Understanding Real Time Linux

Alex Shi
Agenda

- What’s real time OS
- RTL project status
- RT testing and tracing
- Reasons of latency and solutions for them
- Resources
- Summary
What’s real time OS

● Real time and high performance
  ○ Fastest? Max throughput? Average latency?
  ○ No, predictable latency!

● Types:
  ○ Hard real time
    ■ Car engine, Nuclear power plant
  ○ Firm real time
    ■ Between hard/soft
  ○ Soft real time
    ■ Live audio/video

● RTOS list link
RTL project status

- RTL project
  - Linux Foundation plan to merge all RT code into upstream
  - Main team member of RT: Thomas Gleixner, Sebastian Siewior, Richard Cohran, Anna-Maria Gleixner, Benedikt Spranger
  - David Long, Mathieu Poirier and Alex also work on it.

- Much of code merged into upstream
  - High res timer
  - Threaded interrupt
  - Lock dep / Ftrace

- A few code out of kernel
  - 400+ patches, < 20,000 lines
  - Sleep locking
  - Interrupt preemptable -- transfer to local lock
  - Optimized for Cpu hotplug Timers; memory alloc; Softirq; Rcu
RT testing and tracing

- **Kernel setting**
  - # CONFIG_PREEMPT_NONE
  - # CONFIG_PREEMPT_VOLUNTARY
  - # CONFIG_PREEMPT__LL
  - # CONFIG_PREEMPT_RT
  - CONFIG_PREEMPT_RT_FULL=y

- **Testing**
  - Benchmarks: rt-tests, rt-app, ltp-realtime
  - ./cyclitest --smp -p98 -m
  - System stress: hackbench -l 10000
  - Results on my hikey620 and dragonboard 410
    - Maximum latency less than 200 us in a busy system

- **Tracing**
  - Function trace
  - Latency trace
Latency tracing

```bash
# echo 0 > options/function-trace; echo wakeup_rt > current_tracer; echo 1 > tracing_on; chrt -f 5 sleep 1; echo 0 > tracing_on
# cat trace
# tracer: wakeup_rt
#
# wakeup_rt latency trace v1.1.5 on 4.11.12-rt13
# --------------------------------------------------------
# latency: 786 us, #4/4, CPU#5 | (M:preempt VP:0, KP:0, SP:0 HP:0 #P:8)
# ****************
# | task: irq/38-f72c0000-1321 (uid:0 nice:0 policy:1 rt_prio:50)
# ****************
#
# cmd     pid    | time   | caller
# <idle>-0  0d...h4..  1us : 0:120:R   + [005]  1321: 49:R irq/38-f72c0000
# <idle>-0  0d...h4..  3us!: try_to_wake_up <-wake_up_process
# <idle>-0  5d...3..  780us : __schedule <-schedule
# <idle>-0  5d...3..  784us : 0:120:R ==> [005]  1321: 49:R irq/38-f72c0000
```
Reasons/Solutions of latencies

- CPU Capacity and System load
- Memory
- Devices
- Interrupt
- Schedule
- Firmware/BIOS routines
- Power management side effect
- SMP sync up
Reasons/Solutions of latencies

- CPU Capacity and System load
  - 1 Ghz CPU freq required*
  - Busy system load
- Way to make real time friendly ENV
  - Isolated cpu
  - Dedicated resource for RT tasks
  - System tuning
    - Documentation/kernel-per-CPU-kthreads.txt
    - Taskset etc
    - isolcpus=
Reasons/Solutions of latencies

- **Memory**
  - Page fault/reclaim cost a lot
  - Swap cause disk level response time
  - Fragment issue for larger memory allocation

- **Workaround**
  - Pre-allocate memory
  - Use slub for defragmentation
  - Reduce the page alloc locking region
Reasons/Solutions of latencies

● Devices/interrupt
  ○ DMA may holding bus for long time
  ○ Interrupt:
    ■ time cost to trap into kernel and stay long
    ■ break the RT task;
    ■ Interrupt disable region -- irq_save

● solutions
  ○ threaded interrupt,
    ■ Prioritize
    ■ preemptable
  ○ Irq disable -> local spin lock.
Reasons/Solutions of latencies

- **Firmware/BIOS**
  - Some device routine defined in firmware/bios
  - Secure system, Trust zone

- **Power management cost**
  - Cpu/clusters c-status, p-state -- use PM QoS or don’t use them
  - Thermal setting -- cooling system
Reasons/Solutions of latencies

- Schedule cost
  - tlb/cache flush
  - Page table refill
  - System load

- Optimize
  - Priority tasks: FIFO, RR, deadline
  - Control normal task preempt
    - Preempt in spin_lock -- lazy preempt
  - Lazy mm replace
  - TLB/Cache flush range
Reasons/Solutions of latencies

- SMP sync up
  - Spin_lock, rw_lock etc.
- Solution:
  - Sleepable/preemptable spin_lock/rwlock: rt_mutex
    - Migration disable
  - Rwlock: reader bias rwlock or use rcu instead.
Priority inversion

- Priority of RT task
  - Necessary to preempt normal task

- Priority inversion
  - Priority inherit

---

```
grab lock L1 (owned by C)
A ----+
   C preempted by B
   |
C  +-----+

B        +-------->
B now keeps A from running.

E->L4->D->L3->C+
   +->L2+-
   |     |
G++  +->B->L1->A
   |     |
F->L5+-
```
Resource links

- RT tree
  - git://git.kernel.org/pub/scm/linux/kernel/git/rt/linux-rt-devel.git

- Real time wiki
  - https://wiki.linuxfoundation.org/realtime/start
  - https://rt.wiki.kernel.org/index.php/Main_Page

- RT mailing list
  - https://wiki.linuxfoundation.org/realtime/communication/mailinglists

- RT testing
  - https://git.linaro.org/power/rt-app.git

- IRC
  - #linux-rt on irc.oftc.net
Summary

- **RTL status**
  - All upstream is on the way

- **Kernel changes for latency**
  - Sleepable spin_lock/rwlock
  - preemptable Irq_save

- **Resources of RT**
  - [https://wiki.linuxfoundation.org/realtime/start](https://wiki.linuxfoundation.org/realtime/start)
Thank You

Alex Shi <alex.shi@linaro.org>

#SFO17

BUD17 keynotes and videos on: connect.linaro.org
For further information: www.linaro.org