XenSummit Asia

November 2-3, 2011
Seoul, Korea

I/O Scalability in Xen

Kevin Tian kevin.tian@intel.com

Eddie Dong eddie.dong@intel.com

Yang Zhang yang.zhang@intel.com

Sponsored by:

SAMSUNG & LIBERTAS JUSTITIA KOREA UNIVERSITY VERITAS & kt

Agenda

Overview of I/O Scalability Issues

- Excessive Interrupts Hurt
- I/O NUMA Challenge

Proposals

- Soft interrupt throttling in Xen
- Interrupt-Less NAPI (ILNAPI)
- Host I/O NUMA
- Guest I/O NUMA

Retrospect...

2009 Xen Summit (Eddie Dong, …)

Extending I/O Scalability in Xen

Covered topics

- VNIF: multiple TX/RX tasklets, notification frequency
- VT-d: vEOI optimization, vIntr delivery
- SR-IOV: adaptive interrupt coalescing (AIC)

Interrupt is the hotspot!

New Challenges Always Exist

Interrupt overhead is increasingly high

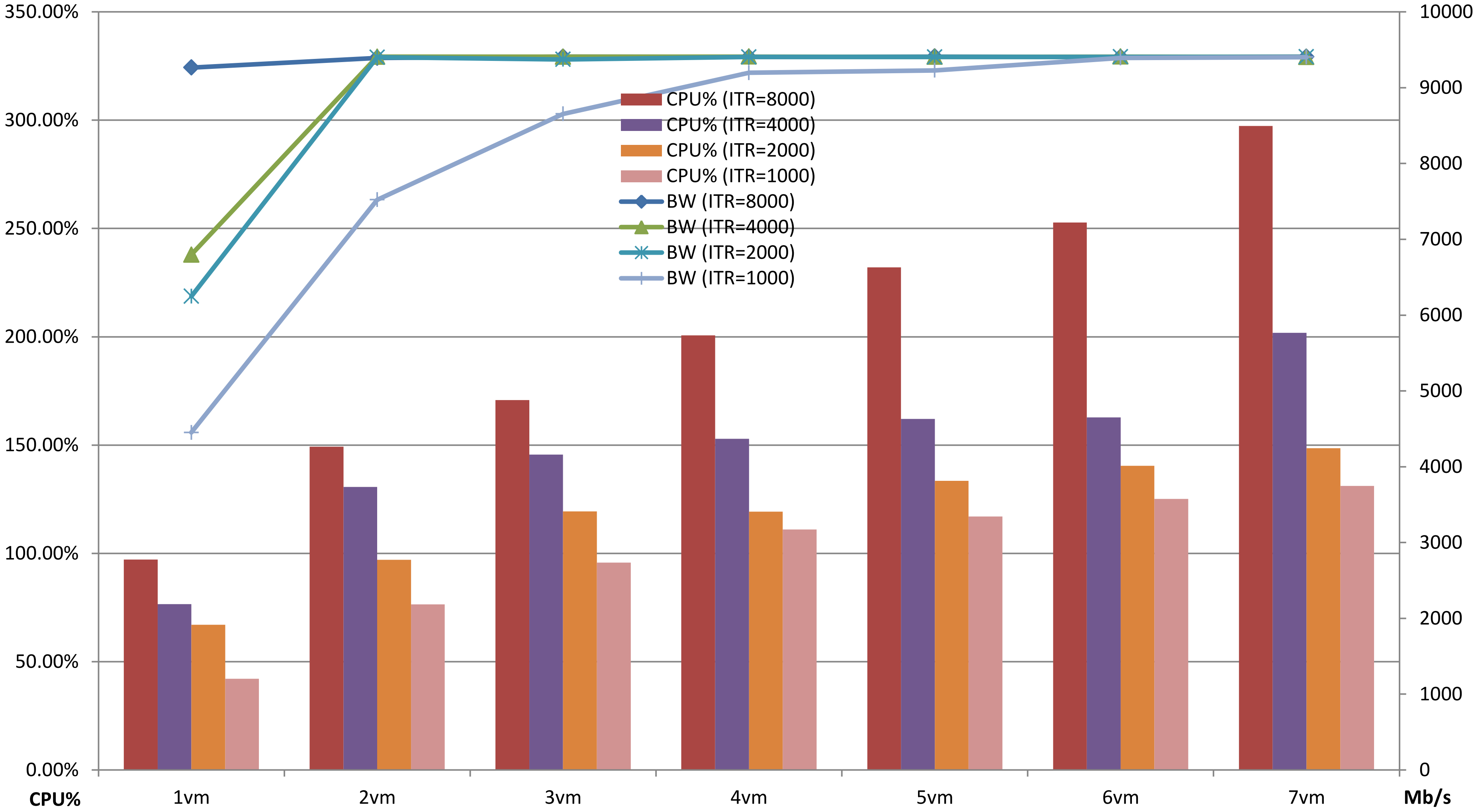
- One 10G Niantic NIC may incur 512k intr/s
  - 64 (VFs + PF) x 8000 intr/s
  - Similar for dom0 when multiple queues are used
- 40G NIC is coming

Prevalent NUMA architecture (even on 2-node low end server)

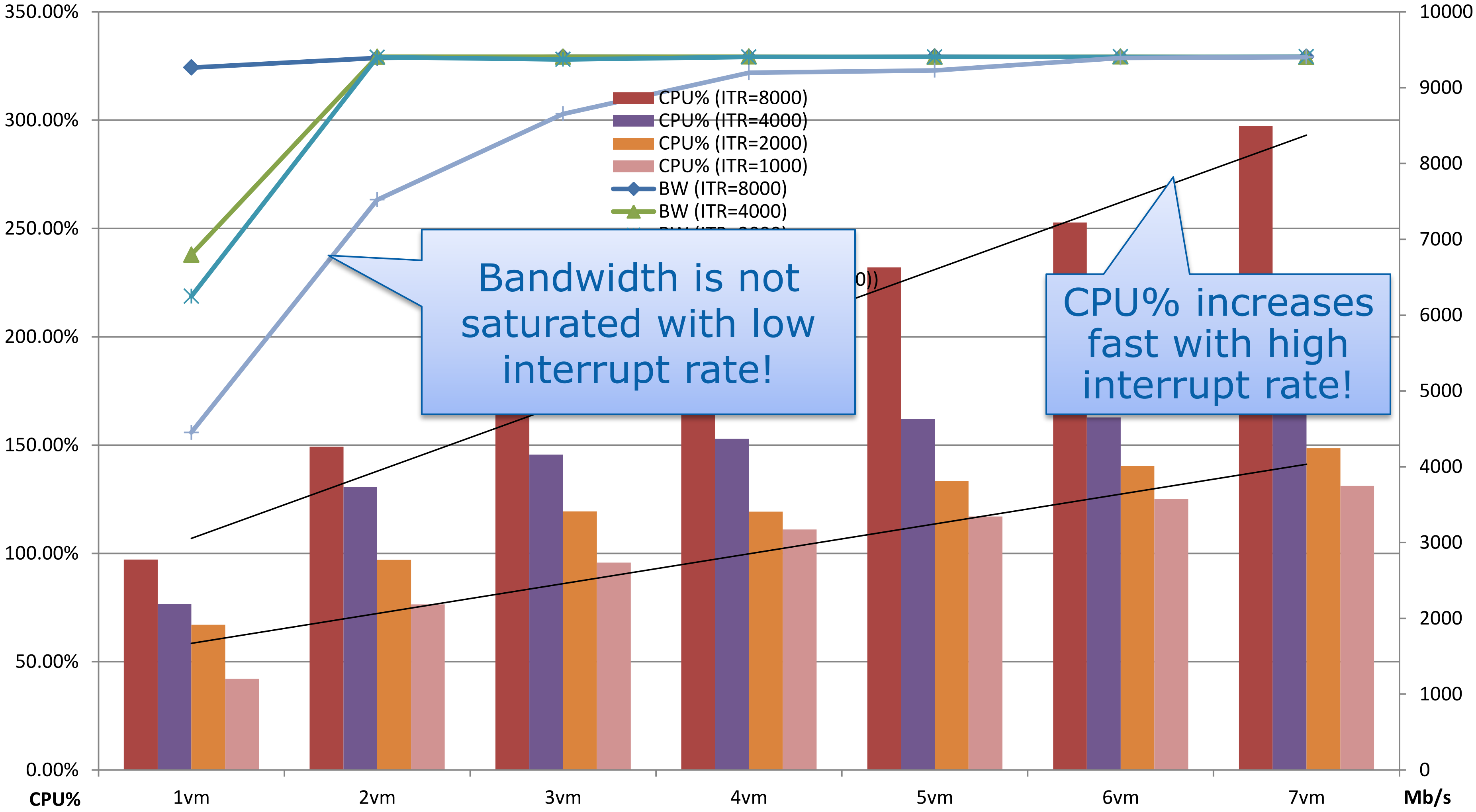
- The DMA distance to memory node matters (I/O NUMA)
- w/o I/O NUMA awareness, DMA accesses may be suboptimal

Need breakthrough in software architecture

Excessive Interrupts Hurt! (SR-IOV Rx Netperf)



Excessive Interrupts Hurt!



Bandwidth is not saturated with low interrupt rate!

CPU% increases fast with high interrupt rate!

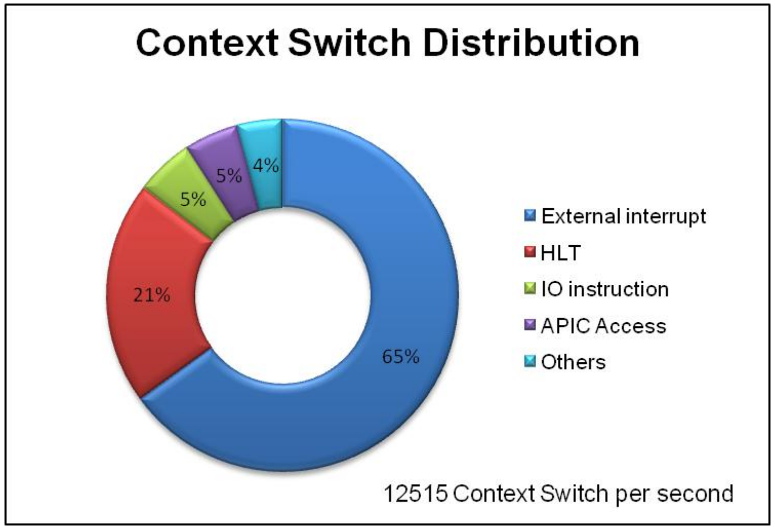
Excessive Interrupts Hurt! (Cont.)

Excessive VM-exits (7vm as example)

External Interrupts	35k/s
APIC Access	49k/s
Interrupt Window	7k/s

Excessive context switches

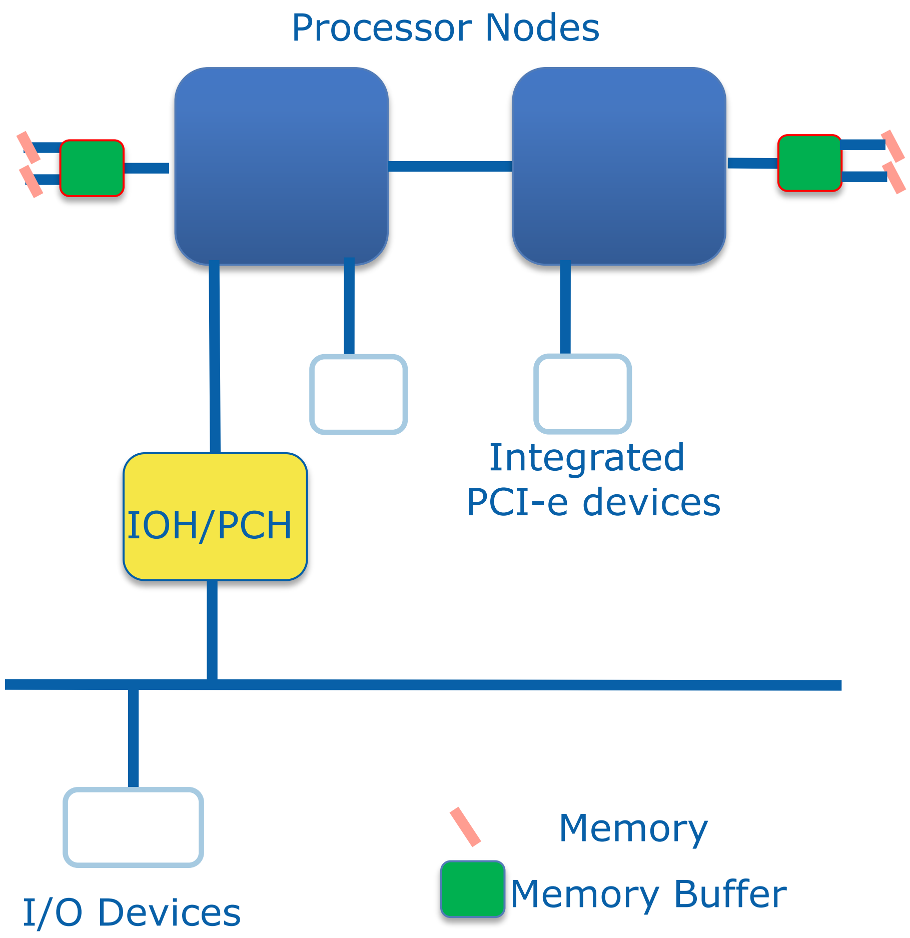
- “Tackling the Management Challenges of Server Consolidation on Multi-core System”, Hui Lv, Xen Summit 2011 SC



Excessive ISR/softirq overhead both in Xen and guest

Similar impact for dom0 using multi-queue NIC

NUMA Status in Xen



Host CPU/Memory NUMA

- Administrable based on capacity plan

Guest CPU/Memory NUMA

- Not supported
- But extensively discussed

Lack of manageability for

- Host I/O NUMA
- Guest I/O NUMA

NUMA Related Structures

An integral combo for CPU, memory and I/O devices

- System Resource Affinity Table (SRAT)
  - **Associates CPUs and memory ranges, with proximity domain**
- System Locality Distance Table (SLIT)
  - **Distance among proximity domains**
- _PXM (Proximity) object
  - **Standard way to describe proximity info for I/O devices**

Solely acquiring _PXM info of I/O devices is not enough to construct I/O NUMA knowledge!

Host I/O NUMA Issues

No host I/O NUMA awareness in Dom0

- Dom0 owns the majority of I/O devices
- Dom0 memory is first allocated by skipping DMA zone
- DMA memory is reallocated for continuity later
- Above allocations are made within node_affinity mask round-robin
  - **No consideration on actual I/O NUMA topology**

Complex and confusing if dom0 handles host I/O NUMA itself

- Implicates physical CPU/Memory awareness in dom0 too
  - **Virtual NUMA vs. Host NUMA?**

Xen however has no knowledge of _PXM()

Guest I/O NUMA Issues

Guest needs I/O NUMA awareness to handle assigned devices

- Guest NUMA is the premise

Guest NUMA is not upstream yet!

- Extensive talks in previous Xen summits
  - "VM Memory Allocation Schemes and PV NUMA Guests", Dulloor Rao
  - "Xen Guest NUMA: General Enabling Part", Jun Nakajima
- Already extensive discussions and works…
- Now time to push into upstream!

No I/O NUMA information exposed to guest

Lack of I/O NUMA awareness in device assignment process

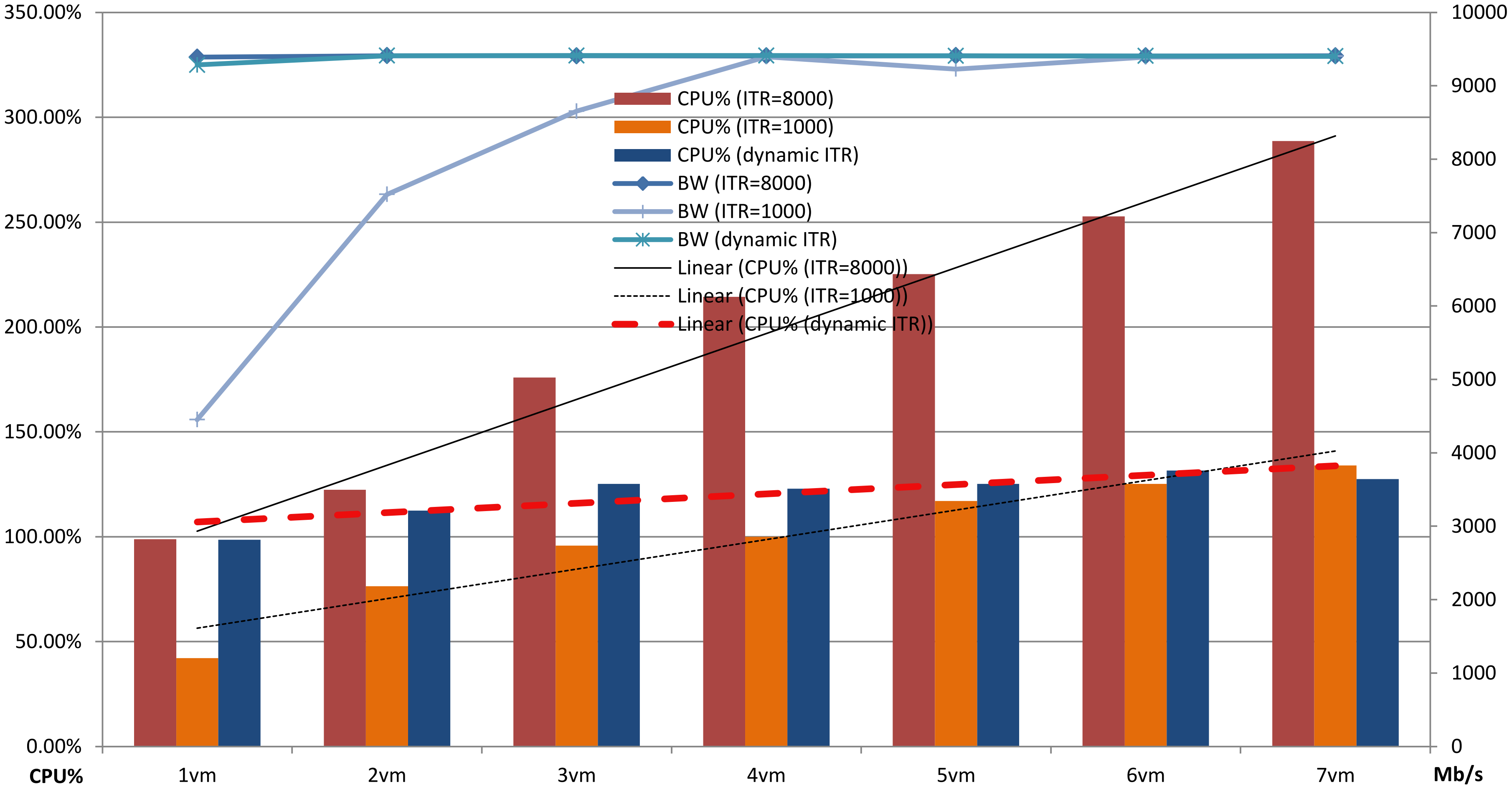
Proposals

Per-interrupt overhead has been studied extensively!

Now we want to reduce the interrupt number!

The Effect of Dynamic Interrupt Rate

A manual tweak on ITR based on VM number (8000 / vm_num)



Software Interrupt Throttling in Xen

Throttle virtual interrupts based on administrative policies

- Based on shared resources (e.g. bandwidth/VM_number)
- Based on priority and SLAs
- Apply to both PV and HVM guests

Fewer virtual interrupts reduces guest ISR/softirq overhead

It may further throttle physical interrupts too!

- If the device doesn't trigger a new interrupt when an earlier request is still pending

Interrupt-Less NAPI (ILNAPI)

NAPI itself doesn't eliminate interrupts

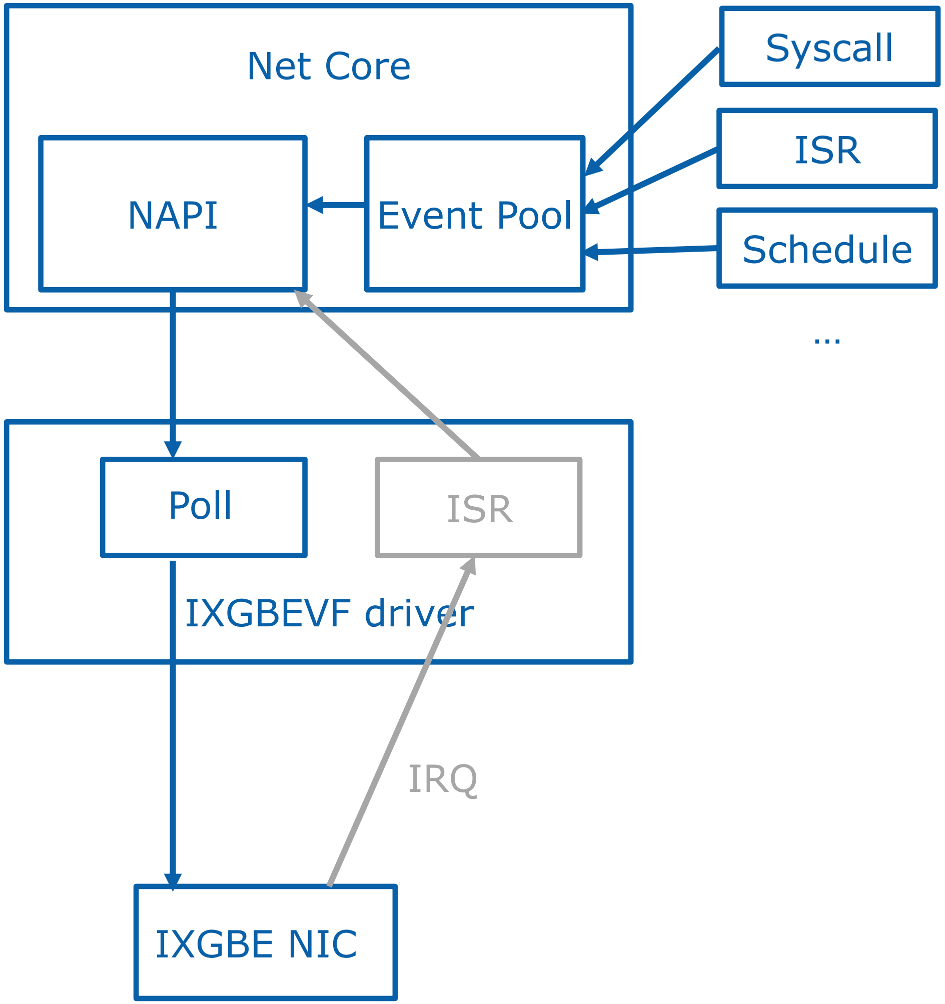
- NAPI logic is scheduled by rx interrupt handler
  - **Mask interrupt when NAPI is scheduled**
  - **Unmask interrupt when NAPI completes current poll**

What about scheduling NAPI w/o interrupts?

- If we can piggyback NAPI schedule on other events…
  - **System calls, other interrupts, scheduling, …**
- Internal NAPI schedule overhead is much less than a heavy device->Xen->VM interrupt path

Yes, that's … "Interrupt-Less NAPI (ILNAPI)"

Interrupt-Less NAPI (Cont.)



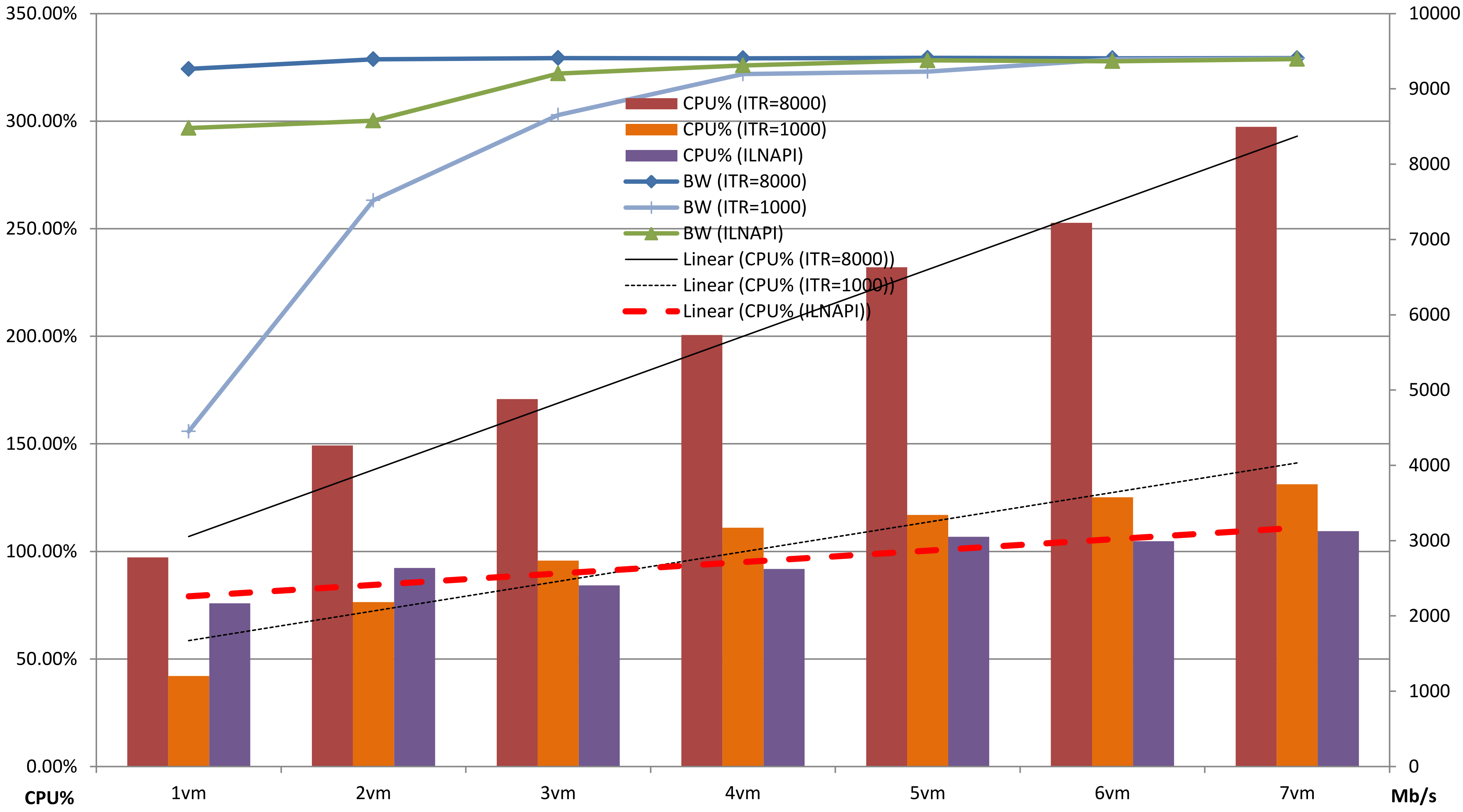
ILNAPI_HIGH watermark:

- When there're too many notifications within the guest
- Serve as the high watermark for NAPI schedule frequency

ILNAPI_LOW watermark:

- Activated when there're insufficient notifications
- Serve as the low water mark to ensure a reasonable traffic
- May move back to interrupt-driven manner

Interrupt-Less NAPI (Cont.)



Interrupt-Less NAPI (Cont.)

Watermarks can be adaptively chosen by the driver

- Based on bandwidth/buffer estimation

Or an enlightened scheme:

- Xen may provide guidance through shared buffer
  - Resource utilization (e.g. VM number)
  - Administrative policies
  - SLA requirements
- ILNAPI can be turned on/off dynamically under Xen’s control
  - E.g. in case where latency is much concerned

Proposals

We need close the Xen architecture gaps for both host I/O NUMA and guest I/O NUMA!

Host I/O NUMA

Give Xen full NUMA information:

- Xen already sees SRAT/SLIT
- New hypercall to convey I/O proximity info (_PXM) from Dom0
    - **Xen need extend _PXM to all child devices**
- Extend DMA reallocation hypercall to carry device ID
    - **May need Xen version for set_dev_node**
- Xen reallocates DMA memory based on proximity info

CPU access in dom0 remains NUMA-unaware…

- E.g. the communication between backend/frontend driver

Guest I/O NUMA

Okay, let’s help guest NUMA support in Xen! ☺

IOMMU may also spans nodes

- ACPI defines Remapping Hardware Status Affinity (RHSA)
  - **The association between IOMMU and proximity domain**
- Allocate remapping table based on RHSA and proximity domain info

Guest I/O NUMA (Cont.)

Make up guest I/O NUMA awareness

- Construct _PXM method for assigned devices in DM
  - Based on guest NUMA info (SRAT/SLIT)
- Extend control panel to favor I/O NUMA
  - Assign devices which are in same proximity domain as specified nodes of the guest
  - Or, affine guest to the node where assigned device is affined
  - The policy for SR-IOV may be more constrained
    - E.g. all guests sharing same SR-IOV device run on same node
  - Warn user when optimal placement can't be assured

Summary

I/O scalability is always challenging every time when we re-examine it! ☺

Excessive interrupts hurt I/O scalability, but there’re some means both in Xen and in guest to mitigate it!

CPU/Memory NUMA has been well managed in Xen, but I/O NUMA awareness is still not in place!

Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

All dates provided are subject to change without notice.

Intel is a trademark of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

intel®

Leap ahead™