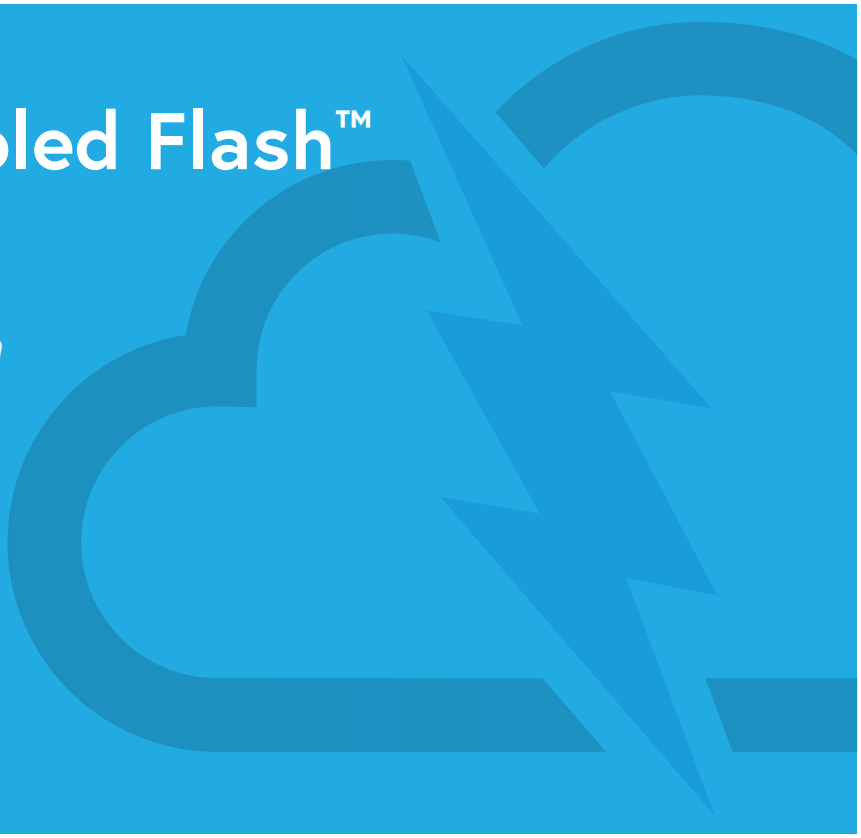
SEF

SOFTWARE-ENABLED FLASH™

Software-Enabled Flash™ Technology:

Presenting Use Cases in Hyperscale and Beyond



TECHNICAL BRIEF

Software-Enabled Flash (SEF) technology is a new and powerful way of deploying and using flash memory in the data center. It is managed by the Software-Enabled Flash Project as an open source initiative hosted by The Linux Foundation®. The technology combines purpose-built hardware (the SEF Unit), with an open source software layer that runs on the host and can turn flash memory into a software-defined flash-based storage resource. SEF can maximize the value of flash memory in a wide variety of use cases from a single server up to an entire cloud infrastructure. This technical brief showcases some examples of how SEF technology can change the way applications are deployed and run, such as:

- *Avoiding the 'RAID Tax' on Cloud-scale NoSQL Databases*
- *Optimizing Virtual Machine (VM) and Container Ephemeral Storage with Data Placement Control*
- *Customizing Input/Output (I/O) Service Levels for VMs and Containers*
- *Fine-tuning Performance Characteristics for Storage Arrays*
- *Maximizing Content Delivery Network (CDN) Edge Caching*

Avoiding the 'RAID Tax' on Cloud-scale NoSQL Databases

Scale-out NoSQL databases are used in many cloud organizations to store and keep critical data available even in the presence of hardware failures. They do this by ensuring multiple replicas of the stored data are kept across one or more data centers. If a server fails, a network link dies, or a data center goes completely offline, the stored data can still be retrieved from one of the remaining online replicas.

In a traditional SSD, the data in each of those replicas is protected from errors in the onboard flash memory by a form of on-drive RAID striping (aka cross-die RAID). One or more dies (out of the many in a super block) are assigned to hold parity bits, not user data, as is comparable in a traditional RAID array where one or more complete drives is assigned parity function (Figure 1). For single-server, single-instance data, this is a very good tradeoff. If a flash die fails, the data can still be recovered using the parity bits. However, for replicated NoSQL data, setting aside this flash set for parity (or 'RAID tax') is not necessary. This enables the flash memory to be better utilized for storing more user data.

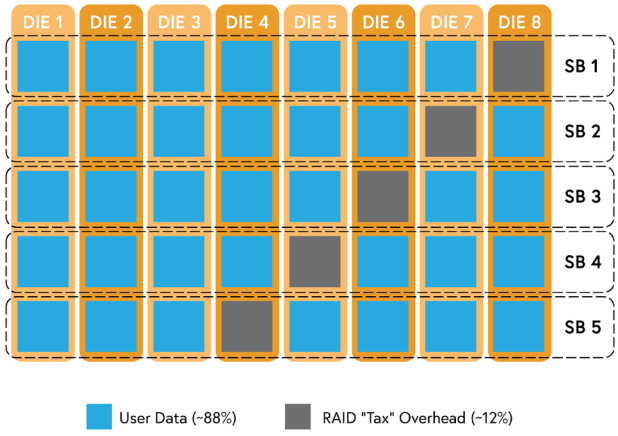


Figure 1: Example of cross-die 'RAID tax'
(Used with permission from KIOXIA America, Inc.)

How SEF Can Avoid 'RAID Tax'

SEF allows the application to eliminate cross-die RAID giving more of the onboard flash memory for storing usable data (Figure 2). If flash memory errors occur, the NoSQL database can simply redirect the read operation to another server with a copy. At cloud scale, where flash memory is a significant portion of infrastructure costs, freeing this capacity for user data can significantly improve flash economics. Furthermore, in cases where data is not replicated between servers, and there is a benefit from the extra data reliability of cross-die RAID, it can still be implemented under application control.

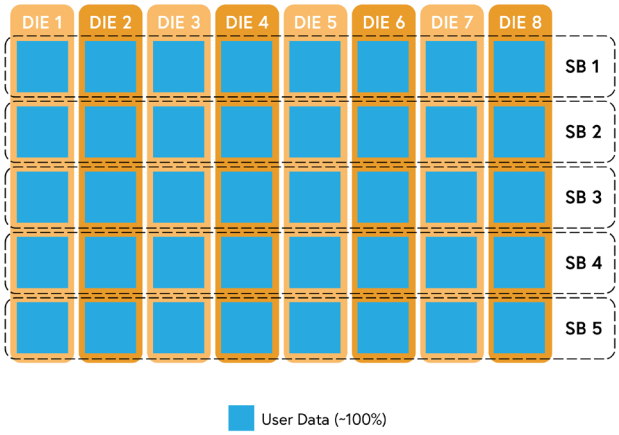


Figure 2: Example of no cross-die 'RAID tax' with SEF for use in replicated NoSQL databases
(Used with permission from KIOXIA America, Inc.)

Optimizing VM and Container Ephemeral Storage with Data Placement Control

Many jobs in the cloud are virtualized, using either complete virtual machines, containers, or a combination of the two. These jobs can often require local storage for tasks associated with temporary or intermediate files. After the task completes, the allocated ephemeral storage needs to be freed, allowing its reuse by another task.

This churn of task data in legacy block storage often results in data from one task residing alongside data from another task. Since flash memory is written in large units containing data from many tasks, this means that recovering the free space from one task may require doing garbage collection (GC) operations on flash super blocks containing many other tasks' data (Figure 3). Performing these tasks on intermixed user of application data can increase flash wear and result in inconsistent performance for the remaining tasks as garbage collection proceeds.

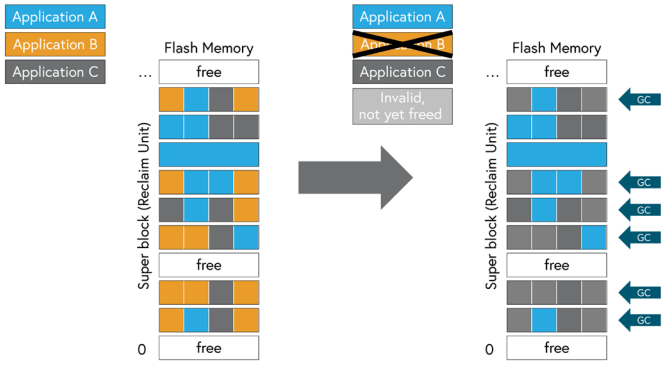


Figure 3: Intermingled data requires garbage collection (GC) when a task completes
(Used with permission from KIOXIA America, Inc.)

How SEF Can Optimize Ephemeral Storage Management

SEF can avoid this wasted performance and lifetime by isolating task data from all others at the super block (i.e. the garbage collection unit) level. This powerful placement control ensures that even if multiple tasks are writing to the SEF Unit at the same time, their data is never mixed. When a task terminates and its ephemeral storage needs to be reclaimed for the next task, no garbage collection is required, only simple super block erases (Figure 4). This frees up storage bandwidth, minimizes the data churn, and eliminates or reduces the need for garbage collection, all of which increases the amount of flash lifetime that is usable by the application.

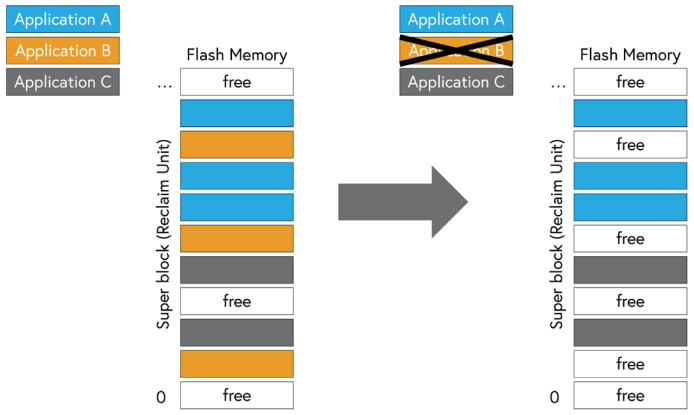


Figure 4: SEF data placement control removes the need for garbage collection on task completion
(Used with permission from KIOXIA America, Inc.)

Customizing I/O Service Levels for VMs and Containers

Providing the right level of I/O responsiveness is another common concern with VMs and containers. In both public and private clouds, not all jobs are of equal importance. Some require instantaneous responses, such as those in backend systems used for high-speed trading, while other jobs can handle large latency spikes and variations, such as scheduled backups.

The first step in providing these different service levels is to isolate the data using placement control, as already discussed. The next critical step is to control the priority of different I/O commands as they are being processed in the drive itself. Without this additional step, a long-running, low priority I/O could end up blocking a higher priority one I/O (Figure 5).

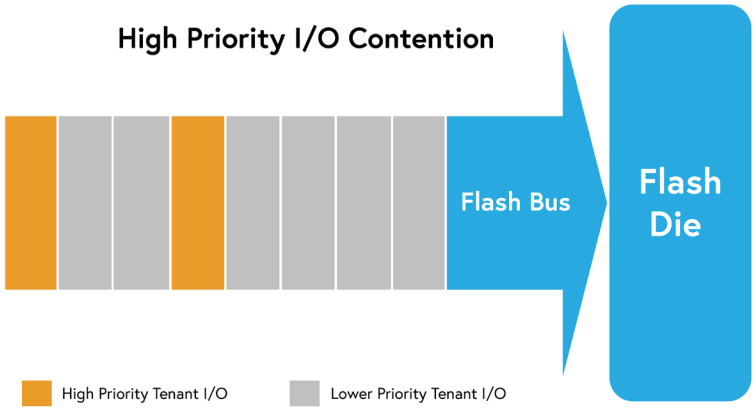


Figure 5: A low priority I/O can delay processing of a more important one
(Used with permission from KIOXIA America, Inc.)

How SEF Can Customize I/O Service Levels

SEF natively supports multiple weighted I/O queues within the SEF Unit. By using the advanced die-time weighted fair queueing available to applications, individual I/Os from multiple applications can be prioritized (Figure 6). Since this queueing mode is based on die-time (the amount of time an individual I/O operation requires on a flash die), it can provide high priority applications with data quickly while avoiding I/O starvation for lower priority tasks.

Figure 6: SEF features die time weighted fair queues that help high priority I/O avoid bottlenecks
(Used with permission from KIOXIA America, Inc.)

Fine-tuning Performance Characteristics for Storage Arrays

All Flash Arrays (AFAs) generally combine many flash drives and then split them, under user control, into multiple LUNs of various sizes. One or more LUNs are connected to an application over a communications protocol such as iSCSI or NVMe over Fabrics™ (NVMe-oF™). The performance characteristics of any LUN are often simply a function of its size and not under the control of the application. For applications such as high-throughput transactional databases, a higher than average sustained write performance may be needed.

One way of delivering higher sustained write performance is to deploy specialized SSDs with higher flash overprovisioning. Flash overprovisioning sets aside a certain amount of raw flash memory, not visible to the end user, to allow the SSD to handle write operations more efficiently. While increasing overprovisioning can improve sustained write performance, it can also significantly increase the cost per usable capacity of a drive.

To achieve different hardware overprovisioning levels without SEF, an AFA would need to deploy different SSD pools with various overprovisioning levels and assign LUNs from the appropriate pool. This process is complicated and can increase storage costs since the worst-case amount of high overprovisioned flash memory would need to be installed.

How SEF Can Fine-tune Storage Array Performance
SEF allows AFA LUNs to have customized performance characteristics without needing to deploy different drive types. Just as VMs or containers can have their I/O service levels tailored by SEF, different LUNs can be tailored to deliver different I/O priorities through SEF queueing capabilities. High priority jobs can, programmatically, have preferential access to the SEF Unit flash memory, while still allowing lower priority jobs to progress.

Using the SEF Software Development Kit, a single type of SEF Unit can be separated into different, customized regions, each with its own customized overprovisioning level. Instead of needing to statically deploy multiple SSD types to deliver different sustained write performance levels, the AFA can dynamically manage a homogenous pool of SEF Units to assign the right amount of write performance to each LUN as needed.

Maximizing CDN Edge Caching

Content delivery networks work like a global cache to accelerate delivery of web sites, games, video, and more to end users. They are built with a hierarchy of storage that includes highly available master copies of files to be served users, spread across hundreds or thousands of caching servers near the edge (to minimize end user latency). The more data that can be stored in the caching layer, the better the overall end user experience.

How SEF Can Maximize CDN Edge Caching
By using the control over flash memory that SEF technology provides, CDNs can also avoid the 'RAID tax' and maximize the useful data stored in their deployed flash memory. For cached data at the edge, no redundancy or additional error correction is necessary since it can always be re-requested and rewritten from the master copies upstream. Cache servers can skip the use of cross-flash die parity and make use of every available byte of flash memory to store actual data. This increases the cache's effective size and can deliver better and faster end user experiences with the same amount of hardware.

Summary: Making Flash Memory More Valuable in the Data Center

By combining the power of software-defined storage with the flexibility of a new hardware platform with built-in queueing and isolation capabilities, SEF technology can make the flash memory deployed in data centers more valuable and better suited for a wide variety of use cases. The use cases presented in this technical brief demonstrate just a few of the ways that SEF can maximize the value of flash memory in the data center.

For More Information

The Software-Enabled Flash Project, managed under the Linux Foundation, provides a vendor neutral collaborative environment with a repository of documentation, source code and discussions related to this new technology. For more detailed information, or to get involved with the project, visit https://softwareenabledflash.org.