Measuring the impacts of the Preempt-RT patch

maxime.chevallier@smile.fr

October 25, 2017

Smile
OPEN SOURCE SOLUTIONS

RT Linux projects

- **Simulation platform** : bi-xeon, lots ot RAM
  `200µs wakeup latency, networking`
- **Test bench** : Intel atom
  1s max latency, I/O and networking
- **Embedded telematic board** : i.mx6q
  Never lose incoming data
- **Image processing** : Intel i3
  Process each frame with a deadline

What is a RTOS ?

Real Time : Determinism

- **Bounded Latencies**
  We need guaranties on the reaction time
- **RT Scheduler**
  We want absolute priorities for the tasks
- **Handle the complex cases**
  Priority Inversion, Starvations, etc.

Linux

We have :

- **RT Scheduler** `SCHED_FIFO, SCHED_RR, SCHED_DEADLINE`
- **PI mutexes** *futex, rt-mutex*
- **Preemptible kernel** (almost)
- **High resolution timers** *nanosleep*

We lack :

- **Full kernel preemption**
  A lot of critical sections are present
- **Some worst case scenario optimisations**
  Mostly arch/driver specific, to be mainlined

Preempt RT - Internals

- **Force threaded interrupts**
  Allows to prioritize interrupt handlers
- **Make locks sleepable and RT-aware**
  rt_spinlocks, rt_mutexes, semaphores, RCU
- **Remove critical sections**
  Avoid disabling preemption, interrupts, spinlocks, etc.

What about non-RT tasks ?

- The kernel internals are changed
- Kernel-userspace API/ABI stays the same
- We have what is left of the resources :
  - SCHED_OTHER runs when no RT tasks run, whatever their priority
  - User configuration might dedicate some resources to RT tasks

Firt steps

- **Am I really running the RT patch ?** uname -a
  `cat /sys/kernel/realtime`
- **More tasks are running** htop
  Threaded IRQs - beware of load-avg

perf

Performance analysis tool for Linux (from manpage)

- Uses the kernel performance counters
- Generate traces
- Versatile tool :
  - debugging
  - profiling
  - benchmarking

perf - Vanilla linux

ping -f <ip> -c 1000000

3.26% ping **_raw_spin_lock_irqsave**
2.40% ping entry_SYSCALL_64
2.33% ping _raw_spin_lock
2.26% ping fib_table_lookup
1.87% ping insert_work
1.62% ping _raw_spin_unlock_irqrestore
1.60% ping __ip_route_output_key_hash
1.56% ping __netif_receive_skb_core
1.53% ping queue_work_on

perf - RT Linux

ping -f <ip> -c 1000000

5.53% ping check_preemption_disabled
4.29% ping migrate_enable
3.29% ping __bitmap_equal
2.56% ping migrate_disable
2.55% ping **rt_spin_lock**
2.30% ping preempt_count_add
2.29% ping **rt_spin_unlock**
1.81% ping entry_SYSCALL_64
1.28% ping preempt_count_sub

pidstat, vmstat, mpstat

Event analysis tools

- Analyse context switching
- Interruptions
- Cache misses
- Page faults
- branch prediction

*stat

vmstat 1

```
r in cs
1 2841 696381
2 2134 686653
2 1511 740010
```

pidstat -w 1

```
cswch/s nvcswch/s Command
70443 76 stress-ng-fifo
70571 61 stress-ng-fifo
70587 52 stress-ng-fifo
```

- **vmstat**
  Global memory stats
- **mpstat**
  per processor stats
- **pidstat**
  per task stats

Another example : ping -f

vmstat

vanilla

```
in cs
14363 218
14565 283
14340 91
```

Smile
OPEN SOURCE SOLUTIONS

Another example : ping -f

vmstat

vanilla

```
in cs
14363 218
14565 283
14340 91
```

Preempt RT

```
in cs
14414 29091
14397 29052
14390 29007
```

Another example : ping -f

vmstat

vanilla

```
in cs
14363 218
14565 283
14340 91
```

Preempt RT

```
in cs
14414 29091
14397 29052
14390 29007
```

mpstat -w

```
cswch/s Command
14280 irq/35-enp14s0
```

Another example : ping -f

vmstat

vanilla

```
in cs
14363 218
14565 283
14340 91
```

Preempt RT

```
in cs
14414 29091
14397 29052
14390 29007
```

mpstat -w

```
cswch/s Command
14280 irq/35-enp14s0
```

- Effect of threaded interrupts
- iperf show no bandwidth difference
- This IRQ can now be prioritized

stress-ng

- Has stressors for a lot of components
- Can be used as a 'rough' benchmarking tool
- use `--XXX-ops` and compare execution time
- Beware, extreme scenarios unlikely to happen in real-life

stress-ng

stress-ng

- Has stressors for a lot of components
- Can be used as a 'rough' benchmarking tool
- use `--XXX-ops` and compare execution time
- Beware, extreme scenarios unlikely to happen in real-life

stressor

`cpu`
`fault`
`fifo`
`futex`
`hdd`

stress-ng

stress-ng

- Has stressors for a lot of components
- Can be used as a 'rough' benchmarking tool
- use `--XXX-ops` and compare execution time
- Beware, extreme scenarios unlikely to happen in real-life

stressor	**vanilla**
`cpu`	`11.23 s`
`fault`	`8.94 s`
`fifo`	`8.24 s`
`futex`	`13.11 s`
`hdd`	`8.75 s`

stress-ng

stress-ng

- Has stressors for a lot of components
- Can be used as a 'rough' benchmarking tool
- use `--XXX-ops` and compare execution time
- Beware, extreme scenarios unlikely to happen in real-life

stressor	vanilla	preempt RT
`cpu`	`11.23 s`	`11.26 s`
`fault`	`8.94 s`	`14.51 s`
`fifo`	`8.24 s`	`69.44 s`
`futex`	`13.11 s`	`7.85 s`
`hdd`	`8.75 s`	`8.88 s`

Performance impacts : Preempt-RT

- Syscalls : Expect an overhead
- Locks : Futexes are made faster
- Fifos, mqueues, pipes : Tend to get slower

Performance impacts : Platform-dependent tweaking

- **CPU Idle states** : Use Poll or C1
  Increase power consumption
- **Dynamic Voltage and Frequency Scaling** : Use a fixed frequency
  Might increase power consumption
- **Hyperthreading** : Disable it
  Less processing power

cpuidle, cpufreq

cpuidle in sysfs : `/sys/devices/system/cpu/cpuX/stateY/`

- `name`
- `latency` : *wakeup latency*
- `residency` : *sleep time needed to enter*
- `power` : *power consumed in that state*

powertop

Allows to see C-state and frequency usage

Useful resources

- Who needs a Real-Time Operating System (Not You!)
  Steven Rostedt, Kernel Recipes 2016
- Understanding a Real-Time System (More than just a kernel)
  Steven Rostedt, Kernel Recipes 2016
- SCHED_DEADLINE: It's Alive!
  Juri Lelli, ELC 2016
- Real-Time Linux on Embedded Multicore Processors
  Andreas Ehmanns, ELC 2016
- IRQs: the Hard, the Soft, the Threaded and the Preemptible
  Alison Chaiken, ELCE 2016

That's it

Thank you !

Smile
OPEN SOURCE SOLUTIONS