M2-Xen: Multi-Mode Virtualization for Soft Real-Time Systems

Haoran Li, Meng Xu, Chong Li, Chenyang Lu, Chris Gill, Lihn Phan, Insup Lee, Oleg Sokolsky
Washington University in St. Louis, University of Pennsylvania
Real-Time Virtualization

- Real-time systems integration
  - Integrate multiple real-time systems on a common platform as VMs.
- Automotive: ~100 ECUs → ~10 multicore processors.
  - Real-time: Infotainment, safety-critical control, driver assistance...
  - Virtualization: Integrity Multvisor, Nautilus (Xen Automotive)

Real-time schedulers in the Xen hypervisor.
Provide real-time guarantees to tasks in VMs.
**Xen**: Real-Time Deferrable Server (**RTDS**) scheduler.
Compositional Scheduling

- Real-time guarantees to tasks in Virtual Machines (VMs).
- VM **resource interfaces** (RI)
  - A set of VCPUs each with an interface `<budget, period>`
    - bandwidth=budget/period
  - Computed based on compositional scheduling analysis
  - Hide task-specific information
Challenge: Multi-Mode Systems

- System may operate in different modes
  - VM may have different tasksets in different modes
  - Example: Infotainment adapts to user behavior in vehicles
  - Example: Edge clouds for agile manufacturing

- Statically provisioning each VM based on its **worst-case** workload
  - Underutilize resource
  - Cannot adapt to workload changes
M2-Xen: Multi-Mode Virtualization

- Dynamic resource allocation to VMs based on current mode
  - Adapt to workload changes
  - Maintain real-time performance in different system modes
  - Efficiently utilize CPU resources

- Multi-mode virtualization platform
  - Multi-mode resource interface for VMs
  - Overload avoidance during mode switching
  - Fast mode switching
  - Implemented and evaluated in Xen 4.8 and Linux 4.4.19
Multi-Mode Resource Interfaces

- Each VM specifies a resource interface in each mode
  - Pre-computed based on the taskset in each mode
- M2-Xen switches its resource interfaces in response to mode change
Resource Efficiency

Static RI based on worst-case tasksets → need more PCPUs

Multi-Mode RI: both modes can be accommodated with 15 PCPUs
M2-Xen: Architecture
M2-Xen: Architecture
Overload Avoidance: VM Level

- Avoid overloading VM during mode transition
  - Keep VM’s bandwidth > taskset utilization
- Solution: ordering the changes to RI and taskset
  - Increase load: increase RI → switch taskset
  - Reduce load: switch taskset → reduce RI
Overload Avoidance: PCPU Level

- Keep total bandwidth of all VMs < system capacity.
  - Schedule the mode changes of VMs subject to capacity constraints
  - Batch: reconfigure VMs with decreasing bandwidth first, then the others.

Example (one PCPU):

<table>
<thead>
<tr>
<th>Bandwidth of Guest</th>
<th>Mode 0</th>
<th>In-Transition</th>
<th>Mode 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1</td>
<td>30%</td>
<td></td>
<td>5%</td>
</tr>
<tr>
<td>VM2</td>
<td>20%</td>
<td></td>
<td>10%</td>
</tr>
<tr>
<td>VM3</td>
<td>15%</td>
<td></td>
<td>30%</td>
</tr>
<tr>
<td>VM4</td>
<td>10%</td>
<td></td>
<td>40%</td>
</tr>
<tr>
<td>Total Bandwidth</td>
<td>75%</td>
<td></td>
<td>85%</td>
</tr>
</tbody>
</table>
Mode Change Latency

Breaking down latency
- VM0 changes the RI of VM
- **VM0 sends message to VM**
- VM changes taskset

- **Inter-VM communication dominates latency!**
  - Caused by budget replenishment
  - VM cannot receive the message until its VCPU is scheduled
The rt-task exhausted all budget of its VCPU. The VCPU budget replenished at next period. Then, the DMM can be scheduled on VCPU.

How to mitigate the impact of budget replenishment on mode change latency?
Reduce Mode Change Latency

- **User-level:** *mitigate* the impact of communication delay
  - **Batch:** send requests to multiple VMs concurrently
  - **Greedy:** schedule VM mode change when bandwidth becomes available.
  - Do not reduce budget replenishment delay

- **Hypervisor-level:** *reduce* communication latency
  - **Boost** VCPU to receive message for mode switching
  - Need to modify the Xen kernel
  - Deviate from compositional scheduling (but in a bounded manner)
Boost the VCPU

When VM0 sends a mode switch request to a VM
- Immediately schedule the receiving VCPU
- Promote the VCPU’s priority to the highest
- Reset the VCPU’s priority when receiving its ACK (or timeout)

In Dom0

Send Request

Receive ACK

Time

Schedule VCPU immediately

Revoke the privilege of the VCPU
Boost Reduces Latency

RTDS incurs budget replenishment delay

Boost removes budget replenishment delay
Mode Change Latency

- Batch & Greedy statistically reduce mode change latency by scheduling concurrent communication from VM0 to VMs.
- Boost drastically reduces median/tail latency through kernel scheduling.

Intel E5-2863v4 16-core @ 2.1GHz, 64GB Memory, Dom0 pinned to PCPU0.
Conclusion

- Multi-mode virtualization adapts to workload changes
  - Maintain real-time performance in different system modes
  - Efficiently utilize CPU resources

- M2-Xen: multi-mode virtualization platform on Xen
  - Multi-mode resource interface for VMs
  - Avoid overload at both VM and PCPU levels during mode switching
  - Fast mode switching through user-space or hypervisor scheduling

- Future work
  - Multi-mode in distributed cloud platforms
  - Mode switch protocols for hard real-time systems