

Realtime In the Enterprise

(definitely not your grandfather's realtime)

*Clark Williams
Team Lead, MRG Realtime
Red Hat Inc.*

Realtime

Back in the Day...

- Usually about meeting strict timing deadlines
- Special purpose hardware
 - Uniprocessor
 - Many times a relatively weak 8 or 16-bit processor
 - Limited RAM
 - Limited mass storage (usually EPROM)
 - Serial communications
 - Single purpose system
 - I/O devices were generally purpose-built for a task

Realtime *Today*

- Less about meeting deadlines and more about reducing the time between an event and the servicing of that event (latency)
- High-end, general-purpose systems
 - Multi-core processors
 - Gigabytes of RAM
 - Large mass storage array
 - High-speed network connections
 - More use of standardized I/O devices

General Purpose Systems Make RT Hard to Do

- SMP complicates the scheduler
- Large numbers of interrupts + large numbers of cores complicate interrupt handling
- Migration costs cause us to lose performance as number of processor cores rise
- Work is being done on different schedulers and new ways of mutual exclusion (EDF and RCU)
- RT is a *magnifier* of existing kernel problems
 - “If RT sees it now, mainline will see it in 3-5 years”

Enterprise Realtime Applications

- Financial
 - Market data (operating on a multicast stream)
 - Trading (transaction oriented)
- Military
 - C³ (Command, Control, Communications)
 - Navigation (image recognition)
- Multimedia
 - Audio recording/playback (millisecond deadlines)
- Seismic Analysis

Realtime Applications differ from Most Regular Applications

- Fairness goes out the window
 - Some threads are more important than others
 - Realtime priorities used to indicate relative importance of threads (SCHED_FIFO)
 - Resources are locked down (e.g. mlock(2))
- Systems are *partitioned*
 - Core (or groups of cores) dedicated to applications
 - Interrupt affinity used to move important interrupts to lightly used core(s)
 - Tuna or taskset is useful for this

Issues with RT Linux on General Purpose Hardware

- Throughput Loss
 - Determinism has its cost
 - 10-20% loss over vanilla kernel in network throughput loads (e.g. netperf)
 - Work is ongoing to identify and remove performance hotspots
- Unexplained Latencies
 - System Management Interrupts (SMIs)

RT Tools

- Many tools being developed for finding/fixing problems and tuning RT Linux
 - perf
 - ftrace
 - rteval
 - hwlat_detector
 - tuna
- All of these tools run on mainstream Linux

perf

- Interface to kernel Performance Events system
- Command line tool with pluggable sub-commands, similar to 'git':
 - perf top – provides updating view of top kernel routines
 - perf record – profiles a command
 - perf report – analyzes data from a 'perf record' run
 - perf annotate – another way to view 'perf record' data
 - perf stat – gather perf counter statistics on a command
- Main use is to analyze kernel performance
- Kept in kernel source tree: tools/perf

ftrace

- Originally grew from Ingo Molnar's latency-tracer
- Steven Rostedt generalized it and added functions
- Many, many tracers available to gather different views of kernel execution
- More info in kernel tree in Documents/tracing
- See Steven's talk this afternoon
 - [git://git.kernel.org/pub/scm/linux/kernel/git/rostedt/linux-2.6-trace.git](https://git.kernel.org/pub/scm/linux/kernel/git/rostedt/linux-2.6-trace.git)

rteval

- Used to measure latency on a system under moderate load
- Two loads:
 - Kernel compile:
 - `make -j<ncores*2> bzImage modules`
 - `hackbench`
- Uses `cyclictest` to measure latency
- Produces XML output with statistical analysis of run data
- [git://git.kernel.org/pub/scm/linux/kernel/git/clkwllms/rteval.git](https://git.kernel.org/pub/scm/linux/kernel/git/clkwllms/rteval.git)

hwlat_detector

- Kernel module written by Jon Masters used to detect hardware related latencies (e.g. SMIs)
 - debugfs interface
- Uses `stop_machine()` to poll system clock, looking for gaps where control transferred away from the OS
- Python wrapper in `rt-tests` package, named `hwlatdetect.py`, drives module
- Tentative plan is to rewrite `hwlatdetect.py` functionality as `perf` sub-command

tuna

- Graphical tool for interactively tuning a running RT system
- Has command line backend, so things done in GUI can be scripted later
- Still under development
 - Python + GTK
 - Should be out in Fedora 12 (already in MRG)
 - [git://git.kernel.org/pub/scm/linux/kernel/git/acme/tuna.git](http://git.kernel.org/pub/scm/linux/kernel/git/acme/tuna.git)

tuna Screenshot

Socket 0			Socket 1			Socket 2			Socket 3		
Filter	CPU	Usage	Filter	CPU	Usage	Filter	CPU	Usage	Filter	CPU	Usage
<input checked="" type="checkbox"/>	1	0	<input checked="" type="checkbox"/>	4	0	<input checked="" type="checkbox"/>	2	0	<input checked="" type="checkbox"/>	3	0
<input checked="" type="checkbox"/>	5	0	<input checked="" type="checkbox"/>	8	0	<input checked="" type="checkbox"/>	6	0	<input checked="" type="checkbox"/>	7	0
<input checked="" type="checkbox"/>	9	0	<input checked="" type="checkbox"/>	12	0	<input checked="" type="checkbox"/>	10	0	<input checked="" type="checkbox"/>	11	0
<input checked="" type="checkbox"/>	13	0	<input checked="" type="checkbox"/>	16	0	<input checked="" type="checkbox"/>	14	0	<input checked="" type="checkbox"/>	15	0
<input checked="" type="checkbox"/>	17	0	<input checked="" type="checkbox"/>	20	0	<input checked="" type="checkbox"/>	18	0	<input checked="" type="checkbox"/>	19	0
<input checked="" type="checkbox"/>	21	0	<input checked="" type="checkbox"/>	0	0	<input checked="" type="checkbox"/>	22	0	<input checked="" type="checkbox"/>	23	0

Socket 4			Socket 5			Socket 6			Socket 7		
Filter	CPU	Usage	Filter	CPU	Usage	Filter	CPU	Usage	Filter	CPU	Usage
<input checked="" type="checkbox"/>	24	0	<input checked="" type="checkbox"/>	25	0	<input checked="" type="checkbox"/>	26	0	<input checked="" type="checkbox"/>	27	0
<input checked="" type="checkbox"/>	28	0	<input checked="" type="checkbox"/>	29	0	<input checked="" type="checkbox"/>	30	0	<input checked="" type="checkbox"/>	31	0
<input checked="" type="checkbox"/>	32	0	<input checked="" type="checkbox"/>	33	0	<input checked="" type="checkbox"/>	34	0	<input checked="" type="checkbox"/>	35	0
<input checked="" type="checkbox"/>	36	0	<input checked="" type="checkbox"/>	37	0	<input checked="" type="checkbox"/>	38	0	<input checked="" type="checkbox"/>	39	0
<input checked="" type="checkbox"/>	40	0	<input checked="" type="checkbox"/>	41	0	<input checked="" type="checkbox"/>	42	0	<input checked="" type="checkbox"/>	43	0
<input checked="" type="checkbox"/>	44	0	<input checked="" type="checkbox"/>	45	0	<input checked="" type="checkbox"/>	46	0	<input checked="" type="checkbox"/>	47	0

Socket 8			Socket 9			Socket 10			Socket 11		
Filter	CPU	Usage	Filter	CPU	Usage	Filter	CPU	Usage	Filter	CPU	Usage
<input checked="" type="checkbox"/>	48	0	<input checked="" type="checkbox"/>	49	39	<input checked="" type="checkbox"/>	50	0	<input checked="" type="checkbox"/>	51	0
<input checked="" type="checkbox"/>	52	0	<input checked="" type="checkbox"/>	53	26	<input checked="" type="checkbox"/>	54	0	<input checked="" type="checkbox"/>	55	0
<input checked="" type="checkbox"/>	56	0	<input checked="" type="checkbox"/>	57	0	<input checked="" type="checkbox"/>	58	0	<input checked="" type="checkbox"/>	59	0
<input checked="" type="checkbox"/>	60	0	<input checked="" type="checkbox"/>	61	0	<input checked="" type="checkbox"/>	62	0	<input checked="" type="checkbox"/>	63	0

IRQ	PID	Policy	Priority	Affinity	Events	Users
0	-1		-1	0-63	4338	timer
4	14323	FIFO	85	0-63	13	serial
8	2257	FIFO	85	0-63	0	rtc0
9	1080	FIFO	85	0-63	0	acpi
14	3323	FIFO	85	0-63	44728	libata
15	3324	FIFO	85	0-63	0	libata
17	2463	FIFO	85	0-63	50	uhci_hcd:usb5
18	2464	FIFO	85	0-63	64	uhci_hcd:usb6
19	2465	FIFO	85	0-63	0	uhci_hcd:usb7
23	2330	FIFO	85	0-63	3	ehci_hcd:usb1,uhci_hcd:usb4
46	3125	FIFO	85	0-63	943	megasas
116	2466	FIFO	85	0-63	52	uhci_hcd:usb9
117	2467	FIFO	85	0-63	18371	uhci_hcd:usb10
118	2468	FIFO	85	0-63	0	uhci_hcd:usb11
122	2331	FIFO	85	0-63	839	ehci_hcd:usb2,uhci_hcd:usb8
142	3127	FIFO	85	0-63	1225	megasas
212	2469	FIFO	85	0-63	0	uhci_hcd:usb13
213	2470	FIFO	85	0-63	59	uhci_hcd:usb14
214	2471	FIFO	85	0-63	0	uhci_hcd:usb15
218	2332	FIFO	85	0-63	55	ehci_hcd:usb3,uhci_hcd:usb12
238	3057	FIFO	85	0-63	29489	loc0
2254	12253	FIFO	85	0-63	7	eth0(bnx2)
2255	12252	FIFO	85	0-63	0	eth0(bnx2)
2256	12251	FIFO	85	0-63	21	eth0(bnx2)
2257	12250	FIFO	85	0-63	6395	eth0(bnx2)
2258	12249	FIFO	85	0-63	180	eth0(bnx2)

PID	Policy	Priority	Affinity	VolCtxtSwitch	NonVolCtxtSwitch	Command Line
1	OTHER	0	0-63	5455	9642	init [5]
2	OTHER	0	0-63	1543	2609	kthreadd
3	FIFO	99	0	986	0	migration/0
4	FIFO	99	0	2	0	posixcpumr/0
5	FIFO	70	0	2	0	sirq-high/0
6	FIFO	70	0	4228459	0	sirq-timer/0

RT Things We Want to Get Upstream

- Finish up threaded IRQ support
 - Lots of drivers need to be evaluated/converted
- Full preemption (sleeping spinlocks)
 - Still working on how to indicate that locks will change behavior based on config options
- More tools
 - tuna
 - rteval
 - perf framework needs extending

Questions?