Using Software to Reduce High Tail Latencies on SSDs

Kapil Karkra
Principal Engineer
Intel Non-volatile Memory Solutions Group (NSG)
Notices and Disclaimers

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

No computer system can be absolutely secure.

Performance results are based on testing as of July 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase.

Intel, the Intel logo, Intel Optane, Xeon, and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

*Other names and brands may be claimed as the property of others.

© 2018 Intel Corporation.
Assume one in a 1000 queries to an SSD will result in a longer (tail) latency event.

At 1 SSD it is

\[ 1 - \left(\frac{999}{1000}\right)^1 = 0.1\% \]

At 100 SSDs it is 10%

\[ 1 - \left(\frac{999}{1000}\right)^{100} = 10\% \]

At 1000 SSDs it is 63%

\[ 1 - \left(\frac{999}{1000}\right)^{1000} = 63\% \]

What causes tail latency problem in SSDs?
Read Tail Latency in SSDs

The main sources of high tail latency are:

1. Host reads colliding with host writes (includes garbage collection)
2. Host reads colliding with other host reads
3. Host reads colliding with asynchronous background operations (includes Read Disturb (RD) and Background Data Refresh (BDR))

Separate host reads from host writes to avoid read on write collisions
Minimize garbage collection and other background activities
Avoid high queue depth reads and writes to avoid too many collisions and resource utilization

<table>
<thead>
<tr>
<th>% of IOs</th>
<th>rw mix</th>
<th>rd only</th>
<th>rd only no BDR/RD</th>
<th>rw mix</th>
<th>rd only</th>
<th>rd only no BDR/RD</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>9.152</td>
<td>3.184</td>
<td>3.11</td>
<td>3.856</td>
<td>0.181</td>
<td>0.181</td>
</tr>
<tr>
<td>99.9</td>
<td>10.944</td>
<td>3.44</td>
<td>3.344</td>
<td>4.08</td>
<td>0.183</td>
<td>0.183</td>
</tr>
<tr>
<td>99.99</td>
<td>17.024</td>
<td>4.32</td>
<td>3.888</td>
<td>7.904</td>
<td>0.185</td>
<td>0.185</td>
</tr>
<tr>
<td>99.999</td>
<td>21.632</td>
<td>5.536</td>
<td>4.192</td>
<td>13.376</td>
<td>1.736</td>
<td>0.185</td>
</tr>
</tbody>
</table>

Configuration: Cliffsdale P4500 480GB, custom firmware suppressing BDR and RD
Benchmark: Fio, 12h, 4k random reads, 4k random writes (a) qd=64 (b) qd=1
I/O Determinism SSD Capabilities Overview

Key IO determinism capabilities

1. NVM Sets and Endurance Groups
2. Deterministic/Non Deterministic (D/ND) Windows

How can software take advantage of these capabilities to achieve I/O Determinism?
Software Approach #1: Solving tail latency using data redundancy, a D/ND I/O scheduler, and a write-back cache

The approach achieves (a) read write separation (b) reduces read on read collisions (c) eliminates the read-on-background-write collisions

What if you don’t like the capacity tradeoff or cost of an additional write buffer and don’t have the I/O determinism capabilities on your SSD?

*Other names and brands may be claimed as the property of others.
SSD Resource Utilization and Tail Latency

Keep the NAND dies idle

The more the resources are utilized, the higher the latency spike

SSD Transfer Buffer

- read1
- read2
- read3
- read4
- Write1
- Write2

Read IOPs

Write BW

Die utilization <5%
QoS requirements met

Die utilization <5%
QoS requirements NOT met

Die utilization >5%
QoS requirements NOT met

SSD can’t accept new read5

Keep writes small for the transfer buffers idle
Software Approach #2: I/O Shaping

1. Throttling/Trickling writes: Die Idle

<table>
<thead>
<tr>
<th>Desired</th>
<th>Sets</th>
<th>Sets</th>
<th>Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>p99</td>
<td>3</td>
<td>5.088</td>
<td>4.896</td>
</tr>
<tr>
<td>p99.9</td>
<td>5</td>
<td>6.88</td>
<td>6.688</td>
</tr>
<tr>
<td>Wr BW (MB/s)</td>
<td>67</td>
<td>57</td>
<td>41</td>
</tr>
</tbody>
</table>

Configuration: Cliffdale P4510 1TB Sets
Benchmark: FIO, 12h, 4k random reads, qd=10, two threads of 25% random write bursts; thread1: 512k writes with a thinktime of 2.56s, thread2: 512k writes with thinktime of 1.28s

2. Chopping writes: Transfer Buffer efficiency

<table>
<thead>
<tr>
<th>Sets</th>
<th>Sets</th>
<th>Sets</th>
<th>Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=1.28, B=160, BS=512k</td>
<td>T=1.28, B=160, BS=128k</td>
<td>T=1.28, B=640, BS=32k</td>
<td>T=1.28, B=5120, BS=4k</td>
</tr>
<tr>
<td>p70</td>
<td>0.111</td>
<td>0.112</td>
<td>0.111</td>
</tr>
<tr>
<td>p99</td>
<td>2.544</td>
<td>1.544</td>
<td>1.464</td>
</tr>
<tr>
<td>p99.9</td>
<td>6.752</td>
<td>4.384</td>
<td>4.192</td>
</tr>
<tr>
<td>p99.99</td>
<td>17.536</td>
<td>11.072</td>
<td>10.56</td>
</tr>
<tr>
<td>Wr BW (MB/s)</td>
<td>15</td>
<td>15.43</td>
<td>15.38</td>
</tr>
</tbody>
</table>

Configuration: Cliffdale P4510 2TB Sets
Benchmark: FIO, 12h, 4k random reads, qd=10, two threads of 25% random write bursts; thread1: writes with a thinktime of 2.56s, thread2: writes with thinktime of 1.28s, varying block sizes from 512k down to 4k

This approach trades off write bandwidth for read determinism
1. Workloads that mix different lifetime data (e.g., mixing frequently updated data with static data) increase garbage collection and thus tail latency.

<table>
<thead>
<tr>
<th>Workload</th>
<th>Different Data Lifetime Streams: 3 Sequential Streams and 1 Random Stream; Large velocity delta among sequential streams, random partition only 5%, uniform random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Lifetime classifier</td>
<td>LBA Ranges different for three sequential workers with 1x, 7x, and 17x velocity difference (QD=4) while a single random stream of 1x velocity (QD=16)</td>
</tr>
<tr>
<td>WAF improvement</td>
<td>60% (2.9→1.2)</td>
</tr>
<tr>
<td>Performance Improvement</td>
<td>3.5x</td>
</tr>
<tr>
<td>Read QoS (P9999)</td>
<td>34%</td>
</tr>
<tr>
<td>Device</td>
<td>P4500</td>
</tr>
</tbody>
</table>

2. Write bursts in a workload also cause high tail latency.

Can we shape workload to (a) avoid mixing different lifetime data streams (b) eliminate bursts from the workload?
Software Approach #3: Intel® Optane™ SSD Write Buffer

Solution Details:

- All writes from application go to Intel® Optane™ SSD buffer
- Data in buffer partitioned, based on lifetime classifier
- Flushing of data performed in buckets of size equal to NAND drive erase unit size
  - Only one bucket at a time
  - Erase unit filled with data with same stream (e.g. same velocity)
- Flushing throttle – TB algorithm to prevent write bursts on NAND drive

The approach achieves (a) write amp reduction through lifetime classification (b) eliminates bursts (c) Intel® Optane™ Technology can absorb multiple reads and writes to improve tail latency
Conclusion

- Software techniques (redundancy, IO shaping, and caching/buffering) built on top of hardware capabilities (Intel® Optane™ SSD and IO Determinism capable SSDs) are a powerful tool to cut the tail.