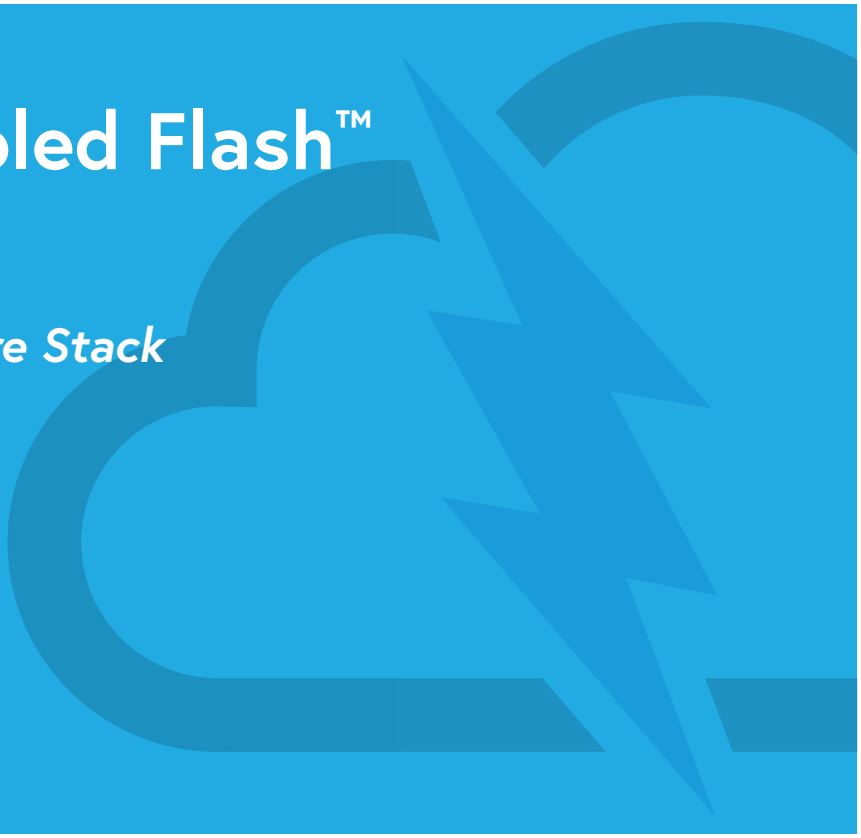
SEF

SOFTWARE-ENABLED FLASH™

Software-Enabled Flash™ Technology:

Introducing the Software Stack



TECHNICAL BRIEF

Software-Enabled Flash technology is a combination of purpose-built hardware and an open source software stack that provides unprecedented application control over flash memory behaviors that include:

- *Hardware/software-based isolation*
- *Application-directed data placement*
- *Latency outcome management via multiple dynamic queueing modes*
- *Simplified migration between flash generations and vendors*

These capabilities do not coexist in most other storage interfaces, so it's worthwhile to examine its components and how they work together.

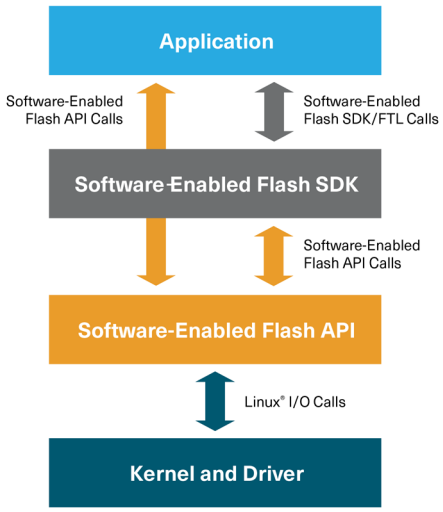


Figure 1: Software-Enabled Flash Software Stack

Full Software Stack

The software interfaces and components that comprise a full stack Software-Enabled Flash application is shown in Figure 1. An application can choose to use the Software Developers Kit (SDK), which will then communicate through the Software-Enabled Flash Application Programming Interface (API) and Software-Enabled Flash Unit (the actual hardware that implements the Software-Enabled Flash technology). Or, applications can use the API directly for more customized control.

Software-Enabled Flash Kernel and Device Driver – *Figure 2*

- *Handles communication to and from the Software-Enabled Flash Unit over the PCIe® bus*
- *Runs at elevated privileges and manages security features*
- *Does not require application changes between different flash technologies or vendors*
- *Converts user input/output (I/O) requests to the Software-Enabled Flash Unit hardware*

Figure 2: Software-Enabled Flash Kernel and Device Driver

Software-Enabled Flash technology requires: (1) Linux® kernel (version 5.10 or newer) for communicating with Software-Enabled Flash Units; and (2) A set of patches to enable optimizations in the NVMe® driver. The ***Software-Enabled Flash Device Driver*** communicates with the Software-Enabled Flash Unit controller, and the standard Linux 'syscall/ioctl' operations when interfacing with applications or libraries. Engineers in charge of provisioning servers will need to ensure the proper patched kernel and drivers are installed and enabled, but again, application developers generally will not need to concern themselves with this level of the stack.

Software-Enabled Flash API – *Figure 3*

- *Wraps the low level Software-Enabled Flash Command Set operations into a functional interface*
- *Requires very low overhead and memory*
- *Communicates to the kernel using standard system calls and to the upper layers through the API and callbacks*

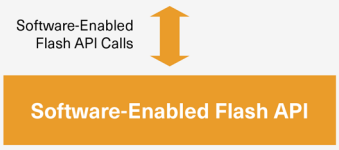


Figure 3: Software-Enabled Flash API

The ***Software-Enabled Flash API*** implements low level wrappers around the Software-Enabled Flash Command Set that make it easier for applications to communicate with the Software-Enabled Flash Unit. This open source API is compatible between different Software-Enabled Flash implementations and vendors. The API represents the lowest level of abstraction that application developers should normally utilize through the documented APIs.

The Software-Enabled Flash API provides calls which read, write and manipulate data directly on flash memory. Write operations are managed using a Nameless Write (NLW) concept that allows the Software-Enabled Flash Unit with the flexibility to assign the best possible flash location for application use without causing bottlenecks or contention while the operation is being performed. Copy operations between flash memory locations can be offloaded using a Nameless Copy (NLC) concept similar to NLW. Read operations are managed through physical flash addresses, not from traditional logical block addresses (LBAs).

Developers looking to implement their own unique flash protocols or Flash Translation Layer (FTL) into their applications will need to use this level of the software stack, while more general purpose applications built around the SDK will often not need to use the API directly.

Software-Enabled Flash SDK – *Figure 4*

- *Implements a reference FTL for use by applications*
- *Uses the Software-Enabled Flash API to perform operations on the Software-Enabled Flash Unit*
- *Exposes new I/O calls (similar to the asynchronous I/O calls already present in the Linux OS) to an application*

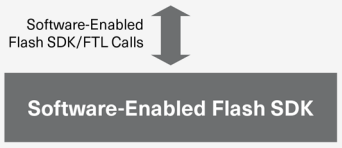


Figure 4: Software-Enabled Flash SDK

The ***Software-Enabled Flash SDK*** consists of an open source library that is built on top of the Software-Enabled Flash API and features a reference FTL that can be used by application developers to craft their own FTLs. The reference FTL handles the LBA to physical address mapping, metadata persistence, overprovisioning and garbage collection for an application. The SDK also provides unique features such as data placement control to help minimize flash management overhead. A Command Line Interface (CLI) based management tool allows the Software-Enabled Flash Unit to be provisioned upon deployment and can dynamically adjust Quality of Service (QoS) domains and FTL settings.

There are reference virtual I/O device drivers for QEMU[1] which allow virtualized machines (VMs) to use a Software-Enabled Flash Unit without changing a line of application code, while maintaining the technology's capabilities of tenant isolation, advanced queueing and latency outcome control. Many application developers will interface with this level of the stack, either directly through the documented SDK calls or via the virtual I/O device drivers in a virtualized system.

Software-Enabled Flash Application – *Figure 5*

- *Supports the use of the Software-Enabled Flash API, Software-Enabled Flash SDK, or both*
- *Performs I/O operations asynchronously*
- *Enables priorities and queueing options to come from applications, hypervisors and orchestrators*

Figure 5: Software-Enabled Flash Application

Applications which support Software-Enabled Flash technology have a broad array of software architecture options to choose from: virtualized access, general purpose application usage and advanced application usage (Figure 6).

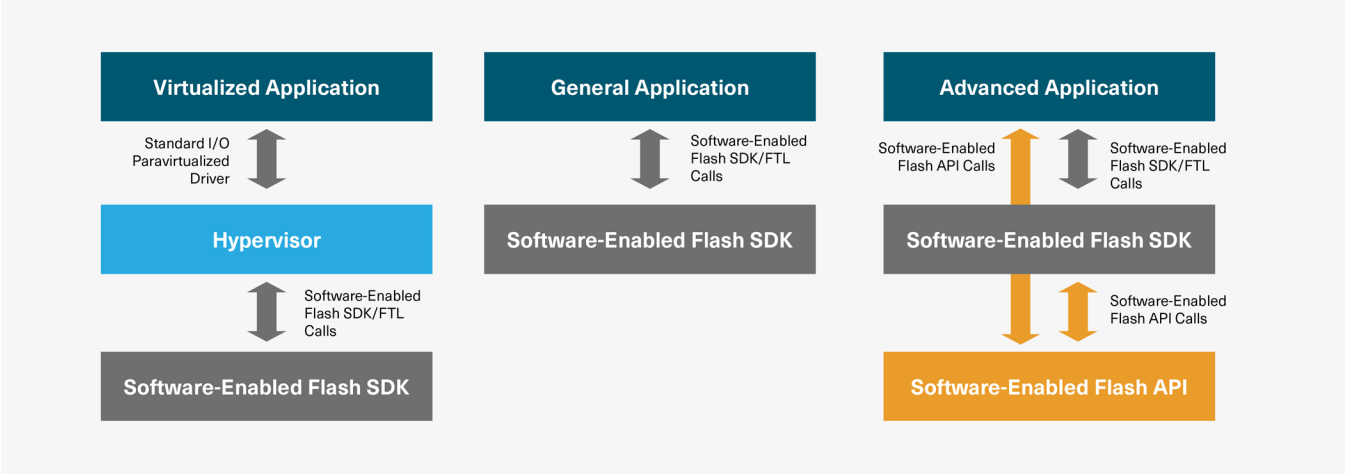


Figure 6: Software-Enabled Flash technology supported usage options

Virtualized Applications

Virtualized applications (Figure 6) often will not be aware that they are running on a Software-Enabled Flash Unit since the hypervisor may present paravirtualized storage devices[2] directly to VMs in operation. In this architecture, virtualized applications do not need to be modified to take advantage of Software-Enabled Flash technology. The hypervisor is modified to support the Software-Enabled Flash SDK, manages different VM FTLs and dynamically balances queue settings for running VMs. This enables the hypervisor to provide different isolation zones and classes of service on an as-needed basis.

General Applications

General applications (Figure 6), such as scale-out databases or caching systems, can make use of the Software-Enabled Flash SDK to provide storage accessed in fundamentally the same manner as traditional storage. The SDK provides a host-based FTL that the application can use instead of traditional block I/O, and can add tuning options such as garbage collection and overprovisioning. Applications with well-defined background processes (such as database compaction or time-based invalidation) can use SDK features to deprioritize those operations, which in turn can deliver better application response times. Applications can take advantage of SDK placement control to ensure that different portions of related data can be stored together, minimizing write amplification and garbage collection overhead. Additionally, the application itself or an orchestration tool can manage priorities between multiple applications on a single server.

Advanced Applications

Applications that can benefit from higher performance and finer control over flash memory via the Software-Enabled Flash API directly are advanced applications (Figure 6). Often these applications will find it useful to not implement a traditional FTL due to their unique requirements. Applications can avoid unnecessary levels of indirection as well as minimize their memory needs through the API, and can take advantage of the technology's offload features.

A Software Stack Built for Cloud-Scale

The fusion of software and hardware that make up Software-Enabled Flash technology empowers software engineers and cloud providers with unprecedented control over flash memory. Each layer of the Software-Enabled Flash software stack has been built with the needs of cloud-scale applications in mind, from the lowest level command set to the high level SDK. Developers are free to choose the level of abstraction that best meets their application needs while availing themselves of the latency control, isolation and other features provided by the technology.

Software-Enabled Flash technology is an open source project that aligns the unique and untapped capabilities of flash memory with the specific requirements of cloud applications and hyperscale data center environments. More information is available at the Software-Enabled Flash technology home page.

KIOXIA, member of The Linux Foundation, released an API definition and specification document to the open source software (OSS) community that is downloadable from the KIOXIA repositories on the GitHub® site. The site provides solutions that can help developers, architects and end users solve their unique flash storage challenges. KIOXIA also provides updates to the Software-Enabled Flash technology home page as a Linux Foundation individual supporter.

NOTES:

[1] QEMU is an acronym for Quick Emulator and defined as a software module that supports full virtualization by providing emulation of various hardware devices. QEMU is a component of the hypervisor platform.

[2] Paravirtualized storage devices are virtual storage devices that provide storage access to VMs with fast I/O performance and low latency.

TRADEMARKS:

DISCLAIMERS: