A Tail of Latency, IOPs, & IO Priority

Adam Manzanares, Filip Blagojevic, Cyril Guyot

adam.manzanares@wdc.com
You want what from my HDD?

• Capacity -> Here is a 12 TB HDD
• Low $/TB -> Here is a 12 TB HDD

• IOPs – 200 4k Rand Read IOPs at QD 32
• Throughput – 243 MB/s
• Better tail latency – lets work together
How do we enable each other?

- HDD is peripheral
  - Under management by OS
  - Well defined interface for OS to HDD communication
    - SATA, SCSI

- Application runs on OS
  - Well defined interface to storage
    - File System, Block Access

- Let’s deliver application semantics to the HDD
  - What, How, and Why

- Our Example Today
  - Binary priority to the HDD
  - 12 TB SATA HDD Supports NCQ Priority

- Linux Supports IOPriority
  - Not so fast
  - Priority used for One Particular Block Scheduler
  - Priority Is Not Passed To The Drive

- This talk discusses
  - Why Priority To The Device Is Important
  - What We Did to Get Priority To The Device
  - Results
Where To Handle Priority In The IO Stack

Priorities?

Block Queue/s

Device Queue/s
## Block Layer Priorities

### PROS

- No support needed from device
- Flexibility in implementation
  - Scheduler can be modified rather easily
  - Linux source code is available

### Cons

- Request makes it to device
  - Priority no longer exists
- Devices may have large queue sizes
  - Your high priority request can be delayed significantly
- Implemented by time slicing
  - High priority requests could be significantly delayed
    - Before hitting the device queue
Device Priorities

**PROS**

- Device will service high priority requests first
  - Should improve latency
- Block layer scheduler irrelevant?
  - Hints of this with NVMe

**Cons**

- Device must support priority
- What does device do with priority?
  - This can not be changed easily
  - Most don’t have access to FW
SATA NCQ & Priority

- **Serial Advanced Technology Attachment**
  - BUS protocol for HDD to host communication

- **Native Command Queuing**
  - SATA feature that defines mechanism for multiple outstanding IOs between host and device
  - Currently supports 31 outstanding commands

- Drive is allowed to reorder commands
  - Achieves better throughput
  - User loses control of when a particular IO is completed.

- NCQ Supports Priority Bit In Read and Write NCQ commands

- What does the drive do with prioritized NCQ commands?
QD vs IOPs & Tail Latency

The graph shows the relationship between QD (Queue Depth) and IOPs (Input/Output Operations per Second) with corresponding tail latency. As QD increases from 1 to 32, the IOPs also increase, and the tail latency shows a noticeable trend, indicating a higher latency with higher QD.
How Are IO Priorities Used In Linux Currently

• System call exists to set IO priorities
  – ioprio_set
  – Priority info currently only used by CFQ scheduler
  – Set in kernel io context which is associated with a process, process group, or user
  – ionice utility can be used

• User Space IO Converted to BIO in Kernel
  – read(), write(), pread(), pwrite() ... eventually generate BIO inside of the kernel

• BIO mapped to a request
  – Request queues are managed by block schedulers

• Request structure
  – Has a priority field
  – IO context only relevant if CFQ scheduler is used

• Request based device drivers convert requests to device specific commands
  – May be a layered path to device, SCSI to libata, SCSI to mpt3 SAS, ...

• CFQ Scheduler
  – Creates queues based on priority classes
  – Time slices across these queues
What we Want to Do, Why, and How

• IO priorities passed to the drive
  – Not just to the CFQ scheduler, allowing use of all kernel schedulers
  – Make sure common request submission pathways pass IO priority information to devices.

• Drive level priority should enable finer grained control
  – Currently, priorities not passed to drive
    • Low priority IO equivalent to high priority IO at the drive
  – Guarantees high priority IO is respected in the drive queues

• Grab priority from IO context
  – Done when BIO is converted to request
  – Request has an io priority field
    • System call does not set this field

• Lower layer leverages request priority
  – Libata builds ata commands with high priority
  – mpt3sas (Broadcom HBAs) flag command as being high priority
    • HBA correctly builds high priority SATA command
  – MicoSemi HBAs TBD

• Libata
  – Recognize IO priority in SCSI command and build SATA command with priority information
Pass priority to request

Recognize request priority and build SATA command with priority
Some Block Layer Code

```c
static blk_qc_t blk_queue_bio(struct request_queue *q, struct bio *bio)

Function takes bio and places it on a request queue. Eventually makes a call to:

```c
static struct request *__get_request(struct request_list *rl, int op, int op_flags,
                                      struct bio *bio, gfp_t gfp_mask)
```

Which gets a free request and initializes it. We added a call to the following in __get_request:

```c
static inline void blk_rq_set_prio(struct request *rq, struct io_context *ioc)
{
    if (ioc)
        rq->ioprio = ioc->ioprio;
}
```
@@ -1755,6 +1756,8 @@ static unsigned int ata_scsi_rw_xlat(struct ata_queued_cmd *qc)
 {
     struct scsi_cmnd *scmd = qc->scsicmd;
     const u8 *cdb = scmd->cmnd;
+    struct request *rq = scmd->request;
+    int class = IOPRIO_PRI0_CLASS(req_get_ioprio(rq));
 }

 int ata_build_rw_tf(struct ata_taskfile *tf, struct ata_device *dev, u64 block, u32 n_block,
                     unsigned int tf_flags,
                     unsigned int tag)
 {
     tf->flags |= ATA_TFLAG_ISADDR | ATA_TFLAG_DEVICE;
     tf->flags |= tf_flags;
@@ -785,6 +786,12 @@ int ata_build_rw_tf(struct ata_taskfile *tf, struct ata_device *dev,
     tf->device = ATA_LBA;
     if (tf->flags & ATA_TFLAG_FUA)
         tf->device |= 1 << 7;
+     if (dev->flags & ATA_DFLAG_NCQ_PRIO) {
+         if (class == IOPRIO_CLASS_RT)
+             tf->hob_nsect |= ATA_PRIO_HIGH << ATA_SHIFT_PRIO;
+     }
 }
But My HDD is Not Connected Through AHCI

- We have tested and completed patches to enable priority when using a couple of Broadcom HBAs
- Changes merged into 4.10
- Investigating how to get this done with MicroSemi HBAs
- HBA Vendors Contact Us
How to set IO Priorities In Applications

• Turn on sata ncq priority support
  – SATA
    • /sys/devices/pci0000:00/0000:00:1f.2/ata8/host7/target7:0:0/7:0:0:0/ncq_prio_enable
  – MPT3SAS
    • /sys/device/*/sas_ncq_prio_enable

• ionice
  – Linux Utility
  – set/get process I/O scheduling class and priority
  – Use process id, process group id, or a user id

• ioprio_set, ioprio_get
  – Systems calls to get/set scheduling class and priority

• Do you want a quick test with fio
  – Use the prioclass and prio arguments
    
    sudo fio --ioengine=libaio --iodepth=1 --rw=randread --bs=4k --direct=1 --numjobs=1
    --time_based --runtime=300 --filename=/dev/sdb --randrepeat=0 --prioclass=1 --prio=7
Benchmark

• Background IO
  – FIO Random Read
  – 4KiB Block Size
  – Run time = 5m
  – Queue Depth = 32
  – ioengine = libaio

• Foreground IO
  – FIO Random Read
  – 4KiB Block Size
  – Run time = 5m
  – ioengine = libaio
  – Queue Depth = \{1,4,16,32\}

• Schedulers used
  – Deadline with/without drive priority
  – NOOP with/without drive priority
  – CFQ with/without priority & CFQ with priority and drive priority
Deadline Scheduler

Graph showing IOPs and p99.99 Latency ms for different QD values. The graph compares four categories:
- DEAD.F
- DEAD.B
- DEADLINE.F.P.D
- DEADLINE.B.P.D

The y-axis represents IOPs and p99.99 Latency ms, while the x-axis represents QD values from 1 to 32.
CFQ Scheduler

- **CFQ.F**
- **CFQ.B**
- **CFQ.F.P**
- **CFQ.B.P**
- **CFQ.F.P.D**
- **CFQ.B.P.D**

The graph illustrates the IOPS (Input/Output Operations Per Second) and 99.99 latency (ms) for different QD (Queue Depth) values. The x-axis represents the QD values ranging from 1 to 32, while the y-axis shows the IOPS and 99.99 latency in milliseconds.
Some Kernel Files Modified

- block/blk-core.c
  - When request is initialized for a bio we now set the iopriority from the io context of the current process

- One block patch committed
  - 5dc8b362a2374d007bc0db649b7ab6a79dd32bda

- drivers/ata/libata-scsi.c
  - ata_scsi_rw_xlat()
    - grab request from scsi command, use this to get priority class

- Four libata/ahci patches committed
  - 8e061784b51ec4a4efed0deaafbf5bd9725bf5b06
  - 84f95243b5439a20c33837075b88926bfa00c4ec
  - 4e647d960c51e0d5cd700058fb8ddd529c390ee
  - 9f56eca3aeab699a7dbfb397661d2eca4430e94

- drivers/ata/libata-core.c
  - ata_build_rw_tf
    - Add priority class, if device supports priorities build ATA tf with high priority
  - ata_dev_config_ncq_prio
    - Check identify log page priority support
  - ata_id_has_ncq_prio
    - check if identify priority bit is set
    - added functions for adding a sysfs entry
    - also changed includes to ata device flag for indicating if prioritized commands should be used

- drivers/scsi/mpt3sas/*
  - HBA command set as high priority when request is set as high prio
  - sysfs entries to enable/disable feature
  - scsi probe functions to discover

- Once mpt3sas patch committed
  - 307d9075a02b696e817b775c565e45c4fa3c32f2
Future Directions

• How do schedulers/devices/priorities mix?
  – CFQ has some unexpected results
    • What causes this and can device priority help?
  – BFQ algorithms may be improved with priority
    • Currently looking into this

• How many priorities does an application need?
  – Currently high/regular

• Is a latency cut off a better mechanism to control tail latency?

• How does this work translate to distributed storage systems?

• Should repair priority be a function of the state of the cluster?
  – Enough redundancy low priority rebuild
  – High priority rebuild when necessary

• Do classes of users map cleanly into priorities?

• Will this be mapped onto NVMe devices using prioritized queues or using prioritized commands?
Thanks for your attention.

Any additional questions? Contact me at:

adam.manzanares@wdc.com