SEF

SOFTWARE-ENABLED FLASH™

Software-Enabled Flash™ Technology:

Introducing Flexible Data Placement Capabilities for Flash Memory

TECHNICAL BRIEF

Software-Enabled Flash (SEF) technology unlocks the powerful digital and parallel capabilities of flash memory for hyperscale and storage developers. It combines purpose-built hardware, called the SEF Unit, with an open source software layer that runs on the host (Figure 1) enabling the flash memory to become a software-defined and controlled element of the application with powerful data placement, workload isolation and latency control capabilities. This new open source technology is managed by the Software-Enabled Flash Project under the auspices of The Linux Foundation®.

Data placement control capabilities, under the name Flexible Data (or Direct) Placement (aka FDP) are being defined by the NVM Express™ technical work group (as of this publication date). This technical brief examines how the native data placement control of SEF technology can be used to implement the data placement functionality of this in-development standard and can provide additional control over flash memory to storage developers.

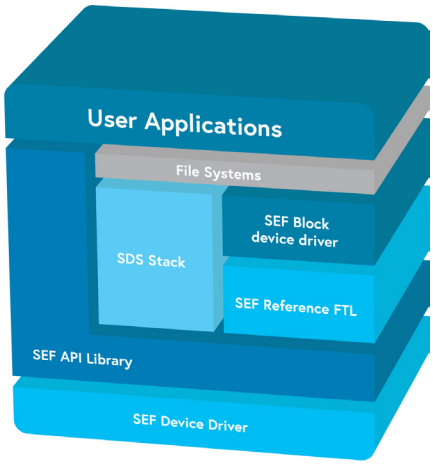

Figure 1: Software-Enabled Flash host software block diagram
(Used with permission from KIOXIA America, Inc.)

Why is Data Placement Important for Flash Memory?

Data placement control is foundational for both the proposed FDP feature as well as SEF technology (which has significantly more control capabilities). It has several key benefits to storage developers such as (1) Longer effective flash memory lifetime; (2) More consistent and tunable latency; and (3) Instant flash memory reclamation for terminated applications, as discussed below.

Longer Effective Flash Memory Lifetime

One of the driving forces behind data placement development is to decrease the flash Write Amplification Factor (WAF). WAF is a measure of the amount of useful data that is written to flash memory (consuming write lifetime) compared to the actual amount of data that the controller needs to write to flash memory in order to handle background operations and the underlying characteristics of the flash memory. In simplest terms, WAF is the ratio of data written to flash memory versus data written by the host.

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

While the underlying flash memory can support the same number of physical writes with or without FDP, more of those physical writes can be for useful application information and not used for flash memory housekeeping if the WAF is minimized. The optimal goal is when the WAF is equal to 1.0, meaning that all data that's ever written to flash memory comes from the application and not from any background maintenance operations. Minimizing WAF enables the flash memory to be deployed longer before running into lifetime write limits.

More Consistent and Tunable Latency

When the same flash memory address is written multiple times by an application, a traditional SSD simply marks the old copy of the data as invalid and writes the new data to a new available portion of flash memory. To actually free the newly invalid space for writes, garbage collection needs to be performed to coalesce the valid data and erase large chunks of invalid data. Garbage collection only occurs on a flash memory super block - a group similar to a RAID stripe containing multiple flash dies. If data from multiple applications are mixed in that super block, then one application can inflict garbage collection on neighboring applications. This activity can reduce performance of even well-behaved applications and can occur randomly, making it difficult to provide application-level guarantees of performance in shared systems.

With SEF technology, application data may be separated from neighboring application data, minimizing or eliminating this problem. All data in any super block can either be owned by one application or deemed invalid. As such, the application itself has the ability to minimize background garbage collection operations.

Instant Flash Memory Reclamation for Terminated Applications

Many cloud servers today support very large multi-tenant configurations. Whether virtualization or container-based orchestration is used, it has become common to have tens or hundreds of different applications running on a single server. When an application terminates, its ephemeral data can be discarded and the storage space can be reclaimed for use by other applications. The reclamation of this flash memory space often requires a garbage collection process (since data from the terminated application is co-mingled with other application data at the super block level). The garbage collection process not only consumes SSD CPU cycles and flash memory bandwidth, but also squanders flash memory write lifetime.

SEF allows an application's data to be separated from every other application's data in different super blocks. One application's super blocks can be immediately freed for erasure without needing to copy any other application's data. The result is an almost instant reclamation of a terminated application's flash memory space as depicted in the images below. Without data placement control, reclaiming terminated application flash memory requires multiple garbage collection (GC) operations (Figure 2). The gray boxes represent invalidated but not reclaimed flash memory space as only a complete horizontal bar (i.e., flash memory super page) can be reclaimed. With data placement control, terminated application flash memory can be immediately freed (Figure 3).

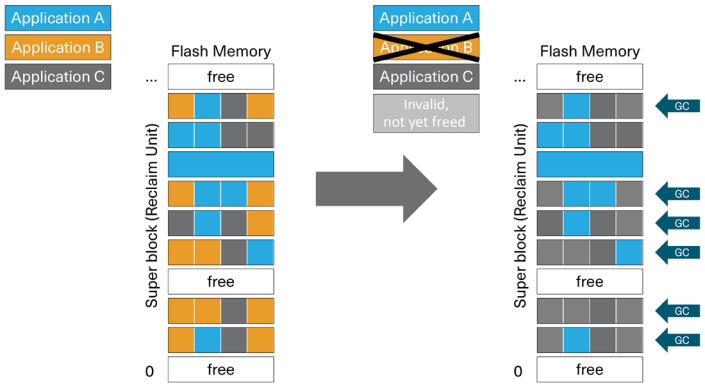

Figure 2: Flash memory space reclamation without FDP
(Used with permission from KIOXIA America, Inc.)

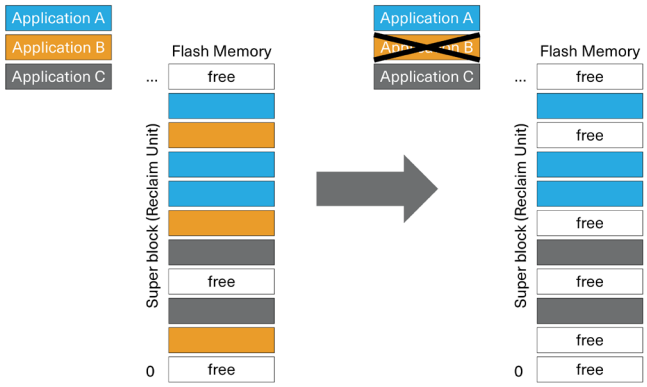

Figure 3: Flash memory space reclamation with FDP
(Used with permission from KIOXIA America, Inc.)

In the examples showcased by Figures 2 and 3, without data placement control for Application B (orange), there are 12 GC operations required and 6 super block erasures. With data placement control, there are no GC operations and only 3 super block erasures required to reclaim the flash memory.

How is Data Placement Functionality Implemented in SEF Technology?

There are two ways of implementing data placement-type control using a subset of SEF technology: (1) Native Software Development Kit (SDK) Flash Translation Layer (FTL) placement control using 'Placement IDs;' and (2) Paravirtualized FDP support with drivers that could implement forthcoming NVMe® extensions if the ongoing FDP proposals are approved by NVM Express, Inc., as described below:

Native SDK Data Placement

The SEF SDK has always supported the concept of data isolation through the use of 'Placement IDs.' Placement IDs are effectively tags that accompany any write operation and allow the open source FTL (included with the SDK) to keep application data separated. The SEFMultiContext Input/Output (I/O) operation structure (Figure 4) allows developers to explicitly assign a placement ID for each write operation, or to use the default one assigned to the Quality of Service Domain.

```
struct SEFMultiContext {
    SEFBlockHandle blockHandle;     /**< SEF Block handle to be used for access to the block instance */
    struct SEFMultiContext *parent; /**< A pointer to an instance used for compound operations */
    void (*completion)(struct SEFMultiContext *); /**< The function that is called when transaction done */
    void *arg;                                    /**< A pointer that can be used by caller for any reason */
    uint64_t lba;                                 /**< Logical block address */
    uint32_t lbc;                                 /**< Logical block count */
    enum SEFBlockIOType ioType;                   /**< The I/O Type that needs to be performed */
    uint8_t flags;                               /**< I/O flags enum SEFBlockIOFlags */
    char reserved[2];
    struct iovec *iov;  /**< A pointer to the scatter/gather list */
    int iovcnt;         /**< The number of elements in the scatter/gather list */
    uint32_t iovOffset; /**< Starting byte offset into iov array */
    struct SEFPlacementID placementID; /**< Placement ID for writes */
    atomic_int transferred; /**< Counter denoting number of bytes transferred for the transaction */
    atomic_int count;       /**< Reference count, I/O is completed -> 0 */
    atomic_int error;       /**< First error for the transaction */
    int cancel;             /**< Set to indicate cancel in progress */
};
```

Figure 4: The SEFMultiContext I/O operation structure with a built-in placement ID (highlighted).

The number of placement IDs can be chosen when the Quality of Service Domain is created and are managed by the SDK software and the SEF Unit. When garbage collection is performed, placement IDs are managed on a per-placement ID basis which means that application data will continue to remain separated even after the process terminates. Keeping this separation by placement ID helps ensure consistent, long-term performance and makes powerful features, such as instant reclamation possible even on a long-running system.

Paravirtualized FDP Support

The SEF SDK currently includes source code for drivers that support QEMU, a popular virtualization system, enabling virtual machines (VMs) to benefit from SEF technology without requiring any changes. Existing driver support includes emulating a standard block device, a zoned namespace (ZNS) device, or a raw NVMe device (Figure 5).

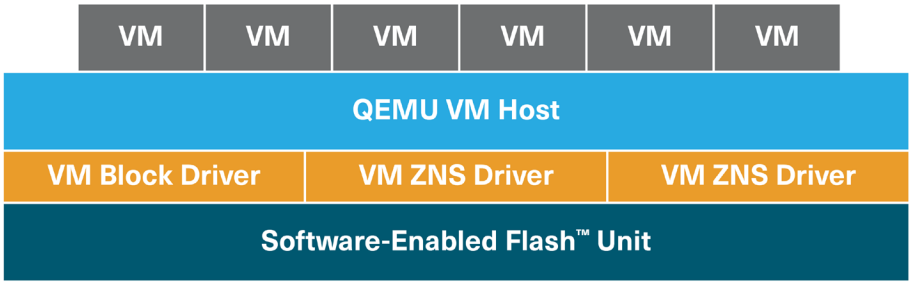

Figure 5: QEMU VM I/O driver block diagram
(Used with permission from KIOXIA America, Inc.)

If and when FDP is ratified as an extension to the NVMe specification, similar sample driver support may be considered by the SEF project. In addition, it will be possible for developers to add the new NVMe commands to the source code of the SDK's NVMe paravirtualized driver. Once implemented, virtualized applications would have access to the same FDP capabilities as bare metal applications.

SEF Technology Beyond Data Placement Control

SEF technology supports data placement control, today, along with many powerful features and functions beyond this capability. Examples include capabilities such as: (1) Hardware and software isolation; (2) Advanced queueing modes for better latency outcome control; and (3) On-drive copy offload for better host resource utilization, as described below:

Hardware and Software Isolation

SEF technology allows applications to partition a single drive into Virtual Domains, which are multiple, physically separate storage units consisting of one or more flash memory dies. These Virtual Domains ensure that workloads in one domain cannot physically interfere with workloads in another domain because they will always occupy different flash dies. On top of this physical isolation, an additional dynamic and software/firmware-based isolation is provided in the form of Quality of Service Domains. This higher-level isolation layer lets multiple applications share a physical flash die while isolating one workload's data from other workloads (Figure 6).

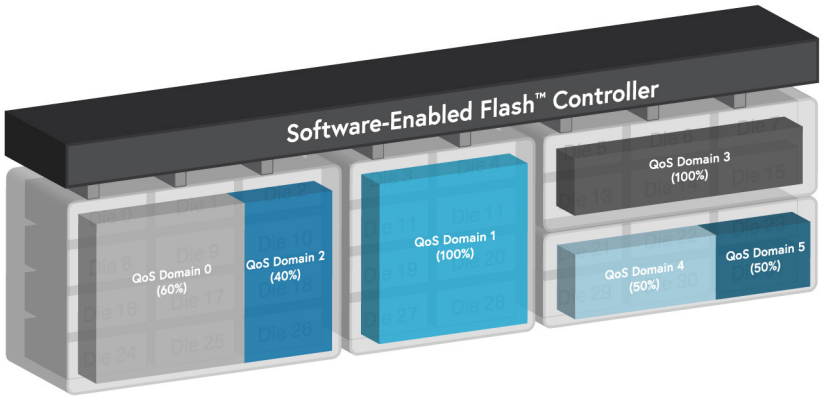

Figure 6: Example of SEF Quality of Service Domains configuration
(Used with permission from KIOXIA America, Inc.)

Advanced Queueing Modes

Workloads which reside in Quality of Service Domains are not just isolated from one another; the priority of their I/O operations can also be adjusted at an application level - at a granularity as small as an individual I/O level. For multi-tenant servers, as commonly used by public and private cloud providers, this capability allows individual workloads to be prioritized dynamically. For applications, advanced queueing allows individual I/Os to be prioritized to deliver the best user experience. For example, an application that performs a housekeeping operation on data can prevent I/Os associated with this background operation from impacting foreground interactive application performance.

On-Drive Copy Offload

The on-drive copy offload support within SEF technology allows for complicated on-drive data transfers, such as host-controlled garbage collection or database compaction, to be performed without host intervention. A single command can move massive amounts of data while the host attends to other work. This copying is completely handled on-drive, requiring no host memory or host PCIe® bandwidth to perform it.

More Information

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 detailed information and to get involved with the project, visit https://softwareenabledflash.org.