KIOXIA

Technical Brief

KIOXIA Software-Enabled Flash™ Technology:

How Can Hyperscale Environments Best Enable QLC Flash Memory?

Hyperscale data centers are deploying massive amounts of flash memory to power their real-time applications and services. Forward Insights market research predicts that hyperscale data centers will deploy over 196 exabytes of flash memory by 2024[1]. This incredible growth is fueled by the latest flash technologies, of which quad-level cell (QLC) flash memory is at the forefront. While it delivers unique benefits and business advantages in hyperscale environments, obtaining the most from QLC flash memory can require a different approach than from prior flash memory generations. Software-Enabled Flash™ technology developed by KIOXIA simplifies the transition to QLC flash memory while enabling hyperscale environments to maximize its capabilities and value.

Optimizing QLC Flash Memory

Hyperscale developers optimize their data centers to improve application performance - from the highest level algorithm to the lowest-level mechanical server rack design. QLC flash memory can assist in this optimization as it delivers a 33% increase in bit density over triple-level cell (TLC) flash memory (Figure 1), providing a great value proposition for hyperscale data centers.

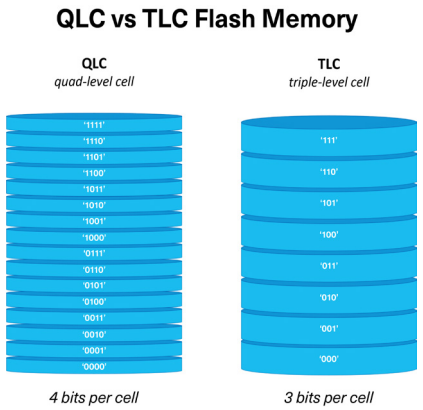

Figure 1: QLC flash memory delivers a 33% increase in bit density over TLC

Solid state drives (SSDs) based on QLC flash memory are designed to address the needs of a wide variety of applications, but hyperscale developers need more control to achieve optimal results for different application classes. This includes tight latency control in interactive server clusters to help meet critical service-level agreements (SLAs). It also includes fine-grained control of data placement in streaming clusters with heavy write workloads to help maximize flash memory lifecycles. Additionally, hyperscale developers need to manage asynchronous drive-initiated background processes to help support predictable performance under varying conditions.

Software-Enabled Flash technology provides tools that help hyperscale developers to meet these needs and achieve optimal performance from their flash deployments, including those based on QLC flash memory.

Introducing Software-Enabled Flash Technology

Software-Enabled Flash technology fundamentally redefines the relationship between hyperscale applications and solid-state storage media. It consists of purpose-built, media-centric flash hardware (Figure 2) focused on hyperscale requirements, and features an open source API and libraries (Figure 3) that enable flash memory to be used to its full potential.

The software API enables hyperscale applications to direct and manage flash allocation with additional capabilities that automatically control low-level details such as wear leveling, flash memory programming algorithms, and more. The ability to abstract and manage low-level flash tasks not only improves performance and extends flash memory life, but also simplifies application portability between flash memory generations and flash storage vendors.

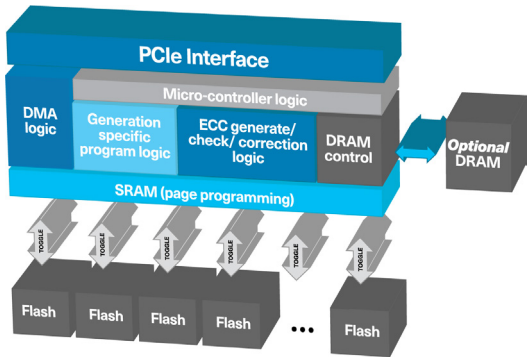

Figure 2: Software-Enabled Flash technology consists of purpose-built, media-centric flash hardware

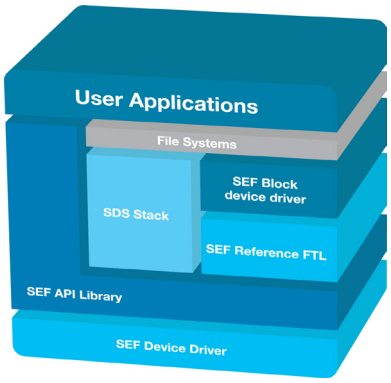

Figure 3: Software-Enabled Flash technology host software block diagram

Unique Software-Enabled Flash Technology Capabilities

Software-Enabled Flash technology has a powerful method of protecting hyperscale applications from interfering with each other, delivering flash hardware isolation with virtual devices, and software isolation with Quality of Service (QoS) domains. Virtual devices (Figure 4) provide application-controlled isolation with granularity that can be as small as individual flash dies or as large as entire drives. This hardware isolation ensures that individual hyperscale applications will not have to compete for flash resources, minimizing latency excursions.

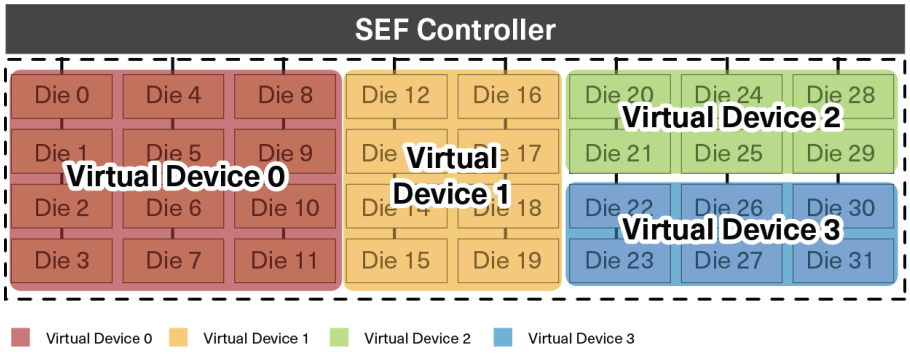

Figure 4: Software-Enabled Flash technology supports virtual devices and application-controlled isolation

QoS domains isolate drive operations on a software-defined, thin-provisioned, application-controlled basis, and are logically layered on top of virtual devices. In this scenario, applications that are assigned to specific QoS domains have tuned queueing parameters, specified flash quotas, and segregated data, all while sharing an individual virtual device. These QoS domains (Figure 5) provide a method of isolating drive operations between tenants and help to ensure consistent service levels.

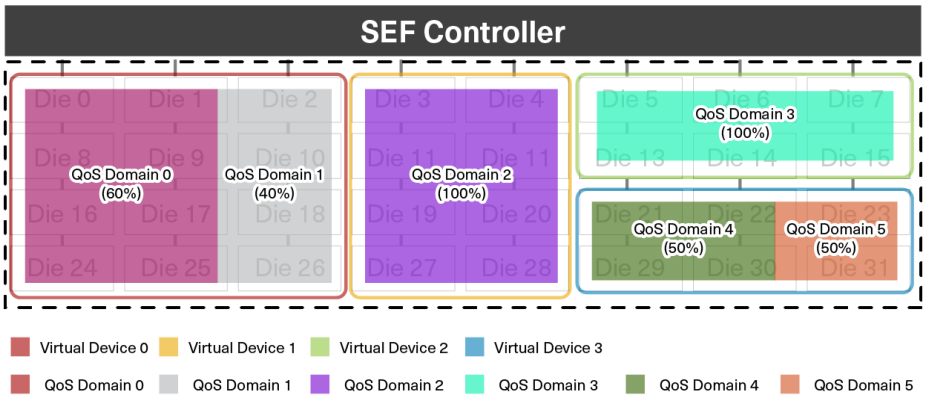

Figure 5: QoS domains isolate drive operations and provide advanced queueing capabilities

Software-Enabled Flash technology also features intelligent offload capabilities: *Nameless Write and Nameless Copy*.

Nameless Write	Nameless Copy
Enables applications to perform logical writes to the drive, while enabling Software-Enable Flash technology to select the best and healthiest location on flash memory to store the data using its knowledge of flash technology, write lifetime, and isolation requirements.	Extends the Nameless Write capability further by enabling offloaded data movement between flash memory locations (for processes that may include garbage collection) without requiring host CPU intervention or wasteful data transfer over the PCIe® bus. A single nameless copy command can perform hundreds of thousands of data movements that dramatically reduce host CPU usage.

Maximizing QLC Flash Lifetime with Application-Assisted Data Organization

Due to its physical structure, flash memory can only be written a fixed number of times before it loses the ability to faithfully store data. This includes all writes to flash memory, whether directly from the application and operating system, or from internal drive background processes (such as garbage collection). Hyperscale applications can reduce the amount of data written with careful algorithmic design, but they lack the ability to control the amount of data processed silently by the drive itself while managing the flash memory. In effect, a single application write can result in multiple flash memory writes. The ratio between flash memory writes and application requested writes is called the Write Amplification Factor (WAF). Minimizing the WAF can maximize flash memory life in any application.

Write Amplification Factor
$WAF = \frac{\textit{Data written to flash memory}}{\textit{Data written by the host}}$

When data with different lifetimes is placed in a single block of flash memory, the WAF will increase. As portions of the data grow old and become invalidated, the drive needs to perform a garbage collection read and write operation to reclaim the stale data's space. Writing the same data multiple times is inefficient, impacts performance and consumes QLC flash memory life. Software-Enabled Flash technology enables applications to minimize WAF and maximize QLC flash memory lifetime with mechanisms to group data with similar lifetimes at the flash memory block level, and with QoS domains and placement IDs.

As discussed earlier, QoS domains isolate drive operations on a software-defined, thin-provisioned, application-controlled basis. Individual applications can be assigned to one or more QoS domains, with the open source API ensuring that data from one QoS domain is not mixed with data from any other QoS domain within a block. This prevents any tenant from impacting the WAF of any other tenant (since processes such as garbage collection operate on units of flash blocks).

Multiple placement IDs within each QoS domain provide an even finer-grained method of isolation that enables an application to notify a drive to place data with similar lifetimes in specific allocation units. This results in less fragmentation of valid data in a block, which in turn requires less garbage collection and minimizes the WAF.

Software-Enabled Flash technology also implements additional techniques to maximize QLC flash memory lifetime by managing its write process without application involvement. Using its proprietary knowledge of the QLC flash cells (including the history and state of all flash memory on the drive), the drive chooses the most appropriate flash block for each QoS domain and placement ID. It implements QLC flash specific programming techniques (such as multi-phase programming), without application changes. Choosing the most appropriate flash block in combination with the specific programming techniques available with Software-Enabled Flash technology helps ensure that the flash memory within each deployed drive wears evenly and that QLC flash memory life is maximized.

Advanced Queueing Modes Enhance QoS

Within an idle system, ensuring QoS is relatively easy since no other operations can interfere with the resource needs of an application. In a hyperscale data center, systems are never expected to be idle and there always seems to be contention for resources, whether it's the CPU, DRAM, networking, I/O or storage. Queueing helps to manage requests for limited resources and is employed extensively at the operating system and application level. However, not all requests, and not all applications, are equally important, and latency spikes imposed by queueing delays can result in cluster-wide impacts on QoS.

For example, a cluster operation (such as a search query or distributed database update) generally cannot be completed until all cluster members report back with their individual results. The slowest member of the cluster dictates overall server performance. Therefore, if just a single server experiences an extended queueing delay on a read operation, the entire cluster of hundreds of servers can suffer performance dips. These latency outliers can be caused by variances in individual read or write timings due to resource conflicts within the drive itself. Hyperscale applications can partially compensate for latency outliers by overprovisioning performance and capacity to minimize these conflicts, but that can limit flash memory effectiveness.

Noisy neighbors[2] are another cause of latency outliers. Since hyperscale applications are highly threaded and containerized, a single host could be tasked with running many jobs at once, and those individual tasks can interfere with each other. For example, if a big data analytical process begins to stream data from a local drive, other real-time applications on that server could suffer unacceptable latency impairments from this co-tenant's monopolization of the storage resource.

Software-Enabled Flash technology's advanced queuing capabilities can help avoid these kinds of latency spikes and deliver three runtime configurable queueing modes for flash memory: ***priority-based queueing***, ***simple round-robin queueing*** and ***weighted fair queueing (WFQ)***. The advanced queueing capability also implements multiple, parallel sets of flash operation queues that enable application I/O requests and background operations to be segregated from each other and individually prioritized.

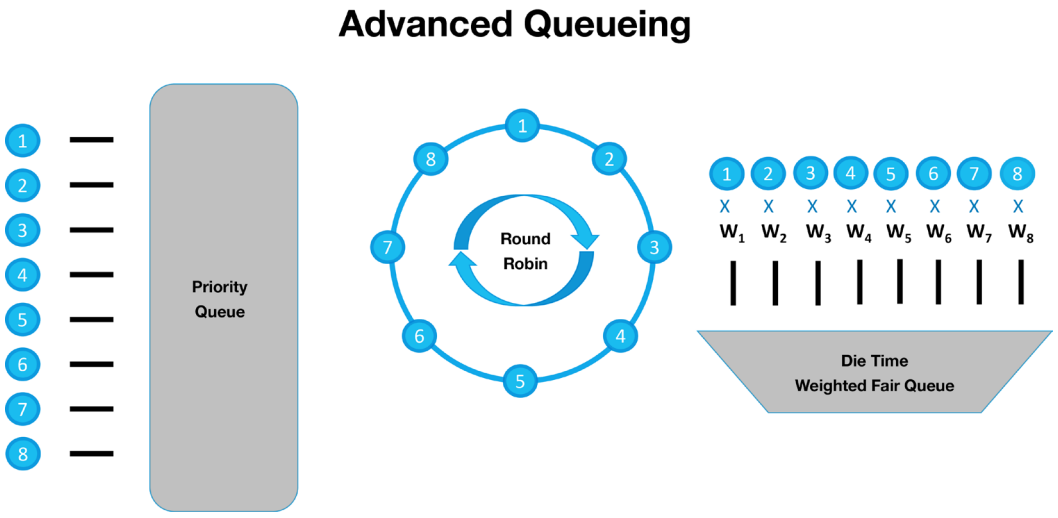

Figure 6: Software-Enabled Flash technology supports three configurable queueing modes

The advanced queueing capability also supports multiple, individual sets of flash queues to help reduce the queue head-of-line blocking[3] phenomenon. This is important for all flash, due to the difference in read and write latencies imposed by the flash memory construction, but can be critical when moving to QLC flash memory due to the larger difference between QLC flash read and write latencies. By using multiple queues, applications can help ensure that a critical flash memory read operation doesn't get delayed by a long-running, less important flash memory write operation on a different queue.

Controlling Background Processes at the Application Level

Garbage collection is a resource-intensive, asynchronous background process to reclaim space from unused or invalid data. Valid data must be copied to a new flash block (and the old one erased) before unused space can be reclaimed. The copying process consumes flash memory lifetime and can block other operations that could lead to unpredictable latency spikes.

Software-Enabled Flash technology mitigates these issues by providing full control of background processes and implements the Nameless Copy offload mechanism discussed earlier. It allows hyperscale applications to avoid the garbage collection process altogether on expired data or to defer the garbage collection process until absolutely necessary as a means of maintaining application performance. If garbage collection does need to be performed, the host or application can use the Nameless Copy offload mechanism (which obeys the advanced queueing policies already described) to collate and relocate data on the drive without involving the host CPU, host memory, or PCIe bandwidth, ensuring that application responsiveness is minimally impacted.

Summary

QLC flash memory is a key advancement for driving hyperscale data center efficiencies, and Software-Enabled Flash technology is a powerful method of maximizing its value. The technology provides software-controlled abstractions that help to solve real-world problems and control low-level flash tasks that maximize performance and improve flash memory life. It also simplifies application portability between flash memory generations and flash storage vendors.

KIOXIA Software-Enabled Flash technology is an open source project that aligns the unique and untapped capabilities of flash memory with the specific requirements of hyperscale environments. The project includes the KIOXIA Software-Enabled Flash technology home page with white papers, presentations and demonstrations, available at https://softwareenabledflash.com, as well as the open source API definition and specification document downloadable from the KIOXIA repositories on the GitHub® site.

NOTES:

[1] Source: Forward Insights, 'SSD Insights Q4/20,' Report No. FI-NFL-SSD Q420, published November 2020.

[2] A noisy neighbor is a hyperscale co-tenant that monopolizes bandwidth, SSD I/O, CPU and other resources, and can negatively affect other co-tenants' performance.

[3] Queue head-of-line blocking occurs when a series of I/O operations in an I/O queue are delayed by a long-running operation in the first slot of that queue.