Real-Time Failure

Real-time unix has been used successfully since at least the late 1980's in many diverse areas, including audio, video, manufacturing, finance, test and measurement, and military applications. Linux support for real-time has been actively developed and maintained in the community since 2004, is included in several commercial distributions, and is partially in the kernel.org tree, with features from the out of tree patch set continuing to flow into the kernel.org tree. Despite its relative youth, real-time Linux is very capable, but as with the other real-time unix kernels there are many ways to fail when attempting to create a real-time Linux solution. This presentation probes some causes of failure that can be avoided.

Frank Rowand, Sony Corporation of America

April 12, 2009

Real-Time Failure

Real-time Linux is very capable, but there are many ways to fail when attempting to create a real-time Linux solution.

Real-Time Failure

Real-time Linux is very capable, but there are many ways to fail when attempting to create a real-time Linux solution.

This presentation looks at some causes of failure that can be avoided.

Real-Time Failure

Real-time Linux is very capable, but there are many ways to fail when attempting to create a real-time Linux solution.

This presentation looks at some causes of failure that can be avoided.

The primary focus is Linux. But some examples will not be Linux specific.

Caveats

- There are many ways to cause failure. This talk only mentions a few of them.

- The "facts" presented are likely to be strongly dependent on the kernel version. This information is mostly based on 2.6.23 – 2.6.30.

section 1

Definitions and Concepts

What is Real Time?

It is determinism (being able to respond to a stimulus before a deadline) with a given load.

What is Real Time?

It is determinism (being able to respond to a stimulus before a deadline) with a given load.

It is NOT fast response time.

What is Real Time?

It is determinism (being able to respond to a stimulus before a deadline) with a given load.

It is NOT fast response time.

The specific real time application deadlines determine how short the maximum response time must be to deliver real time behavior.

Some examples of deadlines are one second, one millisecond, or five microseconds.

What is Real Time?

It is NOT fast response time.

But in MY world -- embedded consumer electronics -- the processors are as slow as possible, to reduce the cost of the product and to minimize power consumption.

What is Real Time?

It is NOT fast response time.

But in MY world -- embedded consumer electronics -- the processors are as slow as possible, to reduce the cost of the product and to minimize power consumption.

Thus achieving fast enough response time is a challenge.

What is Real Time?

It is NOT fast response time.

So a common strategy to avoid failure of real time products is to focus on decreasing response time (by reducing overhead and latency).

What is Real Time Linux?

For this talk:

kernel.org Linux + RT preempt patches

It is not:

Xenomai
RTAI
Adeos

These are interesting, but not enough time to discuss them.

OS Design Trade offs

Batch

- maximize throughput
- sacrifice responsiveness

OS Design Trade offs

Batch

- maximize throughput
- sacrifice responsiveness

OLTP

- maximize transactions per second
- minimize average response time
- sacrifice determinism

OS Design Trade offs

Batch

- maximize throughput
- sacrifice responsiveness, determinism

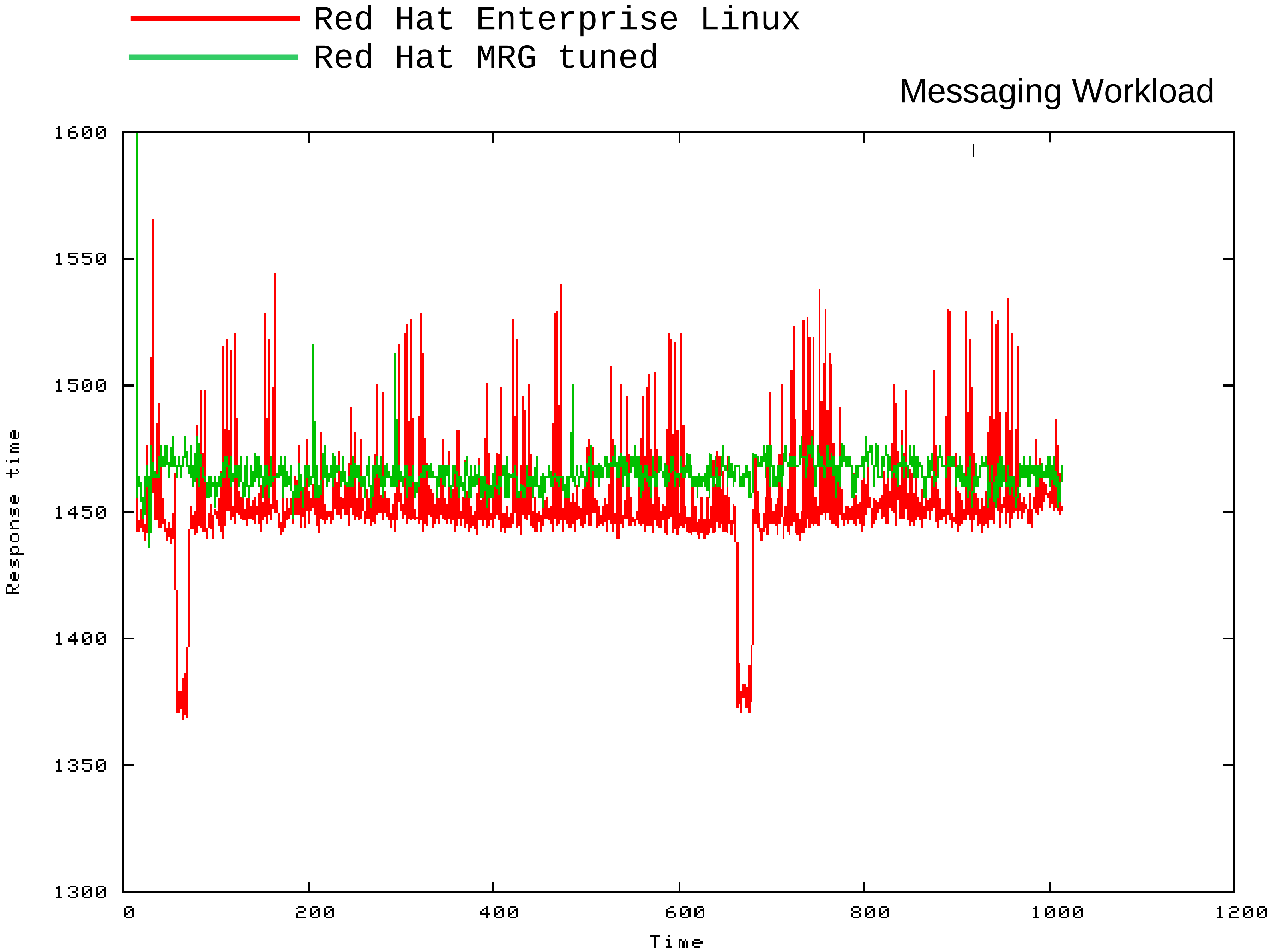
OLTP

- maximize transactions per second
- minimize average response time
- sacrifice throughput, determinism

Real Time

- maximize determinism
- minimize worst case latency
- sacrifice throughput, average response time, minimum latency

Messaging Workload



source: Red Hat

OS Design Trade offs

Contradictory attributes can not be achieved.

Unrealistic expectations will lead to failure.

section 2

Process

Related Issues

Classic Method of RT Design

Design to be RT

Classic Method of RT Design

Design to be RT

Resource budget (eg cpu scheduling) strictly allocated

Classic Method of RT Design

Design to be RT

Resource budget (eg cpu scheduling) strictly allocated

Precise analysis and review of algorithm and code

Classic Method of RT Design

In theory, it is “easy” to follow a process or formula to guarantee that the RT system will meet the design goals.

Ad hoc Method of RT Design

Modify existing application to become RT

- Use RT priorities
- Preallocate resources and lock in place
- Remove obvious blocking and contention

Ad hoc Method of RT Design

Modify existing application to become RT
- Use RT priorities
- Preallocate resources and lock in place
- Remove obvious blocking and contention

Iterative process to fix problems
- Detect existence of problem
- Instrument and measure
- Debug cause
- Fix cause
- Repeat

Ad hoc Method of RT Design

There is not a process or formula to guarantee that the RT system will meet the design goals.

Classic Method of RT Design

Add things that create solutions.

Classic Method of RT Design

Add things that create solutions.

Ad hoc Method of RT Design

Remove things that cause problems.

Classic Method of RT Design

Add things that create solutions.

Ad hoc Method of RT Design

Remove things that cause problems.

It is much harder to prove that nothing bad remains to remove, than to prove that you have only added good.

section 3

HARDWARE

Related Issues

Hardware

Clock Speeds

Most of the mainline development seems to be focused on systems with high clock rates (>= 1 Ghz).

Hardware

Clock Speeds

Most of the mainline development seems to be focused on systems with high clock rates (>= 1 Ghz).

If your target hardware has low clock rates (eg 100 – 500 Mhz), you may need to modify the mainline kernel to achieve acceptable latencies. For example, the scheduler.

Hardware

Clock Speeds

Most of the mainline development seems to be focused on systems with high clock rates (>= 1 Ghz).

If your target hardware has low clock rates (eg 100 – 500 Mhz), you may need to modify the mainline kernel to achieve acceptable latencies. For example, the scheduler.

But real-time is not fast, it is determinism.

Non-deterministic Hardware

Memory Cache

TLB

Memory Bus Contention

BIOS with SMI handlers enabled

Input / Output

External Interrupt Prioritization

SMP

Virtualization

Non-deterministic Hardware

Memory Cache & TLB

These technologies have been present in successful RT systems for decades.

Non-deterministic Hardware

Memory Cache & TLB

These technologies have been present in successful RT systems for decades.

- If the statistical behavior of the system is good enough.

Non-deterministic Hardware

Memory Cache & TLB

These technologies have been present in successful RT systems for decades.

- If the hardware can be made deterministic for the real time application.

Non-deterministic Hardware

Memory Cache & TLB

These technologies have been present in successful RT systems for decades.

- If the hardware can be made deterministic for the real time application:

  + Lock application in TLB and cache
  + Dedicated high speed memory system
  + uclinux (for systems without an MMU)

Non-deterministic Hardware

Memory Cache & TLB

These technologies have been present in successful RT systems for decades.

- If RT application must be locked in TLB or cache then vanilla RT Linux is not the solution.

Non-deterministic Hardware

Memory Cache & TLB

These technologies have been present in successful RT systems for decades.

- If RT application must be locked in TLB or cache then vanilla RT Linux is not the solution.

- It could be possible to modify the kernel to provide these features (architecture specific).

Non-deterministic Hardware

BIOS with SMI handlers enabled

Steals the CPU from the Linux kernel

SMI == System Management Interrupt

Non-deterministic Hardware

BIOS with SMI handlers enabled

Steals the CPU from the Linux kernel

Examples of SMI activities:

- thermal management
- memory errors
- legacy ISA devices
- USB (ps2 emulation)

Non-deterministic Hardware

BIOS with SMI handlers enabled

- How to detect:
  - lkml thread: [RT] [RFC] simple SMI detector
    Jon Masters
    1/24/09 – 1/27/09

Non-deterministic Hardware

- BIOS with SMI handlers enabled
  - How to detect:
    - lkml: [PATCH 0/1] Hardware Latency Detector
      (formerly SMI detector)
      Jon Masters
      Thu, 11 Jun 2009 00:58:29 -0400
    - Not yet in kernel.org as of 2.6.34-rc3

Non-deterministic Hardware

BIOS with SMI handlers enabled

How to detect:

lkml: [PATCH 2.6.34-rc3] A nonintrusive SMI sniffer for x86 (resend)
Joe Korty
Tue, 6 Apr 2010 13:06:05 -0700

Non-deterministic Hardware

BIOS with SMI handlers enabled

Possible Fix:

- Do not use the hardware that requires the SMI handlers (eg USB ps2 emulation)

Non-deterministic Hardware

BIOS with SMI handlers enabled

Possible Fix:

- Do not use the hardware that requires the SMI handlers (eg USB ps2 emulation)
- Use a system that does not have BIOS with SMI handlers.

Non-deterministic Hardware

BIOS with SMI handlers enabled

Possible Fix:

- Work with BIOS and system vendors to replace SMI handlers in BIOS with custom kernel or user space equivalent.

Non-deterministic Hardware

BIOS with SMI handlers enabled

Possible Fix:

- Work with BIOS and system vendors to replace SMI handlers in BIOS with custom kernel or user space equivalent.

  Example:

  How can I improve event response times (latency) for my realtime kernel on Intel-based HP ProLiant G6 systems?

  http://kbase.redhat.com/faq/docs/DOC-19297

Non-deterministic Hardware

BIOS with SMI handlers enabled

Example Fix:

lkml: [RFC][Patch] IBM Real-Time
"SMI Free" mode driver
Keith Mannthey
02/10/09 16:37

Not yet accepted as of 2.6.34-rc3

Non-deterministic Hardware

BIOS with SMI handlers enabled

Example Fix:

lkml: [RFC][Patch] IBM Real-Time
"SMI Free" mode driver

“This driver supports the Real-Time Linux (RTL) BIOS feature. The RTL feature allows non-fatal System Management Interrupts (SMIs) to be disabled on supported IBM platforms”

Non-deterministic Hardware

BIOS with SMI handlers enabled

Example Fix:

http://linuxplumbersconf.org/2009/slides/Keith-Mannthey-SMI-plumers-2009.pdf

Current Support
- various IBM systems
- Redhat MRG
- SUSE SLERT

Non-deterministic Hardware

Input / Output

For example:

- Networking
- USB
- Video
- sdhci Secure Digital Host Controller Interface
- i2c
- media (disk, flash device)

Input / Output

Drivers may be non-deterministic,

or may just create large latencies.

Input / Output

Assume that all drivers are

NOT

real time safe

Input / Output

Assume that all drivers are

NOT

real time safe

Until you have verified otherwise

Input / Output

Assume that all drivers are

NOT

real time safe

Until you have verified otherwise

Most drivers are not created with a real time goal

USB

Linux example: USB2Serial

lkml: “Real time USB2Serial devices and behaivor”
Mark Gross
2008-03-26 15:25:59 GMT

“I'm just starting to look into the behavior now but has anyone looked at the RT'ness of USB2Serial + USB stack yet?”

USB

Linux example: USB2Serial

“USB is not 'deterministic', and these cheap USB to serial devices introduce a very big lag that also is not deterministic.”

“The generic usb serial driver is KNOWN TO BE A VERY SLOW DRIVER!
...
The code was not designed to be fast, only get the job done.”

USB

Linux example: USB2Serial

“I'd think that in a controlled environment (fixed set of USB connections) USB should be able to meet fairly chosen "real time" latency ceilings.

The stack probably needs a few semantic updates to make it happen -- e.g. URB Completions are issued in_irq() -- but it shouldn't be insurmountable.”

USB

From the USB 2.0 and 3.0 specifications for an Interrupt Transfer:

- The host controller polls for “interrupts”
- The minimum poll period is 125 μs
- If an error is detected the transfer is attempted one period later

USB

From the USB 2.0 and 3.0 specifications for an Interrupt Transfer:

- The host controller polls for “interrupts”
- The minimum poll period is 125 µs
- If an error is detected the transfer is attempted one period later

==> Hardware latency could be 125 µs (no error)
Hardware latency could be 250 µs (one error)
etc...

USB

From the USB 2.0 and 3.0 specifications for an Interrupt Transfer:

- The host controller polls for “interrupts”
- The minimum poll period is 125 μs
- If an error is detected the transfer is attempted one period later

But real-time is not fast, it is determinism.

So, is USB fast or deterministic?

Video

“VGA text console causes very large latencies, up to more than hundreds of microseconds.”

source:
http://rt.wiki.kernel.org/index.php/HOWTO:_Build_an_RT-application

Video

“VGA text console causes very large latencies, up to more than hundreds of microseconds.”

source: http://rt.wiki.kernel.org/index.php/HOWTO:_Build_an_RT-application

This is a good example of how a driver can impact a real time task, even if the real time task is not directly using the driver.

sdhci

Secure Digital Host Controller Interface

lkml: sdhci can turn off irq up to 200 ms
Matthieu CASTET
Wed, 1 Jul 2009 15:15:48 +0200

Non-deterministic Hardware

Input / Output

Possible Fix:

Defer I/O to a non-realtime thread

Non-deterministic Hardware

External Interrupt is highest priority

All external interrupts have better priority than all real time processes.

Non-deterministic Hardware

External Interrupt is highest priority

- All external interrupts have better priority than all real time processes.
- Uncontrolled external events are capable of preempting real time processes for an infinite length of time.

Non-deterministic Hardware

External Interrupt is highest priority

Possible Fix:

- Control the external environment
- Implement polled event handling (eg NAPI) for problem drivers
- Mask problem interrupts while RT processes are runnable (theoretical, not implemented)

Non-deterministic Hardware

SMP

Current state in real time Linux:

- Not brand new, but still room for increased experience and improvement
- Not yet predominate platform for real time, but increasingly common

Non-deterministic Hardware

SMP

Current state in real time Linux:

- Marketed by commercial vendors, examples:

  MontaVista Software
  http://www.mvista.com/product_detail_cge.php

  Red Hat Enterprise MRG
  http://www.redhat.com/mrg/

  SUSE Linux Enterprise Real Time
  http://www.novell.com/industries/financial/realtime/

Non-deterministic Hardware

SMP

- Mainstream developers are aware of SMP.
- The real time scheduler supports SMP.

Non-deterministic Hardware

SMP

Linux SMP scheduler research exists,
for example “ARTiS, Asymetric Real-Time Scheduling”:

http://www.lifl.fr/west/artis
https://gna.org/projects/artis

Non-deterministic Hardware

- SMP
  - My opinion:
    - Examples of SMP Linux real time are available, but its relative youth suggests that it should be approached with caution.

Non-deterministic Hardware

SMP

Example area of concern

Even if a cpu is dedicated to a real time process, activity on other cpus can impact it.

For instance, for_each_cpu(,,wait) impacts both sender and receiver cpu

Non-deterministic Hardware

SMP

```
int on_each_cpu(,, int wait)
{
        preempt_disable();
        ret = smp_call_function(func, info, wait);
        local_irq_save(flags);
        func(info);
        local_irq_restore(flags);
        preempt_enable();
```

Non-deterministic Hardware

SMP

```
void smp_call_function_many(,,, bool wait)
{

    /* Send a message to all CPUs in the map */
    arch_send_call_function_ipi_mask(...);

    /* Optionally wait for the CPUs to complete */
    if (wait)
        csd_lock_wait(&data->csd);
```

Non-deterministic Hardware

SMP

- Example area of concern
  - The existing scheduler algorithms might be adequate, but do not be surprised if your real time workload is not handled well by default.

Non-deterministic Hardware

SMP

Possible Fix:

- Adjust scheduler tunables.

Non-deterministic Hardware

SMP

Possible Fix:

- Help improve the mainline and RT preempt scheduler (test, report problems, implement fixes).

Non-deterministic Hardware

SMP

Possible Fix:

- Reduce the scheduler overhead
  - + pin processes to cpu (or other workload partitioning)
  - + simplify the scheduler to remove overhead and latency

Non-deterministic Hardware

SMP

Possible Fix:

- Isolate cpu to reduce or eliminate impact from other cpus

Non-deterministic Hardware

SMP

Possible Fix:

- Isolate cpu to reduce or eliminate impact from other cpus

  One example, that led to a long discussion on improving the current scheduler:

  linux-rt-users: “RFC: THE OFFLINE SCHEDULER”
  raz ben yehuda
  Sun, 23 Aug 2009 02:27:51 +0300

Non-deterministic Hardware

Virtualization

Virtualization

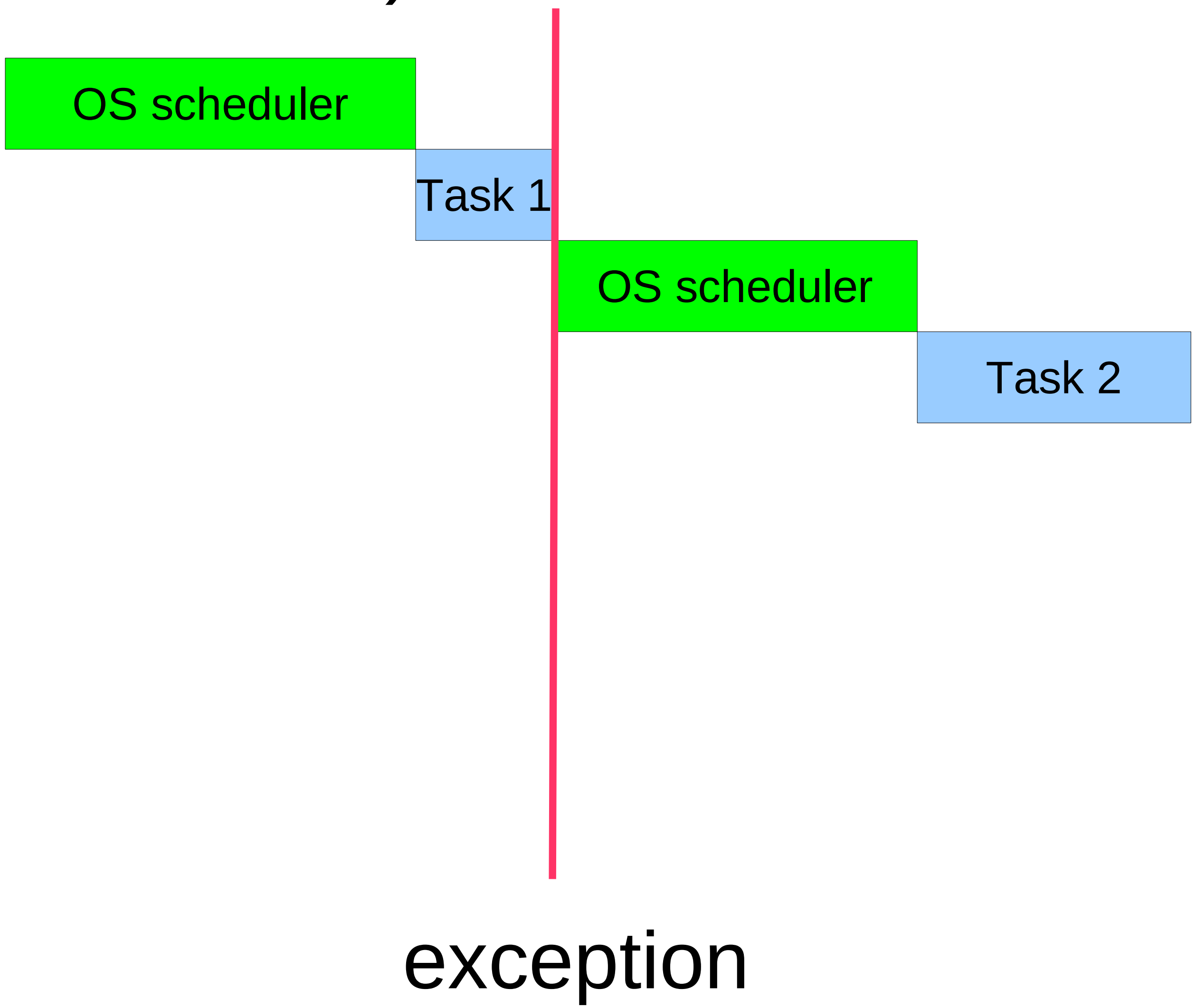
Guest Operating System executes in a "Virtual Machine".

Virtualization

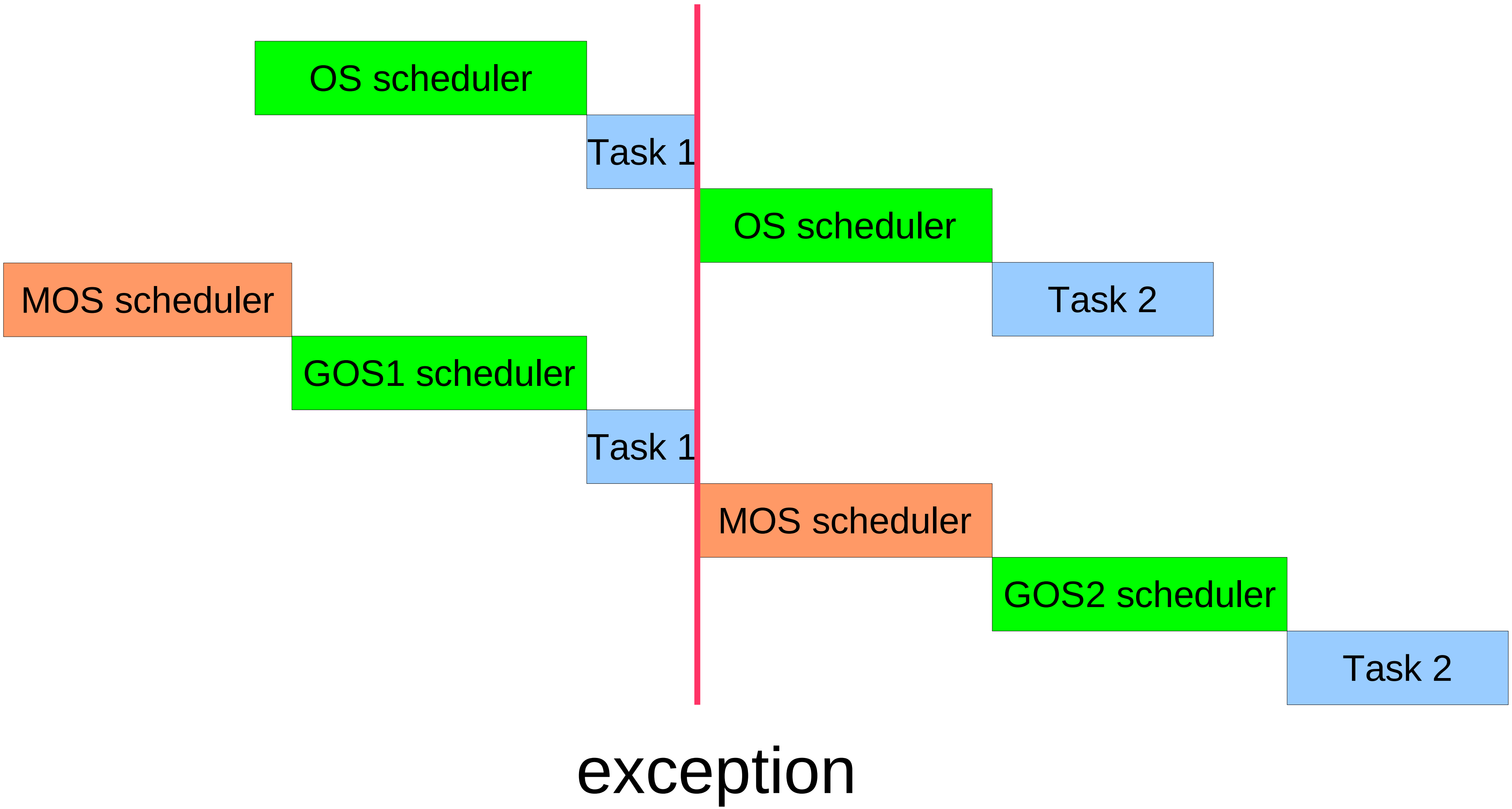
Example Issue 1

Additional overhead of meta operating system (eg hypervisor) mediating between guest operating systems (GOS).

Example of scheduler overhead
(no virtualization)



Example of scheduler overhead
(with virtualization)



Virtualization

Example Issue 1

Additional overhead of hypervisor

In this specific example:

- second scheduling layer
- additional context switches between hypervisor and guest OS's

Virtualization

Example Issue 1

Additional overhead of hypervisor

But real-time is not fast, it is determinism.

Virtualization

Example Issue 1

Additional overhead of hypervisor

But real-time is not fast, it is determinism.

So if the deadlines are met, the extra overhead is not a problem.

Virtualization

Example Issue 2

Guest Operating System can not provide resource guarantees to its real-time tasks, unless the Guest Operating System is given resource guarantees by the meta operating system.

Virtualization
and
Real Time

Frank's viewpoint:

On this path lies insanity.

Virtualization
and
Real Time

Frank's viewpoint:

On this path lies insanity.

But there are people who are braver than Frank.

Braver Than Frank, 1

lkml: [ANNOUNCE] AlacrityVM hypervisor project
Gregory Haskins
Mon, 03 Aug 2009 09:53:40 -0400

We are pleased to announce the formation of the AlacrityVM project and the availability of v0.1 of the code. AlacrityVM is a hypervisor based on KVM targeted specifically at performance sensitive workloads such as HPC and real-time.

You can find more information on the AlacrityVM wiki, available here:

http://developer.novell.com/wiki/index.php/AlacrityVM

Anyone who may be interested in further developments surrounding this project is encouraged to subscribe to one or both of the following lists:

https://lists.sourceforge.net/lists/listinfo/alacrityvm-users
https://lists.sourceforge.net/lists/listinfo/alacrityvm-devel

Braver Than Frank, 1

http://lwn.net/Articles/345296/

“... virtualization ... tends to suffer from performance problems, particularly I/O performance.”

“By shortening the I/O path for guests, AlacrityVM seeks to provide I/O performance near that of 'bare metal' hardware.”

(highly edited – please see original source)

Braver Than Frank, 1

http://developer.novell.com/wiki/index.php/AlacrityVM

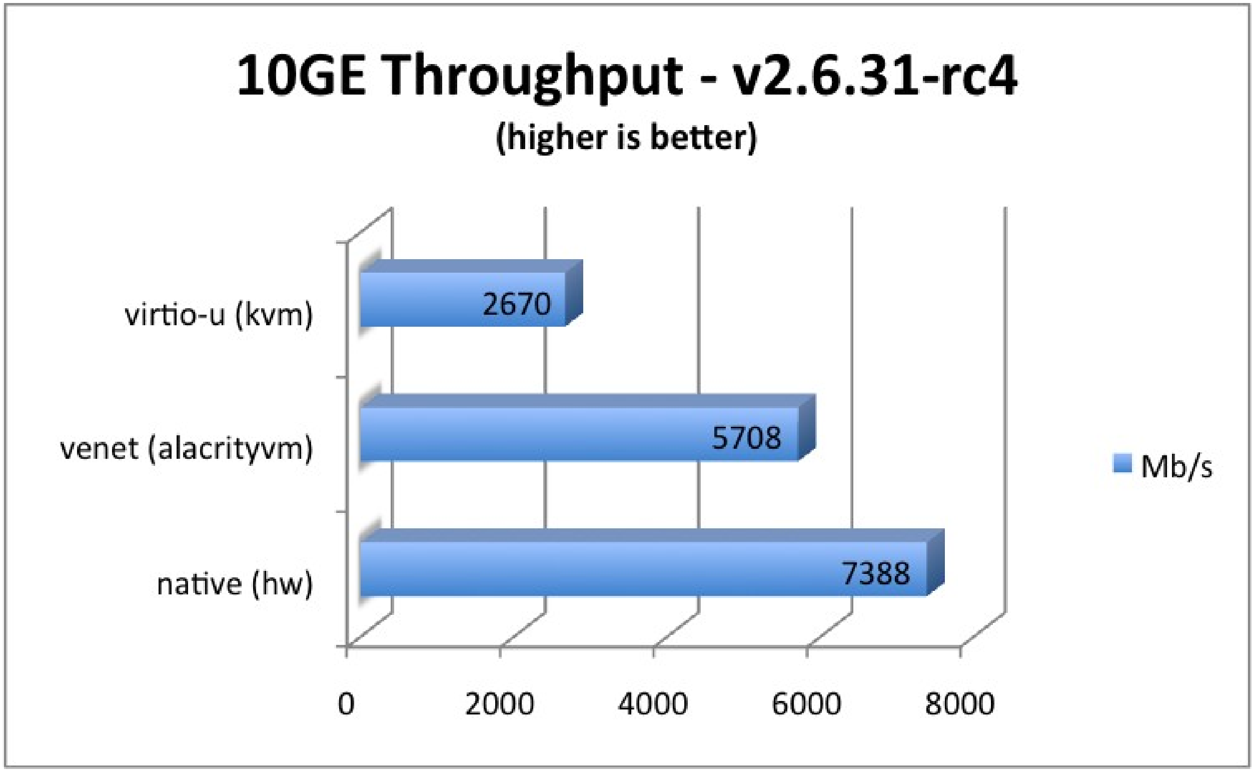
“AlacrityVM is a hypervisor ... which aims to serve a high-performance niche, such as ... HPC and Real-Time workloads in the Data-Center.”

“It achieves this by utilizing a ... high performance IO fabric”

(highly edited – please see original source)

Braver Than Frank, 1

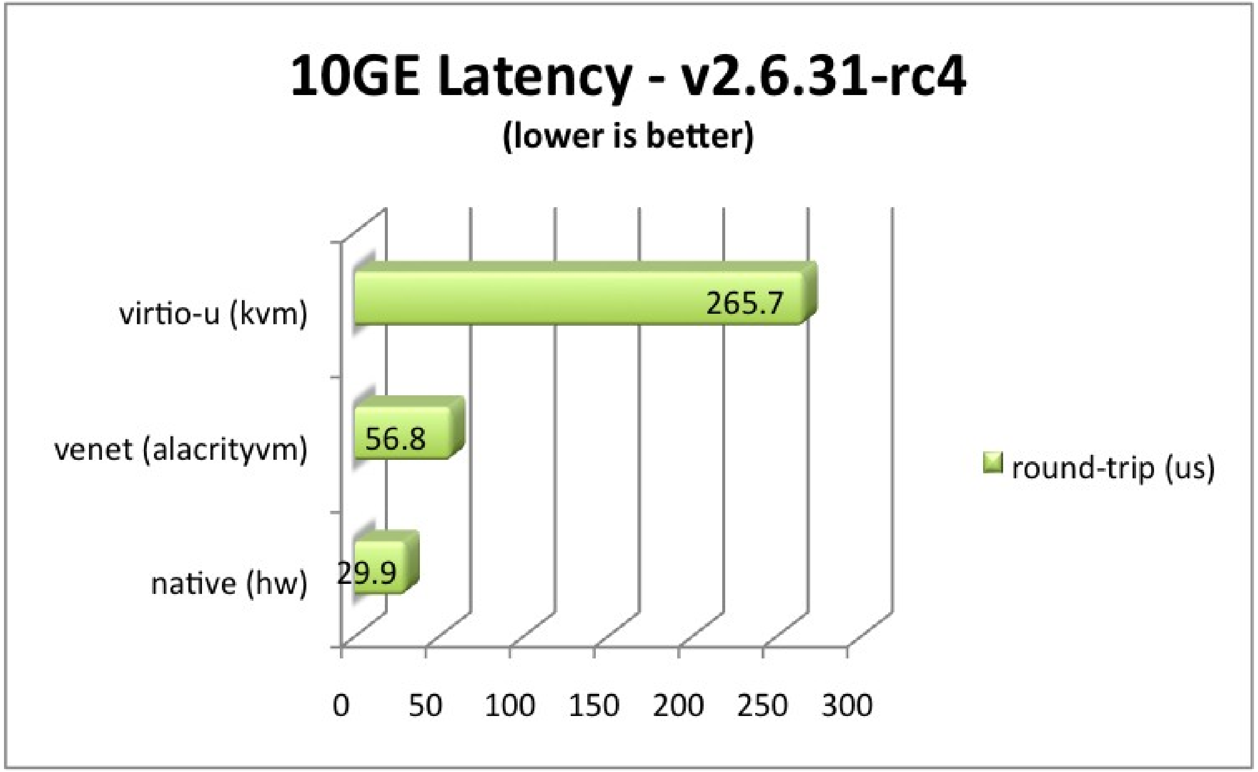
Example of results



source (14 August 2009):
http://developer.novell.com/wiki/index.php/AlacrityVM

Braver Than Frank, 1

Example of results



source (14 August 2009):
http://developer.novell.com/wiki/index.php/AlacrityVM

Braver Than Frank, 1

Example of results

Note that the graphs on the previous two slides are now ancient history and do not reflect current performance of venet and virtio-u.

Braver Than Frank, 2

http://www.osadl.org/Abstract-20-Towards-Linux-as-a-Real-Tim.rtlws11-abstract20.0.html

Eleventh Real-Time Linux Workshop on September 28 to 30,
in Dresden, Germany

Towards Linux as a Real-Time Hypervisor
Jan Kiszka, Siemens AG, Corporate Technology

In this paper, we will present our research work on improving the real-time qualities the Linux hypervisor KVM can provide to its guests. We will specifically focus on a new paravirtualized scheduling interface. It allows guests to influence the scheduling parameters of their virtual CPUs (VCPU) on the host. This, in turn, enables the Linux host to account for real-time load inside guest systems by prioritizing VCPUs properly so that batch load both in other guests as well as on the host itself does not unacceptably interfere.

Braver Than Frank, 2

http://www.osadl.org/Abstract-20-Towards-Linux-as-a-Real-Tim.rtlws11-abstract20.0.html

“... research work on improving the real-time qualities the Linux hypervisor KVM can provide to its guests.”

“... a new paravirtualized scheduling interface ... allows guests to influence the scheduling parameters of their virtual CPUs (VCPU) on the host.”

“This ... enables the ... host to account for real-time load inside guest systems ...”

Hyperthreading

Similar to SMP and Virtualization

...but different!

Hyperthreading

Similar to SMP and Virtualization

...but different!

Do not underestimate the negative impacts on real-time performance.

section 4

Some Random Thoughts

Not All Kernels Are Equal

Some versions of the RT Preempt Patches are less robust.

Not All Kernels Are Equal

Some versions of the RT Preempt Patches
are less robust.

- Some have more radical restructuring
- Some are more experimental
- Recent are based on -tip and pull in origin.patch
- Some have less developer attention
- Some pull previous RT preempt forward
  to newer base kernel without a lot of validation
- Some have more focus on stabilization
- Some have support for more architectures

Not All Kernels Are Equal

Possible Fix

- Use a vendor supported and tested distribution, such as
  - MontaVista Software
  - Red Hat
  - SUSE
  - Wind River

Not All Kernels Are Equal

Possible Fix

Use a stable RT version

eg. OSADL stable rt-linux recomendation

http://www.osadl.org/Realtime-Linux.projects-realtime-linux.0.html

(stable on April 9, 2010 is 2.6.31.12-rt21)

Not All Kernels Are Equal

Possible Fix

Use the RT Preempt Patches on top of a kernel.org tree and schedule sufficient time to tune and stabilize the RT Preempt Patches for your target.

(And submit improvements back to the RT Preempt project.)

section 5

Kernel Features

Resource Allocation

Allocate before beginning real time operation.
For example:

- Create processes.

- Allocate memory.

- Lock memory.

printk()

On PREEMPT_RT kernel printk() may sleep.

kernel/printk.c:

```
/*
 * On PREEMPT_RT kernels __wake_up may sleep, so wake syslogd
 * up only if we are in a preemptible section. We normally dont
 * printk from non-preemptible sections so this is for the emergency
 * case only.
 */

#ifdef CONFIG_PREEMPT_RT
        if (!in_atomic() && !irqs_disabled())
#endif

        if (wake_klogd)
                wake_up_klogd();
```

printk()

On PREEMPT_RT kernel printk() may sleep.

Do not call it from real time context.

- No longer true as of commit b845b517
  Fri Aug 8 21:47:09 2008 +0200
  (2.6.28)
  - wake_up_klogd() no longer calls
    wake_up_interruptible()

Kernel Thread Priorities

Default kernel thread priorities are not likely to be optimal.

Kernel Thread Priorities

Default kernel thread priorities are not likely to be optimal.

Determine proper priorities for:

- IRQ handler threads
- Softirq threads
- real time application kernel threads
- real time application user space threads

Power Management

Frequency Scaling

- Latency while changing frequency.
- Unexpectedly executing slower.

Power Management

CPU Sleep Latency

The wake up latency from cpu sleep increases for deeper levels of sleep.

drivers/cpuidle/* attempts to balance power saving and latency.

Power Management

CPU Sleep Latency

Implementation of balancing power saving and latency is nicely documented by the comment at the top of drivers/cpuidle/governors/menu.c

as of commit 69d25870 2009-09-21 or see http://lwn.net/Articles/352180/ for an earlier version.

Power Management

For optimal real time latency, disable power management.

Power Management

Frequency Scaling, Sleep Mode

CONFIG_APM
CONFIG_ACPI_PROCESSOR
CONFIG_CPU_FREQ
CONFIG_CPU_IDLE

Documentation/cpuidle/*
Documentation/cpu-freq/*

Kernel Configuration Options

Many config options can strongly affect latencies

CONFIG_APM
CONFIG_ACPI_PROCESSOR
CONFIG_CPU_FREQ
CONFIG_CPU_IDLE

CONFIG_NO_HZ
Desirable for cpu isolation, but increases latency.

... and many more – inspect your config!

RT Group Scheduling

Default:

Limit cpu use of real time processes to 95%

RT Group Scheduling

Default:

Limit cpu use of real time processes to 95%

Argument for usage:

Prevents runaway RT process from locking up the system.

RT Group Scheduling

Default:

Limit cpu use of real time processes to 95%

Argument for usage:

Prevents runaway RT process from locking up the system.

Disabling:

echo -1 > /proc/sys/kernel/sched_rt_runtime_us
(Documentation/scheduler/sched-rt-group.txt)

RT Group Scheduling

Issues:

- Scheduler overhead

RT Group Scheduling

Issues:

- Scheduler overhead
- Group sched lock contention

RT Group Scheduling

Issues:

- Scheduler overhead
- Group sched lock contention
- Throttled cpu will attempt to borrow runtime from other cpus.

  A process that can not migrate may have an actual cpu limit that is lower than 95% when other cpus borrow runtime.

RT Group Scheduling

Issues:

- Scheduler overhead
- Group sched lock contention
- Throttled cpu will attempt to borrow runtime from other cpus.
- Reduce headroom by 5%.
  Why eliminate that safety margin?

stop_machine()

Freezes all cpus, except one which executes a specified function.

Interrupts are disabled while the cpus are frozen. Interrupt latency can become very large.

stop_machine()

Users (things to avoid during real time operation):

- module install and remove
- cpu hotplug
- memory hotplug
- ftrace
- hwlat_detector
- xen suspend

highmem

```
#ifdef CONFIG_PREEMPT_RT
# define kmap_atomic(page, type) \
   ({ pagefault_disable(); kmap(page); })
```

- Possible IPI
- Possible sleep

Conclusion: just don't use it...

Recap

- Real time is deterministic, not fast. But typically tune to be fast.

- Hardware issues (memory system, SMI, I/O, external interrupts, SMP, virtualization, other).

- Kernel version

- Kernel specific (priorities, config options, power management, scheduler, stop_machine(), other)

QUESTIONS?

Getting a Copy of the Slides

1) frank.rowand@am.sony.com

2) leave a business card with me