RT-Xen: Real-Time Virtualization from embedded to cloud computing

Chenyang Lu
Cyber-Physical Systems Laboratory
Department of Computer Science and Engineering
Real-Time Virtualization for Cars

- Consolidate 100 ECUs \(\rightarrow\) 10 multicore processors.
- Integrate multiple systems on a common platform \(\rightarrow\) virtualization.
  - Infotainment on Linux or Android
  - Safety-critical control on AUTOSAR
  - Example: COQOS, Integrity Multivisor, Xen automotive branch
- Must preserve real-time guarantees on a virtualized platform!

Right: [http://community.arm.com/docs/DOC-4610](http://community.arm.com/docs/DOC-4610)
Real-Time Cloud

- Internet of Things $\rightarrow$ Cyber-Physical Systems
  - Smart manufacturing, smart transportation, smart grid.
  - Internet-scale sensing and control $\rightarrow$ real-time cloud computing.

- Example: Intelligent Transportation Systems
  - Data center collects data from cameras and roadside detectors.
  - Control the traffic signals and message signs in real-time.
  - Transportation information feed to drivers.
  - SCATS @ Sydney: controlling 3,400 signals at 1s round-trip latency.

- Latency-sensitive applications, e.g., cloud gaming
  - Xbox One: cloud offloading computation of environmental elements
  - Sony acquired Gaikai, an open cloud gaming platform.
Virtualization is **not** real-time today

- Existing hypervisors provide no guarantee on latency
  - Xen: credit scheduler, [credit, cap]
  - VMware ESXi: [reservation, share, limitation]
  - Microsoft Hyper-V: [reserve, weight, limit]

- Public clouds lack service level agreement on latency
  - EC2, Compute Engine, Azure: #VCPUs

Current platforms provision resources, not latency!
Challenges

- Support real-time applications in a virtualized environment.
  - Latency *guarantees* to tasks running in virtual machines (VMs).
  - Real-time performance *isolation* between VMs.

- Multi-level real-time performance provisioning
  - Virtualization within a host
  - Communication and I/O
  - Cloud resource management
RT-Xen

- Real-time hypervisor based on Xen
  - Real-time VM scheduling
  - Real-time communication

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

- Open source
  - Xen patch in progress

- RT-OpenStack: cloud management based on RT-Xen
Xen Virtualization Architecture

- **Xen**: type-1, baremetal hypervisor
  - Domain-0: drivers, tool stack to control VMs.
  - Guest Domain: para-virtualized or fully virtualized OS.

- **Xen scheduler**
  - Guest OS runs on VCPUs.
  - Xen schedules VCPUs on PCPUs.
  - Credit scheduler: round-robin with proportional share.
Compositional Scheduling

- Analytical real-time guarantees to tasks running in VMs.
- VM resource interfaces
  - Hides task-specific information
  - Multicore: <period, budget, #VCPU>
  - Computed based on compositional scheduling analysis

Diagram:
- Hypervisor
  - Resource Interface
  - Scheduler
- Virtual Machines
  - Resource Interface
  - Scheduler
  - Workload
Real-Time Scheduling Policies

- **Global scheduling**
  - Shared global run queue
  - Allow VCPU migration across cores
  - Work conserving
  - Migration overhead and cache penalty

- **Partitioned scheduling**
  - Assign and bind VCPUs to cores
  - Schedule VCPUs on each core independently
  - Not work conserving: cores may idle when others have work pending
  - No migration overhead or cache penalty

- **Priority scheme**
  - Static priority: Rate Monotonic
  - Dynamic priority: Earliest Deadline First (EDF)
Scheduling VCPU as “Server”

- T2 (5, 3)
- T1 (5, 3)

Periodic Server (5,3)

Budget

Not work conserving

Deferrable Server (5,3)

Budget

back-to-back execution
Run Queues

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

A run queue
- Holds VCPUs that are runnable (not idle)
- Divided into two parts: with budget; without budget
- Sorted by priority (DM or EDF) within each part
Scheduling Functions

- **do_schedule**
  - triggered every 1 ms
  - consume current VCPU’s budget
  - replenish a VCPU’s budget if it is the end of its period
  - return runnable VCPU with the highest priority

- **context_switch**

- **context_saved**
  - global scheduler only
  - insert VCPU back to run queue after context switch

- **wake_up**
  - triggered by timer or I/O
  - call do_schedule if necessary
RT-Xen: Real-Time Scheduling in Xen

- Single-core RT-Xen 1.0
- Single-core enhanced RT-Xen 1.1
- Multi-core scheduling RT-Xen 2.0
  - RT-global
  - RT-partition

Diagram:

- Fixed Priority (RM)
  - Deferrable
  - Periodic
- Dynamic Priority (EDF)
  - Deferrable
  - Periodic
- Global Scheduling
  - Deferrable
  - Periodic
- Partitioned Scheduling
  - Deferrable
  - Periodic
- Work Conserving Periodic
- Capacity Reclaiming Periodic
- Polling
- Sporadic

Single-core RT-Xen 1.0
Single-core enhanced RT-Xen 1.1
Multi-core scheduling RT-Xen 2.0

RT-global
RT-partition
Experimental Setup

- **Hardware**: Intel i7 processor, 6 cores, 3.33 GHz
  - Allocate 1 VCPU for Domain-0, pinned to PCPU 0
  - All guest VMs use the remaining cores

- **Software**
  - Xen 4.3 patched with RT-Xen
  - Guest OS: Linux patched with LITMUS

- **Workload**
  - Period tasks: synthetic, ARINC 653 avionics workload (RT-Xen 1.1)
  - Allocate tasks → VMs
RT-Xen 2.0: Credit Scheduler

- Credit misses deadlines at 22% of CPU capacity.
- RT-Xen delivers real-time performance at 78% of CPU capacity.
RT-Xen 2.0: Scheduling Overhead

- rt-global has extra overhead due to global lock.
- Credit has poor max overhead due to load balancing.
RT-Xen 2.0: Theory vs. Experiments

• $g_{EDF} > p_{EDF}$ empirically, thanks to work-conserving global scheduling.
• $g_{EDF} < p_{EDF}$ theoretically due to pessimistic analysis.
Work-conserving wins empirically!

- Deferable Server (DS) > Periodic Server.
- \( gEDF+DS \rightarrow \) best real-time performance.
RT-Xen 2.0: How about Cache?

- gEDF > pEDF for cache intensive workload.
- Benefit of global scheduling dominates migration cost.
- Shared cache mitigates cache penalty due to migration.
Real-Time Inter-VM Communication

- Real-time communication between high priority VMs under contention from other VMs

- Local communication between VMs on a same host
  - ~20 VMs on a single host is not rare.
  - Co-locate VMs to reduce communication cost.

- Network communication (on-going)
Is real-time VM scheduling enough?

CDF Plot

RT–Xen, Original Dom 0

VMM Scheduler

Dom 0

Dom 1 → Dom 2

Dom 3 → Dom 4 → Dom 5...

5K data points
Significant priority inversion in Domain 0

✘ Packets are fetched in a round-robin order.
✘ Sharing one queue in softnet_data.
Preserve prioritization in Domain 0
- Packets are fetched by priority, up to batch size
- Queues are separated by priority in softnet_data
Real-Time Inter-VM Communication

<table>
<thead>
<tr>
<th>Domain-0</th>
<th>Median</th>
<th>75&lt;sup&gt;th&lt;/sup&gt; percentile</th>
<th>95&lt;sup&gt;th&lt;/sup&gt; percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original</td>
<td>RTCA</td>
<td>Original</td>
</tr>
<tr>
<td>Base</td>
<td>68</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td>Light</td>
<td>5183</td>
<td>60</td>
<td>5803</td>
</tr>
<tr>
<td>Medium</td>
<td>9621</td>
<td>61</td>
<td>9780</td>
</tr>
<tr>
<td>Heavy</td>
<td>9872</td>
<td>69</td>
<td>10095</td>
</tr>
</tbody>
</table>

RTCA maintains low latency for high-priority VMs despite contention from low-priority VMs
Conclusion

- Diverse applications demand real-time virtualization.
  - Embedded real-time systems.
  - Internet-scale cyber-physical systems.
  - Latency-sensitive cloud applications.

- RT-Xen provides real-time performance
  - Efficient implementation of diverse real-time scheduling policies.
  - Leverage compositional scheduling theory \( \rightarrow \) analytical guarantee.
  - Resource interface \( \rightarrow \) resource allocation for latency bounds.
  - RTCA supports real-time inter-VM communication.

- On-going
  - RT-Xen patch for Xen core distribution.
  - RT-OpenStack: integration with OpenStack on the way.
Check out RT-Xen

https://sites.google.com/site/realtimexen/

Team

- Sisu Xi, Justin Wilson, Chong Li, Chris Gill, Washington University
- Meng Xu, Jaewoo Lee, Insup Lee, Lihn Phan, Oleg Solosky, University of Pennsylvania