Testing real-time Linux: What to test and how.

Sripathi Kodi
sripathik@in.ibm.com
Agenda

- What is a real-time Operating System?
- Enterprise real-time
- Real-Time patches for Linux
- Testing real-time kernel: What to test
- Testing real-time kernel: Setup
- Tips for writing test cases for real-time kernel
- Existing real-time tests
- How can you contribute?
- References
What is a real-time Operating System (OS)?

- Real-time OSes provide *predictable* performance
- They don't necessarily provide the best *throughput*
  - Real-time OSes often sacrifice throughput for predictability
- Used in environments where timely action is critical
  - Defense, medical systems, embedded devices
- Applications to run are carefully chosen
- Hard vs soft real-time
Enterprise real-time systems: The convergence

Today's Systems

- Historical Realtime:
  - Few CPUs
  - Latency Guarantees
  - Non-Standard

OR

- Historical SMP:
  - Many CPUs
  - No Guarantees
  - Standard (and OSS)

Emerging Systems

- SMP Realtime:
  - Many CPUs
  - Latency Guarantees
  - Standard (and OSS)

- User Demand (DoD, Financial, Gaming, ...)
- Technological Changes Leading to Commodity SMP
  - Hardware Multithreading
  - Multi-Core Dies
  - Tens to Hundreds of CPUs per Die – Or More

But Not Both!!!

Source: Paul McKenney
Real-time patches for Linux

- Various approaches
- CONFIG_PREEMPT_RT (RT) patch by Ingo Molnar
- Soft real-time
- Is a patch on top of Linux mainline kernel
- All APIs remain same, hence apps work as-is
- Concentrates on some areas of the kernel
  - Process subsystem, scheduling, Interrupt handling, synchronization
  - Cannot cater to all parts of the kernel. Example: disk IO
Major Components of RT patch

- High Resolution Timers
- Priority Inheritance
- Threaded Interrupts
  - Some interrupts are not threaded
- Threaded softirqs
- Preemptible spin locks
  - raw spin locks
- Read-write locks
Testing the real-time kernel: What to test

- Functionality
  - Similar to mainline kernel
  - Standard functional tests are valid on real-time kernel
  - Test specific features of RT patch
Testing the real-time kernel: What to test

- Functionality
  - Similar to mainline kernel
  - Standard functional tests are valid on real-time kernel
  - Test specific features of RT patch

- Throughput / performance
  - Throughput likely to be less than that on mainline kernel
  - Compare performance between real-time and mainline
  - Compare different versions of real-time kernel
Testing the real-time kernel: What to test

- Functionality
  - Similar to mainline kernel
  - Standard functional tests are valid on real-time kernel
  - Test specific features of RT patch

- Throughput / performance
  - Throughput likely to be less than that on mainline kernel
  - Compare performance between real-time and mainline
  - Compare different versions of real-time kernel

- Latency
  - Most important category
  - Measure latency and variation in latency over various conditions
Testing the real-time kernel: Set up

- **Hardware**
  - Supported architectures
  - Size of RAM
  - SMIs
  - Virtualization
Testing the real-time kernel: Set up

- **Hardware**
  - Supported architectures
  - Size of RAM
  - SMIs
  - Virtualization

- **Kernel**
  - Latest RT patches
  - Configuration options
    - HZ, NO_HZ, CPU_FREQ, Debug options under “Kernel Hacking”
Testing the real-time kernel: Set up

- **Hardware**
  - Supported architectures
  - Size of RAM
  - SMIs
  - Virtualization

- **Kernel**
  - Latest RT patches
  - Configuration options
    - HZ, NO_HZ, CPU_FREQ, Debug options under “Kernel Hacking”

- **Priorities of IRQs and softirqs**
  - chrt command
  - rtctl, set_kthread_prio
Output of 'ps' showing IRQs and softirqs

```
# ps -eo comm,pid,class,rtprio | grep -i irq
sirq-high/0  5 FF  30
sirq-timer/0  6 FF  30
sirq-net-tx/0  7 FF  90
sirq-net-rx/0  8 FF  90
sirq-block/0  9 FF  30
sirq-tasklet/0 10 FF  30
sirq-sched/0  11 FF 30
sirq-hrtimer/0 12 FF  92
sirq-rcu/0    13 FF  30
sirq-high/1   18 FF  30
...             ...
IRQ-11         119 FF  95
IRQ-12         408 FF  95
IRQ-1          409 FF  95
IRQ-8          420 FF  95
IRQ-19         438 FF  95
IRQ-26         489 FF  95
IRQ-6          1079 FF 95
IRQ-24         8230 FF 95
IRQ-4          10449 FF 95
IRQ-3          10451 FF 95
```
Tips for writing test cases for real-time kernel

- Run as SCHED_FIFO
- Limit to available memory
- Page faults
  - Can't be prevented during application start-up
  - Methods to prevent page faults in critical code paths
    - mlockall()
    - Don't create threads
    - Don't allocate dynamic memory
    - Avoid IO
- Take the number of cpus into account
- Interrupt shielding
  - /proc/irq/<n>/smp_affinity
Tips for writing test cases: continued

- Use proper calls to record time
  - `clock_gettime (CLOCK_MONOTONIC)`

- Understand PI mechanism

- Run large number of iterations
  - Find statistics like max, min, average and percentiles
  - Worst case behavior more important than average

- Use proper priorities
  - Possibility of hung system
  - User space now has power. Power imparts responsibility!

- Avoid `sched_yield()`
What happens?

```c
while (1) {
    if (check_something())
        break;
    sched_yield();
}
```
Existing real-time tests

- LTP contains a real-time test suite
- testcases/realtime directory in LTP
  - Functional tests under testcases/realtime/func
  - Utility functions under testcases/realtime/lib
  - Stress tests under testcases/realtime/stress
  - ...

- Some of these tests were written to test functionality while RT patches were being stabilized
- Some were written to analyze specific latency problems
  - Regression test bucket
- rt-tests by Thomas Gleixner: Outside the LTP
Example of a real-time test: pthread_kill_latency

**Main Thread**
- Start
  - Start Signal Sending thread
  - Start Signal Receiving thread
- Wait for other threads to finish

**Signal Sending (SS) Thread**
- Start
  - Wait for SR Thread to Initialize
  - \texttt{start=rt_gettime()}
  - Send a Signal to SR thread
  - Wait until SR thread processes the signal
- \texttt{Loops Over?}
  - No
  - \texttt{Loops Over?}
    - Yes
      - End
  - Yes

**Signal Receiving (SR) Thread**
- Start
  - SR Thread Initialized
  - Wait to receive signal from SS thread
  - Signal received from SS thread
  - \texttt{end = rt_gettime()}
  - Calculate delta = end - begin
  - Min Delta and Max delta
- \texttt{Signal Processed}
  - \texttt{Loops Over?}
    - No
    - \texttt{Loops Over?}
      - Yes
        - End
Performance vs predictability

pthread kill latency (2.6.24.7 vanilla kernel)
Performance vs predictability

pthread kill latency (2.6.24.7 vanilla kernel)

Latency (us) vs Iterations

pthread kill latency (2.6.24.7-rt14)

Latency (us) vs Iterations
How can you contribute?

- Use existing test cases in LTP to test the real-time kernel
  - Share test results, learning

- Procedure to report bugs is same as that on mainline kernel
  - Report bugs on linux-rt-users ML as well as LKML

- Contribute to tests in LTP
  - Improve existing test cases. They are far from perfect
  - Write new tests for missing functionality
  - Submit patches to LTP mailing list

- Improve stress tests on real-time kernel

- Help debug latency problems on real-time kernel
  - Latency tracer, Logdev
### References


- **Real-time kernel howto on the RT wiki**

- **Thomas Gleixner’s real-time tests**

- **linux-rt-users mailing list:**

- **Linux Test Project:** [http://ltp.sourceforge.net/](http://ltp.sourceforge.net/)

- **LTP mailing list:** [https://lists.sourceforge.net/lists/listinfo/ltp-list](https://lists.sourceforge.net/lists/listinfo/ltp-list)

- **IBM’s press release:**
Legal Statement

- This work represents the view of the authors and does not necessarily represent the view of IBM.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.