Real-Time Performance of Linux

- “A Measurement-Based Analysis of the Real-Time Performance of Linux” (L. Abeni, A. Goel, C. Krasic, J. Snow, J. Walpole)

OS Latency

**Definition [OS Latency]**

Let T be a task belonging to a time-sensitive application that requires execution at time t, and let t' be the time at which T is actually scheduled; we define the OS latency experienced by T as \( L = t' - t \).
Sources of OS Latency

- Timer Resolution ($L_{\text{timer}}$)
  - Timer are generally implemented using a periodic tick interrupt. A task that sleeps for an arbitrary amount of time can experience some timer resolution latency if its expected activation time is not on a tick boundary.

- Scheduling Jitter ($L_{SJ}$)
  - Task is not highest in scheduling queue.

- Non-Preemptable Portions ($L_{NP}$)
  - Latency can be caused by non-preemptable sections in kernel and in drivers. (e.g. ISRs, bottom halves, tasklets).

Timer Resolution

- Standard Linux timers are triggered by a periodic tick interrupt.
- On x86 machines it is generated by the Programmable Interval Timer (PIT) with period $T_{\text{tick}} = 10\text{ms}$.

- How about decreasing $T_{\text{tick}}$?

- High-resolution timers using aperiodic interrupt capabilities in modern APICs (Advanced Programmable Interrupt Controller).

- Timer resolution possible in range of 4-6musec.
Non-Preemptible Section Latency

- **Standard Linux**
  - Monolithic structure of kernel.
  - Allows execution of at most one thread in kernel. This is achieved by disabling preemption when an execution flow enters the kernel, i.e., when an interrupt fires or when a system call is invoked.
  - Latency can be as large as 28ms.

- **Low-Latency Linux**
  - Insert explicit preemption points (re-scheduling points) inside the kernel.
  - Implemented in RED Linux and Andrew Morton’s low-latency patch.

- **Preemptable Linux**
  - To support full kernel preemptability, kernel data must be explicitly protected using mutexes or spinlocks.
  - Linux preemptable kernel patch disables preemption only when spinlock is held.
  - Latency determined by max. amount of time for which a spinlock is held plus maximum time taken by ISRs, bottom halves, and tasklets.

- **Preemptable Lock-Breaking Linux**
  - Spinlocks are broken by releasing spinlocks at strategic points.

---

Timer Latency

**Figure 1.** Inter-Activation times for a task that is woken up by a periodic signal with period 100µs on a high resolution timer Linux.

**Figure 2.** PDF of the difference between inter-activation times and period, when \( T = 100\,\mu s \).
Figure 3. OS non-preemptable section latency measured on a high-resolution timer Linux. This test is performed with heavy background load.

<table>
<thead>
<tr>
<th>Memory Stress</th>
<th>I/O Stress</th>
<th>PrexFS Stress</th>
<th>Lock Stress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolithic</td>
<td>1.564</td>
<td>283.1k</td>
<td>320</td>
</tr>
<tr>
<td>Low-Latency</td>
<td>224</td>
<td>207</td>
<td>320</td>
</tr>
<tr>
<td>Preemptible</td>
<td>1.584k</td>
<td>282</td>
<td>320</td>
</tr>
</tbody>
</table>

Table 1. OS non-preemptible section latencies (in μs) for different kernels under different loads (test run for 25 seconds).

<table>
<thead>
<tr>
<th>Memory Stress</th>
<th>I/O Stress</th>
<th>PrexFS Stress</th>
<th>Lock Stress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolithic</td>
<td>1.564</td>
<td>283.1k</td>
<td>017</td>
</tr>
<tr>
<td>Low-Latency</td>
<td>224</td>
<td>207</td>
<td>320</td>
</tr>
<tr>
<td>Preemptible</td>
<td>1.584k</td>
<td>282</td>
<td>320</td>
</tr>
</tbody>
</table>

Table 2. OS non-preemptible section latencies (in μs) for different kernels under different loads (tests run for 16 hours).
Non-Preemptible Portion Latency

Figure 4. CDF of the latency measured on different versions of Linux (with high resolution timers). This test is performed with the I/O stress in background.

Latencies

Figure 5. Audio/Video Skew on standard Linux. Heavy kernel load is run in the background.

Figure 6. Audio/Video Skew for lock-breaking preemptable Linux with high resolution timers. Heavy kernel load is run in the background. The Audio/Video skew is clustered around 3, and the maximum skew is less than 8 (note that the scale is different from Figure 5).
Inter Frame Times

Figure 7. Inter-Frame times for standard Linux. Heavy kernel load is run in the background.

Figure 8. Inter-Frame times for lock-breaking preemption Linux with high resolution timers. Heavy kernel load is run in the background.