

DOI:10.1145/3286588

Programmable software-defined solid-state drives can move computing functions closer to storage.

BY JAEYOUNG DO, SUDIPTA SENGUPTA, AND STEVEN SWANSON

Programmable Solid-State Storage in Future Cloud Datacenters

THERE IS A major disconnect today in cloud datacenters concerning the speed of innovation between application/operating system (OS) and storage infrastructures. Application/OS software is patched with new/improved functionality every few weeks at “cloud speed,” while storage devices are off-limits for such sustained innovation during their hardware life cycle of three to five years in datacenters. Since the software inside the storage device is written by storage vendors as proprietary firmware not open for general application developers to modify, the developers are stuck with a device whose functionality and capabilities are frozen in time, even as many of them are modifiable in software. A period of five years is almost eternal in the cloud computing industry where new features, platforms, and application program interfaces (APIs) are evolving every couple of

months and application-demanded requirements from the storage system grow quickly over time. This notable lag in the adaptability and velocity of movement of the storage infrastructure may ultimately affect the ability to innovate throughout the cloud world.

In this article, we advocate creating a software-defined storage substrate of solid-state drives (SSDs) that are as programmable, agile, and flexible as the applications/OS accessing from servers in cloud datacenters. A fully programmable storage substrate promises opportunities to better bridge the gap between application/OS needs and storage capabilities/limitations, while allowing application developers to innovate in-house at cloud speed.

The move toward software-defined control for IO devices and co-processors has played out before in the datacenter. Both GPUs and network interface cards (NICs) started as black-box devices that provide acceleration for CPU-intensive operations (such as graphics and packet processing). Internally, they implemented acceleration features with a combination of specialized hardware and proprietary firmware. As customers demanded greater flexibility, vendors slowly exposed programmability to the rest of the system, unleashing the vast processing power available from GPUs and a new level of agility in how systems can manage networks for enhanced functionality like more granular traffic management, security, and

>> key insights

- A fully programmable storage substrate in cloud datacenters opens up new opportunities to innovate the storage infrastructure at cloud speed.
- In-storage programming is becoming increasingly easier with powerful processing capabilities and highly flexible development environments.
- New value propositions with the programmable storage substrate can be realized, such as customizing the storage interface, moving compute close to data, and performing secure computations.

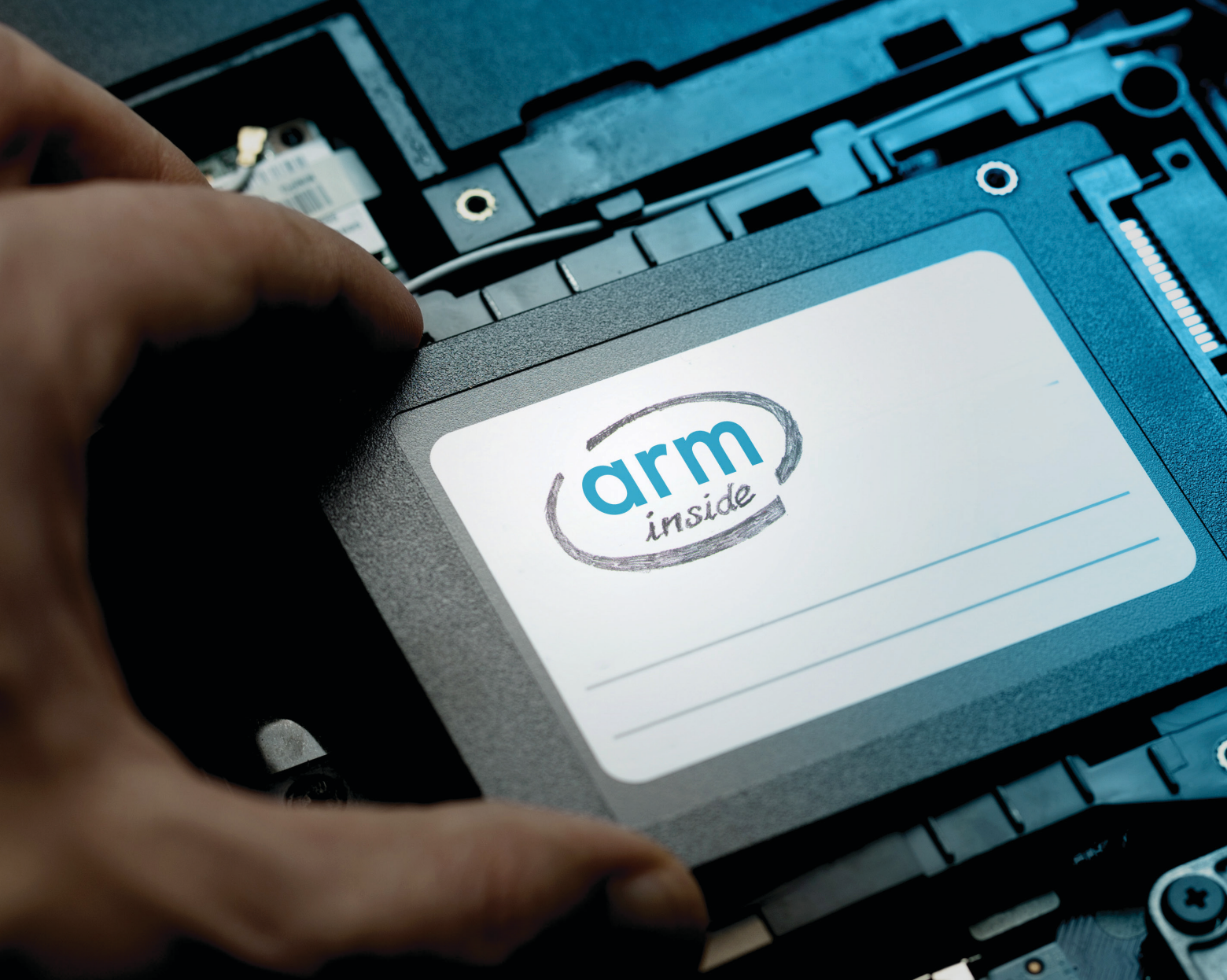
云数据中心的应用/OS更新与存储设备更新存在极大的不匹配性，更新周期为前者几周，后者三至五年；存储设备内部的软件有供应商编写为专用固件，即使本身可被编程，但仍无法为通用应用开发者所用；上述不匹配性阻碍了云技术发展

落后速度  
最终

底层  
灵活

这篇文章的贡献

举例：对于IO设备及协处理器的软件控制的先例



deep-network telemetry. <sup>遥测数据</sup>

Storage is at the cusp of a similar transformation. Modern SSDs rely on sophisticated processing engines running complex firmware, and vendors already provide customized firmware builds for cloud operators. Exposing this programmability through easily accessible interfaces will let storage systems in the cloud datacenters adapt to rapidly changing requirements on the fly.

### Storage Trends

The amount of data being generated daily is growing exponentially, placing more and more processing demand on datacenters. According to a 2017 marketing-trend report from IBM,<sup>a</sup> 90% of the data in the world in 2016 has been created in the last 12 months of 2015.

Such large-scale datasets—which generally range from tens of terabytes to multiple petabytes—present challenges of extreme scale while achieving very fast and efficient data processing: a high-performance storage infrastructure in terms of throughput and latency is necessary. This trend has resulted in growing interest in the aggressive use of SSDs that, compared with traditional spinning hard disk drives (HDDs), provides orders-of-magnitude lower latency and higher throughput. In addition to these performance benefits, the advent of new technologies (such as 3D NAND enabling much denser chips and quad-level-cell, or QLC, for bulk storage) allows SSDs to continue to significantly scale in capacity and to yield a huge reduction in price.

There are two key components in SSDs,<sup>4</sup> as shown in Figure 1—an SSD controller and flash storage media.

The controller that is most commonly implemented as a system-on-a-chip (SoC) is designed to manage the underlying storage media. For example, SSDs built using NAND flash memory have unique characteristics in that data can be written only to an empty memory location—no in-place updates are allowed—and memory can endure only a limited number of writes before it can no longer be read. Therefore, the controller must be able to perform some background management tasks (such as garbage collection) to reclaim flash blocks containing invalid data to create available space and wear leveling to evenly distribute writes across the entire flash blocks with the purpose of extending the SSD life. These tasks are, in general, implemented by proprietary firmware running on one or more embedded processor cores in

<sup>a</sup> <https://ibm.co/2XNvHPk>



Figure 1. Internal architecture of a modern flash SSD.

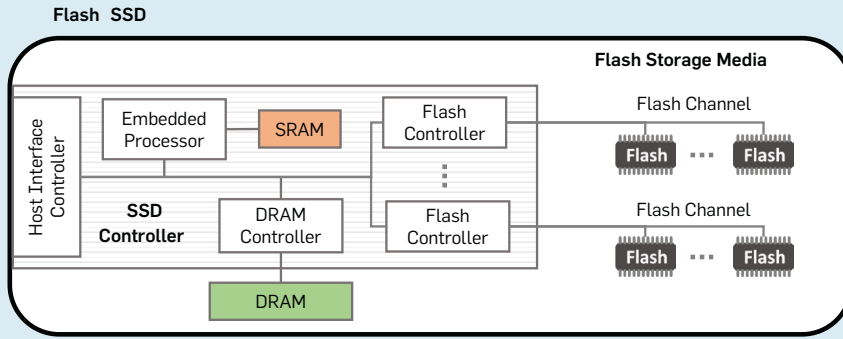
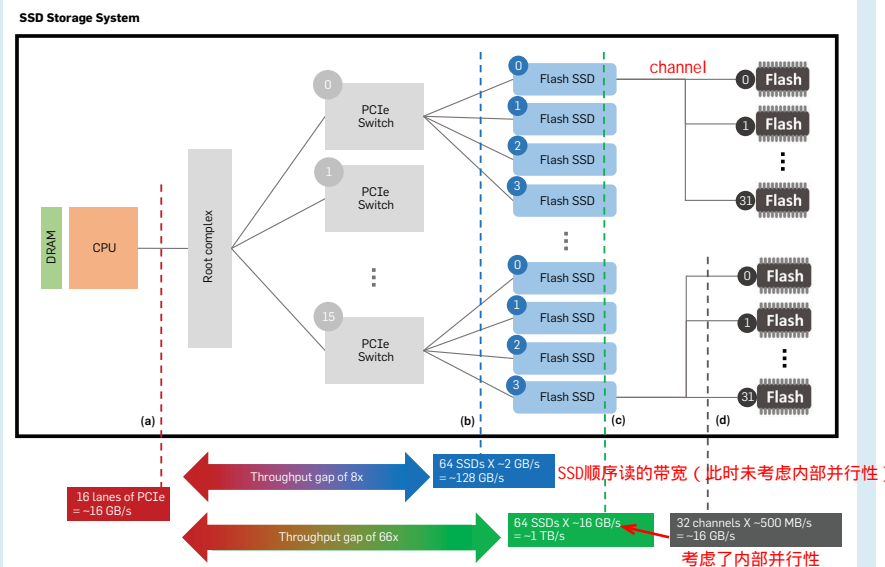


Figure 2. Example conventional storage server architecture with multiple NVMe SSDs.



the controller. In enterprise SSDs, large SRAM is often used for executing the SSD firmware, and both user data and internal SSD metadata are cached in external DRAM.

Interestingly, SSDs generally have a far larger aggregate internal bandwidth than the bandwidth supported by host I/O interfaces (such as SAS and PCIe). Figure 2 outlines an example of a conventional storage system that leverages a plurality of NVM Express (NVMe)<sup>b</sup> SSDs; 64 of them are connected to 16 PCIe switches that are mounted to a host machine via 16 lanes of PCIe Gen3. While this storage architecture provides a commodity solution for high-capacity

storage server at low cost (compared to building a specialized server to directly attach all SSDs on the motherboard of the host), the maximum throughput is limited to 16-lane PCIe interface speed (see Figure 2a), which is approximately 16GB/sec, regardless of the number of SSDs accessed in parallel. There is thus an 8x throughput gap between the host interface and the total aggregated SSD bandwidth that could be up to roughly ~2GB/sec per SSD<sup>c</sup> X 64 SSDs = ~128GB/sec (see Figure 2b). More interestingly, this gap would grow further if the internal SSD performance is considered. A modern enterprise-level SSD usually consists

b A device interface for accessing non-volatile memory attached via a PCI Express (PCIe) bus.

c Practical sequential-read bandwidth of a commodity PCIe SSD.

of 16 or 32 flash channels, as outlined in Figure 2. Since each flash channel can keep up with ~500MB/sec; internally each SSD can be up to ~500MB/sec per channel X 32 channels = ~16GB/sec (see Figure 2d); and the total aggregated in-SSD performance would be ~16GB/sec per SSD X 64 SSDs = ~1TB/sec (see Figure 2c), a 66x gap. Making SSDs programmable would thus allow systems to fully leverage this abundant bandwidth.

## In-Storage Programming

Modern SSDs combine processing—embedded processor—and storage components—SRAM, DRAM, and flash memory—to carry out routine functions required for managing the SSD. These computing resources present interesting opportunities to run general user-defined programs. In 2013, Do et al.<sup>6,17</sup> explored such opportunities for the first time in the context of running selected database operations inside a Samsung SAS flash SSD. They wrote simple selection and aggregation operators that were compiled into the SSD firmware and extended the execution framework of Microsoft SQL Server 2012 to develop a working prototype in which simple selection and aggregation queries could be run end-to-end.

That work demonstrated several times improvement in performance and energy efficiency by offloading database operations onto the SSD and highlighted a number of challenges that would need to be overcome to broadly adapt programmable SSDs: First, the computing capabilities available inside the SSD are limited by design. The low-performance embedded processor inside the SSD without L1/L2 caches and high latency to the in-SSD DRAM require extra careful programming to run user code in the SSD without producing a performance bottleneck.

Moreover, the embedded software-development process is complex and makes programming and debugging very challenging. To maximize performance, Do et al. had to carefully plan the layout of data structures used by the code running inside the SSD to avoid spilling out of the SRAM. Likewise, Do et al. used a hardware-debugging tool to debug programs running inside the SSD that is far more primitive than reg-

SSD管理功能的实现均由控制器中的处理器来完成;

某一工作: 将数据库选择/聚合功能编译到处理器中以实现ISP

有以下问题:

SSD中的嵌入式处理器没有一二级缓存且低性能高延时

即: 可供利用的计算能力有限

嵌入式软件开发过程复杂, 编程和调试具有极大的挑战性

随技术发展进步，未来趋势：可编程型SSD愈发简单

ular debugging tools (such as Microsoft Visual Studio) available to general application developers. Worse, the device-side processing code—selection and aggregation—had to be compiled into the SSD firmware in the prototype, meaning application developers would need to worry about not only the target application itself but also complex internal structures and algorithms in the SSD firmware.

On top of this, the consequences of an error can be quite severe, which could result in corrupted data or an unusable drive. Workaday application programmers are unlikely to accept the additional complexity, and cloud providers are unlikely to let untrusted code run in such a fragile environment.

Application developers need a flexible and general programming model that allows easily running user code written in a high-level programming language (such as C/C++) inside an SSD. The programming model must also support the concurrent execution of multiple in-SSD applications while ensuring that malicious applications do not adversely affect the overall SSD operation or violate protection guarantees provided by the operating and file system.

In 2014, Seshadri et al.<sup>20</sup> proposed Willow, an SSD that made programmability a central feature of the SSD interface, allowing ordinary developers to safely augment and extend the SSD semantics with application-specific functions without compromising file system protections. In their model, host and in-SSD applications communicate via PCIe using a simple, generic—not storage-centric—remote procedure call (RPC) mechanism. In 2016, Gu et al.<sup>7</sup> explored a flow-based programming model where an in-SSD application can be constructed from tasks and data pipes connecting the tasks. These programming models provide great flexibility in terms of programmability but are still far from “general purpose.” There is a risk that existing large applications might still need significant redesigns to exploit each model’s capabilities, requiring much time and effort.

Fortunately, winds of change can disrupt the industry and help application developers explore SSD programming in a better way, as illustrated in Figure 3. The processing capabilities

available inside the SSD are increasingly powerful, with abundant compute and bandwidth resources. Emerging SSDs include software-programmable controllers with multi-core processors, built-in hardware accelerators to offload compute-intensive tasks from the processors, multiple GBs of DRAM, and tens of independent channels to the underlying storage media, allowing several GB/s of internal data throughput. Even more interesting and useful, programming SSDs is becoming easier, with the trend away from proprietary architectures and software runtimes and toward commodity operating systems (such as Linux) running on top of general-purpose processors (such as ARM and RISC-V). This trend enables general application developers to fully leverage existing tools, libraries, and expertise, allowing them to focus on their own core competencies rather than spending many hours getting used to the low-level, embedded development process. This also allows application developers to easily port large applications already running on host operating systems to the device with minimal code changes.

All in all, the programmability evolution in SSDs presents a unique opportunity to embrace the SSDs as a first-class programmable platform in the cloud datacenters, enabling

software-hardware innovation inside the SSD. Moreover, going beyond the packaged SSD, because the two major components inside the SSD are each manufactured by multiple vendors,<sup>d</sup> it is conceivable that SSDs could be custom designed and provided in partnership with component vendors<sup>e</sup> (just like how today’s datacenter servers are built and deployed), and even contribute back some of the designs to the community (via forums like the Open Compute project, <https://www.opencompute.org>). For example, the industry is already moving in this direction with introduction of the Open-Channel SSD technology<sup>2,8,f</sup> that moves much of the SSD firmware functionalities out of the black box and into the operating system or userspace, giving applications better control over the device. In an open source project called Denali<sup>g</sup> in 2018, Microsoft proposed a scheme

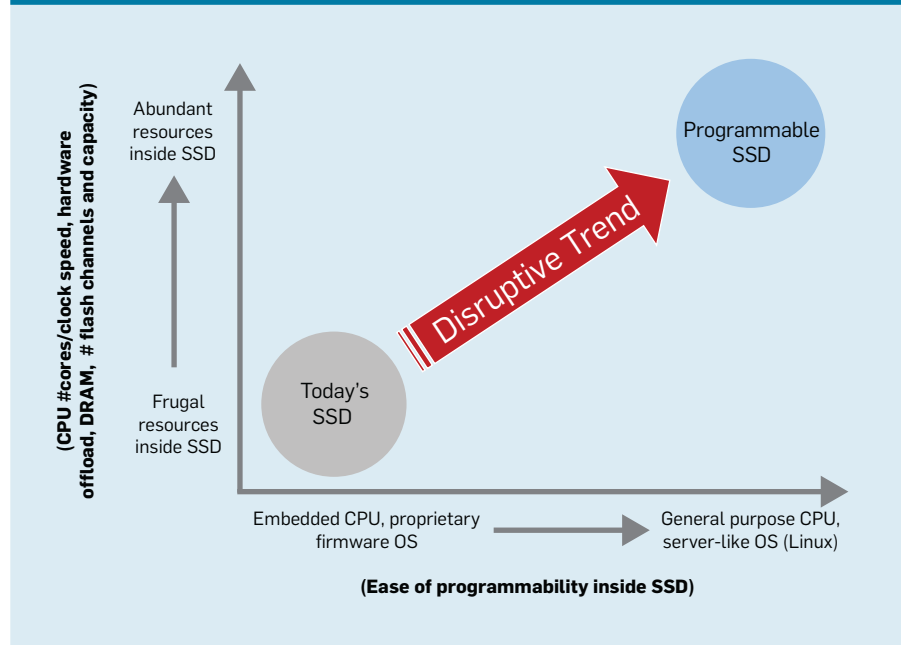
d Several vendors manufacture each type of component in flash SSDs. For example: flash controller manufactured by Marvell, PMC (acquired by Microsemi), Sandforce (acquired by Seagate), Indilinx (acquired by OCZ), and flash memory manufactured by Samsung, Toshiba, and Micron.

e Many large-scale datacenter operators (such as Google<sup>19</sup> and Baidu<sup>16</sup>) build their own SSDs that are fully optimized for their own application requirements.

f The Linux Open-Channel SSD subsystem was introduced in the Linux kernel version 4.4.

g <https://bit.ly/2GCutum>

Figure 3. Disruptive trends in the flash storage industry toward abundant resources and increased ease of programmability inside the SSD.



that splits the <sup>整体的</sup>monolithic components of an SSD into two different modules—one <sup>定制化的</sup>standardized part dealing with storage media and a software interface to handle application-specific tasks (such as garbage collection and wear leveling). In this way, SSD suppliers can build simpler products for datacenters and deliver them to market more quickly while per-application tuning is possible by datacenter operators.

The component-based ecosystem also opens up entirely new opportunities for integrating powerful heterogeneous programming elements (such as field-programmable gate ar-

ray (FPGA)<sup>13,21</sup> and GPU,<sup>h</sup> with storage media) and flash and other emerging new non-volatile memories (such as 3D XPoint, ReRAM, STT-RAM, and PCM) that provide persistent storage at DRAM latencies to deliver high-performance gains. This approach would present the greatest flexibility to take advantage of advances in the underlying storage device to optimize performance for multiple cloud applications. In the near future, the software-hardware innovation inside the SSD can proceed much like the PC, networking hardware, and

<sup>h</sup> <https://bit.ly/2L8LfM4>

GPU ecosystems have in the past. This is an opportunity to rethink datacenter architecture with efficient use of heterogeneous, energy-efficient hardware, which is the way forward for higher performance at lower power.

## Value Propositions

Here, we summarize three value propositions that demonstrate future directions in programmable storage (see Figure 4):

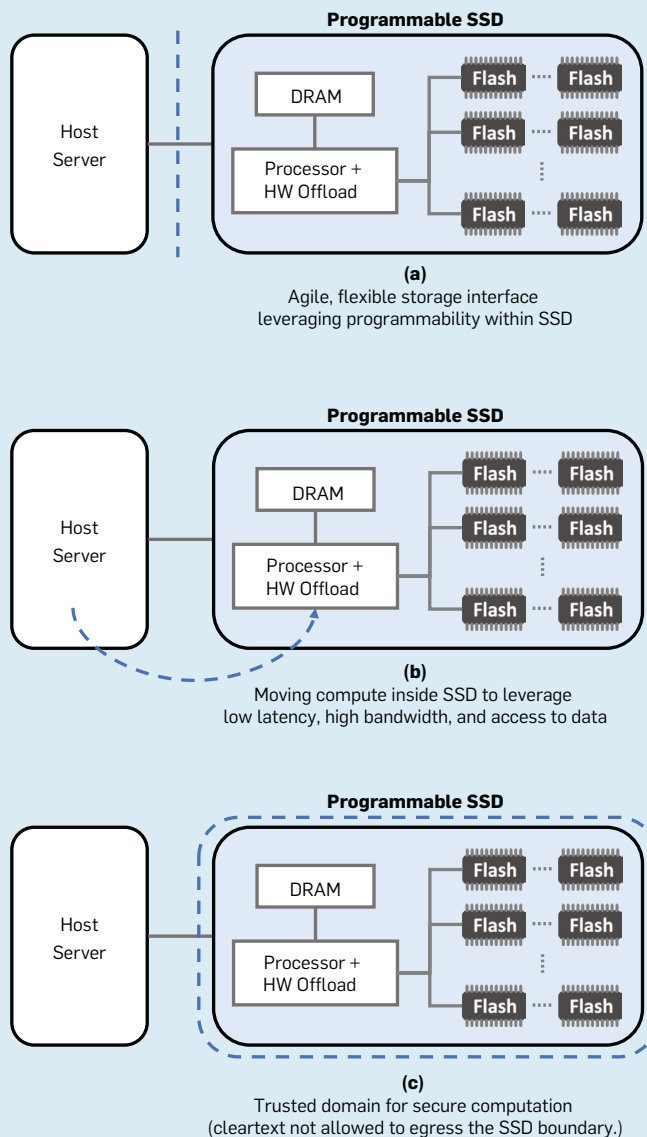
**Agile, flexible storage interface (see Figure 4a).** Full programmability will allow the storage interface and feature set to evolve at cloud speed, without having to persuade standardization bodies to bless them or persuade device manufacturers to implement them in the next-generation hardware roadmap, both usually involving years of delay. A richer, customizable storage interface will allow application developers to stay focused on their application, without having to work around storage constraints, quirks, or peculiarities, thus improving developer productivity.

As an example of the need for such an interface, consider how stream writes are handled in the SSD today. Because the SSD cannot differentiate between incoming data from multiple streams, it could pack data from different streams onto the same flash erase block, the smallest unit that can be erased from flash at once. When a portion of the stream data is deleted, it leaves blocks with holes of invalid data. To reclaim these blocks, the garbage-collection activity inside the SSD must copy around the valid data, slowing the device and increasing write amplification, thus reducing device lifetime.

If application developers had control over the software inside the SSD, they could handle streams much more efficiently. For instance, incoming writes could be tagged with stream IDs and the device could use this information to fill a block with data from the same stream. When data from that stream is deleted, the entire data block could be reclaimed without copying around data. Such stream awareness has been shown to double device lifetime, significantly increasing read performance.<sup>14</sup> In Microsoft, this need of supporting multiple streams in the SSD was identified in 2014, but NVMe incorporated the fea-

在Microsoft中，在2014年就确定了在SSD中支持多个流的需求，但是NVMe仅在2017年末才合并了该功能。

Figure 4. Programmable SSD value proposition.



提出运用OCSSD作为底层设备的方案  
SSD各元件由不同供应商提供  
基于这一点提出



ture only late 2017.<sup>i</sup> Moreover, large-scale deployment in Microsoft data-centers might take at least another

year and be very expensive, since new SSDs must be purchased to essential-

ly get a new version of the firmware. Waiting five years for a change to a

system software component is completely out of step with how quickly

computer systems are evolving today. A programmable storage platform

would reduce this delay to months and allow rapid iteration and refinement of the feature, not to mention

the ability to “tweak” the implementation to match specific use cases.

**Moving compute close to data (see Figure 4b).** The need to analyze and glean intelligence from big data im-

poses a shift from the traditional compute-centric model to data-centric model. In many big data scenarios,

application performance and responsiveness (demanded by interactive usage) is dominated not by the

execution of arithmetic and logic instructions but instead by the requirement to handle huge volumes of data

and the cost of moving this data to the location(s) where compute is performed. When this is the case, moving

the compute closer to the data can reap huge benefits in terms of increased throughput, lower latency, and reduced energy usage.

Big data analytics running inside an SSD can have access to the stored data with tens of GB/sec bandwidth (rivaling

DRAM bandwidth), and with latency comparable to accessing raw non-volatile memory. In addition, large energy

savings can be achieved because processors inside the SSD are more energy efficient compared to the host-server CPU (such as Intel Xeon), and data

does not need to be hauled over large distances from storage all the way up to the host via network, which is more energy-expensive than processing it.

Processors inside the SSD are clearly not as powerful as host processors, but together with in-storage hardware offload engines, a broad range of data processing tasks can be competitively

performed inside the SSD. As an example, consider how data analytic queries are processed in general: When an

analytic query is given, compressed data required to answer the query is first loaded to host, uncompressed,

and then executed using host resources. Such fundamental data analytics primitive can be processed inside the

SSD by accessing data with high internal bandwidth and by offloading decompression to the dedicated engine. Subsequent stages of the query-

processing pipeline (such as filtering out unnecessary data and performing the aggregation) can execute inside

the SSD, resulting in greatly reduced network traffic and saved host CPU/memory resources for other important

jobs. Further, performance and bandwidth together can be scaled by adding more SSDs to the system if the applica-

tion requires higher data rates.

**Secure computation in the cloud (see Figure 4c).** Recent security breach events related to personal, private in-

formation (financial and otherwise) have exposed the vulnerability of data infrastructures to hackers and attackers.

Also, a new type of malicious software called “encryption ransomware” at-

tacks machines by stealthily encrypting data files and demanding a ransom

to provide access to these files.<sup>10</sup> Security is often among the topmost concerns enterprise chief information offi-

cers have when they move to the cloud, as cloud providers are unwilling to take

on full liability for the impact of such breaches. Development of a secure

cloud is not just a feature requirement but also an absolute foundational capability necessary for the future of the

cloud computing model and its business success as an industry.

To realize the vision of a trusted cloud, data must be encrypted while stored at rest, which however, limits

the kind of computation that can be performed on encrypted data without decryption. To facilitate arbitrary

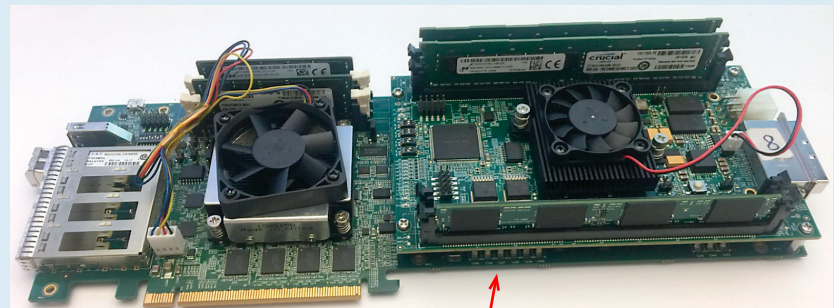
(legitimate) computation on stored data, it needs to be decrypted before computing on it. This requires decrypted

cleartext data to be present (at least temporarily) in various portions of the datacenter infrastructure

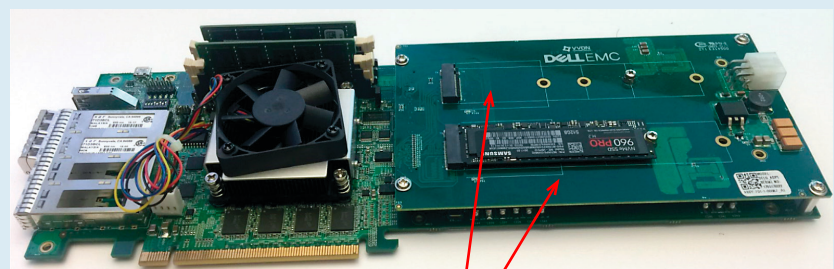
vulnerable to security attacks. Application developers need a way to facilitate secure computation on the cloud

by fencing in well-defined, narrow, trusted domains that can preserve the ability to perform arbitrary com-

Figure 5. A prototype programmable SSD developed for research purposes.



(a) Device with a storage board with an embedded storage controller and DIMM slots for flash or other forms of NVM



(b) Device with a storage board where M.2 SSDs can be plugged into.

<sup>i</sup> Note the multi-stream technology for SCSI/SAS was standardized in T10 on May 20, 2015.

putation on the data.

SSDs with their powerful compute capabilities can form a trusted domain for doing secure computation on encrypted data, leveraging their internal hardware cryptographic engine and secure boot mechanisms for this purpose. Cryptographic keys can be stored inside the SSD, allowing arbitrary compute to be carried out on the stored data—after decryption if needed—while enforcing that data cannot leave the device in cleartext form. This allows a new, flexible, easily programmable, near-data realization of **trusted hardware** in the cloud. Compared to currently proposed solutions like Intel Enclaves<sup>1</sup> that are protected, isolated areas of execution in the host server memory, this solution protects orders of magnitude more data.

### Programmable SSDs

While the concept of in-storage processing on SSDs was proposed more than six years ago,<sup>6</sup> experimenting with SSD programming has been limited by the availability of real hardware on which a prototype can be built to demonstrate what is possible. The recent emergence of prototyping boards available for both research and commercial purposes has opened new opportunities for application developers to take ideas from conception to action.

Figure 5 shows such prototype device, called **Dragon Fire Card** (DFC),<sup>k,3,5</sup> designed and manufactured by Dell EMC and NXP for research. The card is powerful and

flexible with enterprise-level capabilities and resources. It comprises a main board and a storage board. The main board contains an ARMv8 processor, 16GB of RAM, and various on-chip hardware accelerators (such as 20Gbps compression/decompression, 20Gbps SEC-crypto, and 10Gbps RegEx engines). It also provides NVMe connectivity via four PCIe Gen3 lanes, and 4x10Gbps Ethernet that supports remote dynamic memory access (RDMA) over converged ethernet (RoCE) protocol. It supports two different storage boards that connect via 2x4 PCIe Gen3 lanes: One type of board (see Figure 5a) includes an embedded storage controller and four memory slots where flash or other forms of NVM can be installed; and the second (see Figure 5b) an adapter that hosts two M.2 SSDs.

The ARM SoC inside the board runs a full-fledged Ubuntu Linux, so programming the board is very similar to programming any other Linux device. For instance, software can leverage the Linux container technology (such as Docker) to provide isolated environments inside the board. To create applications running on the board, a software development kit (SDK) containing GNU tools to build applications for ARM and user/kernel mode libraries to use the on-chip hardware accelerators is provided, allowing a high level of programmability. The DFC can also serve as a block device, just like regular SSDs. For this purpose, the device is shipped with a flash translation layer (FTL) that runs on the main board.

The SSD industry is also moving

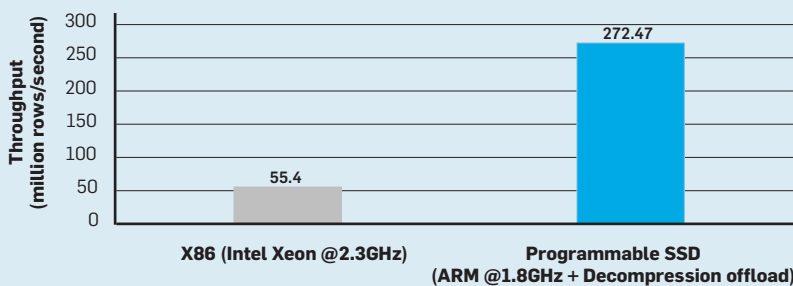
toward bringing compute to SSDs so data can be processed without leaving the place where it is originally stored. For instance, in 2017 NGD Systems<sup>1</sup> announced an SSD called Catalina2<sup>1</sup> capable of running applications directly on the device. Catalina2 uses TLC 3D NAND flash (up to 24TB), which is connected to the onboard ARM SoC that runs an embedded Linux and modules for error-correcting code (ECC) and FTL. On the host server, a tunnel agent (with C/C++ libraries) runs to talk to the device through the NVMe protocol. As another example, ScaleFlux<sup>m</sup> uses a Xilinx FPGA (combined with terabytes of TLC 3D NAND flash) to compute data for data-intensive applications. The host server runs a software module, providing API accesses to the device while being responsible for FTL and flash-management functionalities.

Academia and industry are working to establish a compelling value proposition by demonstrating application scenarios for each of the three pillars outlined in Figure 4. Among them we are initially focused on exploring the benefits and challenges of moving compute closer to storage (see Figure 4b) in the context of big data analytics, examining large amounts of data to uncover hidden patterns and insights.

**Big data analytics within a programmable SSD.** To demonstrate our approach, we have implemented a C++ reader that runs on a DFC card (see Figure 5) for Apache Optimized Row Columnar (ORC) files. The ORC file format is designed for fast processing and high storage efficiency of big data analytic workloads, and has been widely adopted in the open source community and industry. The reader running inside the SSD reads large chunks of ORC streams, decompresses them, and then evaluates query predicates to find only necessary values. Due to the server-like development environment—Ubuntu and a general-purpose ARM processor—we easily ported a reference implementation of the ORC reader<sup>n</sup> to the ARM SoC environment (with only a few lines of code changes) and incorporat-

j <https://software.intel.com/en-us/sgx>  
k <https://github.com/DFC-OpenSource>

**Figure 6. Preliminary results using a programmable SSD yield approximately 5x speedups for full scans of ZLIB-compressed ORC files within the device, compared to native ORC readers running on x86 architecture.**



在DFC中运行一个reader，其能读取闪存中数据，解压缩并得出必要值  
因reader开发环境与ARM处理器环境相似，能较为容易的移植到SSD内嵌主控芯片中


l <http://www.ngdsystems.com>  
m <http://www.scaleflux.com>  
n <https://github.com/apache/orc>

ed library APIs into the reader, enabling reading data from flash and offloading the decompression work to the ARM SoC hardware accelerator.<sup>o</sup> ARM是SSD的主控芯片


Figure 6 shows preliminary bandwidth results of scanning a ZLIB-compressed, single-column integer dataset (one billion rows) through the C++ ORC reader running on a host x86 server vs. inside the DFC card, respectively.<sup>o</sup> As in the figure, we achieved approximately 5x faster scan performance inside the device compared to running on the host server. Given that this is a single device performance, we should be able to achieve much better performance improvements by increasing the number of programmable SSDs that are used in parallel.

In addition to scanning, filtering, and aggregating large volumes of data at high-throughput rates by offloading part of the computation directly to the storage has been explored as well. In 2016 Jo et al.<sup>12</sup> built a prototype that performs very early filtering of data through a combination of ARM and a hardware pattern-matching engine available inside a programmable SSD equipped with a flow-based programming model described by Gu et al.<sup>7</sup> When a query is given, the query planner determines whether early filtering is beneficial for the query and chooses a candidate table as the target if the estimated filtering ratio is sufficiently high. Early filtering is then performed against the target table inside the device, and only filtered data is then fetched to the host for residual computation. This early filtering inside the device turns out to be highly effective for analytic queries; when running all 22 TPC-H queries on a MariaDB server with the programmable device prototyped on a commodity NVMe SSD, a 3.6x speed-up was achieved by Jo et al.<sup>12</sup> compared to a system with the same SSD without the programmability.

Alternatively, an FPGA-based prototype design for near-data processing inside the a storage node for database engines was studied by István et al.<sup>11</sup> in 2017. In this prototype, each storage



**The programmable storage substrate can be viewed as a hyper-converged infrastructure where storage, networking, and compute are tightly coupled for low-latency, high-throughput access, while still providing availability.** 超收敛



node that could be accessed over the network through a simple key-value store interface provided fault tolerance through replication and application-specific processing (such as predicate evaluations, substring matching and decompression) at line rate.

### Datacenter Realization

Each application running in cloud datacenters has its own, unique requirements, making it difficult to design server nodes with the proper balance of compute, memory, and storage. To cope with such complexity, an approach of physically decoupling resources was proposed recently by Han et al.<sup>9</sup> in 2013 to allow replacing, upgrading, or adding individual resources instead of the entire node.<sup>分拆</sup> With the availability of fast interconnect technologies (such as InfiniBand, RDMA, and RoCE), it is already common in today's large-scale cloud datacenters to disaggregate storage from compute, significantly reducing the total cost of ownership and improving the efficiency of the storage utilization. However, storage disaggregation is a challenge<sup>15</sup> as storage-media access latencies are heading toward single-digit microsecond level<sup>p</sup> compared to a disk's millisecond latency, which is much larger than the fast network overhead. It is likely that, in the next few years the network latency will become a bottleneck as new, emerging non-volatile memories with extremely low latencies become available.

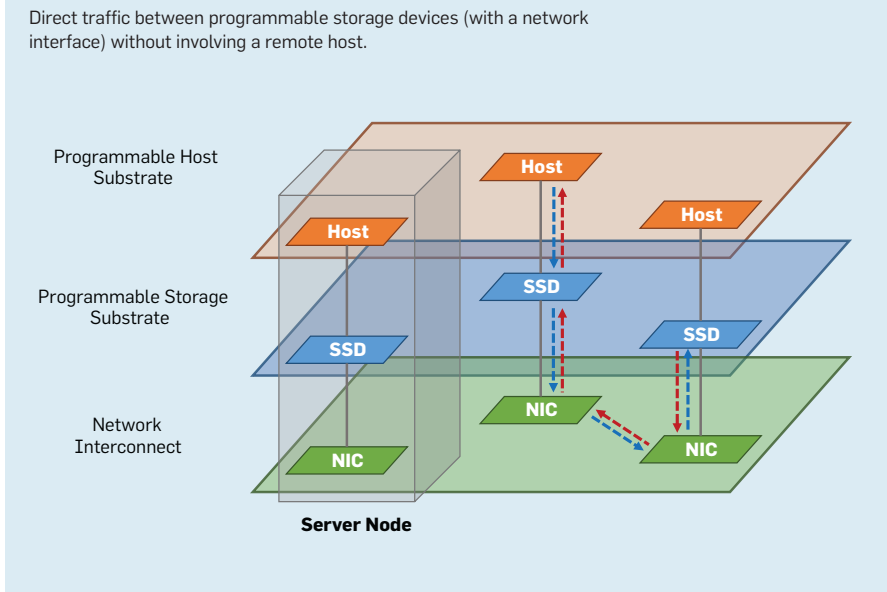
This challenge of storage disaggregation can be overcome by using programmable storage, enabling a fully programmable storage substrate that is decoupled from the host substrate as outlined in Figure 7. This view of storage as a programmable substrate allows application developers not only to leverage very low, storage-medium access latency by running programs inside the storage device but also to access any remote storage device without involving the remote host server where the device is physically attached (see Figure 7) by

<sup>o</sup> Note, to effectively compare data-processing capability in each case—Intel Xeon in x86 vs. ARM + decompression accelerator in the device—only a single core for each processor was used.

<sup>p</sup> For example, the access latency of 3D XPoint can take 5~10 μsec, while NVMe SSD and disk takes ~50~100 μsec and 10 msec, respectively.<sup>8</sup>



**Figure 7. Enabling a programmable storage substrate decoupled from the host substrate.**



using NVMe over Fabrics (NVMe-oF)<sup>q</sup> with RDMA.

With the programmable storage substrate, we can think of going beyond the single-device block interface. For example, a micro server inside storage can expose a richer interface like a distributed key-value store or distributed streams. Or the storage infrastructure can be managed as a fabric, not as individual devices. The programmable storage substrate can also provide high-level datacenter capabilities (such as backup, data snapshot, replication, de-duplications, and tiering), which are typically supported in a datacenter server environment where compute and storage are separated. This means the programmable storage substrate can be viewed as a hyper-converged infrastructure where storage, networking, and compute are tightly coupled for low-latency, high-throughput access, while still providing availability.

## Conclusion

In this article, we have presented our vision of a fully programmable storage substrate in cloud datacenters, allowing application developers to innovate the storage infrastructure at cloud speed like the software application/OS infrastructure. The programmability evolution in SSDs

provides opportunities for embracing them as a first-class programmable platform in cloud datacenters, enabling software-hardware innovation that could bridge the gap between application/OS needs and storage capabilities/limitations. We hope to shed light on the future of software-defined storage and help chart a direction for designing, building, deploying, and leveraging a software-defined storage architecture for cloud datacenters.

## Acknowledgments

This work was supported in part by National Science Foundation Award 1629395.

## References

- Alves, V. In-situ processing. Flash Memory Summit (Santa Clara, CA, Aug. 8–10), 2017.
- Björling, M., González, J., and Bonnet, P. Lightnvm: The Linux open-channel SSD subsystem. In *Proceedings of the 15th USENIX Conference on File and Storage Technologies* (Santa Clara, CA, Feb. 27–Mar. 2). USENIX Association, Berkeley, CA, 2017, 359–374.
- Bonnet, P. What's up with the storage hierarchy? In *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research* (Chaminade, CA, Jan. 8–11), 2017.
- Cornwell, M. Anatomy of a solid-state drive. *Commun. ACM* 55, 12 (Dec. 2012), 59–63.
- Do, J. Softflash: Programmable storage in future data centers. In *Proceedings of the 20th SNIA Storage Developer Conference* (Santa Clara, CA, Sep. 11–14), 2017.
- Do, J., Kee, Y.-S., Patel, J.M., Park, C., Park, K., and DeWitt, D.J. Query processing on smart SSDs: Opportunities and challenges. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (New York, NY, Jun. 22–27). ACM Press, New York, 2013, 1221–1230.
- Gu, B., Yoon, A. S., Bae, D.-H., Jo, I., Lee, J., Yoon, J., Kang, J.-U., Kwon, M., Yoon, C., Cho, S., et al. Biscuit: A framework for near data processing of big data workloads. In *Proceedings of the ACM/IEEE 43rd Annual International Symposium on Computer Architecture* (Seoul, S. Korea, Jun. 18–22). IEEE, 2016, 153–165.

- Hady, F. Wicked fast storage and beyond. In *Proceedings of the 7th Non Volatile Memory Workshop* (San Diego, CA, Mar. 6–8). Keynote, 2016.
- Han, S., Egi, N., Panda, A., Ratnasamy, S., Shi, G., and Shenker, S. Network support for resource disaggregation in next-generation datacenters. In *Proceedings of the 12th ACM Workshop on Hot Topics in Networks* (College Park, MD, Nov. 21–22). ACM Press, New York, 2013, 10.
- Huang, J., Xu, J., Xing, X., Liu, P., and Qureshi, M. K. Flashguard: Leveraging intrinsic flash properties to defend against encryption ransomware. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, TX, Oct. 30–Nov. 3). ACM Press, New York, 2017, 2231–2244.
- István, Z., Sidler, D., and Alonso, G. Caribou: Intelligent distributed storage. In *Proceedings of the VLDB Endowment* 10, 11 (Aug. 2017), 1202–1213.
- Jo, I., Bae, D.-H., Yoon, A.S., Kang, J.-U., Cho, S., Lee, D.D., and Jeong, J. YourSQL: A high-performance database system leveraging in storage computing. In *Proceedings of the VLDB Endowment* 9, 12 (Aug. 2016), 924–935.
- Jun, S.-W., Liu, M., Lee, S., Hicks, J., Ankcorn, J., King, M., Xu, S., et al. BlueDBM: An appliance for big data analytics. In *Proceedings of the ACM/IEEE 42nd Annual International Symposium on Computer Architecture* (Portland, OR, Jun. 13–17). IEEE, 2015, 1–13.
- Kang, J.-U., Hyun, J., Maeng, H., and Cho, S. The multi-streamed solid-state drive. In *Proceedings of the 6th USENIX Workshop on Hot Topics in Storage and File Systems* (Philadelphia, PA, Jun. 17–18). USENIX Association, Berkeley, CA, 2014.
- Klimovic, A., Kozyrakis, C., Thereska, E., John, B., and Kumar, S. Flash storage disaggregation. In *Proceedings of the 11th European Conference on Computer Systems* (London, U.K., Apr. 18–21). ACM Press, New York, 2016, 29.
- Ouyang, J., Lin, S., Jiang, S., Hou, Z., Wang, Y., and Wang, Y. SDF: Software-defined flash for web-scale Internet storage systems. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems* (Salt Lake City, UT, Mar. 1–5). ACM Press, New York, 2014, 471–484.
- Park, K., Kee, Y.-S., Patel, J.M., Do, J., Park, C., and Dewitt, D.J. Query processing on smart SSDs. *IEEE Data Engineering Bulletin* 37, 2 (Jun. 2014), 19–26.
- Picoli, T.L., Pasco, C.V., Jónsson, B.P., Bouganim, L., and Bonnet, P. uFLIP-OC: Understanding flash I/O patterns on open-channel solid state drives. In *Proceedings of the 8th Asia-Pacific Workshop on Systems* (Mumbai, India, Sep. 2–3). ACM Press, New York, 2017, 20.
- Schroeder, B., Lagisetty, R., and Merchant, A. Flash reliability in production: The expected and the unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies* (Santa Clara, CA, Feb. 22–25). USENIX Association, Berkeley, CA, 2016, 67–80.
- Seshadri, S., Gahagan, M., Bhaskaran, S., Bunker, T., De, A., Jin, Y., Liu, Y., and Swanson, S. Willow: A user-programmable SSD. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation* (Broomfield, CO, Oct. 6–8). USENIX Association, Berkeley, CA, 2014, 67–80.
- Woods, L., István, Z., and Alonso, G. Ibex: An intelligent storage engine with support for advanced SQL offloading. In *Proceedings of the VLDB Endowment* 7, 11 (Jul. 2014), 963–974.

**Jaeyoung Do** (jaedo@microsoft.com) is a researcher at Microsoft Research, Redmond, WA, USA. He is leading a project, SoftFlash, which aims to use programmable SSDs in cloud datacenters.

**Sudipta Sengupta** (sudipta@amazon.com) is leading new initiatives in artificial intelligence/deep learning at Amazon AWS, Seattle, WA, USA; the research reported in this article was done while he was at Microsoft Research, Redmond, WA, USA.

**Steven Swanson** (swanson@cs.ucsd.edu) is a professor in the Department of Computer Science and Engineering at the University of California, San Diego, USA.

Copyright held by authors/owners.  
Publication rights licensed to ACM. \$15.00.