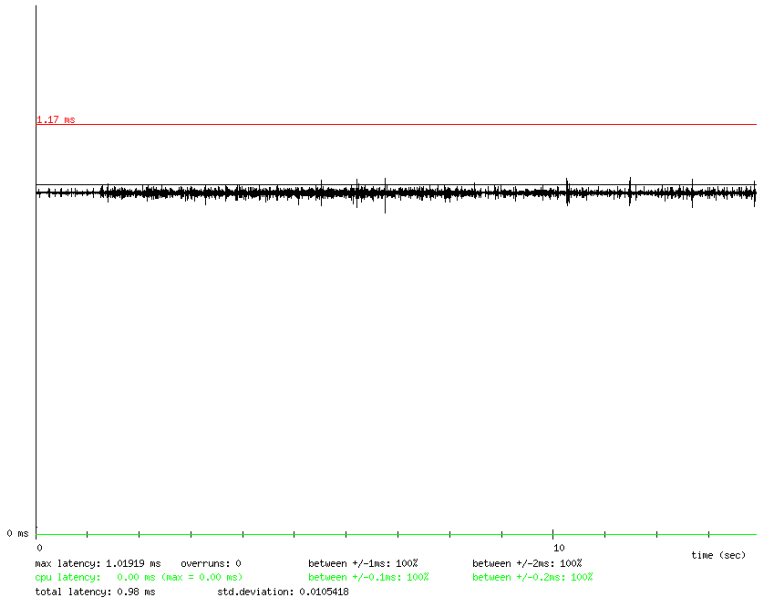


SMP/HT-Kernel (RT / Disk)



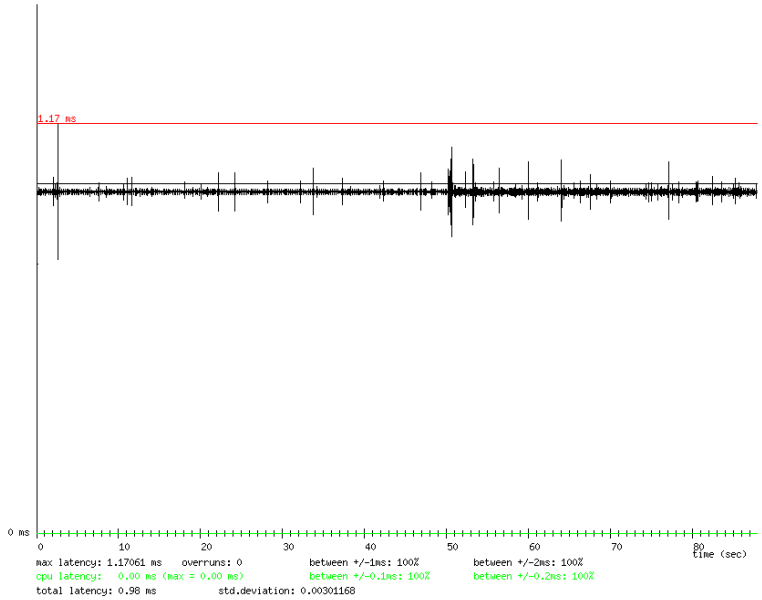


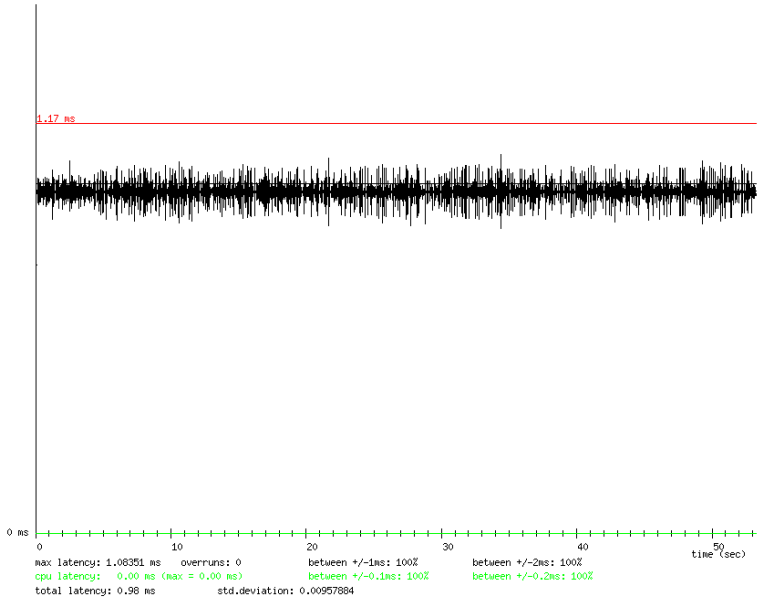




SMP/ +NO+ HT-Kernel (NOPE)

- Almost identical with UP kernel



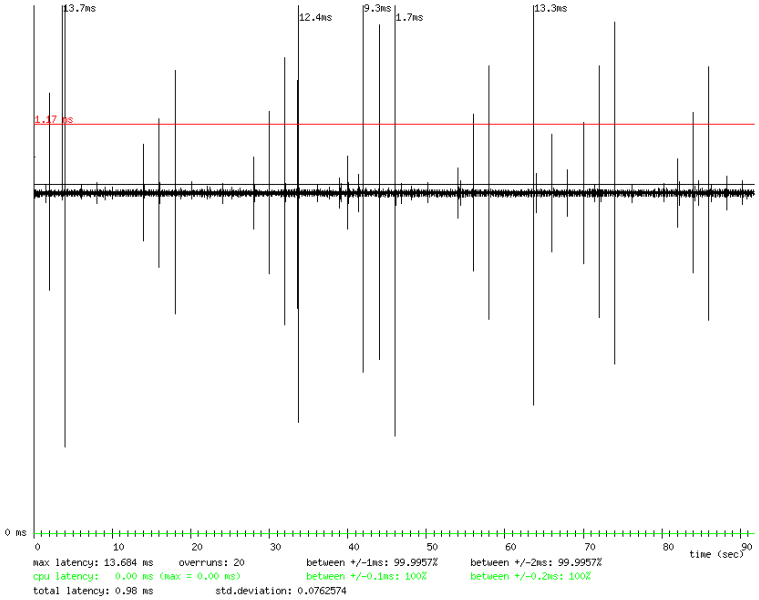


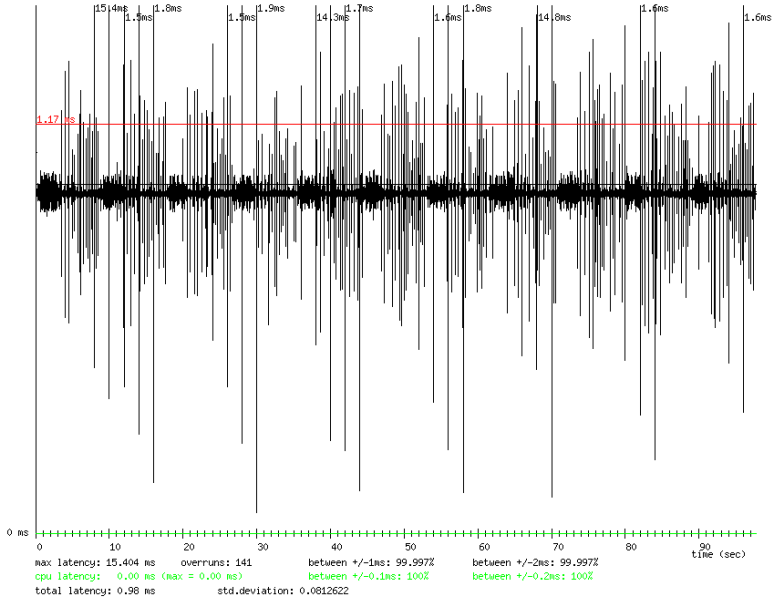
Remarks on Tests with SMP/HT Kernels

- Bad news:
  - SMP/HT kernels are much worse than UP
    - More than 10ms
  - Peaks found in
    - pdflush
    - softirqd with RCU
  - A problem around HT?
- Good news:
  - RT-kernel still shows ~100us latency
- However...

Running on A Laptop

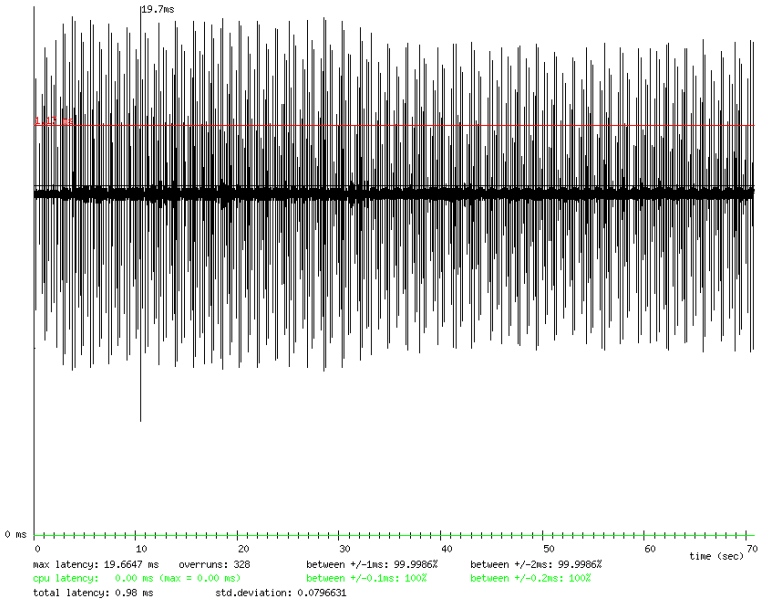
- A worse news: Terrible RT on my laptop!
- ACPI and other weird issues

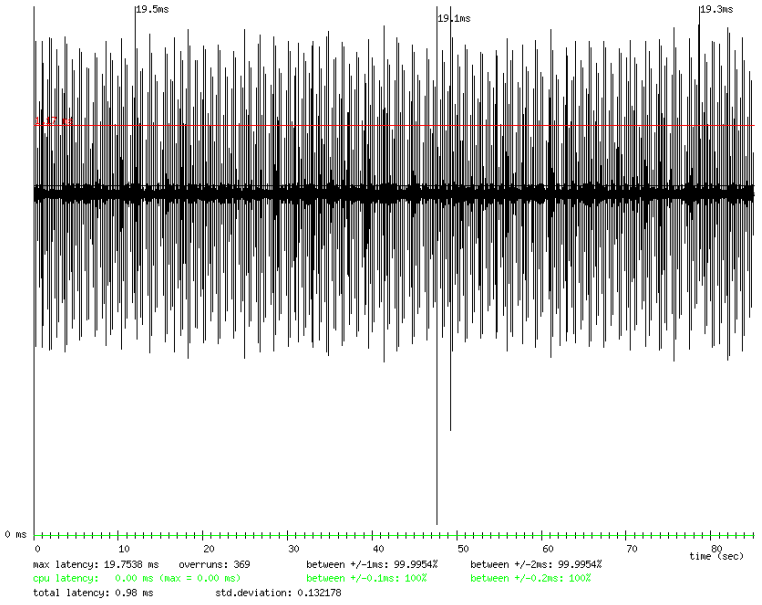




Is RT-Patch Perfect?

- Well, you might have a bad day:
  - depending on patch/kernel versions
  - depending on machines...





Performance Tests with LMBench

- LMBench
  - Measures various performances (latency, bandwidth)
  - Collection of small benchmark tests
    - Memory, Socket, Pipe, Disk I/O, etc.
- Useful to see the difference of kernels, too
- Let’s check it out
  - Running on P4 2.8GHz, 1GB machine
  - With and without IRQ workloads via `ping -f`
  - NOPE, PE, RT kernels
  - Average over 10 runs

LMBench - Elapsed Time

- No loads

UP		SMP/HT	
NOPE	3:08.57	NOPE	3:05.18
VP	3:09.01 (+0.1%)	VP	3:06.09 (+0.4%)
PE	3:09.18 (+0.3%)	PE	3:05.16 (-0.0%)
RT	3:08.37 (-0.1%)	RT	3:43.30 (+20.0%)

- With IRQ loads

UP		SMP/HT	
NOPE	4:59.83	NOPE	3:13.27
VP	4:58.30 (-0.6%)	VP	3:10.56 (-1.4%)
PE	4:59.94 (+0.0%)	PE	3:17.19 (+2.0%)
PE*	5:03.41 (+1.2%)	PE*	3:19.88 (+2.9%)
RT	5:10.28 (+3.4%)	RT	3:52.58 (+20.0%)

PE* = with

LMBench - Benchmarks (UP Kernel)

- Local Communication bandwidths in MB/s
- No loads

Kernel	Pipe	UNIX	TCP	File
NOPE-UP	2271	2776	1539	2225.4
PE-UP	2207	2711	1571	2228.5
RT-UP	2174	2769	1450	2192.2

- With IRQ loads

Host	Pipe	UNIX	TCP	File
NOPE-UP	987.	1195	629.	988.3
PE-UP	1006	1098	621.	985.3
RT-UP	896.	1113	566.	924.8

LMBench - Benchmarks (SMP/HT Kernel)

- Local Communication bandwidths in MB/s
- No loads

Kernel	Pipe	UNIX	TCP	File
NOPE-SMP	886.	2141	1125	2084.0
PE-SMP	900.	2069	1158	2061.9
RT-SMP	1116	1176	469.	1844.8

- With IRQ loads

Kernel	Pipe	UNIX	TCP	File
NOPE-SMP	863.	1896	790.	1639.1
PE-SMP	831.	1768	716.	1682.9
RT-SMP	975.	875.	295.	1521.4

Analysis via OProfile

- 50% spent in user copy during TCP, Unix tests

```
Counted GLOBAL_POWER_EVENTS events
samples  %        image name               symbol name
60218    29.3093  vmlinux-2.6.13-14-smp    __copy_from_user_ll
45478    22.1350  vmlinux-2.6.13-14-smp    __copy_to_user_ll
......
```

- Many cache misses found on PE
  - NOPE

```
Counted BSQ_CACHE_REFERENCE events
samples  %        image name               symbol name
246      42.7083  vmlinux-2.6.13-14-smp    __copy_from_user_ll
112      19.4444  vmlinux-2.6.13-14-smp    __copy_to_user_ll
```

  - PE

```
Counted BSQ_CACHE_REFERENCE events
samples  %        image name               symbol name
3161     57.8196  vmlinux                  __copy_to_user_ll
1651     30.1994  vmlinux                  __copy_from_user_ll
```

Analysis via OProfile (contd.)

- Spinlock mutex in RT kernel

```
Counted GLOBAL_POWER_EVENTS events
samples  %        image name               symbol name
6008     13.7213  vmlinux                  __down_mutex
5888     13.4472  vmlinux                  __copy_from_user_ll
5716     13.0544  vmlinux                  __copy_to_user_ll
3285      7.5024  vmlinux                  __schedule
2107      4.8120  vmlinux                  _raw_spin_lock
......
```

Remarks on LMBench Tests

- Elapsed Time
  - No big difference in all UP kernels without load
    - Don’t set CONFIG_DEBUG_PREEMPT!
  - Bad performance with RT-kernel under IRQ load
  - Much worse performance on RT-SMP/HT kernel
- Benchmarks
  - Differences TCP/Unix-Socket
    - Related with cache misses?
    - Might be improved with the recent patch
  - Pipe is faster on RT
  - Big reduction of TCP/Unix on RT
    - Due to mutexized spinlock
  - PE and RT kernels are obviously not for servers

Interactivity

- Interactivity = Latency of I/O processes
  - Examples
    - Mouse reacts quickly without jerks
    - Sound playback doesn’t skip
    - Play video without drop outs
  - Interactivity != Performance speed
- A typical job: IRQ scheduling
  - Mouse IRQ -> X server -> Redraw a cursor
  - Audio IRQ -> Audio app -> Fill audio data
- A known workaround
  - Renice the process (e.g. X)
  - Scheduler should handle different processes correctly
    - ... but difficult to predict the behavior of each process

Interactivity Tests

- Interbench
  - by Con Kolivas
  - Performs latencies on different loads
    - Audio: latency of 50ms intervals, 5% CPU
    - Gaming: Hog CPU, and measure frames per second
- Latencytest
  - Running without RT-scheduling

Interbench Results: Audio simulation

NOPE / UP

Load	Latency +/- SD (ms)	Max Latency	% Desired CPU	% Deadlines Met
None	0.014 +/- 0.0196	0.152	100	100
Video	1.48 +/- 2.64	7.11	100	100
X	2.44 +/- 4.31	12	100	100
Burn	0.014 +/- 0.0237	0.165	100	100
Write	0.036 +/- 0.12	1.43	100	100
Read	0.021 +/- 0.0275	0.201	100	100
Compile	0.058 +/- 0.205	2.5	100	100
Memload	0.031 +/- 0.0795	0.749	100	100

RT / UP

None	0.012 +/- 0.0129	0.038	100	100
Video	0.014 +/- 0.024	0.179	100	100
X	2.57 +/- 5.14	19.5	100	100
Burn	0.014 +/- 0.0219	0.171	100	100
Write	0.064 +/- 0.502	7.02	100	100
Read	0.023 +/- 0.0313	0.26	100	100
Compile	0.041 +/- 0.147	1.57	100	100
Memload	0.05 +/- 0.151	1.22	100	100

Interbench Results: Gaming simulation

NOPE / UP

Load	Latency +/- SD (ms)	Max Latency	% Desired CPU
None	1.03 +/- 2.48	11.6	99
Video	68.5 +/- 69.7	81.5	59.4
X	105 +/- 201	912	48.8
Burn	198 +/- 314	877	33.6
Write	46.5 +/- 111	802	68.2
Read	5.67 +/- 6.18	8.91	94.6
Compile	240 +/- 367	575	29.5
Memload	4.66 +/- 8.41	45.6	95.5

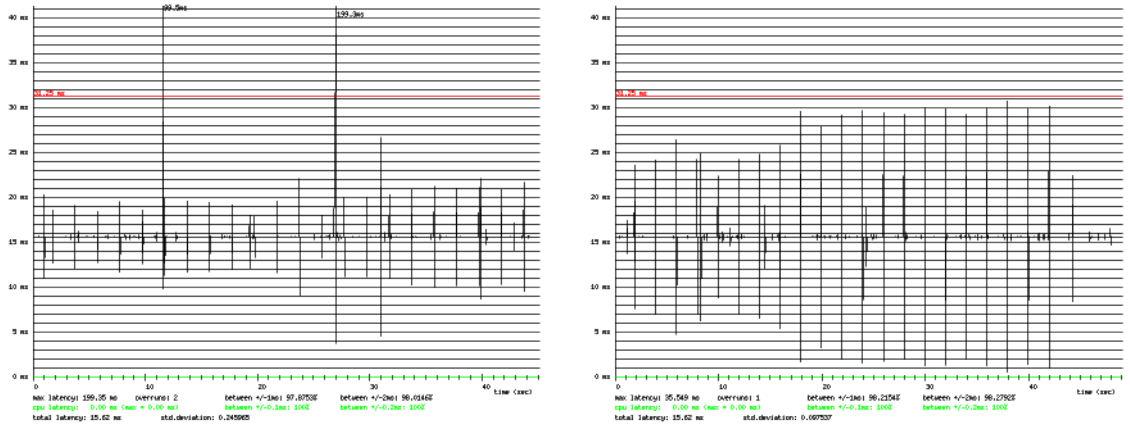
RT / UP

None	0 +/- 0	0	100
Video	43.3 +/- 43.8	47.8	69.8
X	65.9 +/- 124	520	60.3
Burn	184 +/- 268	392	35.2
Write	22.6 +/- 102	727	81.6
Read	0.051 +/- 0.211	1.41	99.9
Compile	218 +/- 332	786	31.5
Memload	2.4 +/- 6.04	38.3	97.7

Thanks, Andrea

- per-task write-throttle patch
  - seems to reduce highest peeks

Left: Default / Right: With per-task-write patch



Remarks on Interactivity Tests

- Good interactivity in general with all tests
- Audio latency tests
  - No significant difference between normal and RT kernels
- Gaming simulations
  - Slightly better results on RT
- Stall by disk access
  - Seems to be the cause of occasional high peaks

Conclusions and TODO

- RT on Linux
  - Works nicely in most cases, even without patches
  - SMP/HT has small problems
  - Glitches on laptops
- RT-Kernel
  - Shows supreme RT latency (~100us)
  - Slightly improves the average interactivity
    - Still high peaks in the same level
  - Much less thruputs than other kernels
- TODO
  - Easier RT-testing
    - Running on virtualized environment?
  - More precise analyses on peaks
  - Comparison with nano-kernel approach
  - More realistic interactivity tests