Getting the Most Out of QLC-based NVMe Storage

NVME-202-1: PCIe/NVMe Storage

Andy Watson
CTO, WekaIO
watson@weka.io, @the_andywatson

How To Leverage QLC's Lower Cost

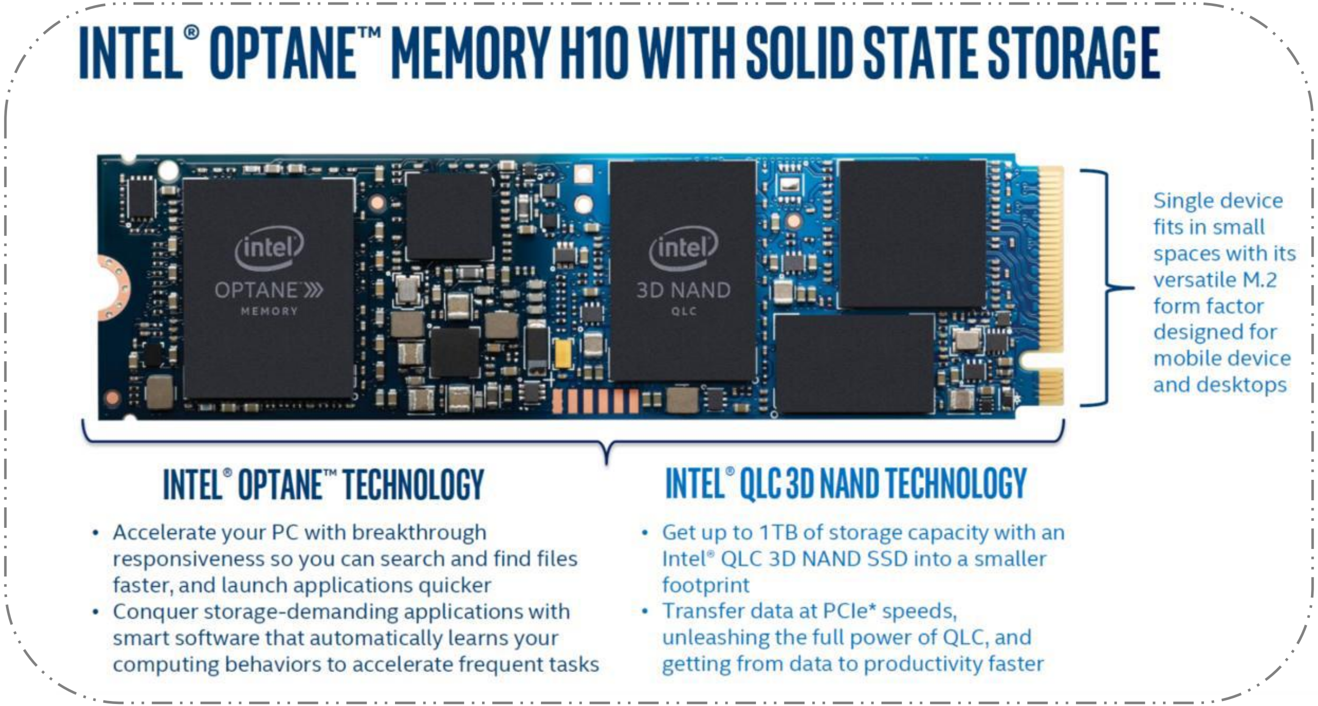
- Combine QLC with another layer of flash
  - Hybrid approach protects QLC durability *and* Accelerates Performance
  - Accelerate performance at lower overall cost

	SCM	SLC	MLC	TLC	QLC
Endurance (Write Cycles)	1M – 3M ~2M	20K – 100K ~50K	3K – 10K ~5K	500 – 2K ~1K	100 – 1K ~0.5K
~Cost per GB 2020+ *	< $1.00	< $0.80	< $0.40	< $0.20	< $0.10

* *These $/GB estimates are very approximate and only for relative comparison purposes*

- **Admittedly, not a completely new idea . . .**

For Laptops, A Shipping Product



The above was captured from https://www.pcworld.com/article/3389742/intel-optane-memory-h10-ssd-review.html

For Fileservers, Not So Simple

- In that simple Intel “H10” hybrid combination —
  - Big files & Sequential IO are sent to the QLC device
  - Small files & Random IO are sent to the Optane device
  - For a single user on a laptop, very workable …
  - … but not appropriate for a petabyte-scale fileserver
- Need a more nuanced hybrid-QLC strategy
  - Server performance expectations are higher
  - Workloads & data sets are more complicated

No Writes Go Directly to QLC

- Incoming Writes go first to non-QLC flash
  - All data lands in “Front Layer” (e.g., TLC or 3D-XPoint)
  - Process data before move or copy to QLC layer
- Concerns: size & frequency of data being written
  - Absorb Write traffic with durable Front Layer
    - Mitigate wearing out QLC with high rate of small Writes
    - TLC and 3D-XPoint also offer lower-latency Write perf
  - Small Writes are more suitable for Front Layer
    - QLC 8-KB page size is larger than Front Layer 4-KB page size

Absorb Volatility in Front Layer

- Filesystem Metadata Updates
  - On a fileserver, directories are often rapidly updated
  - Similarly, other metadata can be aggressively modified
    - Repetitive inode updates (i.e., to *atime*, *ctime*, *mtime*, etc.)
- Read-Modify-Writes and Appends in general
  - Journaling and Logs can be hammered hard
- Strategy: Wait, then Coalesce
  - Commit updates to Front Layer, then subsequently . . .
  - . . . Rewrite to QLC after overwrites *quiesce*

Group Data By ETTL

- Organize data by ETTL (Expected Time To Live)
  - How long until it will be modified or deleted
  - Blocks of QLC pages must be erased together
    - Therefore, ideally data written together in adjacent 8KB pages can “age out” together
- Improved capacity utilization
- Extended Endurance via Reduced Churn

Low-Level QLC Writes

- When writing to QLC layer, if available …
  - … Leverage a PCS (Page Collection Scheme) *
    - – Fill higher % of 8-KB QLC pages (16 512-byte sectors)
- Write whole stripes whenever possible
  - More efficient erasure coding (or RAID)

_*https://thesai.org/Downloads/Volume9No11/Paper_64-Efficient_Page_Collection_Scheme.pdf_

Net Effect of All These Techniques

- Enable embracing QLC for high-performance file-server configurations while:
  - Improving overall system performance
  - Lowering Overall System Cost-per-GB
  - Extending Life of QLC
  - Improving QLC Capacity Utilization

WekaIO Possibility

Incoming Writes

TLC (50–500 TB)

segregate metadata

coalesce, then sort by TTL (Time-To-Live)

prioritize to QLC based on LRU + longest TTL

QLC (1–100's PB)

PCS (Page Collection System)

Background tasks: indexing; global dedupe operations; async replication

S3-API Object Storage *

One or More

On-Premises

or

Cloud-based S3-API systems

**Note: Data can also move directly between the TLC and Object Storage without passing thru QLC*

WEKA.IO
World's Fastest File System

Other Considerations

- Optionally send all writes to 2 QLC devices *
  - Parity-rebuilding a failed 256-GB QLC device could take > *a week*
  - Copying from a surviving mirrored device should take < *2 days*
    - In the era of PCI 4.0, this would be < 1 day
- But mirroring QLC probably will not make sense until prices fall even lower

***Local replication, distinct from remote replication for system-level Disaster Recovery purposes*