SOFTWARE-ENABLED FLASH™

Software-Enabled Flash™ Technology:

Understanding How Data Placement Can Increase Flash Value

TECHNICAL BRIEF

Software-Enabled Flash (SEF) technology redesigns the storage stack allowing hardware to handle low-level flash management tasks while providing applications with the power to create optimized data layouts, to define isolation requirements and to prioritize latency profiles. The foundation of this refinement is a new flash memory hierarchy that starts at the lowest level flash dies, combines them into new Virtual Devices and divides them into upper-layer Quality of Service (QoS) Domains and placement IDs. Understanding how this new hierarchy works together can enable developers to build more robust and responsive applications, delivering the best value from the flash memory installed in their data centers.

Flash Hierarchy in a SEF Unit

A SEF Unit consists of tens to hundreds of flash dies connected to an onboard controller. The controller communicates to these dies over multiple shared flash busses and manages low-level flash operations such as erasing, programming and error correction coding (ECC). Figure 1 is an example configuration with multiple flash busses and a PCIe® interface for communicating with the host.

When configuring a SEF Unit, the user can choose the number of flash dies needed for specific tasks or isolation requirements, and aggregate them into a Virtual Device. These dies may share flash communication busses, but they are independent and can perform internal flash operations such as erasing or writing in parallel. A SEF Unit may have one or many Virtual Devices, depending on the needs of the application, as shown in Figure 2.

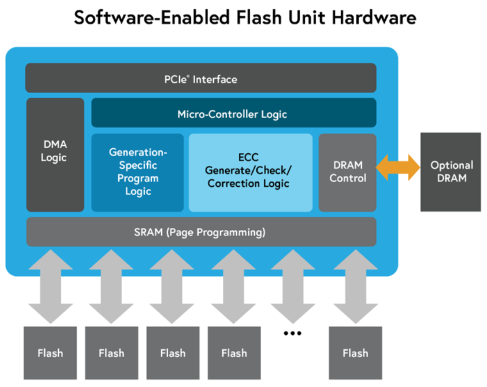

Figure 1: Example of a SEF Unit hardware configuration.
(Used with permission from KIOXIA America, Inc.)

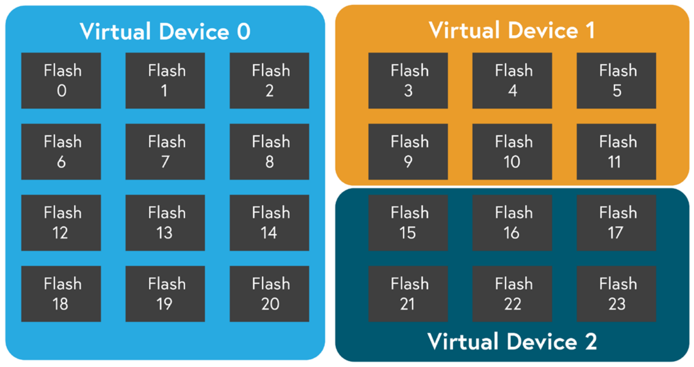

Figure 2: Example of twenty-four flash dies configured into three separate Virtual Devices.
(Used with permission from KIOXIA America, Inc.)

A pool of super blocks are contained within a Virtual Device, as shown in Figure 3, where a super block is the standard garbage collection reclaim and erase unit. It may contain from tens to thousands of addressable data units (ADUs). A single size of super block is defined for each Virtual Device. The super block width configuration can be smaller than the number of dies in a Virtual Device, but never greater. There is a tradeoff between super block size and garbage collection processes: The larger a super block, the fewer blocks that are available for garbage collection, and a larger amount of work to clean them.

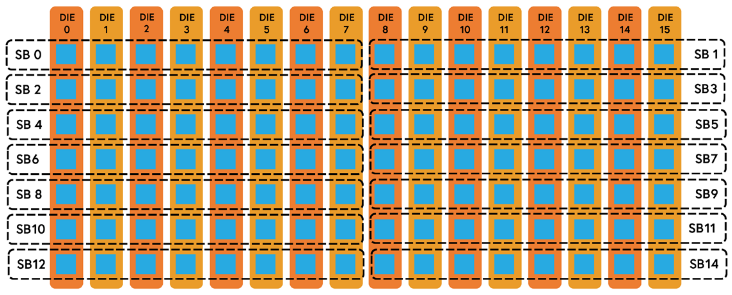

Figure 3: Example super blocks spanning eight flash dies in a Virtual Device containing sixteen total dies.
(Used with permission from KIOXIA America, Inc.)

The super block pool in a Virtual Device is available to applications via a QoS Domain abstraction. A single Virtual Device supports multiple QoS Domains. QoS Domains are thin-provisioned, which means it is possible to set minimum reserved and maximum allowable quotas. As data is written to QoS Domains, the SEF Unit allocates super blocks from the Virtual Device pool. When the data in these blocks are garbage collected, freed or the QoS Domains destroyed, they return to the Virtual Device pool for use by any other QoS Domain.

The abstraction of the QoS Domain includes an additional layer of isolation referred to as placement IDs. This capability allows data from a single application working in a QoS Domain to be isolated into separate physical locations in flash memory. Placement IDs are most useful in applications where there are multiple data streams with very different write and update characteristics. For example, a relational database may store its transaction logs and table data using different placement IDs. As super blocks are allocated from the Virtual Device pool for placement ID writes, they are charged against the QoS Domain quota to which they belong. While the super block abstraction is for writing and erasing data on flash memory by the SEF Unit, read operations from flash memory can be as small as an individual ADU.

SEF Unit Manages Flash Memory / Application Organizes Data

The SEF Unit handles all management of Virtual Devices, the super block pool, QoS Domains and their quotas. Flash housekeeping operations, such as managing the super block pool, are completely contained in the SEF Unit. It also performs the low-level operations necessary to program, erase, read, and error correct data from flash memory.

An application developer never needs to be aware of the actual physical layout of data in flash memory or low-level flash operations. Developers are free to concentrate on the different lifecycles and access patterns of their data, giving the SEF Unit hints via the QoS Domain and placement IDs on the best way to organize it. This separation of concerns help to make the migration between different flash memory vendors, different flash technologies, and even different generations of the same flash memory easier.

Lifecycle of Virtual Devices and QoS Domains

Virtual Devices and QoS Domains have different lifecycles. A Virtual Device is created when a server is deployed because internal flash processes, like wear leveling, would not be feasible if the Virtual Device was modified on the fly. Furthermore, if multiple Virtual Devices are required, all of them must be created at the same time.

To create a Virtual Device requires using the SEF application programming interface (API) or the SEF command line interface (CLI) as part of server deployment. Several major decisions are required at this time regarding flash memory use, such as:

- ***Number of dies to use***
  Defines the maximum super block size and performance. Generally, all dies will be allocated among one or many Virtual Devices set up because any flash die that is not allocated to a Virtual Device can't be used to store user data and will be wasted.

- ***Die organization***
  Defines organizational items such as flash channel sharing. This is normally not a major concern because the flash die operations are significantly longer than the shared bus cycles needed to communicate with them. However, if all dies in a Virtual Device are on the same flash channel then parallel performance may suffer.

- ***Super block sizes***
  Defines the super block size for all QoS Domains that are part of the Virtual Device. Larger super blocks may result in higher performance for write intensive workloads at the expense of the garbage collection effort. Smaller super blocks may give equivalent performance in most cases, dramatically easing the garbage collection efforts. If an application can benefit from both large and small super block sizes, it will need to generate two separate Virtual Devices to allow setting different super block sizes.

- ***Number of read queues***
  Defines the number of read queues to generate and share amongst the contained QoS Domains. There may be specific SEF Unit limitations as to the total number of queues supported, but in general the higher the number of queues, the better.

- ***Default weights***
  Defines default weights for flash operations such as read, write, copy and erase. The default weights can be overridden by individual QoS Domains and are only used when the QoS Domain generation does not specify overrides.

QoS Domains and associated placement IDs can be created or destroyed at run time as needed. For example, when a new container uploads, a QoS Domain can be generated and assigned to it for ephemeral data. When that container exits, the QoS Domain can be deleted and its used super blocks released back to the pool.

QoS Domains are generated using the SEF API or SEF CLI. Each QoS Domain has important configuration settings that the application or management layer can tune to meet specific application needs. Examples of these settings include:

- ***Minimum flash capacity***
  This explicitly reserves a portion of the total flash capacity for a QoS Domain. As long as the QoS Domain creation succeeds, this amount of space will be available to an application and not used for any other QoS Domain.

- ***Maximum QoS Domain quota***
  This sets a limit to the size that a QoS Domain may grow. The quota can simplify management operations since it does not require a hard limit on any QoS Domain.

- ***Encryption keys, if applicable***
  This enables the use of unique encryption keys for each QoS Domain, allowing for better isolation and protection against data leakage.

- ***Number of placement IDs and maximum number of open super blocks***
  This setting determines the number of placement IDs available for an application to tailor its data placement at an input/output (I/O) level. For example, it may be possible to use one placement ID for a database transaction log with ordered sequential writes, while another set of placement IDs can be used for random table updates. The ordered writes would be kept in their own set of super blocks, avoiding an I/O blender effect[1] caused by the random table updates. The maximum number of open super blocks has an effect on memory usage since each block requires system state and buffering.

- ***Selection of which read queue in the Virtual Device to use by default***
  This assigns separates different tenants' read workloads to different queues for better control of individual application tail latencies.

- ***Default weights for flash operations like read, write, copy and erase***
  This defines default weights for specific flash operations, but any weight can be overridden on an individual I/O basis if required.

Benefits of Virtual Devices, QoS Domains and Placement IDs Working Together

The combination of Virtual Devices and QoS Domains (including placement IDs) provides many benefits for modern workloads. The hardware-based isolation afforded by assigning different workloads to different dies via Virtual Devices can significantly reduce latency outliers for critical workloads on a shared device, avoiding noisy neighbor problems. The layering of QoS Domains on top of a Virtual Device (Figure 4) provides powerful control over workloads, allowing applications to manage their data placement, and in their latency outcomes, without needing to manage low-level flash operations.

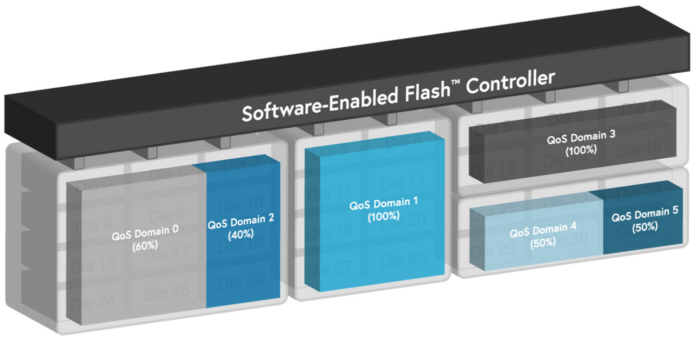

Figure 4: Example of SEF Quality of Service Domains configuration
(Used with permission from KIOXIA America, Inc.)

The Software-Enabled Flash Project, managed under the Linux Foundation®, provides a vendor neutral collaboration environment with a repository of documentation, source code and discussions related to this new technology. For more information about Virtual Devices, QoS Domains, and the rest of the Software-Enabled Flash technology stack, visit https://softwareenabledflash.org.

FOOTNOTES:

[1] The I/O blender effect occurs when multiple virtual machines (VMs) simultaneously send I/O to a storage subsystem causing the sequential I/O to become highly randomized, which in turn can increase latency in the storage system.