# HDD IO Prioritization

How can we get the most performance out of rotational media? How can the software and hardware get smarter together?

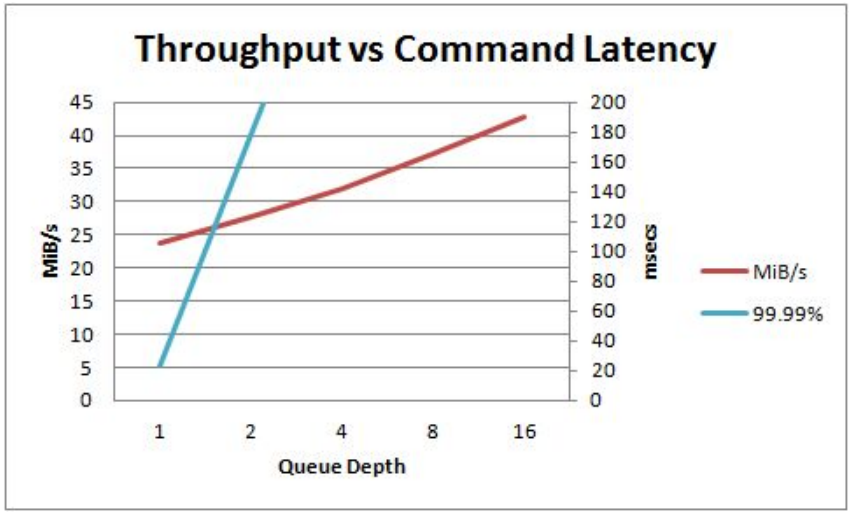
Seagate
Mark Gaertner

# The Problem

Increasing IOPS raises command latency.  Command latency is critical since they are critical to application response time and meeting QoS (Quality of Service) or SLA (Service Level Agreements) for web services

Let’s look at an example workload to highlight this issue

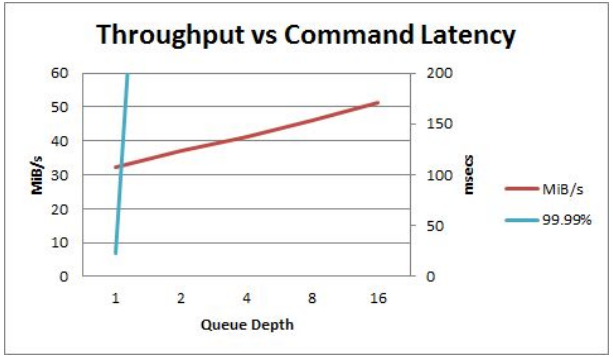
- Workload: Read/Write ratio of 60/40, Read = 256KiB, Write = 512KiB
- SLA:  99.99% read commands must complete within 100ms
- Throughput increases nicely as queue depth increases but command latency rises quickly
- Can not even increase queue depth to 2 and meet the 100ms requirement
- System must limit queue depth to 1, stuck at 23.8MiB/s



# IOPS/Throughput Opportunities

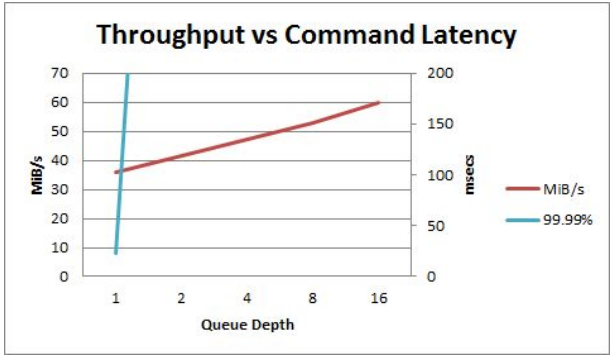
Write Cache (WCE) improves IOPS but still does not allow increasing queue depth

- Big jump to 33.0 MiB/s




Improves system write caching/logging and/or improvements in stack.  Often drivers, kernel, and HBA split up large operations.

- Writes are increased from 512KiB to 2MiB
- Read to Write ratio in terms of bytes transferred remains constant
- Modest jump to 36.9 MiB/s



# “Fast Fail” used for Priority

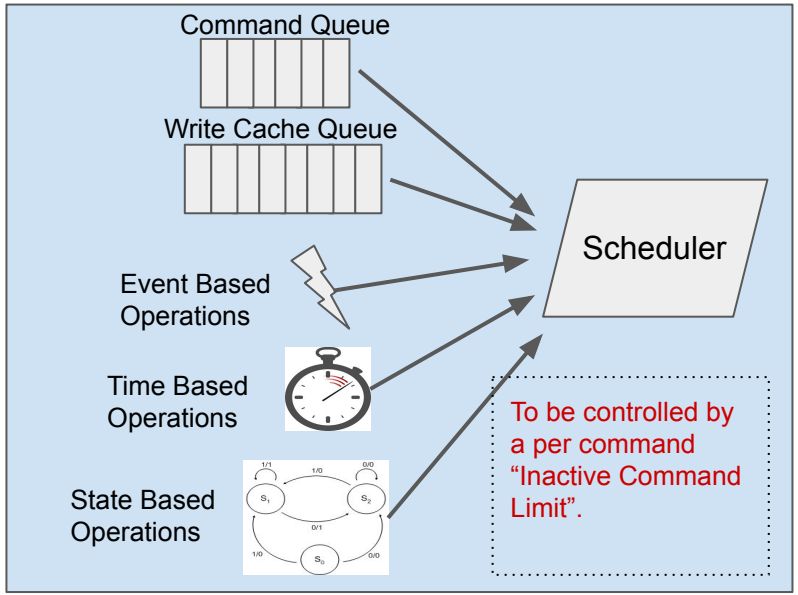
All IOs are not equal.  Some are for internal operations such as data migration, data redundancy generation.  Writes are usually cached by the system and, as a result, HDD write command latency is not critical.  Non-internal reads may have varying QoS requirements.

The “Fast Fail” mechanism is quite capable of differentiating the priority of these commands.

Fast Fail in this context is the protocol and functionality drafted by OCP without any additional capability.  This functionality is directly reflected in T13 and does not include the extensions in T10.

- **T13: Command Duration Limits feature set (see ACS-5) - ratified proposal: T13/f18162r9 (Author: Seagate)**
- T10: Command duration limits (see SPC-6, SBC-4, and SAM-6) -  ratified proposal:  T10/18-089r5 (Author: Western Digital)

# Fast Fail Overview




*Command based*

Active Command(s)

Positioning
Positioning recovery
Data transfer
Data recovery
Data reallocation

To be controlled by a per command “Active Command Limit”.

Command
*Index*

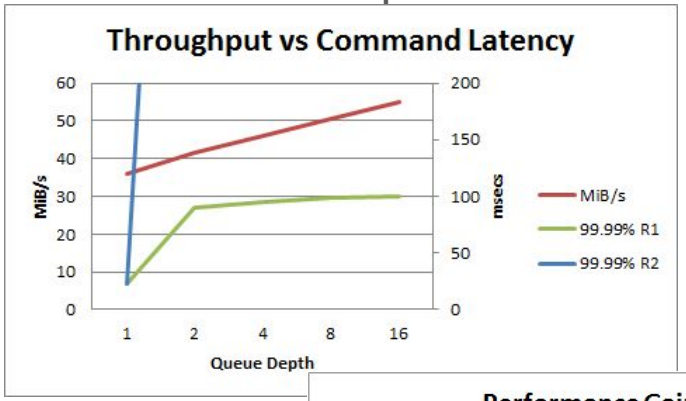
| | Inactive Command Limit | Inactive Policies | Active Command Limit | Active Policies |
|---|---|---|---|---|
| 1 | | | | |
| ..... | | | | |
| 7 | | | | |

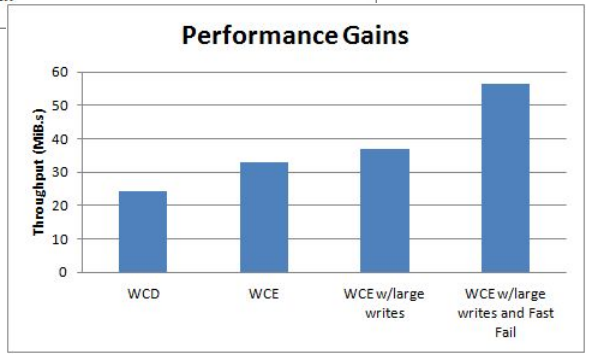
# Fast Fail: IOPS while Preserving Command Latency

To demonstrate value, the previous example is extended by assuming 30% of the reads (R1) have QoS requirement of 100ms at 99.99%. The remaining reads (R2) are bounded to 1000ms and the writes are cached thus response time insensitive.

R1
R2

| | Inactive Command Limit | Inactive Policies | Active Command Limit | Active Policies |
|---|---|---|---|---|
| 1 | 100ms | Best effort | NA | NA |
| ..... | 1000ms | Best effort | NA | NA |
| 7 | | | | |




- Queue depth can now be increased dramatically while maintaining the QoS for R1 reads
- 99.99% command latency of 100ms at QD = 16
- Very significant increase to 56.3 MiB/s
- Performance gains are inversely proportional to the amount of high priority commands

# Summary

Large gains in IOPS/throughput are possible using Fast Fail as a IO priority mechanism

Fast Fail defines 7 QoS levels for reads and 7 for writes

STX position is that this minimal Fast Fail definition is adequate and robust