October 13, 2014

KVM I/O performance and end-to-end reliability

Nicholas Bellinger

Overview

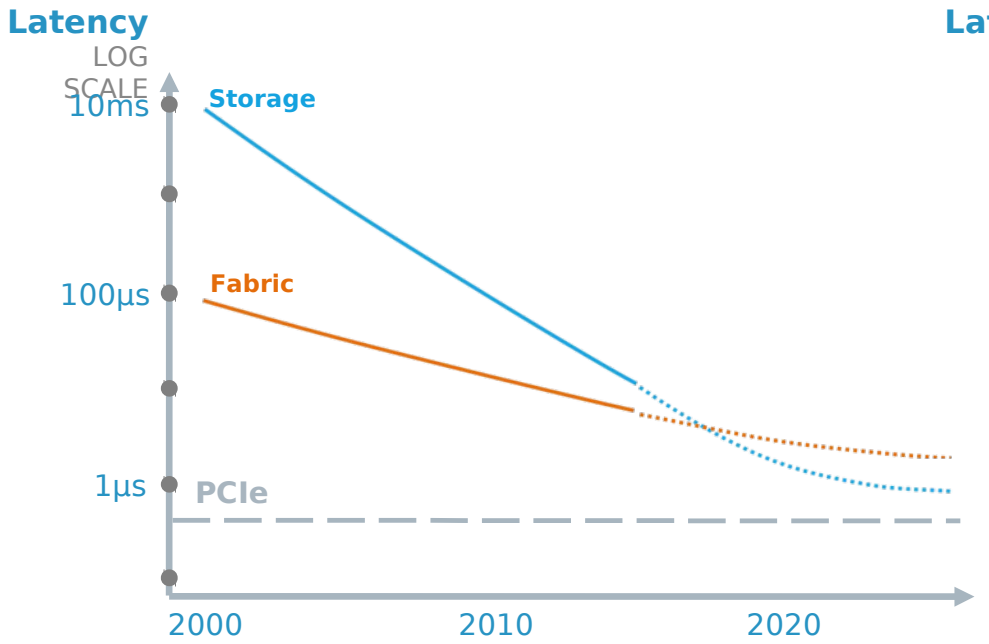
DATERA

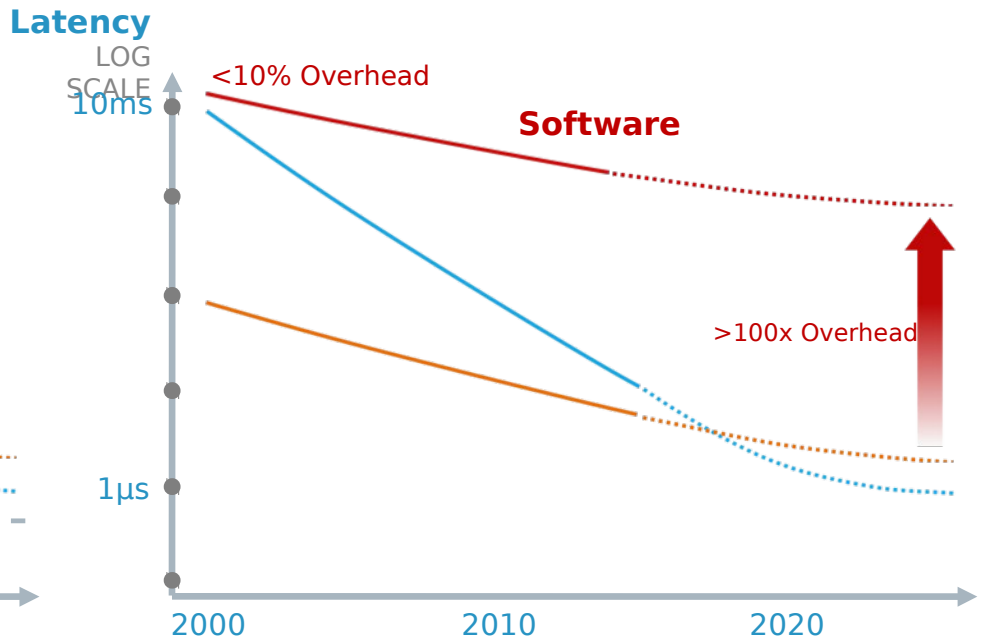
- **Background**
- **Past, present and future**
- **Big changes in Linux Block/SCSI (blk-mq + scsi-mq)**
- **Big changes in HW interface (NVMe-HI)**
- **T10 Data Integrity Field (DIF)**
- **What does it all mean to KVM..?**
- **Performance test configuration**
- **Performance results**
- **Performance summary**
- **Vhost-scsi TODO**
- **Linux I/O ecosystem update (Copy offload)**
- **Linux I/O ecosystem update (T10 DIF syscall interface)**

Background

- **virtio-scsi in QEMU userspace**
  - Originally performance limited by Big QEMU lock
  - Pre v3.17 scsi-mq guests performance also limited by legacy scsi_request_fn() locking overhead
- **virtio-blk-dataplane in QEMU userspace**
  - Multithreaded AIO + O_DIRECT context from host userspace
  - Posix thread per device, avoids Big QEMU lock
  - Supports Live Migration
- **vhost-scsi in KVM host kernel**
  - By-passes second level AIO + O_DIRECT overheads using LIO
  - No changes to guest virtio-scsi LLD
  - Direct passthrough of T10 DIF protection information from virto-scsi
  - Currently missing live migration support

Past, present and future





Opportunity
I/O acceleration is flattening the datacenter
The datacenter fabric becomes the new backplane

Challenge
Software attenuates the I/O acceleration
Software is the performance and management bottleneck

"History teaches us that when the data fabrics change, just about everything else in our industry changes." –Paul Maritz, CEO, Pivotal

Big changes to Linux Block/SCSI

DATERA

- **blk-mq**
  - Generational rewrite of block subsystem by Jens Axboe
  - Percpu software queues mapped to pre-allocated hardware queues
  - Smart NUMA allocation and placement
  - Has scaled up to 10M IOPs to a single null-blk device!
  - Merged in v3.13-rc1
- **scsi-mq**
  - Utilizes blk-mq to by-pass legacy scsi_request_fn() codepath
  - Legacy LLD performance with request_queue->queue_lock and struct Scsi_Host->host_lock overheads limited small block performance to ~250K per LUN with ramdisk
  - Now able to reach 1M IOPs per device to SCSI ramdisk LLD!
  - Merged in v3.17-rc1, thanks to Christoph Hellwig & Co.

Big changes to HW interface

DATERA

- **NVMe Host Interface specification**
  - Effort to standardize HW host interface, allowing for single OS driver to support all hardware out of the box.
  - Backed by Cisco, Dell, EMC, HGST, Intel, LSI, Micron, Netapp, Oracle, PMC-Sierra, Samsung, SanDisk, and Seagate.
- **New NVMe command set**
  - Required to implement commands is only **3** !
  - Optional to implement commands borrow from SCSI heritage, including WRITE_SAME, COMPARE_AND_WRITE, and eventually EXTENDED_COPY.
- **NVMe over Fabrics**
  - Future specification to map NVMe submission and completion queues to RDMA hardware queues.
  - LIO prototype for NVMe-RP dropping in 2015

T10-DIF End-to-end protection

DATERA

- **How..?**
  - Uses extra 8 bytes protection information per 512-4096 byte block
  - Depending upon DIF type, carries Block Guard (CRC), Reference Tag (LBA), and Application Tag (vendor specific area)
- **Why..?**
  - Allows individual software + hardware components to verify DIF metadata against original LBA + payload
  - Prevents misdirected WRITE data corruption, and silent data corruption on READs
  - Identify failures of individual faulty components
- **Who..?**
  - Supported by FC HBAs, (some) RDMA HCAs/NICs and SAS disks
  - Supported by LIO iSER, qla2xxx, and vhost-scsi target drivers
  - Optional to implement feature in NVMe specification

What does it all mean to KVM..?

DATERA

- **I/O stack in guest is no longer bottleneck**
  - blk-mq + scsi-mq is fastest I/O stack on the planet
  - Exposes more bottlenecks elsewhere in paravirtualized I/O stack
- **HW interface on bare-metal is no longer bottleneck**
  - NVMe host interface is designed to scale beyond flash to next generation storage class memory
- **The faster the pipe, the higher the error rates**
  - Undetectable error rates (silent data corruption) is a fact of life.
  - It is not if these errors occur, but when..
- **So what are the new bottlenecks for KVM..?**
  - First, let's see the I/O performance on current state of the art hardware and software..

Performance test configuration

- **Haswell-EP 2697-v3 (28/56 cores/threads)**
  - Grantley chipset, DDR4-2133 memory
  - Posted interrupts reduce APIC software emulation overhead
- **Radian Memory Systems (RMS-200)**
  - /dev/nvme0n1 namespace
  - 56 MSI-X interrupt vectors for single block_device on host
  - 8 GB capacity, combination of NV-RAM fronted SLC flash
- **Device Backends**
  - IBLOCK NVMe namespace
  - brd.ko ramdisks
  - rd_mcp (LIO ramdisk) with TYPE1 T10 Protection (DIF)

Test configuration (cont.)

- **Linux v3.17-rc5**
  - Same kernel on KVM guest + Host
- **QEMU**
  - V2.0.2 + vhost-scsi T10 DIF patches
- **KVM guest setup**
  - 16 vCPUs + 16 GB memory
  - Posted interrupts to reduce VMEXITs
  - PCLMULQDQ instruction offload for DIF generate + verify ops
- **FIO setup**
  - Iodepth=16 + numjobs=2x * $NUM_LUNS
  - Random 4k blocksize read/write
  - AIO + O_DIRECT from virtio guest.

Performance results, NVMe

- **Bare-metal nvme0n1**
  - 1x NVMe controller with 1x LUN: 700k IOPs @ 50 usec
- **virtio-blk-dataplane: nvme0n1**
  - 1x virtio-blk controller with 1x LUN: 135k IOPs @ 235 usec
  - 4x virtio-blk controller with 4x LUN: 350k IOPs @ 360 usec
- **vhost-scsi: nvme0n1**
  - 1x virtio-scsi controller with 1x LUN: 235k IOPs @ 145 usec
  - 4x virtio-scsi controller with 4x LUN: 715K IOPs @ 185 usec
- **KVM guest configuration**
  - Both virtio-blk + virtio-scsi using single virtio queue
  - Virtio-scsi enabled with scsi_mod.use_blk_mq=1
  - Explicit IRQ affinity of virtioX-request MSI-X vectors

Performance results, brd.ko

- **Bare-metal brd:**
  - 1x brd controller with 1x LUN: 680k IOPs @ 50 usec
- **virtio-blk-dataplane: /dev/ramX**
  - 1x virtio-blk controller with 1x LUN: 135k IOPs @ 235 usec
  - 4x virtio-blk controller with 4x LUN: 380 IOPs @ ~325 usec
- **vhost-scsi: /dev/ramX**
  - 1x virtio-scsi controller with 1x LUN: 225k IOPs @ 150 usec
  - 4x virtio-scsi controller with 4x LUN: 680K IOPs @ 185 usec
- **KVM guest configuration**
  - Both virtio-blk + virtio-scsi using single virtio queue
  - Virtio-scsi enabled with scsi_mod.use_blk_mq=1
  - Explicit IRQ affinity of virtioX-request MSI-X vectors

Performance results, T10-DIF

- **Bare-metal rd_mcp + DIF**
  - 1x loopback controller with 1x LUN: 350k IOPs @ 160 usec
- **virtio-blk-dataplane: N/A**
  - Currently no user-space syscalls for attaching T10 PI
- **vhost-scsi: rd_mcp + DIF**
  - 1x virtio-scsi controller with 1x LUN: 170k IOPs @ 185 usec
  - 4x virtio-scsi controller with 4x LUN: 620K IOPs @ 205 usec
- **KVM guest configuration**
  - Virtio-scsi using single virtio queue
  - Virtio-scsi enabled with scsi_mod.use_blk_mq=1
  - Explict IRQ affinity for virtioX-request MSI-X vectors
  - World's first end-to-end paravirtualized I/O stack!

Performance summary:

- **virtio-blk-dataplane:**
  - Currently limited per device by second-level O_DIRECT overheads on KVM host. Yes, O_DIRECT is really that expensive.
  - virtio-scsi-dataplane will see similar performance limitations due to same second level O_DIRECT overheads
  - Other bottlenecks in QEMU..?
- **vhost-scsi:**
  - vhost-scsi is **double** (715k vs. 350k) 4k random IOPs performance, at **half** (185 usec vs. 360 usec) latency to NVMe namespace
  - T10 DIF using rd_mcp is ~12.5% performance overhead vs. NVMe namespace without end-to-end protection
  - virtio-scsi → vhost-scsi → nvme passthrough of T10 DIF metadata should see similar performance overhead
  - Overall I/O efficiency is more important than raw I/O performance

vhost-scsi TODO

- **Live migration**
  - Use existing vhost-net log infrastructure to copy current virtio-scsi register state to migration destination
  - Requirements of blocking I/O on LIO side while migration occurs, use ALUA, PR, or something else..?
  - Who drives the vhost-scsi + LIO backend configuration on destination..?
- **libvirt**
  - Same question, who drives the vhost-scsi + LIO backend configuration on destination.?
- **Openstack Nova**
  - WIP patches to Nova Kilo by Mike Perez (Cinder PTL)
  - Basic vhost controller attach + detach working

Linux I/O ecosystem update

DATERA

- **Copy offload SCSI host interface**
  - SCSI host patches submitted by Martin Petersen, likely a v3.19 item at this point
  - Developed against LIO target EXTENDED_COPY implementation, supporting block-to-block copy using IEEE NAA descriptors
- **Copy offload userspace interface**
  - Syscall entry points for userspace API has been discussed for a while now..
  - According to Zach Brown, these will not be seeing a v3.19 merge, yet.
- **What does this mean to KVM..?**
  - Cloning of disk images is hugely inefficient if blocks have to actually be copied all the way to the host
  - For arrays that support copy offload, cloning can be a matter of just setting COW pointers (eg: zero-second clones)

Linux I/O ecosystem update

DATERA

- **T10 DIF userspace API**
  - Patches proposed by Darrick Wong to extended AIO syscall interface to accept DIF payload from userspace
- **Status for upstream**
  - According to Darrick, currently too many objections to proposed interface. Not considered v3.19 material at this point.
- **What does this mean to KVM..?**
  - Applications in guest can use application-tag field (metadata) in DIF to describe what data blocks actually are.
  - In a storage hierarchy, being able to pass hints from userspace to I/O stack is **hugely** helpful to make intelligent placement decisions
  - Will eventually become standard syscall interface for attaching metadata from userspace, once details are sorted out..

Thank You.

DATERA
2570 W El Camino Real,
Suite 380
Mountain View, CA 94040
nab@datera.io

www.datera.io