Minimizing I/O Latency in Xen-ARM

Seehwan Yoo, Chuck Yoo
and OSVirtual
Korea University
I/O Latency Issues in Xen-ARM

- Guaranteed I/O latency is essential to Xen-ARM
  - For mobile communication,
    - Broken/missing phone calls, long network detection
    - AP, CP are used for guaranteed communication

- Virtualization premises transparent execution
  - Not only instruction execution; access isolation
  - But also performance; performance isolation

- In practice, I/O latency is still an obstacle
  - Lack of scheduler support
    - Hierarchical scheduling nature
      i.e. hypervisor cannot impose task scheduling inside a guest OS
    - The credit scheduler doesn’t match to time-sensitive applications
  - Latency due to the split driver model
    - Enhances reliability, but degrades performance (w.r.t. I/O latency)
    - Inter-domain communication between IDD and user domain

- We investigate these issues, and present possible remedies
Related Work

- **Credit schedulers in Xen**
  - Task-aware scheduler (by Hwanjoo et. al. vee’08)
    - Inspection of task execution at guest OS
    - Adaptively use boost mechanism by VM workload characteristics
  - Laxity-based soft RT scheduler (by Lee min et. al. vee’10)
    - Laxity calculation by execution profile
    - Assign priority based on the remaining time to deadline (laxity)
  - Dynamic core-allocation (by Y. Hu et. al. hpdc’10)
    - Core-allocation by workload characteristics (driver core, fast/slow tick core)

- **Mobile/embedded hypervisors**
  - OKL4 microvisor: microkernel-based hypervisor
    - Has verified kernel – from sel4
    - Presents good performance – commercially successful
    - Real-time optimizations – slice donation, direct switching, reflective scheduling, threaded interrupt handling, etc.
  - VirtualLogix – VLX: mobile hypervisor for real-time support
    - Shared driver model – device sharing model among RT-guest OS and non-RT guest OSs
    - Good PV performance
Background – the Credit in Xen

• Weighted round-robin based fair scheduler
• Priority – BOOST, UNDER, OVER
  – Basic principle: preserve fairness among all VCPUs
    • Each vcpu gets credit periodically
    • Credit is debited as vcpu consumes execution time
  – Priority assignment
    • Remaining credit <= 0 → OVER (lowest)
    • Remaining credit > 0 → UNDER
    • Event-pending VCPU : BOOST (highest)
  – BOOST: for providing low response time
    • Allows immediate preemption of the current vcpu
Fallacies for the BOOST in the Credit

• Fallacy 1) VCPU is always boosted by I/O event
  – In fact, BOOST is sometimes ignored because VCPU is boosted only when it doesn’t break the fairness
    • ‘Not-boosted vcpu’s are observed when the vcpu is in-execution
  – Example 1)
    • If a user domain has CPU job and waiting for execution, then
    • It is not boosted since it will be executed soon, and tentative BOOST is easy to break the fairness

• Fallacy 2) BOOST always prioritizes the VCPU
  – In fact, BOOST is easy to be negated because multiple vcpus can be boosted simultaneously
    • ‘Multi-boost’ happens quite often in split driver model
  – Example 1)
    • Driver domain has to be boosted, and then
    • User domain also needs to be boosted
  – Example 2)
    • Driver domain has multiple pkts that are destined to multiple user domains, then
    • All the designated user domains are boosted
Xen-ARM’s Latency Characteristic

- I/O latency measured throughout interrupt path
  - Preemption latency: until code is preemptible
  - VCPU dispatch latency: until the designated VCPU is scheduled
  - Intra-VM latency: until IO completion
Observed Latency thru intr. path

- I/O latency measured throughout interrupt path
  - Send ping request from external server to dom1
  - VM settings
    - Dom0 : the driver domain
    - Dom1 : 20% cpu load + ping recv.
    - Dom2 : 100% cpu load (CPU-burning workload)

- Xen-netback latency
  - Large worst-case latency
  - Dom0 vcpu dispatch lat. + intra-dom0 lat.

- Netback-domU latency
  - Large average latency
  - Dom1 vcpu dispatch lat.
- Experimental setting
  - Dom1: varying CPU workload
  - Dom2: burning CPU workload
  - Another external host sends ping to Dom1

- Not-boosted vcpus affect I/O latency distribution
  - Dom1 CPU workload 20%: almost 90% ping requests are handled within 1ms
  - Dom1 CPU workload 40%: 75% ping requests are handled within 1ms
  - Dom1 CPU workload 60%: 65% ping requests are handled within 1ms
  - Dom1 CPU workload 80%: only 60% ping requests are handled within 1ms

- When Dom1 has more CPU load
  ➔ larger I/O latency (self-disturbing by not-boosted vcpu)
Observed Multi-boost

- At the hypervisor, we counted the number of “schedule out” of the driver domain

- Multi-boost is specifically when
  - the current VCPU is BOOST state, and it is unblocked
  - Imply that there should be another BOOST vcpu, and the current vcpu is scheduled out

- Large number of Multi-boosts

<table>
<thead>
<tr>
<th>vcpu state</th>
<th>priority</th>
<th>sched. out count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocked</td>
<td>BOOST</td>
<td>275</td>
</tr>
<tr>
<td></td>
<td>UNDER</td>
<td>13</td>
</tr>
<tr>
<td>Unblocked</td>
<td>BOOST</td>
<td>664,190</td>
</tr>
<tr>
<td></td>
<td>UNDER</td>
<td>49</td>
</tr>
</tbody>
</table>
Intra-VM latency

- Latency from usb_irq to netback
  - Schedule outs during dom0 execution

- Reasons
  - Dom0: not always the highest prio.
  - Asynch I/O handling: bottom half, softirq, tasklets, etc.
Virtual Interrupt Preemption

@ Guest OS

• Interrupt enable/disable is not physically operated within a guest OS
  – local_irq_disable(): disables only virtual intr.
• Physical intr. can be occurred
  – might trigger inter-VM scheduling
• Virtual interrupt preemption
  – Similar to lock-holder preemption
  – Perform driver function with interrupt disabled
  – Physical timer intr. triggers inter-VM scheduling
  – Virt. Intr. can be received only after the domain is scheduled again (as large as tens of ms)

Note that the driver domain performs extensive I/O operation that disables interrupt
1. Fixed priority assignment
   - Let the driver domain always run with DRIVER_BOOST, which is the highest priority
     • regardless of the CPU workload
     • Resolves non-boosted VCPU and multi-boost
   - RT_BOOST, BOOST for real-time I/O domain

2. Virtual FIQ support
   - ARM-specific interrupt optimization
   - Higher priority than normal IRQ interrupt
   - vPSR (Program status register) usage

3. Do not schedule out the driver domain when it disables virtual interrupt
   - It will be finished soon, and the hypervisor should give a chance to run the driver domain
     • Resolves virtual interrupt preemption
Enhanced Latency: no dom1 vcpu dispatch latency

Latency (ms)

Cumulative distribution

Xen-netback @ dom0
Xen-icmp @ dom1
Dom0 : IDD
Dom1 : CPU 20%+ ping recv
Dom2 : CPU 100%

Latency (ms)

Cumulative distribution

Xen-netback @ dom0
Xen-icmp @ dom1
Dom0 : IDD
Dom1 : CPU 40%+ ping recv
Dom2 : CPU 100%

Latency (ms)

Cumulative distribution

Xen-netback @ dom0
Xen-icmp @ dom1
Dom0 : IDD
Dom1 : CPU 60%+ ping recv
Dom2 : CPU 100%

Latency (ms)

Cumulative distribution

Xen-netback @ dom0
Xen-icmp @ dom1
Dom0 : IDD
Dom1 : CPU 80%+ ping recv
Dom2 : CPU 100%
Enhanced Interrupt Latency: @ Driver Domain

- **Vcpu dispatch latency**
  - From intr. handler @ Xen to hard irq handler @ IDD
  - Fixed priority dom0 vcpu
  - Virtual FIQ
  - No latency is observed!

- **Intra-VM latency**
  - From ISR to netback @ IDD
  - No virt. intr. preemption (dom0 highest prio.)

- Among 13M intr.,
  - 56K are caught where virt intr preemption happened
  - 8.5M preemption occurred with FIQ optimization
Enhanced end-user Latency: overall result

- Over 1 million ping tests,
  - Fixed priority make the driver domain to run without additional latency (from inter-VM scheduling)
    - Largely reduces overall latency
  - 99% interrupts are handled within 1ms
Conclusion and Possible Future work

• We analyzed I/O latency in Xen-ARM virtual machine
  – Throughout the interrupt handling path in split driver model

• Two main reasons for long latency
  – Limitation of ‘BOOST’ in the Xen-ARM’s Credit scheduler
    • Not-boosted vcpu
    • Multi-boost
  – Driver domain’s virtualized interrupt handling
    • Virtual interrupt preemption (aka. lock-holder preemption)

• Achieve under millisecond latency for 99% network packet interrupts
  – DRIVER_BOOST; the highest priority for driver domain
  – Modify scheduler in order for the driver domain not to be scheduled out while virtual interrupts are disabled
  – Further optimizations (incl. virtual FIQ mode, softirq-awareness)

• Possible future work
  – Multi-core consideration/extension (core allocation, etc.)
    • Other scheduler integration
  – Tight latency guarantee for real-time guest OS
    • Rest 1% holds the key
Thanks for your attention

Credits to OSvirtual @ oslab, KU
Appendix.
Native comparison

- Comparison with native system – no cpu workload
  - Slightly reduced handling latency, largely reduced max.

<table>
<thead>
<tr>
<th></th>
<th>Native</th>
<th>Orig. dom0</th>
<th>Orig. domU</th>
<th>New sched. Dom0</th>
<th>New sched. DomU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>375</td>
<td>459</td>
<td>575</td>
<td>444</td>
<td>584</td>
</tr>
<tr>
<td>Avg</td>
<td>532.52</td>
<td>821.48</td>
<td>912.42</td>
<td>576.54</td>
<td>736.06</td>
</tr>
<tr>
<td>Max</td>
<td>107456</td>
<td>100782</td>
<td>100964</td>
<td>1883</td>
<td>2208</td>
</tr>
<tr>
<td>Stdev</td>
<td>1792.34</td>
<td>4656.26</td>
<td>4009.78</td>
<td>41.84</td>
<td>45.95</td>
</tr>
</tbody>
</table>

![Cumulative distribution graph](http://os.korea.ac.kr)
Appendix. Fairness?

- is still good, and achieves high utilization

CPU burning jobs' utilization

* Setting
Dom1: 20, 40, 60, 80, 100% CPU load
  + ping receiver
Dom2: 100% CPU load
Note that the credit is work-conserving

Normalized throughput

\[
\text{Normalized throughput} = \left( \frac{\text{Measured throughput}}{\text{Ideal throughput}} \right)
\]
Appendix. Latency in multi-OS env.

- 3 domain cases with differentiated service
  - added RT_BOOST priority, between DRIVER_BOOST and BOOST
  - Dom1 assuming RT
  - Dom2 and Dom3 for GP

Credit's latency dist. 3 domains for 10K ping tests (interval 10~40ms)

Enhanced latency dist. 3 doms. for 10K ping tests (interval 10~40ms)
Appendix.

Latency in multi-OS env.

- 3 domain cases with differentiated service
  - added RT_BOOST priority, between DRIVER_BOOST and BOOST
  - Dom1 assuming RT
  - Dom2 and Dom3 for GP