RT-Xen: Real-Time Virtualization

Sisu Xi, Meng Xu, Chenyang Lu, Linh T.X. Phan, Christopher D. Gill, Insup Lee, Oleg Sokolsky
Real-Time Virtualization

Cars: Consolidate ~100 ECUs -> ~10 multicore processors
- Infotainment on Linux or Android
- Safety-critical control on AUTOSAR

Cloud Computing’s Killer App: Gaming [IEEE Spectrum]
- Need to compute and stream 30 to 50 frames per second

Applications must meet real-time performance constraints on virtualized platforms!
RT-Xen: Real-Time Virtualization

- Real-time hypervisor scheduling framework in Xen
  - Implement a suite of real-time scheduling algorithms

- Based on compositional scheduling theory
  - VMs specify resource interfaces
  - Real-time guarantees to tasks in VMs

- Open source
  - RT-Xen patch submitted
  - https://sites.google.com/site/realtimexen/
Xen Virtualization Architecture

- Guest OS runs on VCPUs
- Hypervisor schedules VCPUs on PCPUs
- Credit scheduler
  - [Weight, Cap] per VM
  - Round robin

![Diagram of Xen Virtualization Architecture]

- Real-Time Task
- VCPU
- OS Sched
- VMM Scheduler
- PCPUs
RT-Xen Interface

- VM resource interface
  - A set of VCPUs, each characterized by <period, budget>
  - Optional: use cpumask to specify VCPU affinity with PCPUs
  - Hide task-specific information

- Real-Time scheduling algorithms
  - Ordering of VCPUs? Priority scheme
  - Placement of VCPUs? Global vs. partition
  - Resource isolation? Server mechanisms
Real-Time Scheduling Policies

➢ Priority scheme
  ▹ Static priority: Deadline Monotonic (DM)
  ▹ Dynamic priority: Earliest Deadline First (EDF)

➢ Global scheduling
  ▹ Schedule VCPUs based on global information
  ▹ Allow VCPU migration across cores
  ▹ Flexible use of multiple cores
  ▹ Migration overhead and cache penalty

➢ Partitioned scheduling
  ▹ Assign and bind VCPUs to PCPUs
  ▹ Schedule VCPUs on each core independently
  ▹ May underutilize PCPUs
  ▹ No migration overhead or associated cache penalty
Scheduling a VCPU as a Deferrable Server

- A VCPU receives budget us of CPU resources every period us
  - Budget is replenished at every start of period
  - VCPU consumes budget when running, suspends when no budget left
  - Preserves budget when there is no task
RT-Xen Investigation Roadmap

- Single-core
- Single-core enhanced
- Multi-core

RT-Xen 1.0
RT-Xen 1.1
RT-Xen 2.0
RT-Xen 2.0: Run Queues

- A run queue
  - holds VCPUs that are runnable (have task to run)
  - has two parts: VCPUs with budget and out of budget
  - is sorted by priority (DM or EDF) within each part

- rt-global: all cores share one run queue with a spinlock
- rt-partition: one run queue per core

- Patches for more efficient implementation on the way!
Experimental Setup

- **Hardware**: Intel i7 processor, six cores running at 3.33 GHz
  - Dedicate one PCPU to domain 0
  - All guest VMs use the remaining cores

- **Cache architecture**
  - Each core has dedicated L1 cache (32 KB) and L2 cache (256 KB)
  - All six cores share L3 cache (12 MB)
  - Inclusive L3 cache, all data in L2 cache must also be in L3 cache

- **Software**
  - Xen 4.3 patched with RT-Xen
  - Guest OS: Linux patched with LITMUS$^{RT}$
rt-global has extra overhead due to global lock

credit has high max overhead due to load balancing
credit missed deadline at 22% CPU capacity

RT-Xen delivers real-time performance up to 78%
Global scheduling wins empirically!
gEDF + deferrable server -> best real-time performance
Demo

- YouTube: “RT-Xen Demonstration”
  - https://www.youtube.com/watch?v=wisxWn3mR5s
Patch Status

- Patch RFC v1 & v2
  - gEDF + deferrable server
  - cpupool support
  - July 10th & July 29th

- Patch RFC v3
  - Scheduling trace support
  - Performance improvement (splitting RunQ)
  - Expected on Aug 24th

- Patch RFC v4
  - Performance improvement
    - Timer based budget replenishment
    - Improve the timing resolution of budget
  - Expected before Sep 10th
Conclusion

- Diverse applications demand real-time virtualization
  - Real-time virtualization in embedded area
  - Cloud gaming

- RT-Xen provides real-time performance
  - Efficient implementation of diverse real-time scheduling policies
  - Leverage compositional scheduling theory -> analytical guarantee
  - gEDF + deferrable server wins empirically
Research Contributions

- **RTCA**: S. Xi, C. Li, C. Lu, and C. Gill, *Prioritizing Local Inter-Domain Communication in Xen*, ACM/IEEE International Symposium on Quality of Service (IWQoS) 2013

- RT-Xen patch (gEDF with deferrable server)
Implementation – PCPU

### VCPU Position

<table>
<thead>
<tr>
<th>VCPU Position</th>
<th>✓ Budget</th>
<th>✓ Budget</th>
<th>✓ Task</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RunQ</td>
<td>RunQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Task</td>
</tr>
<tr>
<td></td>
<td>RdyQ</td>
<td>RdyQ</td>
<td></td>
</tr>
</tbody>
</table>

### VCPU Params

(period, budget, priority)

#### Sorted by VCPU Priority

- RunQ
- RdyQ
- RepQ

#### Sorted by Replenishment time

Three Queues within One Physical Core

#### Task arrives within a VCPU

- IDLE

#### VCPU has no task to run

- Periodic

- IDLE
Server Design – Deferrable & Polling

 Servers (Period, Budget, Priority)

S1 (5, 3) with Two Tasks

\[ T1 (10, 3) \]
\[ T2 (10, 3) \]

1. Replenish?
2. Budget but NO task?

Actual Execution

Deferrable Server

Polling Server

Budget in S1
Server Design – Periodic & Sporadic

Servers (Period, Budget, Priority)

S1 (5, 3) with Two Tasks

\( T_1 (10, 3) \)

\( T_2 (10, 3) \)

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budget in S1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Actual Execution

**Periodic Server**

**Sporadic Server**

1. Replenish?
2. Budget but NO task?

---

theory favored

overhead ++
RT-Xen 2.0: Workload

- Periodic task sets: [period, execution time, deadline]

- CPU-intensive, independent tasks

- Randomly generate the task sets until a total task utilization, then distribute tasks to four VMs, and apply compositional scheduling theory to calculate each VM’s resource interface

- 25 task sets per data point, measure fraction of schedulable tasksets
Less than 1 us overhead for the spinlock
RT-Xen 2.0: Theory vs. Experiments

\[ g\text{EDF} < p\text{EDF} \text{ theoretically due to pessimistic analysis} \]
\[ g\text{EDF} > p\text{EDF} \text{ empirically, thanks to global scheduling} \]
RT-Xen 2.0: Theory vs. Experiments

- gEDF > pEDF empirically, thanks to global scheduling
- gEDF < pEDF theoretically due to pessimistic analysis
RT-Xen 2.0: How about Cache?

Benefit of global scheduling dominate migration cost on a shared L3-cache platform.
RT-Xen 2.0: Context Switch

“Fake” context switch with idle VCPUs