The Price of Safety: Evaluating IOMMU Performance

Preliminary Results

Muli Ben-Yehuda

muli@il.ibm.com

IBM Haifa Research Lab

Table of Contents

IOMMUs

- IOMMU - IO Memory Management Unit
- Think MMU for IO devices - separate address spaces!
- IOMMUs enable virtual machines direct hardware access for PV and FV guests
- ... while protecting the rest of the system from mis-behaving or malicious virtual machines
- But at what cost?

Setting the Stage

- Full results to appear at OLS ’07
- Calgary (PCI-X) and Calgary (PCI-e) IOMMUs
- Linux and Xen implementations
- ... with some corroborating evidence from the DART IOMMU on PPC JS21 blades and the IBM Research Hypervisor
- Joint work with Jimi Xenidis, Michal Ostrowski, Karl Rister, Alexis Bruemmer and Leendert Van Doorn
- Utilizing IOMMUs for Virtualization in Linux and Xen, by by M. Ben-Yehuda, J. Mason, O. Krieger, J. Xenidis, L. Van Doorn, A. Mallick, J. Nakajima, and E. Wahlig, OLS ’06

On Comparisons

- What are we comparing against?
  - *not* emulation
  - *not* virtual IO (frontend / backend drivers)
- Direct hardware access without an IOMMU compared with direct hardware access with an IOMMU on the IO path

Lies, Damn Lies and Benchmarks

- These are not official benchmarks in any way shape or form!
- netperf for network performance
- ffsb (`http://sourceforge.net/projects/ffsb/`) for block IO
- Ran each set of tests on baremetal, in Xen dom0 and in Xen domU
- Two sets of runs, with and without an IOMMU
- In both the dom0 and domU cases dom0 is the one driving the IOMMU, to neutralize dom0 scheduling oddities

Prelim. Numbers - Network Baremetal

test	off throughput	off cpu	on tput	on cpu	% tput diff	% cpu diff
64	358.24	12.52/100	345.28	12.52/100	-3.62	0.00
128	632.40	12.61/100	599.77	12.52/100	-5.16	0.71
256	941.34	11.01/100	940.05	12.44/100	-0.14	-12.99
512	941.21	7.85/100	941.29	9.44/100	0.01	-20.25
1024	941.31	5.07/100	941.25	8.04/100	-0.01	-58.58
1460	941.18	6.16/100	941.15	6.84/100	-0.00	-11.04
1480	941.11	6.10/100	941.34	6.67/100	0.02	-9.34
2048	941.34	5.69/100	941.21	7.20/100	-0.01	-26.54
4096	941.17	5.15/100	941.30	6.62/100	0.01	-28.54
8192	941.34	5.10/100	941.20	6.28/100	-0.01	-23.14
16384	941.14	4.90/100	941.34	6.29/100	0.02	-28.37
32768	941.17	4.89/100	941.19	6.25/100	0.00	-27.81
65536	941.21	4.84/100	941.34	6.16/100	0.01	-27.27

Prelim. Numbers - Network dom0

test	off throughput	off cpu	on tput	on cpu	% tput diff	% cpu diff
64	178.23	100.61/800	151.86	105.05/800	-14.80	-4.41
128	282.69	101.54/800	235.07	104.42/800	-16.85	-2.84
256	456.53	100.59/800	353.54	104.33/800	-22.56	-3.72
512	899.39	96.58/800	506.72	104.15/800	-43.66	-7.84
1024	937.29	71.97/800	915.02	101.94/800	-2.38	-41.64
1460	940.13	63.53/800	918.41	95.49/800	-2.31	-50.31
1480	940.02	62.48/800	916.46	93.35/800	-2.51	-49.41
2048	940.11	57.41/800	939.15	86.71/800	-0.10	-51.04
4096	940.06	51.06/800	940.06	79.31/800	0.00	-55.33
8192	940.06	48.03/800	940.03	75.37/800	-0.00	-56.92
16384	940.11	46.20/800	940.05	73.34/800	-0.01	-58.74
32768	940.01	45.58/800	940.03	72.33/800	0.00	-58.69
65536	940.11	45.07/800	940.05	71.68/800	-0.01	-59.04

Prelim. Numbers - Network domU

test	off throughput	off cpu	on tput	on cpu	% tput diff	% cpu diff
64	181.59	47.21/800	154.83	54.19/800	-14.74	-14.79
128	309.91	63.59/800	262.81	74.57/800	-15.20	-17.27
256	483.26	82.09/800	438.64	94.09/800	-9.23	-14.62
512	937.48	59.88/800	935.60	84.31/800	-0.20	-40.80
1024	938.30	60.53/800	938.24	83.72/800	-0.01	-38.31
1460	940.00	58.25/800	939.99	81.60/800	-0.00	-40.09
1480	940.03	58.19/800	940.02	81.58/800	-0.00	-40.20
2048	940.01	58.31/800	939.94	81.72/800	-0.01	-40.15
4096	940.02	58.46/800	939.84	81.83/800	-0.02	-39.98
8192	940.04	58.61/800	939.95	81.77/800	-0.01	-39.52
16384	939.81	58.55/800	939.84	81.98/800	0.00	-40.02
32768	940.05	58.65/800	939.55	81.85/800	-0.05	-39.56
65536	940.00	58.58/800	938.74	81.71/800	-0.13	-39.48

The Straight-forward Implementation

- Map Linux DMA API calls to TCE (translation control entries - think MMU PTEs) map / unmap calls
- Straight-forward implementation
- With the best isolation properties! Only entries in active use are mapped - minimizes window of exposure
- Unfortunately, map / unmap hypercalls are expensive
- ... even on bare metal calling into the DMA API too many times hurts
- Xen multicalls don’t help

Pre-allocating the IO Address Space

- Map the entire guest address space in the IOMMU address space such that the guest pseudo-physical address that maps a given machine frame is equal to the DMA address that maps that machine frame
- Start-up cost but minimal runtime overhead
- Isolates the system from the guest
- But provides no protection inside the guest (guest is oblivious to the IOMMU)
- Precludes (or requires hypervisor involvement for) page flipping, ballooning and anything else that modifies guest P->M translations
- Size of IO address space may be limited - theoretical 4GB limit on Calgary

Allocate in Advance; Free When Done

- Don’t use the streaming DMA API (map / unmap)
- Use the persistent allocation DMA API (allocate / free)
- Goes against standard Linux driver practice
- ... DMA-API is really designed for platforms with limited number of DMA mappings
- Alternative is to cache map / unmap calls in the DMA-API itself and save the hypervisor crossing - definitely beneficial for hypervisor scenario but not sure about baremetal
- Another alternative is to allocate and free in large batches, rather than on a per-buffer basis - add dma_map_multi and dma_unmap_multi and teach drivers and subsystems to batch their DMA mappings

Misc. Optimizations

- Deferred cache flushing due to architectural constraints
- Never free! The mapping may exist until it gets reused, but if the driver is well-behaved and the mapping does not map anyone else's page... who cares?
- Grant table integration: when using PV drivers map and unmap intelligently from the grant table ops rather than from the DMA API. Only applicable for driver domains, not for direct hardware access domains

Conclusions and Future Work

- IOMMUs are useful and necessary
- ... but they have non-negligible costs at the moment - up to 60% more CPU utilization
- ... which we know how to fix!
- What about Intel VT-d and AMD IOMMU?
- Once we get rid of the software inefficiencies, how do we build better IOMMUs?
- `http://xenbits.xensource.com/ext/xen-iommu.hg`
- `http://xenbits.xensource.com/ext/linux-iommu.hg`