IO Priority: To The Device and Beyond

Adam Manzanares
Filip Blagojevic
Cyril Guyot
The Relationship of Throughput and Latency

- Increasing IOPs is positive
- Increase in Latency is not desirable

Environment – Lots of HDDs
Response Time of a given HDD is critical

Queue Depth - # of outstanding requests
IOPs - requests served per second
Latency – time to service one request
p99.99 – 99.99 per cent of request latencies are lower than this value

Drive Queue

From Host
Request 1
Request 2
Request 3
Request 4
To Media

BARS represent IOPs
Line represents tail latency

Increasing QD
Application to device IO traverses many layers in the OS.

Priority may be handled in an OS layer, but requests may be reordered in the device queue.

Device queue priorities are a tuning knob for device queue request de-staging.

Latency should improve, but not as flexible as host level solutions.

OS = Linux
For this work
What We Did

Ensure application to device IO preserves priority throughout the layers in the OS IO stack.

Leveraged existing application priority interface. Mapped this interface down to device through block and device driver layers.

Enables priority to be respected at block layer, device layer, or a combination of the two.

Goal – prioritized application requests are de-staged from device queue faster than non-prioritized IO improving application tail latency.

Enables priority to be respected at block layer, device layer, or a combination of the two.

Goal – prioritized application requests are de-staged from device queue faster than non-prioritized IO improving application tail latency.

Ensure application to device IO preserves priority throughout the layers in the OS IO stack.

Leveraged existing application priority interface. Mapped this interface down to device through block and device driver layers.

Enables priority to be respected at block layer, device layer, or a combination of the two.

Goal – prioritized application requests are de-staged from device queue faster than non-prioritized IO improving application tail latency.
All IOs are small (4KiB) random read requests. This approximates infrequently accessed non cacheable requests. This is typical in large scale storage systems where HDDs are used for long term high capacity storage.
Benchmark Details

- **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

Benchmark approximates small random read requests. Large-scale-enterprise customers indicate that tail latency is a key metric. HDD efficient with large requests. Write requests typically buffered in large scale systems. Lots of data spread over many HDDs, most of the data not hot. Cold data must be retrieved in a bounded, predictable manner.

In the experiments we fix the background at a QD of 32, which fills the drive queue. Foreground is then increased from QD 1-32 while maintaining the background workload of QD 32. We compare the behavior of the application before and after our changes that plumb IO priority to the device.

FIO is the Flexible IO tester, which we use to approximate the user application. We also vary the Linux Block Scheduler that is used in the experiment. For each of the experiments we have results when using drive level priorities and also block level scheduler priorities when they are respected at their respective layers.
Two experiments. Foreground and Background IO with no prioritization. Foreground IO prioritized to the device with Background IO.

Drive prioritization dramatically improves latency with a minimal impact to IOPs.
Deadline Scheduler IOPs Results

Two experiments. Foreground and Background IO with no prioritization. Foreground IO prioritized to the device with Background IO.

Drive prioritization dramatically improves latency with a minimal impact to IOPs.
CFQ Scheduler Latency Results

Two experiments
Foreground IO with host prioritization and Background IO
Foreground IO with host and device prioritization and Background IO

Graphs showing p99.99 latency in milliseconds for different QD values.
Two experiments
- Foreground IO with host prioritization and Background IO
- Foreground IO with host and device prioritization and Background IO
Conclusion and Future Directions

• Priority at the device matters
  • Extra knob to influence tail latency
  • Must be passed through many layers
  • Mechanism now exists to test device behavior

• Host Queue and Drive Queue
  • Interaction between the two can cause unexpected behaviors
  • Working with host scheduler developers

• Work is currently HDD focused
  • HDD is inherently serial
  • How does SSD impact this?

• Work is large-scale enterprise focused
  • What to do with consumer and mobile?
Thanks for your attention