Realtime and Linux

Ingo Molnar, Red Hat
Paul E. McKenney, IBM LTC

2005 Linux Kernel Summit
Ottawa, Canada
July 18-19, 2005
Overview

- Realtime Desiderata
- Linux Realtime Approaches
- Realtime-Related Patches & Components
- Other Things RT Users Might Ask For
- Discussion
Realtime Desiderata

- Quality of Service (Beyond “Hard”/“Soft”)
  - Services Supported
    - Probability of meeting deadline absent HW failure
    - Deadlines supported
  - Performance/Scalability for RT & non-RT Code
- Amount of Global Knowledge Required
- API Provided (POSIX? Ad Hoc?)
- OS/Application Relative Complexity
- Fault Isolation
- HW/SW Configurations Supported

“But Will People Use It?”
# Linux Realtime Approaches

(Violently Abbreviated)

<table>
<thead>
<tr>
<th>Project</th>
<th>Quality of Service</th>
<th>Inspection</th>
<th>API</th>
<th>Complexity</th>
<th>Fault Isolation</th>
<th>HW/SW Configs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla Linux Kernel</td>
<td>10s of ms all services</td>
<td>All</td>
<td>POSIX + RT extensions</td>
<td>N/A</td>
<td>None</td>
<td>All</td>
</tr>
<tr>
<td>PREEMPT</td>
<td>100s of us Schd, Int</td>
<td>All spinlock critsect, preempt- &amp; int-disable</td>
<td>POSIX + RT extensions</td>
<td>N/A</td>
<td>None</td>
<td>All</td>
</tr>
<tr>
<td>Nested OS</td>
<td>~10 us RTOS svcs</td>
<td>RTOS + int-disable</td>
<td>RTOS</td>
<td>Dual environment</td>
<td>Good</td>
<td>All</td>
</tr>
<tr>
<td>Dual-OS / Dual-Core</td>
<td>&lt;1 us RTOS svcs</td>
<td>All RTOS</td>
<td>RTOS</td>
<td>Dual environment</td>
<td>Excellent</td>
<td>Specialized</td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>10s of us Schd, Int</td>
<td>All preempt- &amp; int-disable (most ints in process ctxt)</td>
<td>POSIX + RT extensions</td>
<td>“Modest” patch</td>
<td>None</td>
<td>All (except some drivers)</td>
</tr>
<tr>
<td>Migration Between OSes</td>
<td>? us RTOS svcs</td>
<td>All RTOS + int-disable</td>
<td>RTOS (can be POSIX)</td>
<td>Dual env. (Fusion)</td>
<td>OK</td>
<td>All?</td>
</tr>
<tr>
<td>Migration Within OS</td>
<td>? us RTOS svcs</td>
<td>Scheduler + RT syscalls</td>
<td>POSIX + RT extensions</td>
<td>Small patch</td>
<td>None</td>
<td>All?</td>
</tr>
</tbody>
</table>
Examples of Linux Approaches

- **Nested OS:**
  - RTLinux, L4Linux, I-pipe (latency from RTLinux)
- **Dual-OS/Dual-Core:**
  - Huge numbers of real products, e.g., cell phones
- **Migration Between OSes:**
  - RTAI-Fusion
- **Migration Within OS:**
  - ARTiS (Asymmetric Real-Time Scheduling)
Related Patches & Components

- **High-Resolution Timers (HRT)**
  - Avoids the “three-millisecond shuffle”
  - Additional code provides fine-grained timers
- **Variable idle Sleep Time (VST)**
  - Suppress unneeded timer ticks, CONFIG_VST
  - Also helps virtualization/consolidation
- **fusyn implementation**
  - Priority inheritance for user-level mutexes
    - Such as pthread_mutex
- **isolcpus**
- **Interrupt-shielding patches & config options**
Other Things RT Users Might Ask For

- Deterministic I/O
  - Disk I/O – or, more likely, flash memory
  - Network protocols
    - Datagram protocols (UDP) relatively straightforward
    - “Reliable” protocols (TCP, SCTP) more difficult

- Other priority inheritance
  - Across memory allocation
    - Boost priority of someone who is about to free...
  - Reader-writer locks with concurrent readers
    - Writer-to-reader boosting quite complex...
  - Across RCU
    - Boost priority of RCU readers when OOM
Discussion

• Observations:
  – Many still consider realtime response from a general-purpose OS to be impossible.
    • Might be why we only have 7 different approaches.
  – Incremental philosophy works well
    • Each approach is incremental
    • Approaches not necessarily mutually exclusive
  – Much overlap between realtime and high end
    • CONFIG_VST
    • SMP testing on UP machines with PREEMPT_RT
    • Paul expects increasing need for realtime response from moderate SMP systems