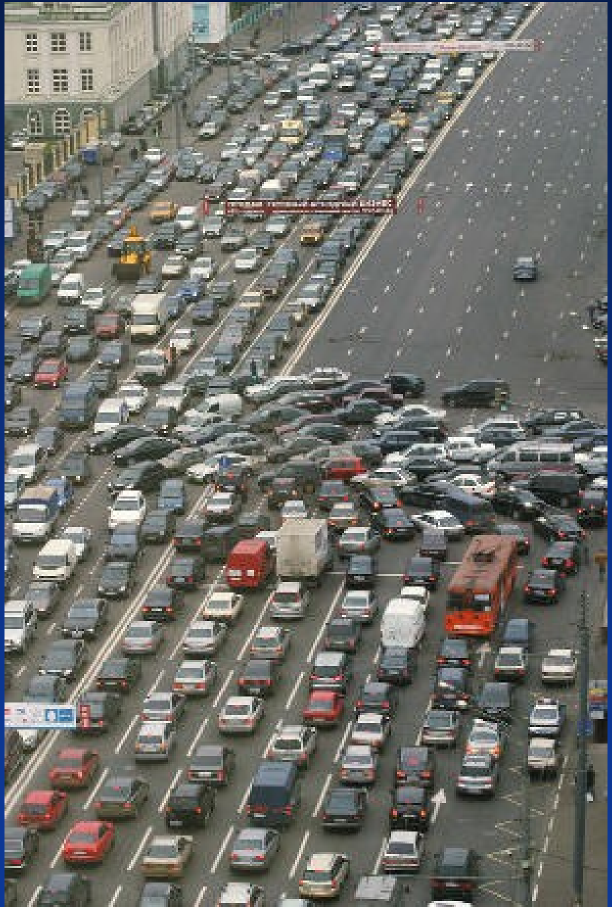
A Case for IO Determinism for Hyperscale Applications Utilizing QLC Flash Memory

Steven Wells – Data Center Architecture Fellow
Jim Ulery – Distinguished SSD Engineer
Toshiba Memory America, Inc.

Flash Memory Summit

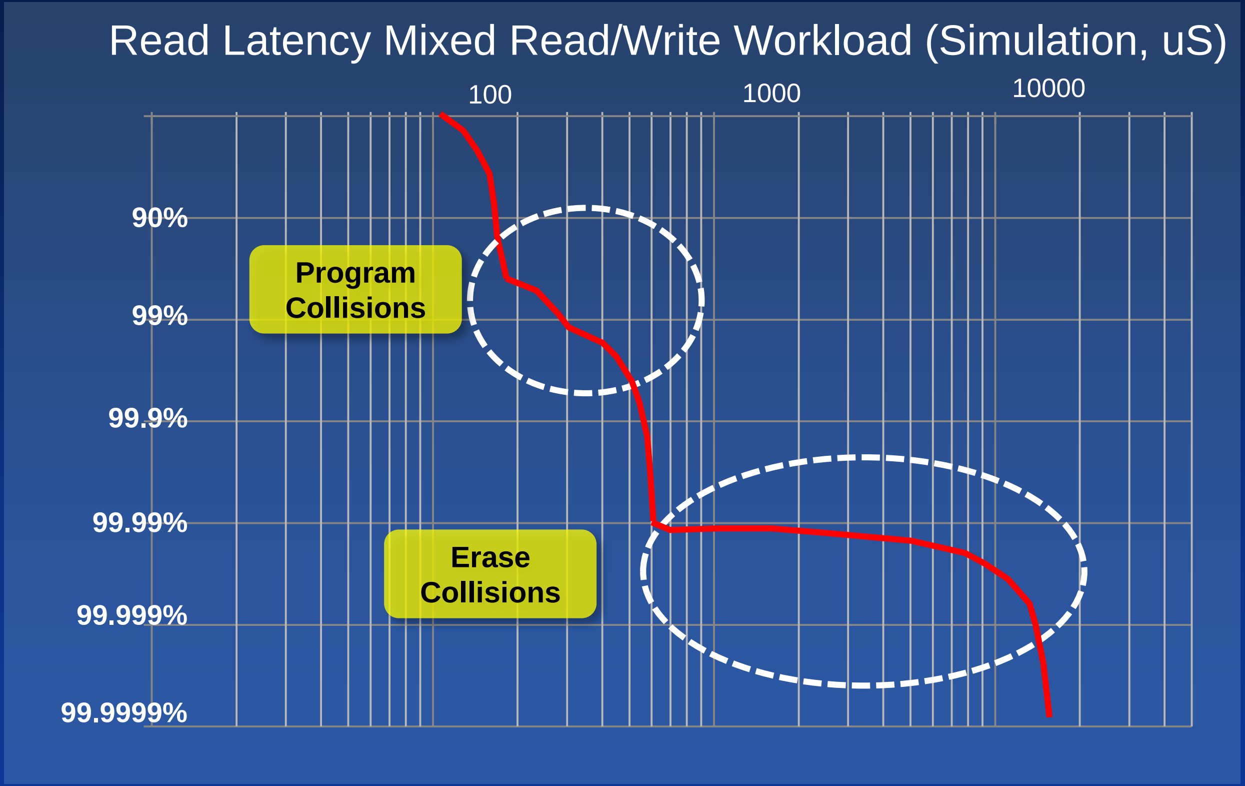
“Don’t worry, we process this in the background”



Introduction

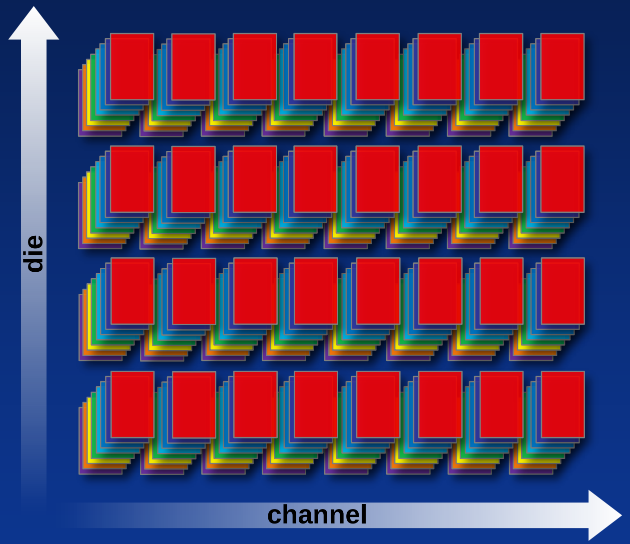
- Last year we introduced hyperscaler challenges with read tail latencies and offered a solution using IO Isolation
- This year, this presentation will expand the concept with what we are coining “Hyperscale Mean Latency”
- Technologies such as QLC intrinsically have higher latency and this presentation will demonstrate how Hyperscale Mean Latency is best mitigated with IO Isolation
- We demonstrate using NVMe™ IO Determinism solution how to mitigate internal operations such as garbage collection and background data refresh.

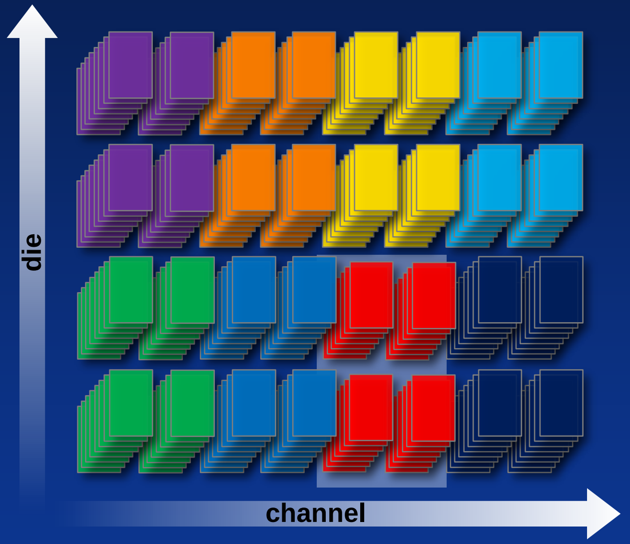
From Last Year – Read latency tails



Simulation data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

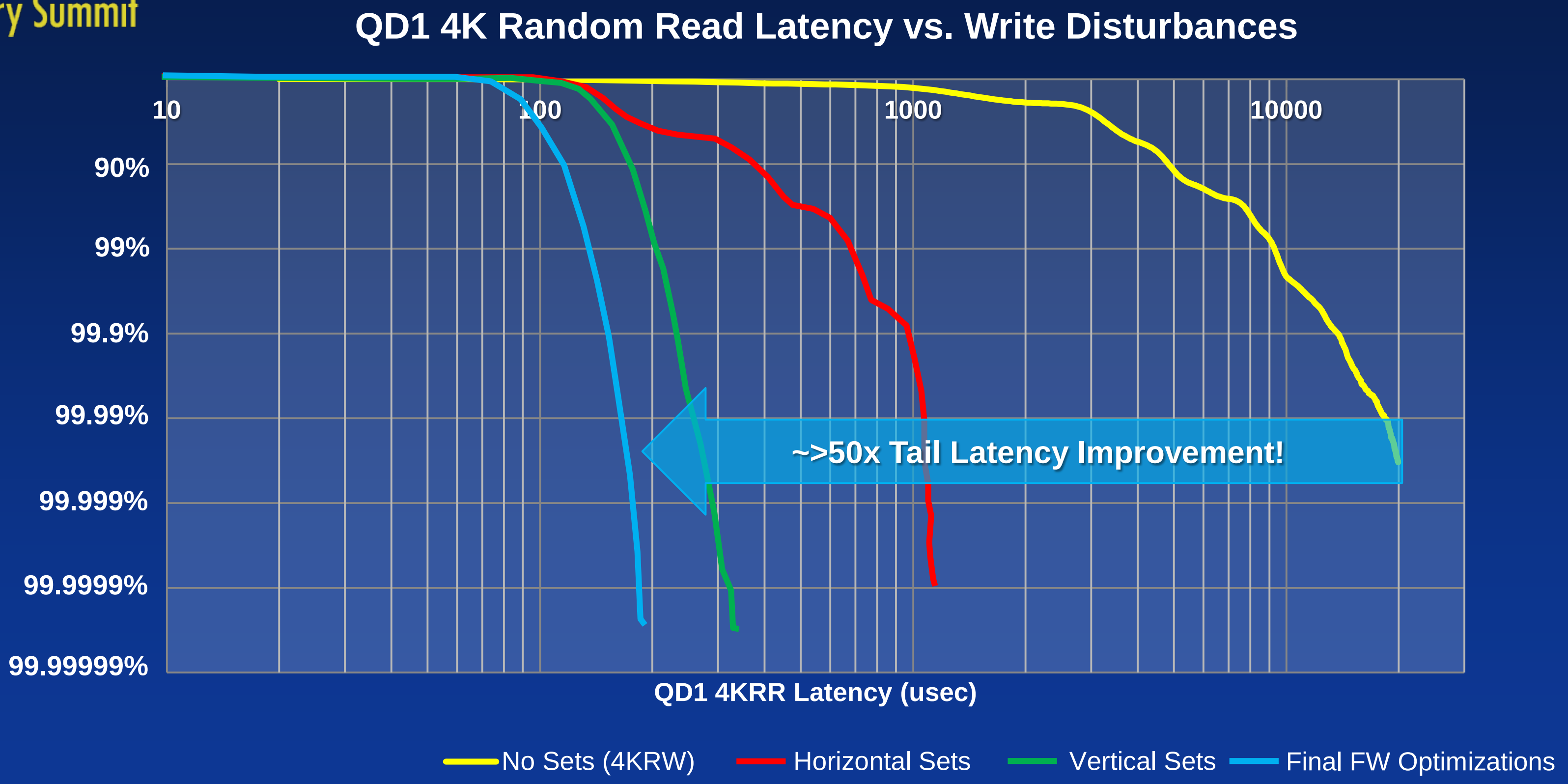
From last year: NVM set isolation concept





- Classic SSD architecture uses “bands” of devices on every channel to maximize bandwidth. Maintenance is also on every die on every channel
- New SSD array architecture creates independent NVM Sets

From last year's POC Set Isolation Result



Lab data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

Further justification for IO Isolation in hyperscale environments

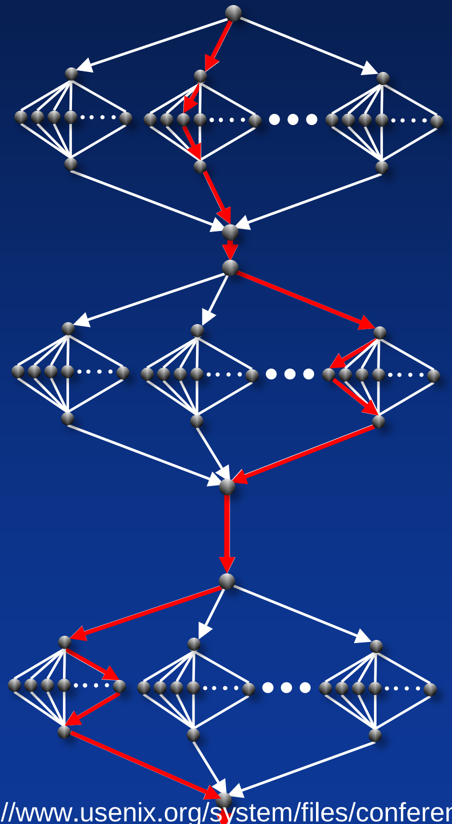
New Concept: Hyperscale Mean Latency (HML)

From last year…

"In practice, a single user request may result in thousands of subqueries, with a **critical path that is dozens of subqueries long**."

"The fork/join structure of subqueries causes latency outliers to have a **disproportionate effect on total latency**, and the large number of subqueries would cause slowdowns or unavailability to quickly propagate…"

Challenges to Adopting Stronger Consistency at Scale
- Ajoux et. Al., (Facebook & USC), 2015

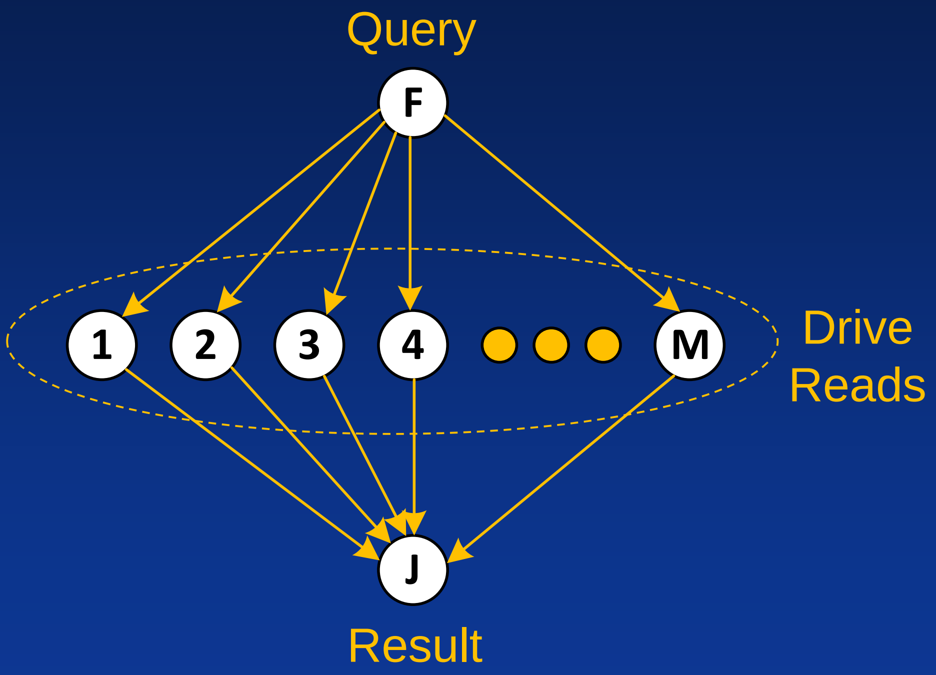


"Topology: Thousands=Hundreds x Dozens"

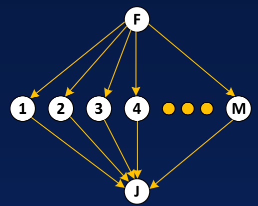
https://www.usenix.org/system/files/conference/hotos15/hotos15-paper-ajoux.pdf

Hyperscale Fork/Join Query Topology

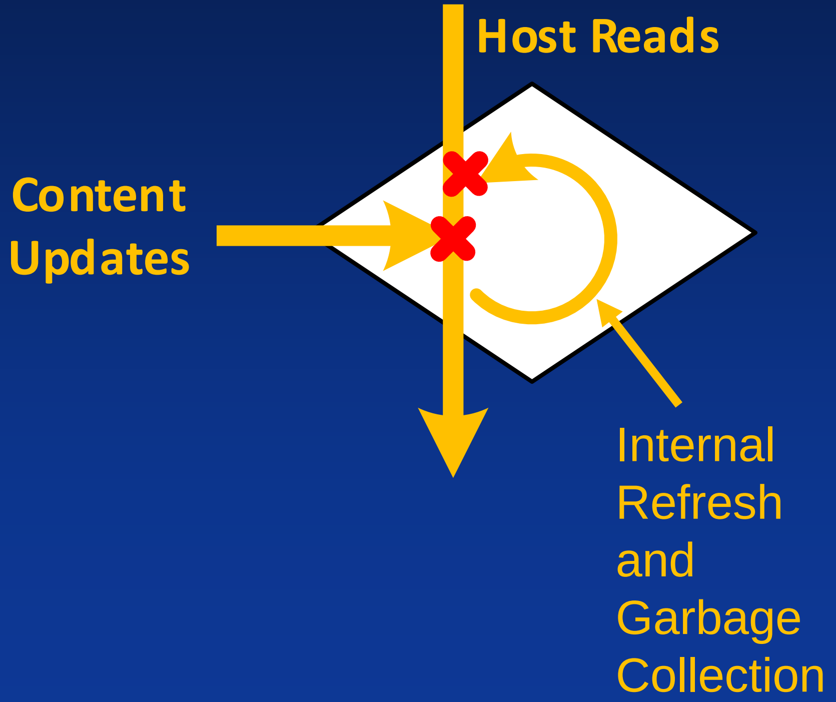
- M parallel drive reads per Fork/Join
- Results compiled @J
- Fork/Join J-latency determined by worst case latency



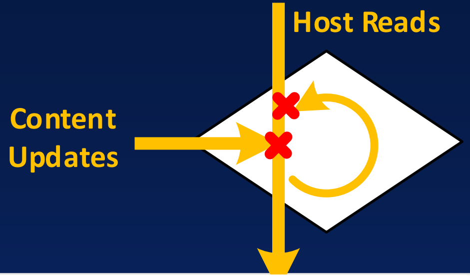
Effects of Content Updates and Internal Refresh on Fork/Join Latencies



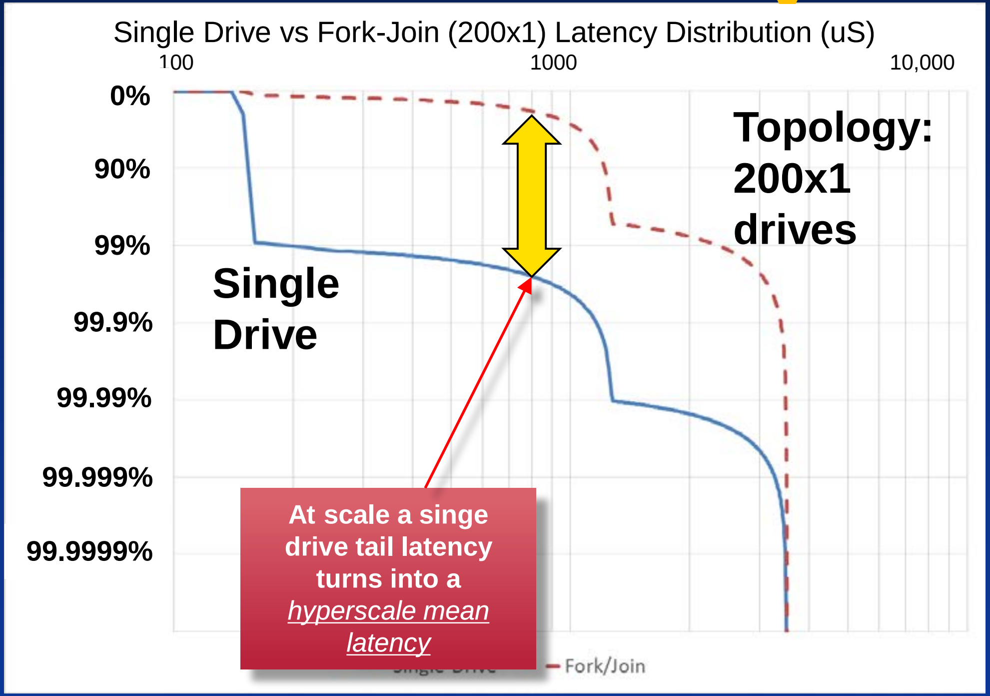
- While the host is doing time critical fork/join queries, each drive element is subject to
  - Host initiated writes to update content.
  - Drive initiated garbage collection and internal refresh



Hyperscale Mean Latency



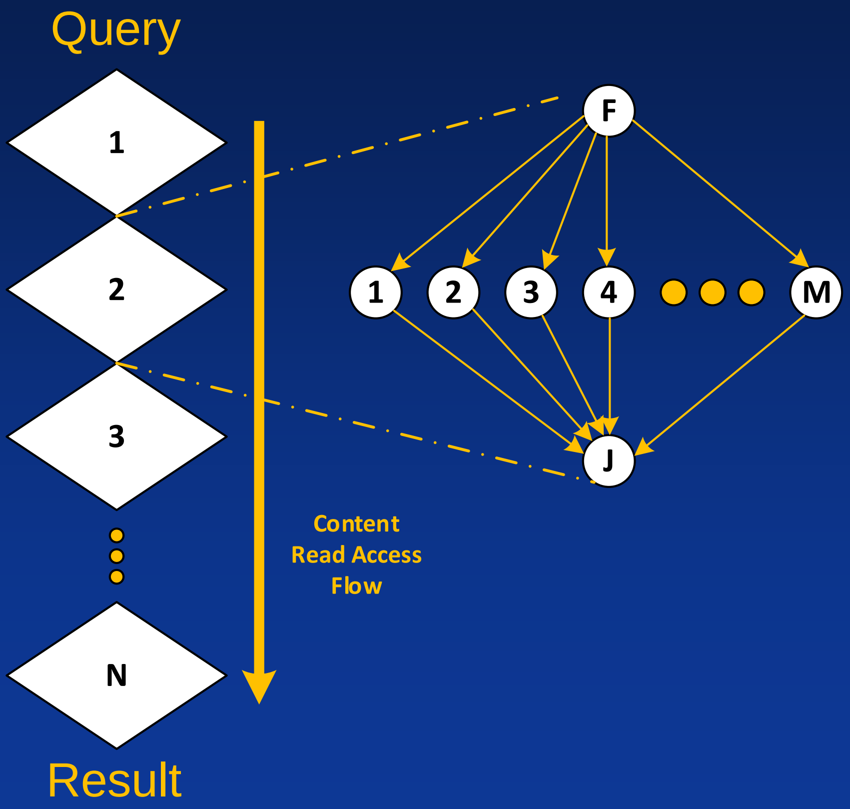
- Attempting to do fork/join queries in an environment with both content updates (writes) along with internal garbage collection and refresh amplify the *mean latencies* as seen from the perspective of the hyperscaler



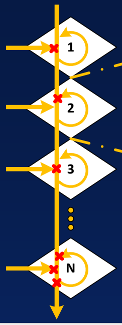
Simulation data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

Cascade Fork/Join Query Topology

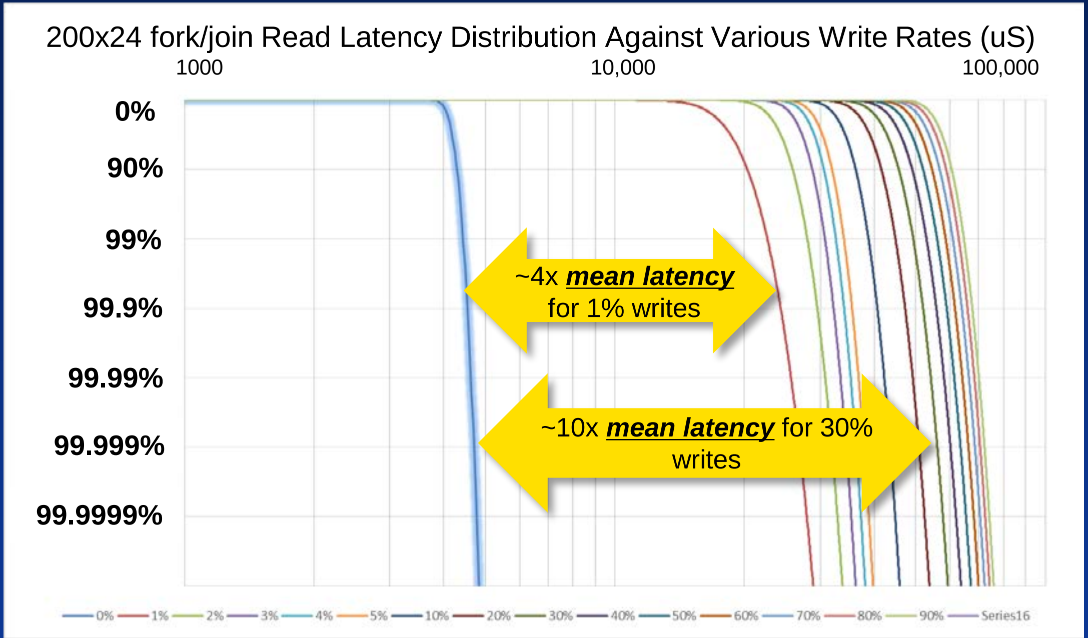
- Cascade of N Fork/Joins
- M parallel drive reads per Fork/Join
- Fork/Join read latency determined by Tall Pole
- ∴Cascade latency is sum of Tall Poles
- For the rest of this paper we'll assume MxN=200x24 as example



Tail Latencies: Real System Impact!



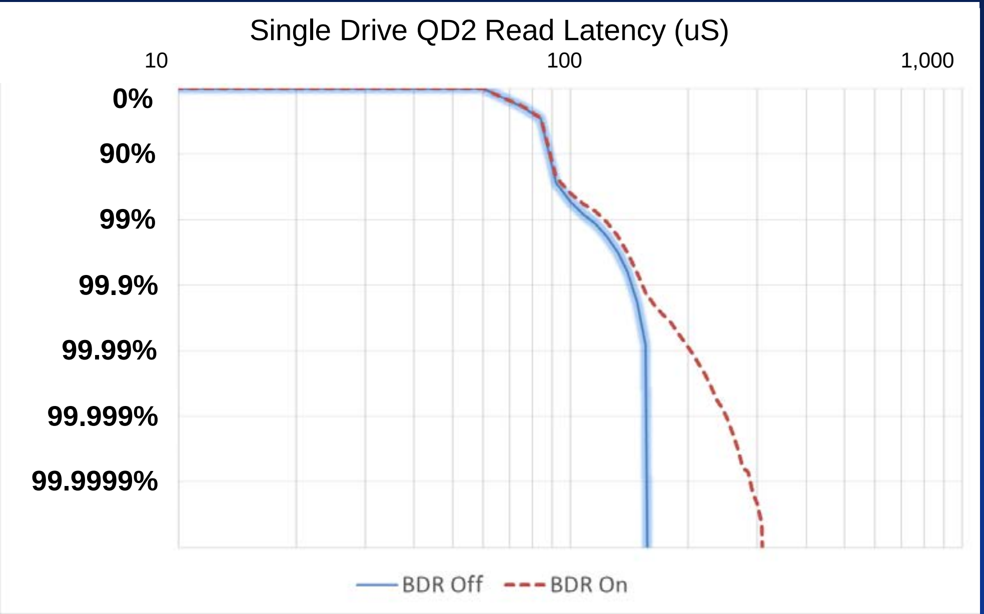
- Even 1% write level impacts hyperscale mean read latency 4x!
- A classical ~70/30 write profile can impact mean read latencies by 10x
- Best system latency is when read set is <u>quiet</u> except host reads
- <u>Solution</u>: IO Isolation



Simulation data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

Background Data Refresh (BDR)

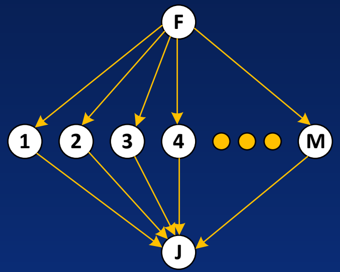
- BDR continuously reads mapped content.
  - Creates read-on-read collisions.
- Relocates weak content.
  - Creates read-on-write/erase collisions.
- Data shows limited mean impact at a single drive level. This is what justifies an SSD designer to think its OK to call it “Background Data Refresh”. But…



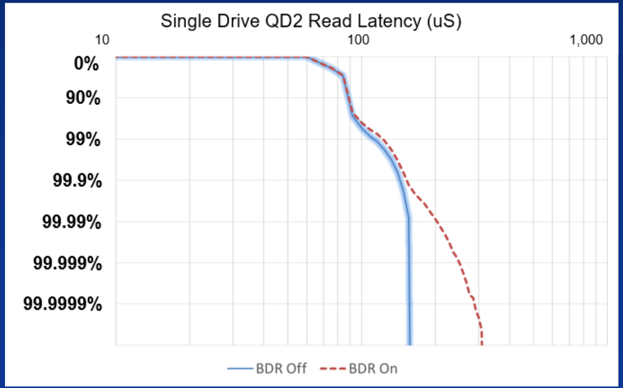
Per-Drive (no relocations)

Lab Data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

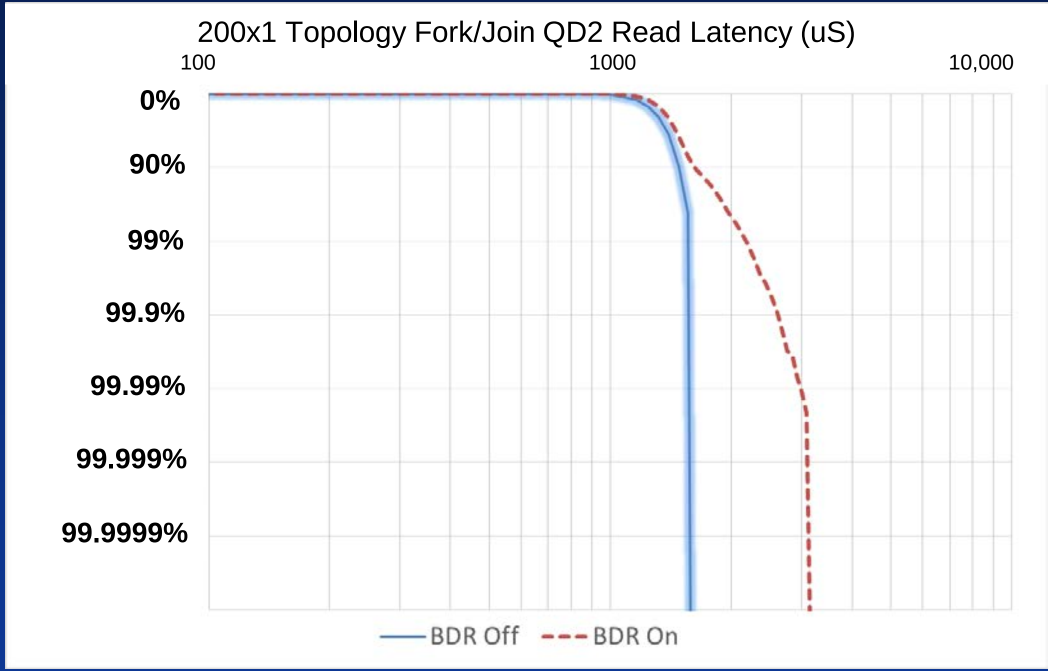
BDR Impact at Hyperscale Level Cannot be Ignored



M=200



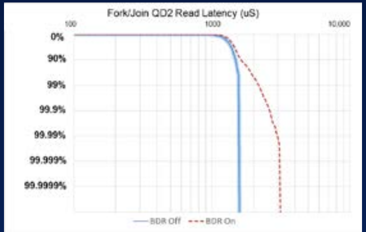
Per-Drive



Per-Fork/Join

Lab and Simulation Data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

IOD BDR Recommendation

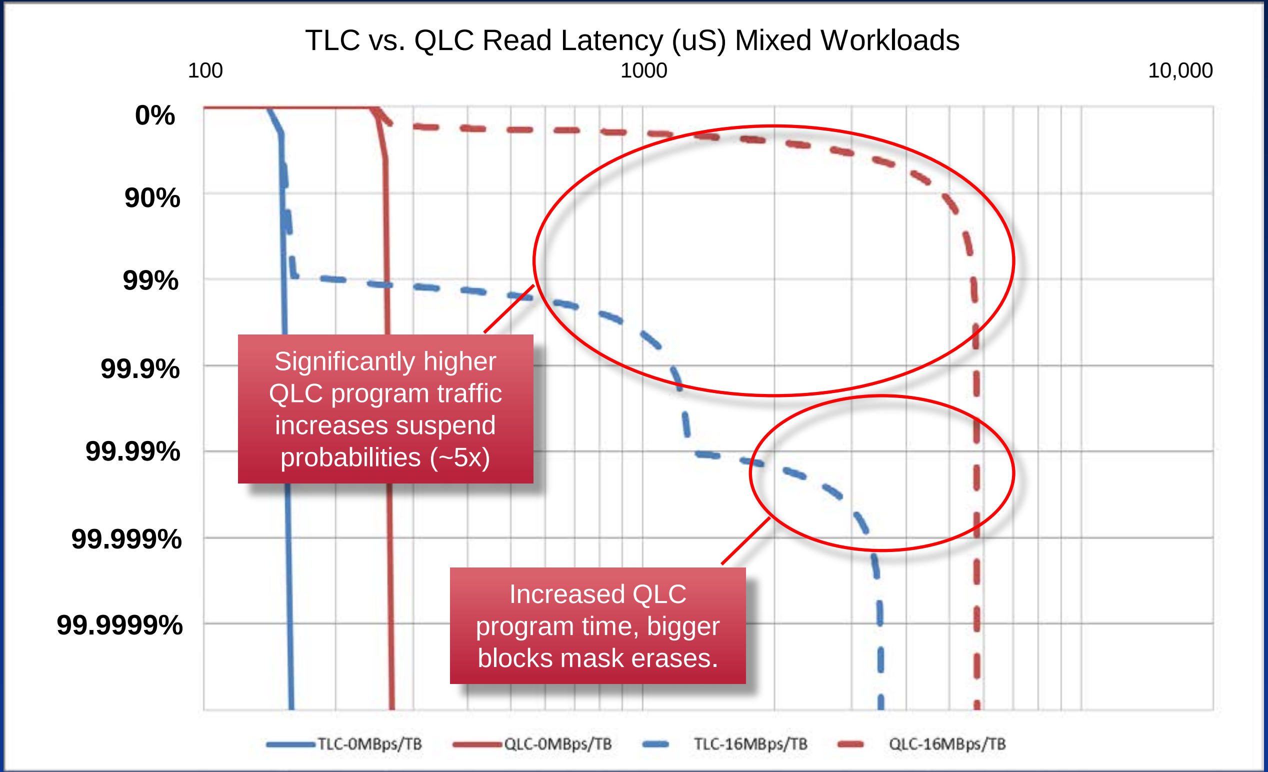


- Suspend BDR scan during DTWIN.
- Requires accelerated BDR scan rate during NDWIN intervals to meet coverage targets

What about QLC and IOD?

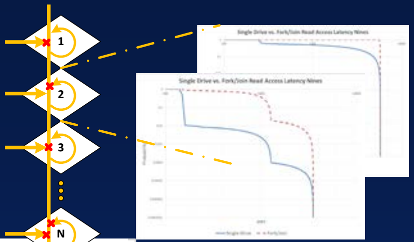
- Assumptions (QLC vs. TLC):
  - Bigger blocks
  - Reads 2x-3x slower
  - Programs 4x-5x slower
  - Erases and suspends “about” the same

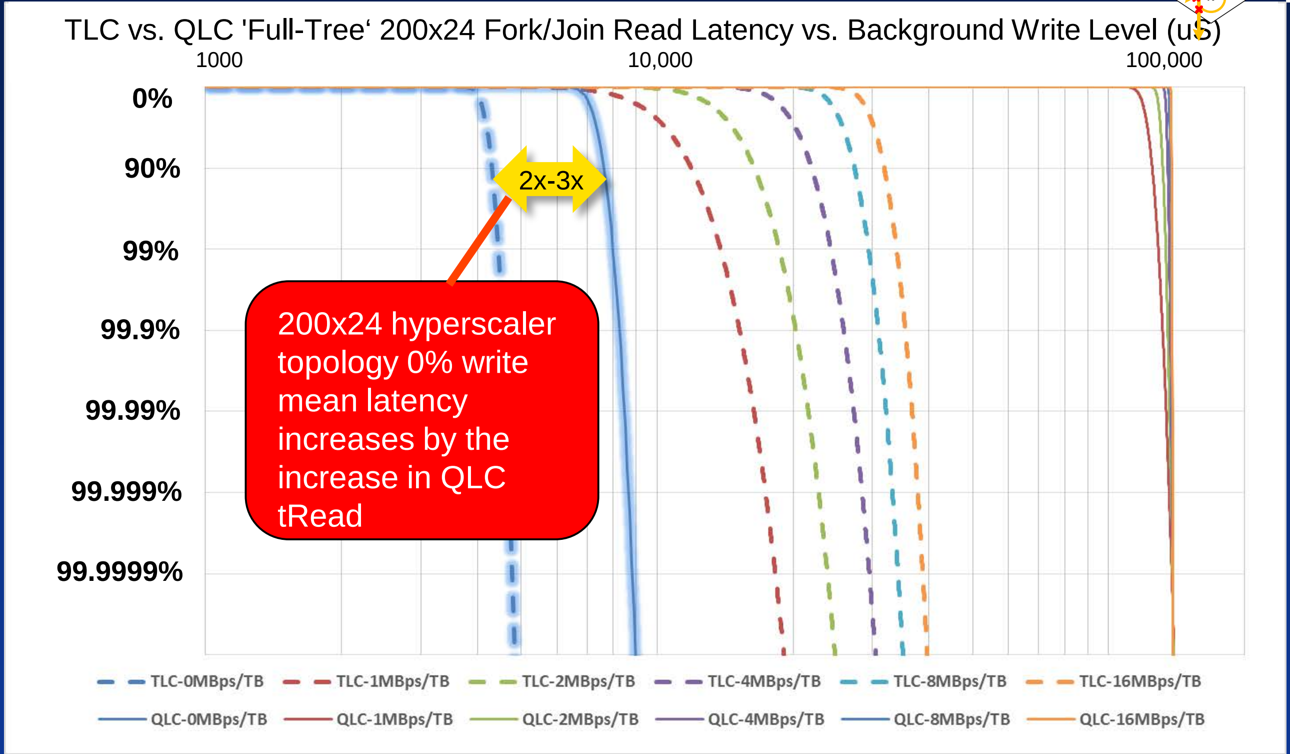
TLC vs. QLC Per-Drive Read Latencies



Simulation Data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

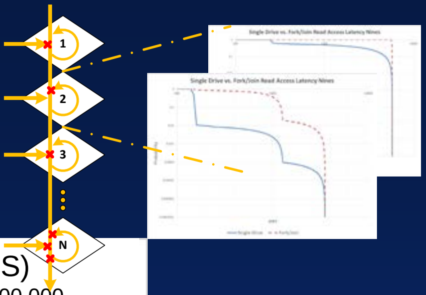
TLC vs QLC in a Hyperscale Environment

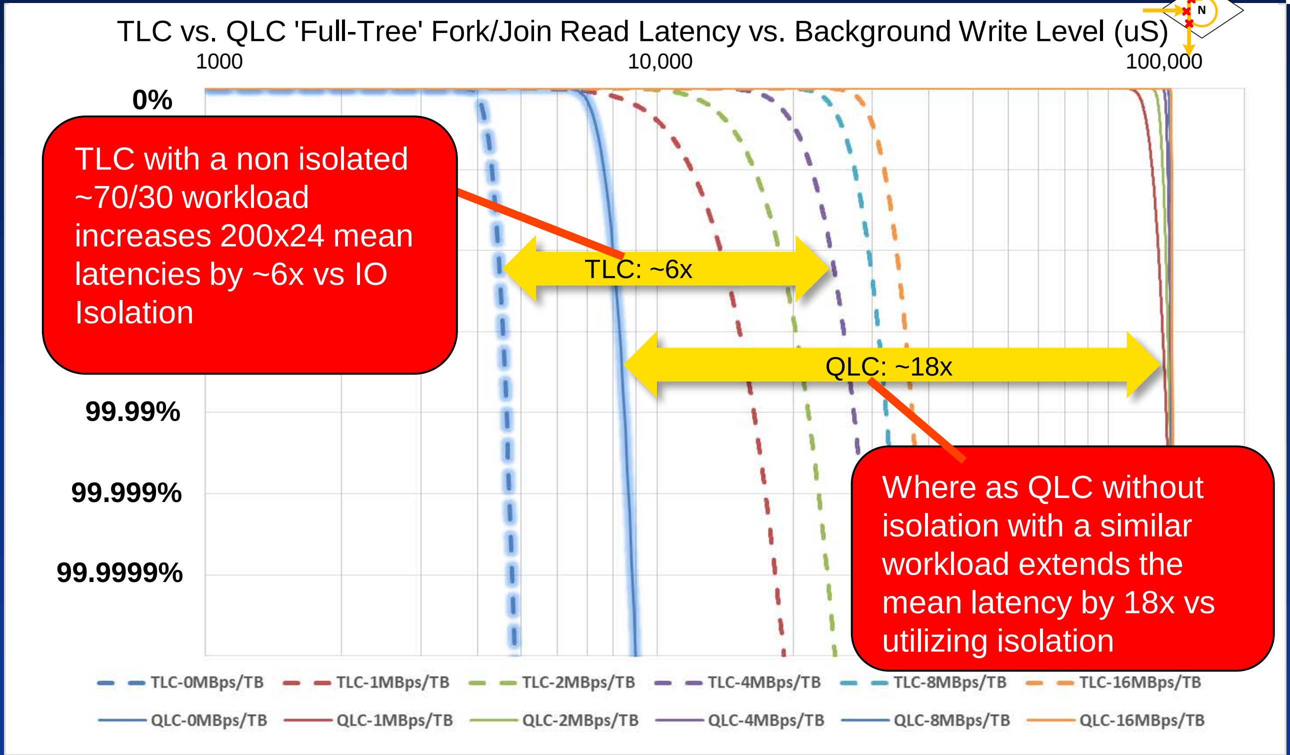




Simulation Data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

TLC vs QLC in a Hyperscale Environment





Simulation Data: As with any test, the results and outcomes herein should not be interpreted as a guarantee or warranty of similar results. Results may vary, depending on the circumstances and conditions.

Summary

- Last year we demonstrated array isolation offering ~50x latency tail improvements
- The concept of Hyperscale Mean Latency (HML) is explored where low probability drive read tail latencies turn into mean latency impacts for hyperscalers given the breadth and depth of fork/join operations.
- Applying HML concepts to a TLC SSD tells us
  - Even 1% write rates without IOD impacts HML by 4x
  - A classical 70/30 workload without can impact HML by ~10x
  - NVMe™ IOD is an idea solution to address HML
- Background data refresh can meaningful impact HML and is recommended to utilize determinism modes of NVMe™ to mitigate
- QLC's longer program latencies can induce further HML latencies and values IO Determinism concepts even more than TLC

Please stop by booth #307 to see the latest offerings and technology demonstrations from Toshiba Memory America

TOSHIBA