

KVM Limits Arbitrary or Architectural?

Ryan Harper – Open Virtualization – ryanh@us.ibm.com Karl Rister – Performance – krister@us.ibm.com IBM Linux Technology Center

June 11, 2008



Linux is a registered trademark of Linus Torvalds.

Outline

- Goals
- Virtual Resources
- IO Subsystem
- Future Work



Goals

- Determine for any resources (physical or virtual) whether the limit is arbitrary, or limited in some way by the architecture.
- X86-centric view
- Examine current limits
- Extend arbitrary limits
- Identify true architectural limits which inhibit scalability



Testing Platform

IBM System x3950 M2



IBM System Storage DS3400



- 2 x3950 M2 servers
 - 2 node configuration
- 64G per node
 - 128G total
- 4 Intel Quad Core
 - 4 sockets per node
 - 32 cores total
- 2 Qlogic 4Gb Fiber
- 2 DS3400 trays
 - 24 * 300G 15k SAS



Virtual Resources

- VCPUs per guest
- Guest max memory
- System devices
 - Emulated or Paravirtual



VCPU Limitations

- Current limit at 16 VCPUs
 - Hard-coded array sizes
 - kernel/include/asm/kvm_host.h (KVM_MAX_VCPUS)
 - libkvm/kvm-common.h (MAX_VCPUS)
 - bios/rombios.h (MAX_CPUS)
- Change limit to 255
 - Max for x86-64 in Linux
- Guest now boots up to 32-way
 - X86 defconfig sets NR_CPUS to 32
- Recompile kernel with NR_CPUS=64
 - Guest boots as 64-way
 - Oops on >64 in cpu_to_node()
 - X86 defconfig sets NODE_SHIFT to 6 limiting CPUS to 64.



VCPU Limitations cont.

- Changing NODES_SHIFT to 8 supports up to 256 CPUs
 - Guests now boot up to 128-way
 - > 128way exits with:

bios_table_end_addr_overflow!

- MADT, MPTABLE, and SMBIOS tables generate percpu data
 - While some portion of the tables must be in low memory, the majority of the data can be relocated to high memory
 - Update BIOS to locate per-cpu data in ACPI_DATA e820 region
- Guest now boots 255-way -- partially
 - Guest wedges after starting up userspace



VCPU Limitations Cont.

| <pre>top - 14:07:43 up 38 days, 21:53, 3 users, load average: 33.66, 33.43, 49.58 top - 14:07:43 up 38 days, 21:53, 3 users, load average: 33.66, 33.43, 49.58 Tasks: 115 total, 2 running, 113 sleeping, 0 stopped, 0 zombie Cpu(s): 11.9%us, 7.2%sy, 0.0%ni, 80.7%id, 0.0%wa, 0.0%hi, 0.1%si, 0.0%st Mem: algo344k total, 4469616k used, 3720728k free, 2427740k cached PTD USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND 3619 root 20 0 7290m 363m 19m S 440 4.5 60:35.53 qemu-system=x86 anthony@npt2: -//harper/git/build/kvm-userspace/gemu/x86_64-softmmu region 4: 0x0000c000 processor : 191 vendor id : AuthenticAMD cpu family : 6 model name : QEMU Virtual CPU version 0.9.1 stepping : 3 cpu MHz : 1995.120 cache size : 512 K8 fpu e : yes fpu exception : yes cpuid level : 2 wp : yes fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx lm pni bogomips : 3990.48 TLB size : 1024 4K pages clflush size : 64 address sizes : 40 bits physical, 48 bits virtual power management: rharper@lewis:-\$ grep ^processor /proc/cpuinfo wc -l 192 rharper@lewis:-\$] </pre> | | |
|---|--|---|
| <pre>Tasks: 115 total, 2 running, 113 sleeping, 0 stopped, 0 zombie Cpu(5): 11.9%us, 7.2%sy, 0.0%ni, 80.7%i(0.0%wa, 0.0%hi, 0.1%si, 0.0%st Mem: 8190344k total, 4469616k used, 3720728k free, 207240k buffers Swap: 6385796k total, 0k used, 6385796k free, 2437740k cached PID USER PR NI VIRT RES SHR 5 %CPU 9MEM TIME+ COMMAND 3619 root 20 0 7290m 363m 19m S 40 4.5 60:35.53 gemu-system-x86 anthony@npt2: ~/harper/git/build/kvm-userspace/gemu/x86_64+softmmuX</pre> | | × |
| <pre>3619 root 20 0 7290m 363m 19m S 40 4.5 60;35.53 gemu-system-x86 anthony@npt2: ~//harper/git/build/kvm-userspace/gemu/x86_64=softmmu</pre> | Tasks: 115 total, 2 running, 113 sleeping, 0 stopped, 0 zombie Cpu(s): 11.9%us, 7.2%sy, 0.0%ni, 80.7%id, 0.0%wa, 0.0%hi, 0.1%si, 0.0%st Mem: 8190344k total, 4469616k used, 3720728k free, 207240k buffers Swap: 6385796k total, 0k used, 6385796k free, 2437740k cached | Î |
| region 4: 0x0000c000 processor : 191 vendor_id : AuthenticAMD cpu family : 6 model : 2 model name : QEMU Virtual CPU version 0.9.1 stepping : 3 cpu MHz : 1995.120 cache size : 512 KB fpu : yes fpu = : yes fpu exception : yes cpuid level : 2 wp : yes flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx lm pni bogomips : 3990.48 TLB size : 1024 4K pages clflush size : 64 cache_alignment : 64 address sizes : 40 bits physical, 48 bits virtual power management: rharper@lewis:~\$ grep ^processor /proc/cpuinfo wc -l 192 | | |
| <pre>processor : 191 vendor_id : AuthenticAMD cpu family : 6 model : 2 model name : QEMU Virtual CPU version 0.9.1 stepping : 3 cpu MHz : 1995.120 cache size : 512 KB fpu : yes fpu_exception : yes cpuid level : 2 wp : yes flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx lm pni bogomips : 3990.48 TLB size : 1024 4K pages clflush size : 64 cache_alignment : 64 address sizes : 40 bits physical, 48 bits virtual power management: rharper@lewis:~\$ grep ^processor /proc/cpuinfo wc -l 192</pre> | | |
| <pre>vendor_id : AuthenticAMD cpu family : 6 model : 2 model name : QEMU Virtual CPU version 0.9.1 stepping : 3 cpu MHz : 1995.120 cache size : 512 KB fpu : yes fpu : yes fpu : yes cpuid level : 2 wp : yes flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx lm pni bogomips : 3990.48 TLB size : 1024 4K pages clflush size : 64 cache_alignment : 64 address sizes : 40 bits physical, 48 bits virtual power management: rharper@lewis:~\$ grep ^processor /proc/cpuinfo wc -l 192</pre> | region 4: 0x0000c000 | |
| | <pre>vendor_id : AuthenticAMD cpu family : 6 model : 2 model name : QEMU Virtual CPU version 0.9.1 stepping : 3 cpu MHz : 1995.120 cache size : 512 KB fpu : yes fpu_exception : yes cpuid level : 2 wp : yes flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx lm pni bogomips : 3990.48 TLB size : 1024 4K pages clflush size : 64 cache_alignment : 64 address sizes : 40 bits physical, 48 bits virtual power management: rharper@lewis:~\$ grep ^processor /proc/cpuinfo wc -l 192</pre> | |

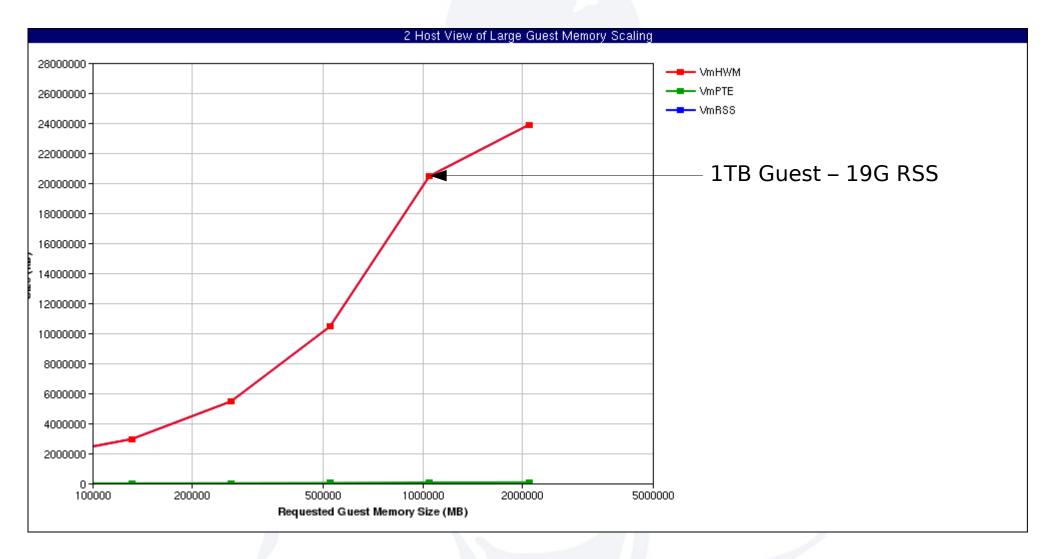


Memory Limitations

- Host VA hardware limits
 - X86 64-bit processors support up to 48-bits
- QEMU imposed limit of 4TB(42 bits) for 64-bit builds
- Bochs BIOS e820 table writes 48-bit values
- KVM pages in memory on-demand (get_user_pages())
- For large memory guests, (1-3%) of total guest memory will be consumed by OS frame table.
 - Booted 256G guest, ran test to consume all memory
 - Consumed 140G (128G of RAM, 12 of swap) and host swapper process started to OOM
 - Booted guest with 2TB of allocated memory
 - Linux detected about 1.4TB
 - Potentially a Host issue with very large swap (4TB)



Frame Table Consumption





PCI Device Limitations

- PCI Specification defines up to 32 slots
- Standard PC guests use 5 slots for typical guests
 - 1 disk, 1 nic, 1 vga, 1 isa, 1 pci host bridge
- Users requesting large disk and nic count support
- Two current approaches
 - Use multi-function devices
 - Add additional emulated PCI bridges
- Existing patch for virtio multi-function
 - Guest boots with 220 disks or nics
 - OSes aren't well tested with 220 PCI devices
- Additional PCI bridges patch
 - Extra bridges failed to work (IRQ delivery)



KVM IO Architecture

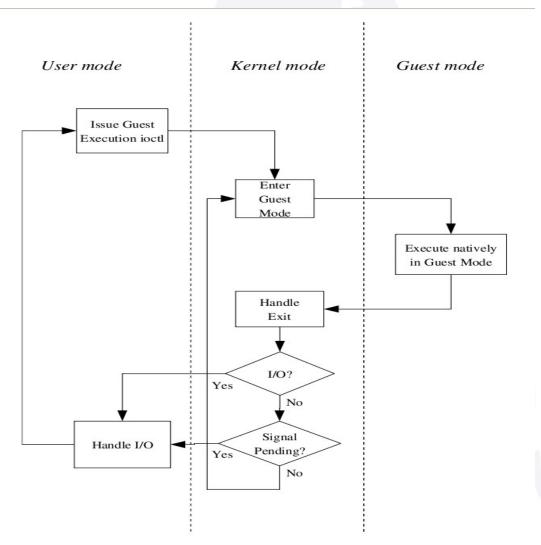


Figure 2: Guest Execution Loop

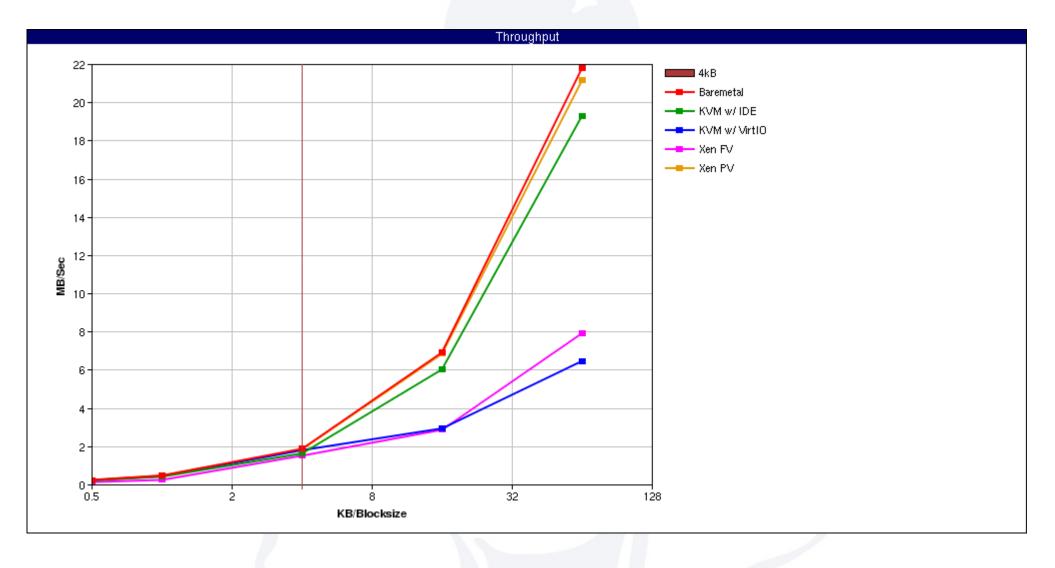


IO Microbenchmarks

- 50/50 Mix of read/write ops randomized
- Varying request sizes from 512b to 64k

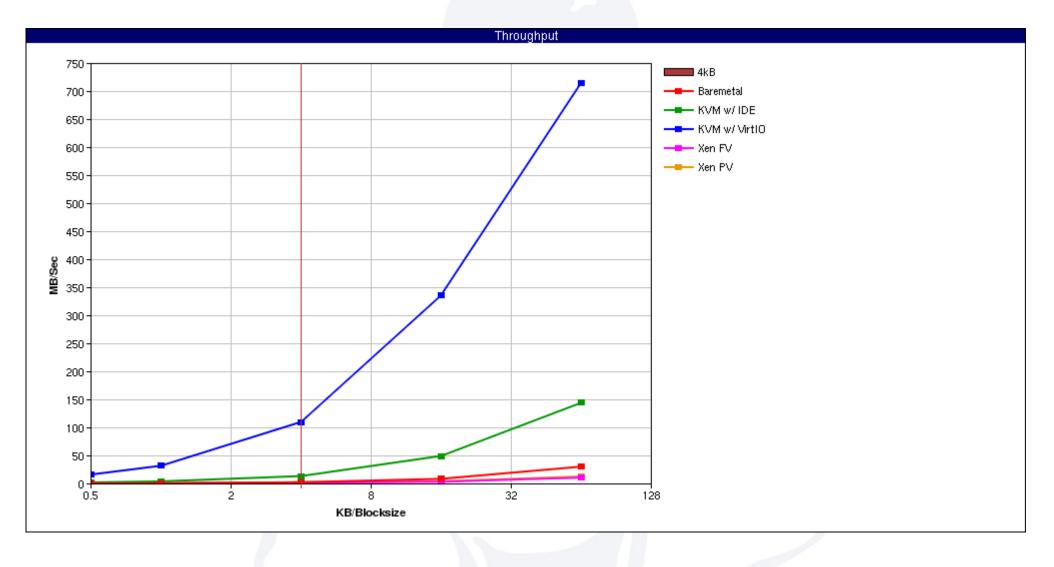


IO Throughput – Cache off



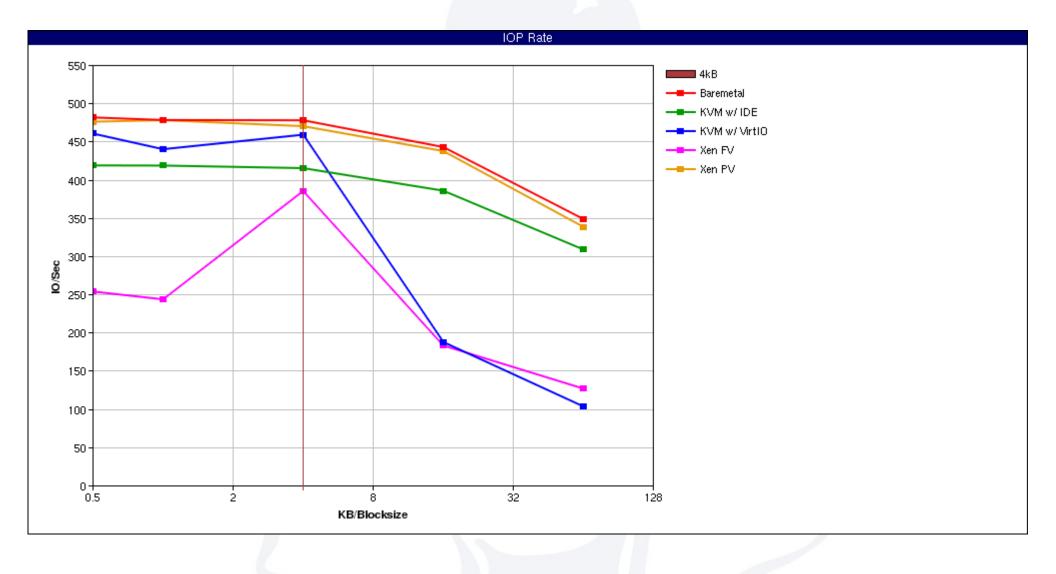


IO Throughput – Cache on



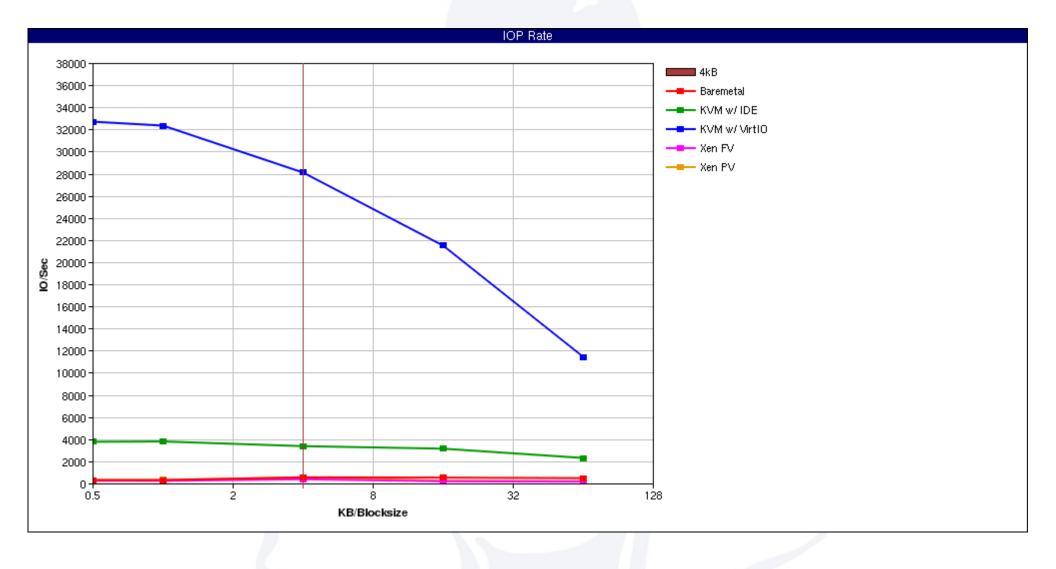


IO OP Rate – Cache off



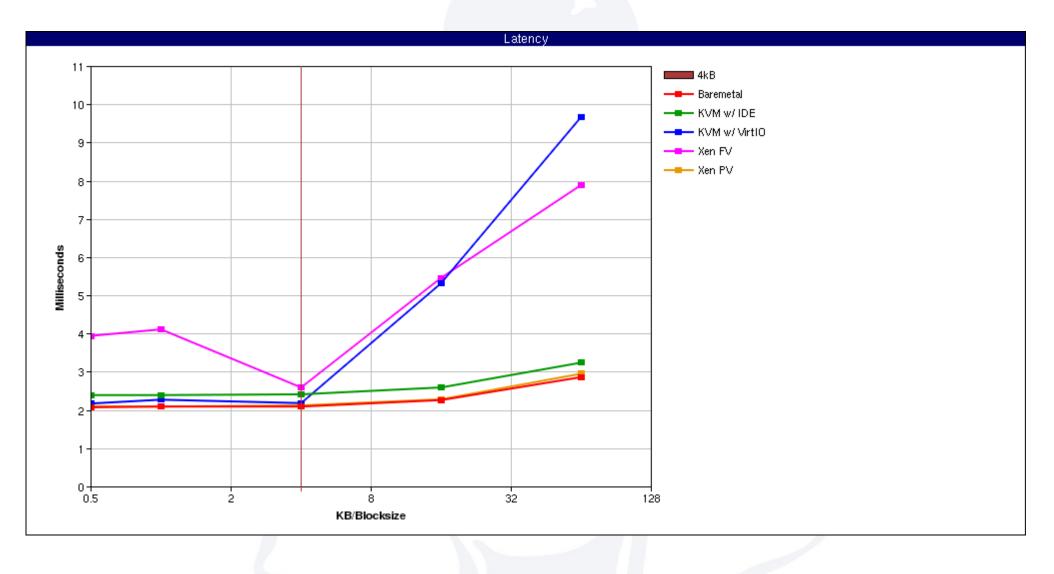


IO OP Rate – Cache on



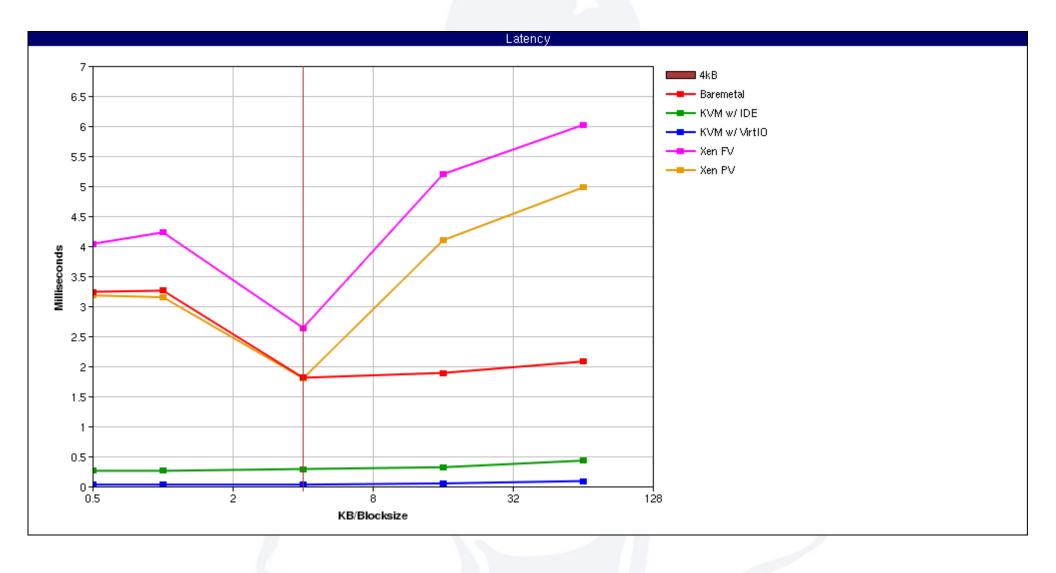


IO Latency – Cache off



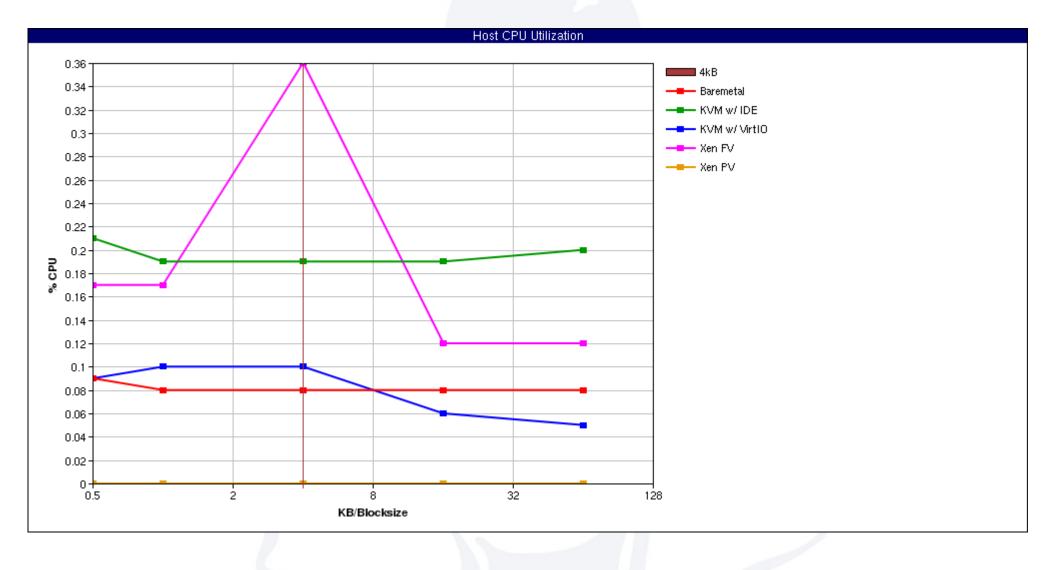


IO Latency – Cache on



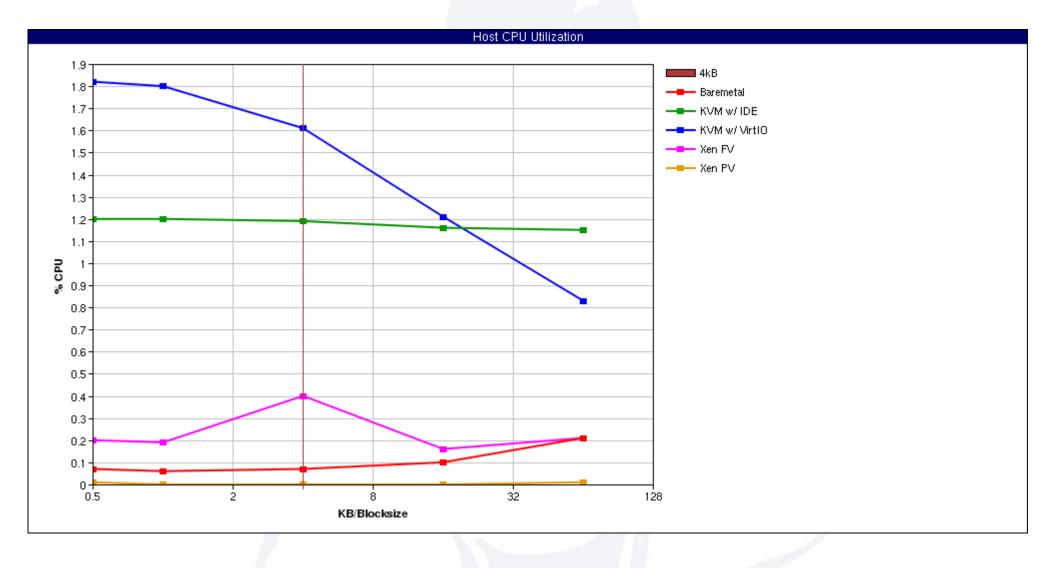


IO Host CPU Load – Cache off





IO Host CPU Load – Cache on





System Scaling

- Idle guest scaling
 - Mildly interesting and useful for finding scaling issues
 - Booted 500 KVM guests
 - ~200 responsive
 - CPU-bound on the Host
 - Oprofile points to time code (hrtimer/jiffies updates)
 - Lockstat output

| class | name | con-bounces | contentions | waittime-min | waittime-max | waittime-total.acq-bounces | acquisitions |
|--------------------|---------|-------------|-------------|--------------|--------------|----------------------------|--------------|
| xtime_lock | 7468921 | 7482457 | 3.47 | 22231.72 | 868397199.73 | 13417786.16 | 2093.8 |
| tty_ldisc_lock | 1217066 | 1477064 | 10.43 | 1042.04 | 2535049.05 | 2380528.81 | 469.62 |
| &sem->wait_lock | 713840 | 734493 | 5.18 | 534.51 | 745288.65 | 7499089.63 | 928.08 |
| &rq->rq_lock_key#2 | 152367 | 159743 | 4.63 | 1288.21 | 657369.39 | 1336917.18 | 1179.79 |
| &rq->rq_lock_key#1 | 126642 | 133424 | 4.63 | 1503.75 | 586686.21 | 1243939.17 | 1276.63 |
| &rq->rq_lock_key#2 | 121815 | 129158 | 5.22 | 1028.95 | 530599.21 | 1177768.17 | 1108.34 |
| &rq->rq_lock_key#1 | 121646 | 127047 | 3.77 | 1428.16 | 533693.92 | 1180284.17 | 1172.43 |
| &rq->rq_lock_key#1 | 117988 | 123783 | 3.59 | 1163.07 | 543945.53 | 1204265.17 | 1477.89 |



System Scaling cont.

- xtime_lock in high contention
 - Called from ktime_get_ts()
 - When KVM_CLOCK is enabled each vcpu_load() invokes kvm_write_guest_time() which calls ktime_get_ts().



Future Work

- Submit patches bumping KVM VCPU limit to 64
- KVM Tasklet patches
- Run System-wide Scaling with new KVM Tasklet patches
- Examine NUMA affects with cpusets/cgroup and page migration
- Run Large # of guest test w/o KVM_CLOCK
- Run System-wide scaling on large NPT/EPT systems when available.

