Memories - Materials, Devices, Circuits and Systems 4 (2023) 100051

Contents lists available at ScienceDirect

Memories - Materials, Devices, Circuits and Systems

journal homepage: www.elsevier.com/locate/memori

A review on computational storage devices and near memory computing for high performance applications

Dina Fakhry [a], Mohamed Abdelsalam [b], M. Watheq El-Kharashi [a,c,*], Mona Safar [a]

[a] Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt
[b] Siemens EDA, Cairo, Egypt
[c] Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada

ARTICLE INFO

Keywords:
Computational Storage Devices (CSDs)
In-Storage Computing (ISC)
Near Data Processing (NDP)
Solid-State Drives (SSDs)

ABSTRACT

The von Neumann bottleneck is imposed due to the explosion of data transfers and emerging data-intensive applications in heterogeneous system architectures. The conventional computation approach of transferring data to CPU is no longer suitable especially with the cost it imposes. Given the increasing storage capacities, moving extensive data volumes between storage and computation cannot scale up. Hence, high-performance data processing mechanisms are needed, which may be achieved by bringing computation closer to data. Gathering insights where data is stored helps deal with energy efficiency, low latency, as well as security. Storage bus bandwidth is also saved when only computation results are delivered to the host memory. Various applications, including database acceleration, machine learning, Artificial Intelligence (AI), offloading (compression/encryption/encoding) and others can perform better and become more scalable if the "move process to data" paradigm is applied. Embedding processing engines inside Solid-State Drives (SSDs), transforming them to Computational Storage Devices (CSDs), provides the needed data processing solution. In this paper, we review the prior art on Near Data Processing (NDP) with focus on In-Storage Computing (ISC), identifying main challenges and potential gaps for future research directions.

1. Introduction

Throughout the years, High-Performance Computing (HPC) applications' computational cost has notably declined. The cost of data transfer however, did not scale down proportionally with computation cost. The system bandwidth availability represented in I/O, network, and CPU memory hierarchy in addition to low data locality hinder the performance of data-intensive applications as sizable energy is consumed when moving around such massive data. Even though data movement is unavoidable in HPC algorithms, attempts to optimize its amount and the network flow in the HPC environment is being actively researched to improve energy consumption and performance. Towards such optimization, the computing paradigm of bringing processing closer to data is very promising with various concepts investigated based on processing location [1].

Near Data Processing (NDP) is considered the big umbrella under which several techniques have been explored, including On-Disk Data Processing (ODDP) [2], In-Memory Computing (IMC) or Processing-In-Memory (PIM) [3–5], and In-Storage Computing (ISC) or In-Storage Processing (ISP) [6,7]. IMC or PIM falls under a bigger frame known as Near Memory Computing (NMC). Fig. 1 demonstrates such NDP classification.

ODDP is associated with the arrangement where secondary storage devices, usually Hard Disk Drives (HDDs), are further equipped with memory capacity and computing to process stored data [2].

On the other hand, IMC/PIM is mainly associated with equipping RAM with a data processing capability [8,9]. Different innovations and PIM-enablers have been thoroughly discussed in literature. These innovations include the emergence of 3D-stacked memory dies [10,11], which augment Dynamic RAMs (DRAMs) with a logic layer as discussed in [12,13]. This allows the addition of processing logic to that layer to exploit the high bandwidth available between the logic layer and the DRAM cell arrays. Other enablers include the emergence of resistive memory technologies that are more computation-friendly and the usage of byte-addressable Non-Volatile Memories (NVM).

ISP refers to processing data where it resides, such that computation tasks are offloaded to storage devices.

With memory hierarchy consisting of storage, main memory, and multiple cache levels, the processing location of data is what determines the used technique. Storage Class Memory (SCM) is a type that bridges the gap between DRAMs and disks [14]. Fig. 2 shows the memory system hierarchy, with parts of it implemented as Volatile Memories (VMs) and the other as Non-Volatile Memories (NVMs). The

* Corresponding author at: Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt.
E-mail address: watheq.elkharashi@eng.asu.edu.eg (M.W. El-Kharashi).

https://doi.org/10.1016/j.memori.2023.100051
Received 27 November 2022; Received in revised form 16 April 2023; Accepted 22 April 2023

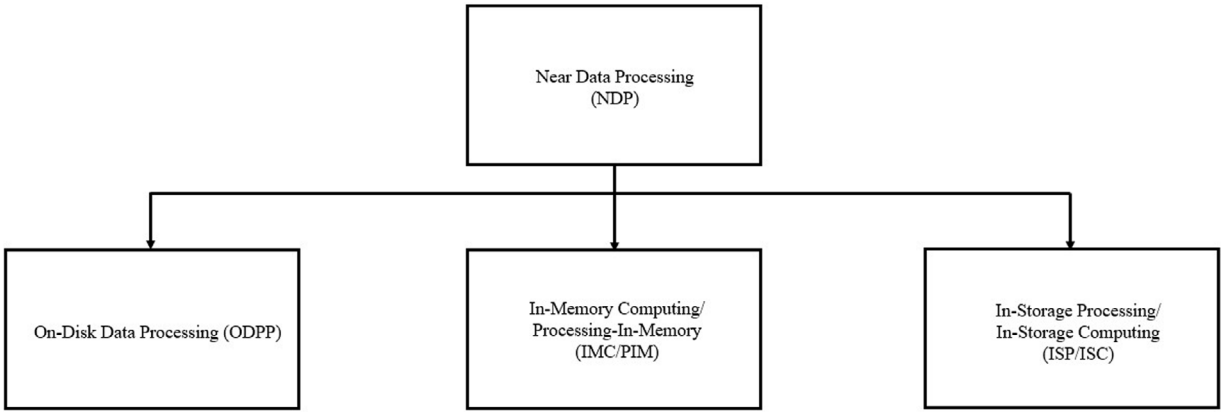


Fig. 1. Near Data Processing (NDP) classification.

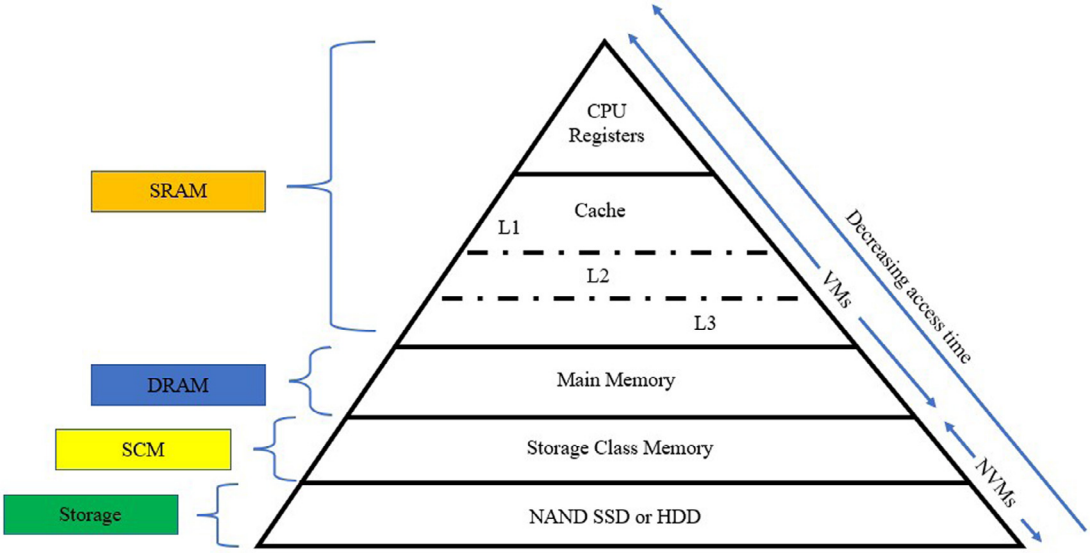


Fig. 2. Memory system hierarchy.

number of cache levels shown in the diagram is 3 however, more levels could exist.

The focus of this research is to review ISP, where data is retrieved from drives before intermediate results are forwarded to computational nodes. Promising benefits can be foreseen with the advent of Solid-State Drives (SSDs). They incorporate, by default, high computing power due to the presence of several embedded cores for controlling the flash memory array and the parallel flash interfaces that allow handling multiple I/O commands simultaneously. Deploying ISP, by placing processing engines within the storage units inside an SSD to run data-intensive tasks, transforms a conventional SSD into a Computational Storage Device (CSD) that does not need to send input data through a complex hardware and software stack to be processed. Fig. 3 shows the modern architecture for SSDs with a Non-Volatile Memory Express (NVMe) being used as the logical-device interface. Multiple SSDs can be connected to the host interface, which is demonstrated by SSDs numbered from 1 to m.

Compared to similar efforts in [15–17], we contribute with the following in this paper:

1. Comprehensive overview on CSDs,
2. Thorough assessment of NDP-related work with a focus on ISP, and
3. Highlight on contemporary challenges facing NDP with proposals for future directions for their adoption.

This paper is organized as follows. In Section 2, we give an overview of CSDs, their classifications, and potential applications. Section 3 reviews notable architectures for different ISP solutions. In Section 4, we briefly discuss challenges facing NDP. Section 5 highlights emerging interconnects that may leverage ISP potentiality. In Section 6, we briefly discuss some future research challenges and directions for NDP.

2. Systematization and classification

The Storage Networking Industry Association (SNIA) [18], a non-profit organization, is a reputed authority for storage leadership, standards, and technology expertise worldwide with a mission to facilitate management, movement, and security of information. The SNIA Computational Storage Technical Work Group (TWG) has been formed to create standards to promote the interoperability of computational storage devices, and to define interface standards for system deployment, provisioning, management, and security.

According to SNIA, architectures that provide computation coupled to storage through the integration of compute resources outside the traditional compute and memory architectures can be termed Computational Storage [19]. They can directly be integrated with storage or between host and storage to reduce data movement. Such architectures allow improvements in application performance and infrastructure efficiency by enhancing bandwidth usage and latency.

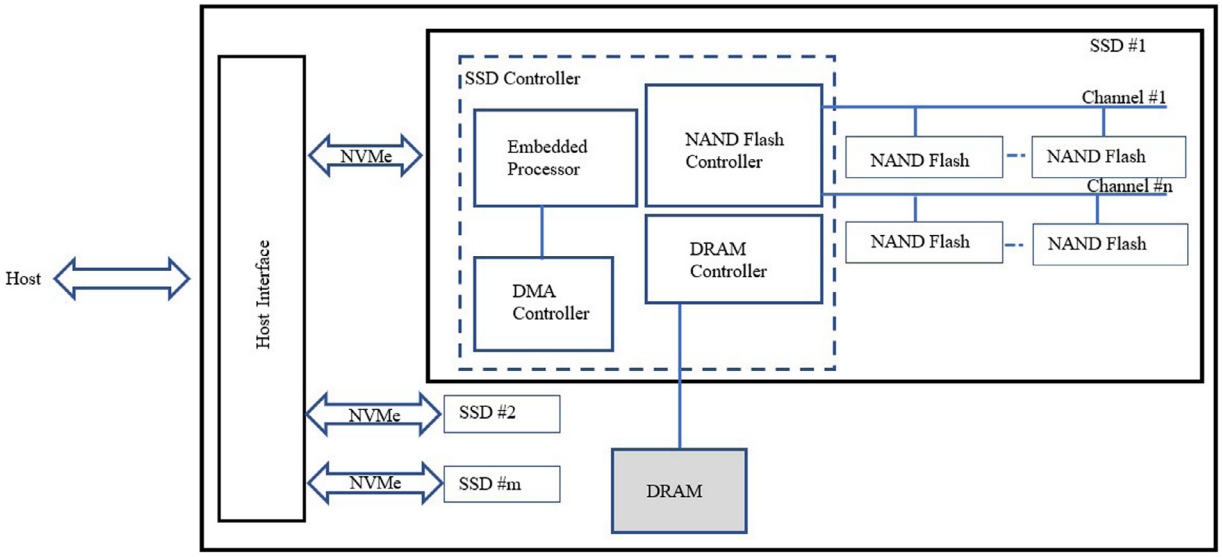


Fig. 3. Modern SSD architecture.

CSDs can be categorized into two groups in terms of the approach used for running applications, with each having its pros and cons [20]: Processor-based CSDs and FPGA-based CSDs.

2.1. Processor-based CSDs

Processor-based CSDs can be equipped either with existing SSD cores that are primarily used to handle normal write/read operations and Flash Translation Layer (FTL) procedures or with general-purpose processing engines to process user data and run applications.

Using existing SSD cores may not always be the best option when viewing processor-based CSDs. That is mainly due to the fact that conventional SSD operations cannot be suspended, in addition to being a heavy workload, which can make sharing cores to run user applications in-situ inefficient [20].

On the other hand, usage of dedicated general-purpose processors provides enough flexibility and even lifts the load from existing cores. However, limitations in performance of storage devices' embedded processors still exist because even though their performance increased, the high-end servers' performance further improved.

2.2. FPGA-based CSDs

For parallel tasks, FPGAs are considered as fast and low power solution. Some studies claim FPGAs can notably enhance big data applications' performance [20,21]. FPGAs provide flexible, fully configurable architectures with customizable data paths in addition to the targeted performance, power consumption, and latency of dedicated hardware and the reconfigurability of software. Even though FPGAs are more cost-effective than Application Specific Standard Products (ASSPs) or Application Specific Integrated Circuits (ASICs) for many applications, they may fall short in some aspects, such as operation frequency. FPGAs are also error-prone with debugging being extremely difficult due to poor visibility, in addition to being time consuming, where a user must redesign the RTL and regenerate the bitstream to achieve a different functionality [22]. There are three different options for deployment of FPGAs along the SSD data path, each providing its own benefits [23]. The different deployment options are listed below:

1. Accelerator integrated with flash, also known as Computational Storage Drive (CSD), which provides plug and play implementation, scaling performance with the number of accelerators, as well as providing FTL customization for specific workloads.
2. Accelerator and storage on the same Peripheral Component Interconnect Express (PCIe) subsystem, also known as Computational Storage Processor (CSP), which allows independent SSD and acceleration scaling in addition to allowing plugging into standard computer slots.
3. Accelerator in-line with storage, also known as Computational Storage Array (CSA). It is an SSD vendor-independent solution that allows independent scaling of accelerators and SSDs, providing bandwidth optimization between them.

Fig. 4 depicts the three different options for deployment of FPGAs along the SSD data path [23]. By using PCIe as the transfer protocol between host and storage systems, substituting the SATA interface, an increase in performance can be expected due to the increased number of channels, which makes it suitable for buffering and caching applications.

2.3. Metrics for ISP-friendly applications

Even though computational SSDs are beneficial to reduce the data movement and overcome the memory wall as well as the bandwidth gap between host and storage, they should only be used when needed. According to SNIA, computational SSDs should be employed whenever bandwidth is a bottleneck with large data transfers associated, data delivery can bypass the host, or software application can be moved to storage [29]. However, if an application is compute-intensive with small data-transfer incurred, if computation itself involves small data, or if algorithm is highly sequential, then the host CPU can be thought as a better processing option.

Through providing an emulation platform, Ruan et al. estimated the extent by which an application could exploit FPGA-enabled CSDs to offload operations [30]. They evaluated 12 common applications to indicate basic criteria for choosing ISP-friendly applications. Farahpour et al. also discussed a similar approach in addition to evaluating the effect on an application performance based on changing the accelerator type and system configuration [31].

To standardize terminologies associated with CSDs, SNIA referred to services provided by such CSDs as Computational Storage Services (CSSs). The type of service could be used as another way to classify CSDs, with ones delivering fixed services that can be configured and used to provide a given function and others that are programmable via code upload to provide one or more CSSs [19]. Code upload may be done through one of the four mechanisms: Operating System

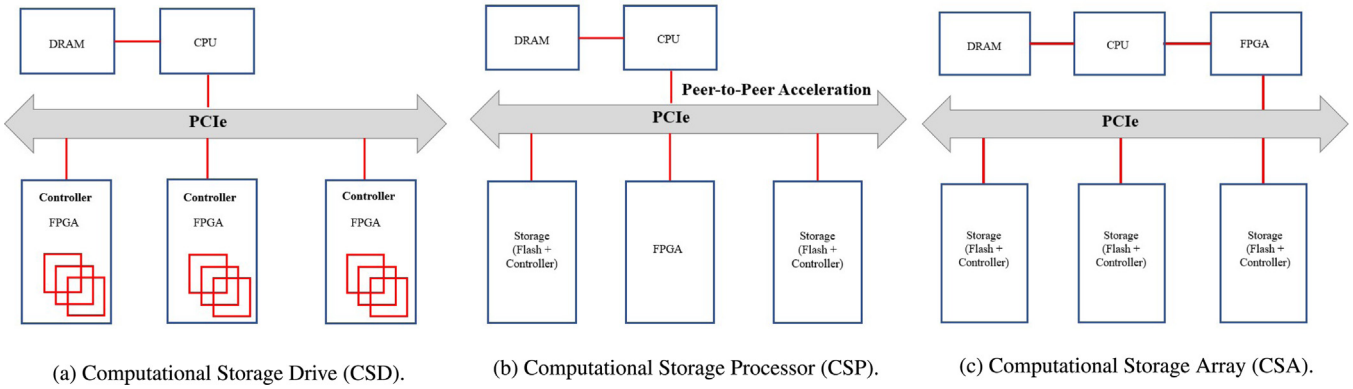


Fig. 4. FPGA deployment options along the SSD datapath for performing applications in-situ [23].

Table 1
Fixed NDP architecture classification.

NDP architecture	Prototype description/ evaluation model	Programming model	Application domain/workload	Host interconnect
Biscuit [24]	Real SSD	Flow-based using APIs	Database	PCIe Gen-3 ×4
Scan and join [25,26]	Simulation model/ ARM Cortex A9 (sim)	–	Database	SATA 2.0, 3.0, PCIe Gen-2 ×16
YourSQL [27]	Real SSD	Biscuit APIs	Database	PCIe Gen-3 ×4
Query analytics [28]	Real SSD	Session-based, compatible with standard SATA/SAS	Database (Microsoft SQL)	SATA/SAS

The "–" in the table indicates insufficient information is available.

(OS) image, containerized application, FPGA bitstream, or extended Berkeley Packet Filter (eBPF) bytecode [32].

Singh et al. listed the different metrics upon which NDP can be evaluated [15]:

- Memory: For our focus on ISP, SSDs are the most commonly used, where they fall in the storage memory hierarchy. Other memory types ranging from DRAMs, Commercial 3D (C3D) memories, Dual In-line Memory Modules (DIMM), Phase Change Memories (PCM), and others are among the viable options for NMC.
- Processing: The above mentioned CSD categories that range from existing SSD cores, general-purpose processors, and FPGAs summarize the different mechanisms that can be used.
- Tool: Various techniques and prototypes to validate ISP potentiality have been explored with some opting for usage of real SSDs [28], others for simulation models [33] or FPGA-based SSDs [34].
- Interoperability: It deals with the main barriers restraining applications of NDP, however, challenges can differ slightly whether NMC or ISP is applied. They can be briefly summarized in having a programming model, achieving cache coherence, supporting virtual memory, and providing effective data mapping mechanisms.
- Application: Determining the application to run on ISP can be tricky as NDP is usually specialized for a particular workload. However, some examples and potential applications include Redundant Array of Independent/Inexpensive Disks (RAID), Erasure coding, Regular Expression (RegExp), database filter, compression, data deduplication, and Scatter-Gather.

3. Processing near Storage Class Memories

In this section, we describe some of the notable ISP architectures. Discussion is dependent on classification of CSDs based on the services they provide, whether fixed or programmable. Table 1 summarizes the different fixed NDP architectures, whereas Table 2 summarizes most of the programmable ones.

3.1. Fixed services CSDs

Gu et al. introduced an NDP framework, Biscuit, which presents general-purpose solutions using high-level APIs composed of two libraries "libslet" and "libsisc" to allow coding data-intensive applications in the form of SSDlets that run using a distributed mechanism on both host and storage system using PCIe Gen-3 ×4 links [24]. Biscuit utilizes ARM Cortex-R7 real-time embedded processors distributed among the SSD firmware and user runtime in addition to hardware pattern matchers to offload and run tasks in place. These ARM cores were initially used for SSD procedures and even though this may have led to some performance leverage, the authors failed to show the effect on performance when both user applications and host I/O requests are running simultaneously. Biscuit follows a flow-based programming model [50] with a principle of separating it into computation and coordination models. Two important features supported by Biscuit's software layer to run ISP applications are: dynamic module loading and dynamic memory allocation. There are some hardware limitations, which pose challenges to Biscuit that include lack of cache coherency, less-than-ideal compute power, restrictive synchronization, and no large enough memory or Memory Management Unit (MMU). Biscuit is also thought of as a fixed-function unit CSD as users are constrained to use specific applications due to the usage of the flow-based programming model. Fig. 5 demonstrates the overall system of Biscuit, including its software and hardware architecture [24].

Kim et al. analyzed ISP benefits on SSD to accelerate key database operations, scan and join, for large-scale data analysis [25,26]. They claimed that embedded processors and memory are the main bottlenecks for storage architectures and hence, augmented flash channels, specifically the Flash Memory Controller (FMC), with a scan function to allow data filtering across the flash data path. With restricted amount of memory, processing power, and real-time constraints, their design is limited for two particular operations, specifically scan and join, preventing generalization. They evaluated ISP performance using simulation environments, varying host interconnects to include PCIe and SATA. Meanwhile, similar proposals regarding database operations targeting external sorting algorithms were proposed to perform on-the-fly merge operations in SSDs [51]. YourSQL is proposed, where it is

Table 2
Programmable NDP architecture classification.

NDP architecture	Prototype description/ evaluation model	Programming model	Application domain/workload	Host interconnect
Smart SSD [35]	OTS SATA SSD equipped with 2 ARM processors	MapReduce	Database	SATA
iSSD [33]	iSSD simulator running on SoC Designer and real SSD	MapReduce	Data-intensive kernels	SATA, PCIe
XSD [36]	Simulation model	MapReduce	MapReduce	–
Willow [37]	FPGA	RPC	Generic	NVMe over PCIe Gen-1.1 ×8
BlueDBM [34]	FPGA	API	Data analytics	Connectal and its PCIe Gen-1
Summarizer [38]	SSD development board (Multi-core ARM processor + FPGA)	API	Database	NVMe over PCIe Gen-3 ×4
Caribou [39]	FPGA	API	Database	Network interface using TCP/IP
CompStor [40]	NVMe SSD/ Quad core ARM A53	API	Search, compression/ decompression	NVMe over PCIe
RISP [41]	FPGA	–	Generic	SATA, PCIe ×4 and ×16
PRINS [42]	Associative processing units	Manual encoding at ASM level	Machine learning	–
INSIDER [43]	FPGA	Virtual file abstraction accessed via POSIX-like file I/O APIs	String matching, decompression, SQL query, Etc.	PCIe Gen-3 ×8 or ×16
Catalina [44]	MPSoC (quad-core ARM Cortex-A53 + FPGA)	MapReduce	Generic	NVMe over PCIe
DeepStore [45]	Simulation model	API	Intelligent-query	PCIe
REGISTOR [46]	FPGA	API	RegExp search	PCIe Gen-3 ×4
THRIFTY [47]	FPGA	INSIDER API	Speech and face recognition, medical diagnosis using HDC	Similar to a PCIe Gen-3.1 connected SSD
POLARDB [48]	FPGA	API	Database	PCIe
NASCENT [49]	FPGA on SmartSSD	API	Database	PCIe Gen-3 ×4

The "–" in the table indicates insufficient information is available.

built on a commodity NVMe SSD connected to the server via PCIe Gen-3 ×4 to enable early filtering, offering user-programmability and, through the usage of a hardware pattern matcher, allow on-the-fly table scan throughput [27]. However, regarding YourSQL, challenges related to code modification are incurred, which may result in security, performance and reliability issues. Significant code modifications were done to MariaDB [52] to incorporate its query planner and storage engine with Biscuit [24] in order to create a baseline system integrated with ISP. YourSQL has also two levels of filtering, both hardware and software, with the software involving some overhead that can affect overall performance.

Do et al. extended Microsoft SQL by making use of Samsung SmartSSD to perform relational analytic query processing [28]. An API-based programming model is provided to ease programming efforts and deal with host commands as well as manage query operation threads and exploit the memory within the SSD and data in the DRAM. Several drawbacks and limitations exist as a realistic database system architecture is not recognized due to unsupported join queries. In addition, there is a need to covert internal representation and layout of data to achieve reasonable speed-up. They have established a SATA/SAS-compatible session-based protocol however, such interfaces are outdated, which may be inconvenient for future systems.

3.2. Programmable services CSDs

Kang et al. proposed Smart SSD that uses an Off-The-Shelf (OTS) SATA-based Samsung MLC SSD equipped with 2 ARM processors [35]. An object-based communication protocol is used to provide flexible APIs to applications and OSs. The Hadoop MapReduce programming model [53] is extended to support the on-device map function, where this extension allows hiding the task handling details in addition to giving the applications the flexibility to run the exact API sets irrelevant of the communication protocol. However, due to supporting such extended Hadoop MapReduce version, the ISP task is broken into units of application called tasklets with cross compilation being required for the C program to allow the usage of ARM cores in the ISP engine. Future insights such as investigating job scheduling across several hosts and Smart SSDs, adopting an application processor for tasklets, and providing increased bandwidth and processing unit caches among the different CPUs and DRAM are yet to be researched. Park et al. discussed the integration of ISC with Hadoop MapReduce framework and the implementation of a unique SCSI command to set off ISC features within the SSD [54]. Design challenges including data representation discrepancy, system interface discrepancy and data split have also been discussed with possibilities to resolve them. One major problem regarding this proposal however, is the firmware's inability to dynamically allocate memory hence, requiring tasklets' preloading and memory space reservation. Making use of the dynamic memory allocator scheme upon which Biscuit was built can be explored as an option [55]. Another drawback that can be observed is the sharing of the internal ARM cores with the conventional flash management functions and the ISP engine, which restricts application development only to trusted people. Usage of a Trusted Execution Environment (TEE) is one way to ensure security through isolation of flash memory routines and in-storage programs.

Cho et al. presented an intelligent SSD (iSSD), which embeds SSD with heterogeneous processing elements [33]. To increase the data processing rate, they proposed to exploit a customized Application Specific Instruction Set Processor (ASIP) and a reconfigurable stream processor

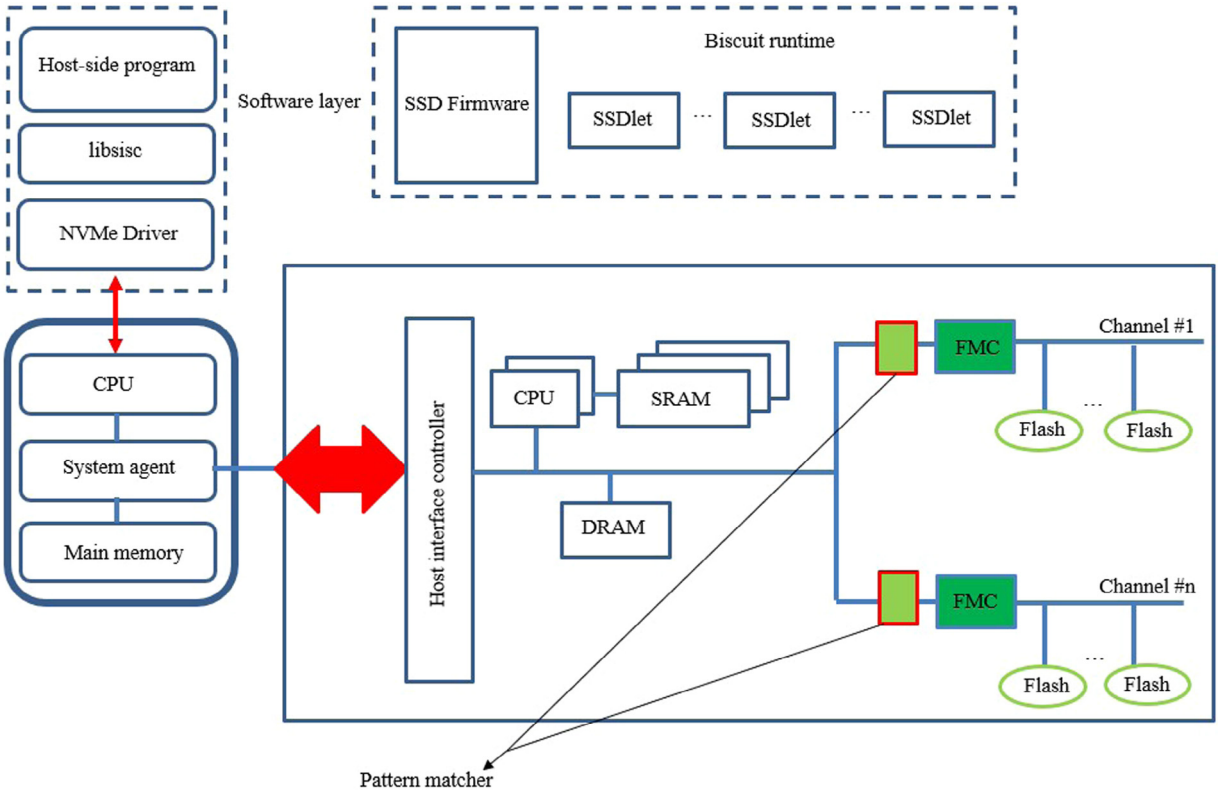


Fig. 5. Biscuit system architecture [24].

as the FMC processing resource since SSDs fall short when comparing their computing power to the data bandwidth hence, increasing parallelism. With the presence of embedded and stream processors, cache misses can be avoided as data can be fetched from a scratchpad memory, not a cache. They also discussed two mechanisms for data processing namely pipelining and partitioning to fully utilize the computing resources. Since the stream processor's bounded resources are entirely managed by the programmer, this leads to complex workloads' programming being more difficult. Validation of their data processing mechanisms was done using two prototypes: iSSD simulator that runs on SoC Designer and real SSD, using SATA and PCIe interfaces.

Cho et al. presented XSD, where they augment the SSD architecture by integrating a Graphics Processing Unit (GPU) near the embedded memory within the SSD, implementing their proposal using a simulator [36]. They demonstrated that adding a powerful CPU is not considered a suitable approach for accelerating data-centric workloads. Limitations for processing large-scale data sets also exist when using discrete GPUs. However, upon exploiting the SSD's high internal bandwidth, using an embedded GPU that captures parallelism and following a streaming scheme, performance independence from storage bandwidth can be achieved. Application parallelism can also be expressed through usage of MapReduce APIs, yet because of a modified MapReduce version, only compatible applications can be offloaded to the XSD. Also, due to sharing buffers between the flash channels and GPU in the DRAM, synchronization is required in addition to a lock-free scheme to allow simultaneous utilization of all channels.

Seshadri et al. proposed a prototype system implemented using FPGA, known as Willow, to allow programmers to extend and augment the SSD semantics with application-specific features without trading-off filesystem protections [37]. The FPGA is connected to the host system over PCIe with Willow providing programmable functionality represented by SSD Apps. Each App is composed of a set of generic Remote Procedure Call (RPC) handlers that provide no storage-specific functionality and can be installed at each Storage Processor Unit (SPU). That is in addition to a library required by the application to allow SSD App access and a kernel module if needed. Some of the challenges associated with Willow include striping the memory across SPUs complicating the RPCs' implementation as changes would be required over multiple memory banks. That is in addition to the limited complexity and performance provided by the Apps especially if concurrent SPU transfers are required and the number of Apps which can execute concurrently. Also, dynamically allocating memory or offloading tasks for running on SSD is unsupported.

Jun et al. presented BlueDBM, a novel system architecture that augments flash-based storage with ISP capability for data analytics and provides an inter-controller network with high-throughput and low latency [34]. An FPGA is used to implement the host and network controllers as well as the flash and in-storage processor. Two network interconnects are used to compose the cluster: one between the hosts and the other is the storage network, which permits one BlueDBM node to retrieve or access data at other nodes, which is demonstrated in Fig. 6. Connectal and its PCIe Gen-1 implementation are used for the host link. Even though from a performance perspective this architecture may be scalable due to the high-speed link provided by the storage network to other CSDs' data, adding more ISP-enabled nodes with the presence of two networks may imply physical scalability problems. Another challenge facing BlueDBM is the filesystem in use which is Remote File Sharing (RFS) [56]. Despite maintaining the mapping information allowing queries to the filesystem to retrieve files from their physical locations on the flash, being restricted to RFS could restrict generic adoptability of the solution. Several applications including string search, graph traversal and Nearest Neighbor Search (NNS) were evaluated. Several other researches were explored for graph analytics [57–59].

Koo et al. proposed Summarizer with the usage of APIs through applying simple modifications to the NVMe interface commands maintaining NVMe compatibility to allow filtering and summarizing stored SSD-data prior to transmitting it back to the host [38]. Summarizer is implemented on an NVMe SSD development platform using the built-in ARM-based cores within the SSD processor with minor modifications

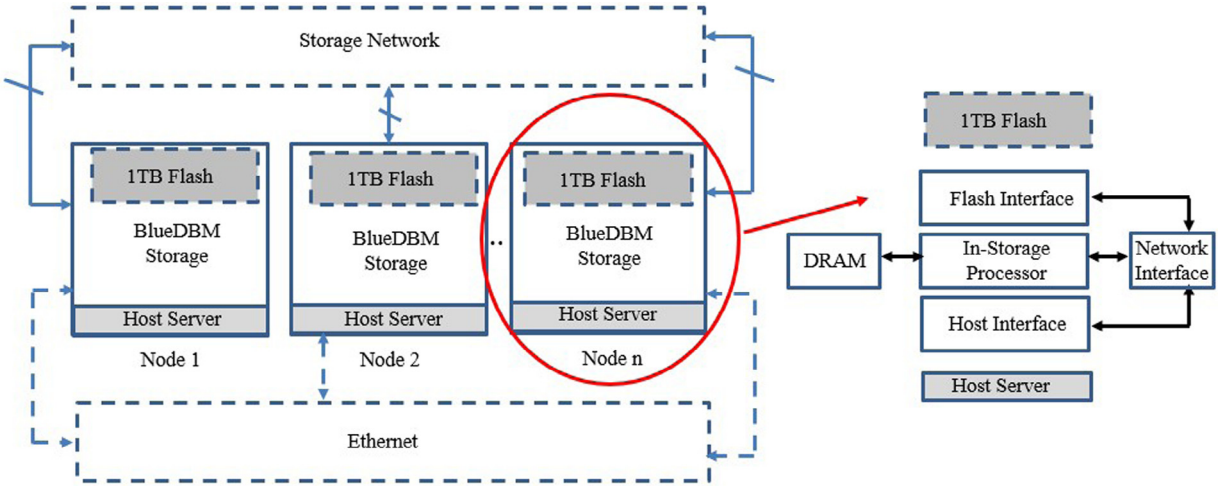


Fig. 6. BlueDBM architecture [34].

made to the NVMe command interpreter and addition of a software module to the SSD controller. Workload at the SSD processor is dynamically monitored by Summarizer enabling appropriate selection of execution whether at the host processor or the SSD processor. They evaluated different strategies with running the complete workload entirely on host or in SSD in addition to collaboratively using both. The main challenge on collaborative distribution is workload compilation to allow its execution on two different ISAs, where an x86 makes up the host processor, whereas the SSD controller is ARM-based.

Zsolt et al. introduced Caribou, an FPGA prototype, where they considered TCP/IP socket utilization to expose a key–value store interface to provide a completely distributed storage layer aiming at NDP on DRAM over the network [39]. Caribou integrates lots of aspects in its design including low latency and high throughput using a slab-based memory allocator hence, filling the allocation gap that exist in other solutions. They have also utilized bitmaps for efficient data scanning and provided selection for structured and unstructured data. Having transparent replication built within the system has increased its fault tolerance. The key–value stores domain has also been explored in other works like Minerva [60], nKV [61], nativeNDP [62], and BlueCache [63].

Torabzadehkashi et al. utilized an NVMe over PCIe SSD to introduce a computational storage architecture called CompStor [40]. CompStor implements a software stack and a dedicated quad-core ARM processor is used to build the in-storage processing subsystem that runs a full-fledged Linux OS. Using an OS has enabled running executables, and shell scripts and allowed dynamic task offloading, where tasks are loaded into the computational SSD at runtime. Two sub-systems constitute CompStor, as shown in Fig. 7, conventional sub-system bringing the flash memories together with the SSD controller modules and the ISP sub-system. The connection between these two subsystems is done through an FPGA mezzanine card, where the isolation between control and data paths ensures simultaneous processing of data and storage functionality without degrading either one's performance. Even though flexibility to the overall design may be achieved through usage of general processors for ISP, it may be insufficient in giving the needed processing power for intensive tasks' computation, since a user application could thwart an SSD's performance and vice-versa. They also explored the usage of NVMe over PCIe in a NAND-based storage system called Solana to demonstrate the effectiveness of ISP in Natural Language Processing (NLP) applications [64].

Lee et al. explored the performance gap between SSDs and DRAMs to develop SmartSAGE, a software/hardware co-design based on an ISP architecture [65]. They studied the possibility of utilizing capacity-optimized NVM SSDs to store Graph Neural Networks (GNN) data to allow training beyond the limits of main memory size.

Mohammadreza et al. also proposed the utilization of Samsung's SmartSSDs as a substitute for DRAM-based systems to implement SmartRec [66]. They offloaded two of the main neural recommendation model computations, sparse length sum and general matrix–matrix multiplication to the FPGA on the smartSSD. The FPGA's DRAM was used as a cache to overcome the slow accesses resulting from SSDs, hence, achieving high energy efficiency while maintaining higher storage capacity than a DRAM-based CPU system.

Song et al. proposed RISP, which employs FPGA, to resolve a common issue among most ISP techniques in which entire storage data processing resources are always working in full gear to serve a data-intensive application regardless of its data processing complexity by reconfiguring them accordingly [41]. A Reconfigurable Unit (RU) is considered to be the main difference between a RISP-based SSD and a conventional one. Every flash channel is augmented with an NVM controller and a data processing cell in addition to a public processing cell that is used by all channels. The public processing cell acts as a coordinator to process RISP channels' intermediate results or provide internal communication among them by serving as a shared memory. They have also compared the impact of host I/O bandwidth using a conventional host-CPU based computing architecture by utilizing different interfaces including SATA, PCIe $\times 4$ and $\times 16$.

Kaplan et al. presented PRINS, a novel processing-in-storage system based on Resistive Content Addressable Memory (ReCAM) system with an architecture not following the von Neumann model [42]. This helps resolve the bandwidth wall encountered in near-data von Neumann architectures, such as 3D DRAMs and CPU stacks or SSDs with embedded CPU. It targets machine learning algorithms' evaluation namely K-Nearest Neighbors (KNN) and K-means. Each ReCAM memory row acts as an associative processing unit with a microcontroller encoded in assembly language (ASM) being in charge of issuing instructions and handling baseline processing. Through the usage of associative processing units rather than boolean logic, execution of any logic or arithmetic operation could be carried out in a fixed number of cycles. PRINS however, suffers from a major drawback, which is the lack of data coherency between the ISP engine and the host's CPU prohibiting storage access from the host's side throughout in-storage task executions.

Ruan et al. helped overcome several challenges through introducing INSIDER, an FPGA-based reconfigurable drive controller as the in-storage computing unit, adopting PCIe Gen-3 as the interconnection [43]. Through using an FPGA, they can target several workloads rather than being constrained to only one. By separating the drive architecture into a control plane responsible for firmware logic and a data plane in the form of an accelerator cluster, security is provided to user data, where the drive is protected from being accessed by unwarranted

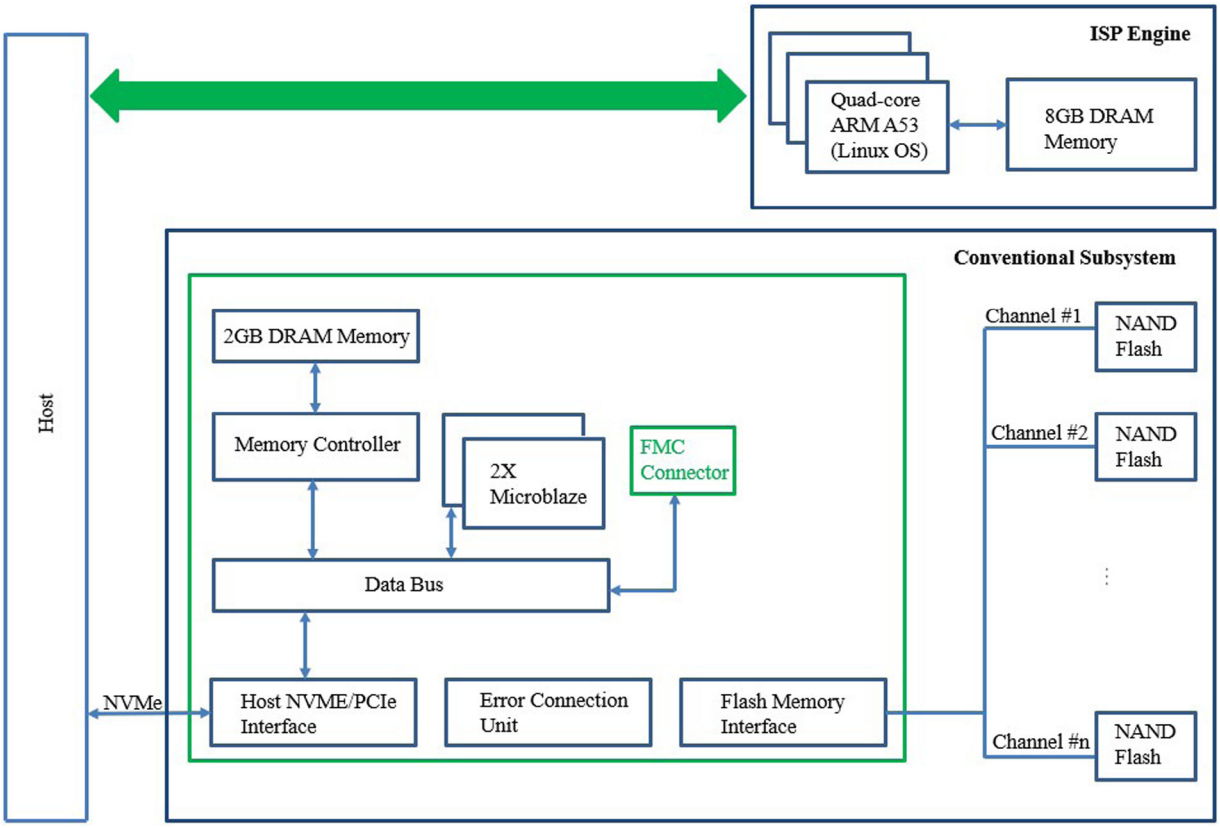


Fig. 7. CompStor architecture [40].

programs. INSIDER introduces virtual file abstraction as the host-side programming model and defines a clear interface at the drive-side to hide all data movement details between the system components and the ISC unit. This helps overcome the need for programmers to develop customized APIs which could be incompatible with an existing system interface, like POSIX.

Torabzadehkashi et al. proposed an architecture called Catalina, illustrated in Fig. 8, which like CompStor is composed of an NVMe over PCIe SSD supporting 64-bit Linux OS [44]. However, the major difference between them is having a single chip incorporating the Catalina controller that manages the different flash procedures besides the ISP engine preventing off-chip data transmission for ISP. Catalina is developed using an MPSoC computational storage platform and is mainly comprised of two sub-systems, the ASIC-based Processing System (PS) and the Programmable Logic (PL). The PS includes the main ISP engine of Catalina, which is ARM-based and equipped with floating-point units and Neon SIMD engines [67], the memory controller, and other data movement and interconnect components. The architecture also incorporates FPGA accelerators, which could be effortlessly tapped to boost specific applications' performance significantly.

Vikram et al. proposed DeepStore through the employment of specialized accelerators in PCIe-based SSDs to support DNN-based intelligent queries providing programming APIs that internally use new NVMe commands [45]. DeepStore is implemented through employing an SSD simulator built with SSD-Sim [68] and SCALE-Sim [69]. Similarity comparison, which is the main component of intelligent queries, is computed using neural-network accelerators rather than the conventional way of embedding multi-core CPUs in an SSD. Other than the employment of in-storage accelerators that consist of a systolic array of Processing Engines (PEs), a scratchpad memory, and a controller, DeepStore consists of a runtime query engine, and a similarity based in-storage query cache.

Pei et al. accelerated RegExp searches in storage devices using a deep pipeline structure known as REGISTOR [46]. Two main parts constitute REGISTOR: hardware architecture represented in the search engine implemented on an FPGA and the software stack. The different hardware components include: a file semantics extractor to retrieve file locations and match expressions across different blocks, a matching candidates finder to hit possible string matches in addition to RegExp Matching Units (REMUs) for exact equivalence and a results organizer. Two levels of APIs are designed to provide users with the needed functions for application development. Another RegExp matching ArChiTecture (REACT) was proposed by embedding an accelerator in the SSD and providing a scheduling algorithm for data access [70].

Gupta et al. proposed an ISP solution, THRIFTY, which executes HyperDimensional Computing (HDC) [71] encoding and training across multiple levels of the storage hierarchy allowing for highly-parallel ISC [47]. Using simple low-power digital circuits, they provide on-chip acceleration for HDC in addition to augmenting it with a top-level FPGA accelerator that further processes data collected from the chips. Such low power circuits are represented in a die-level accelerator and a scratchpad, as shown in Fig. 9 [47]. Every read page is encoded into a hypervector using the on-die accelerator to be accumulated in batches using the top-level accelerator and stored as a projection matrix using the scratchpad. THRIFTY's base system-architecture exploits INSIDER ISC [43], to provide software support. Another difference is also shown when comparing THRIFTY to Deepstore [45] regarding the transferal of hypervectors, which affects execution time. For Deepstore, raw data is encoded into hypervectors before being sent for training to the host which is not the case for THRIFTY hence, relaxing the SSD channel bottleneck faced by THRIFTY. They assumed characteristics of a PCIe-3.1 connected SSD for THRIFTY drive simulation.

A cloud-native Online Transaction Processing (OLTP) database designed by Alibaba Cloud, POLARDB, was proposed by Cao et al. to apply heterogeneous computing to the storage nodes to achieve early predicate evaluation by supporting table scan pushdown [48]. Through enhancements made to the software stack, including the distributed

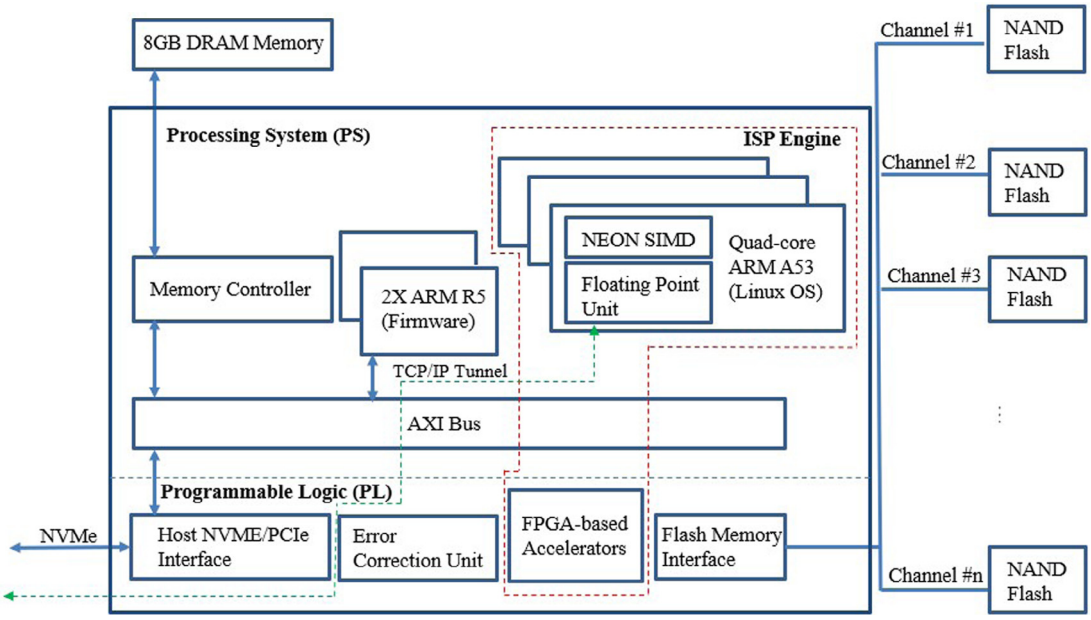


Fig. 8. Catalina architecture [44].

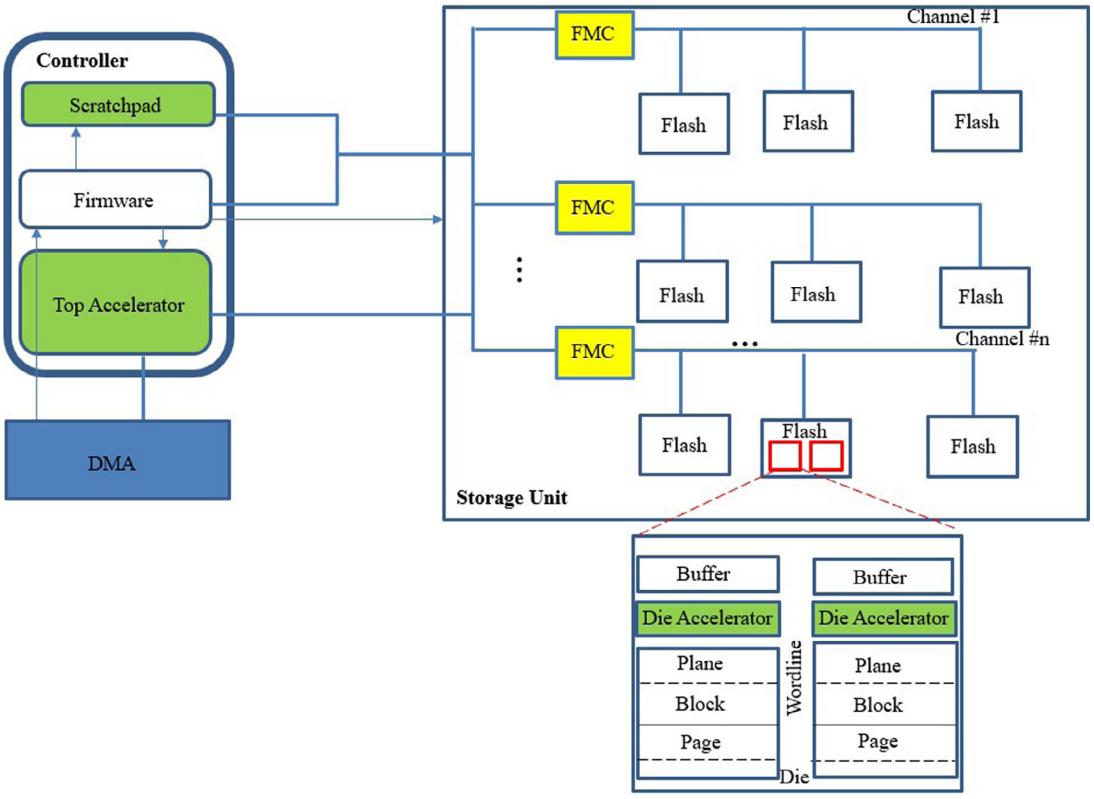


Fig. 9. THRIFTY architecture [47].

filesystem PolarFS [72], they were able to support table scan pushdown and mitigate the first challenge of having the complete storage I/O stack fitting between the POLARDB storage engine and the CSD. An FPGA-based host-managed CSD design strategy is targeted for implementation with modifications to lower the hardware associated cost using PCIe as the storage interface. However, data needs to be checked against complete table scan conditions as not all possible scan conditions are supported.

Salamat et al. utilized Samsung SmartSSD to provide a near storage accelerator, NASCENT, for database sort based on bitonic sort [49,73,74]. They introduced a novel FPGA-based architecture to exploit parallelism with the FPGA connected to the system through PCIe bus. Comparing the conventional architecture to their proposition, the former maintains a single FPGA, which is PCIe-connected to the storage devices limiting the number of simultaneous accesses to the multiple SSDs present. NASCENT on the other hand ensures direct connection

Table 3
Challenges faced by NDP architectures.

NDP challenges	Impacted NDP architectures' examples
Restricted amount of memory causing restrictions to particular applications [25,26]	Scan and join
Cache and data coherency [24,42]	Biscuit, PRINS
Software modifications [27,33,35]: • Code modifications to incorporate MariaDB's query planner and storage engine with Biscuit in order to create a baseline system integrated with ISP • Cross compilation for the C program to allow the usage of ARM cores in the ISP engine • Processor's bounded resources being managed by the programmer leading to difficulty programming complex workloads	YourSQL, Smart SSD, iSSD
File system [34]: • Being limited to a specific one may restrict the generic adoptability of the solution	BlueDBM
Memory allocation [35,37]: • Firmware's inability to dynamically allocate memory would require tasklets' preloading and memory space reservation	Smart SSD, Willow
Using same cores initially used for SSD procedures to run user applications may affect performance and restrict application development to trusted people [24,35]	Biscuit, Smart SSD
Synchronization which arises due to sharing buffers between the flash channels and core within the DRAM [36]	XSD

between a SmartSSD and an FPGA allowing partition sorting on the SSD-level independent of other SmartSSDs. This highly affects the system performance acceleration especially upon increasing the number of storage devices. FANS is also another SmartSSD-based architecture associated with sorting [75].

Table 3 summarizes the challenges faced by architectures studied in this section.

4. Near data acceleration challenges

Several barriers are still hindering the adoption of NDP and restraining their widespread usage. Adoption of new programming paradigms is a key challenge. Providing cache coherency and virtual memory support is also among such challenges, especially for PIM [76]. For ISP concerns, some of the main challenges and needed design goals when proposing a flexible and an efficient CSD architecture are discussed by Torabzadehkashi et al. [77]. Challenges increase especially with commercial workloads' integration with data/graph analytics and AI/ML [78,79]. These workloads require memory solutions, which could ease low-level integration of heterogeneous architectures as well as innovation for system interconnect network fabric that extends out to I/O and storage. Several work addressed cache coherence for PIM with proposals ranging from conventional methods to new approaches [80–84]. Virtual memory challenges, especially because of hosts and PIM modules sharing the same memory, have also been discussed in previous work [76,81,85,86].

In this section, we discuss some of these challenges focusing on programming models, security, and filesystem.

4.1. Programming models

Code-offloading is a real challenge to adoption of NDP especially due to the various modifications that are needed at the host-side, which include ISA modifications and programming effort. Memory level parallelism should be exploited by such programming models, libraries, and languages. There could be some platforms that include several heterogeneous processor cores, and hence, the programming framework should be unified to manage parallelism for such cores and provide co-ordination between heterogeneous devices and host CPUs. Such systems demand full-core implementation and may cause critical overhead with regard to area and power [80].

Multi-threading techniques were adopted for PIM, like OpenMP, MPI, CUDA, and OpenCL, which may be suitable especially for iSSD platforms [33] that include multiple heterogeneous processor cores [85,87–89]. Other proposals have also adopted MapReduce frameworks [90], to enable processing enormous amounts of data in a distributed manner [35,36,44].

Lee et al. developed a PIM framework to match PIM programming and execution concepts with those of parallel programs in multi-core environments [91]. That was done through mapping each bank in memory to a single core with one computing unit per bank, hence, allowing memories with several banks to act as a multi-core system.

SNIA TWG has presented a draft document for CSDs that aims to standardize the software application interface definitions using APIs to provide a way to access compute offload capable devices [92]. SNIA TWG has also published the standard for computational storage architecture and programming model to determine the hardware and software co-design needed for computational storage [19].

4.2. Security

Enforcing security in SSDs especially ones with compute capabilities is extremely important to avoid interference with conventional SSD operations or destruction and modification of data belonging to other applications.

In-storage vulnerabilities and attacks have been discussed with a proposal to defend against them through the usage of a TEE called IceClave [93] that considers the in-storage workload characteristics and unique flash properties. Previous solutions, like Intel's Software Guard Extension (SGX) [94], were adopted however, they come with two main problems: (1) not being supported by modern SSD processors and (2) requiring significant hardware changes. Similar solutions to SGX were also discussed [95,96].

4.3. Filesystem

Data access within storage may be done on the block-level only, limiting available programming model types and driving the need for mechanisms to access the file system metadata inside the ISP engine to retrieve the needed information. Equipping a CSD with an OS like Linux helps mount the filesystem and allows developing applications in any language supported by Linux OS.

Synchronization between the host's OS and the on-drive OS has also been discussed as a needed mechanism to overcome problems that may occur due to both OSs accessing the data stored in the flash memory array at the filesystem-level and simultaneously mounting

Table 4
Emerging interfaces for unleashing potentials from CSDs.

Interconnect	Use cases	Compatibility
NVMe-oF [99]	Cold storage (retention of inactive data)	Extension of NVMe (block-based data transfers)
CXL [100–102]	Acceleration and memory expansion	Reusage of PCIe Gen-5 PHY, adding flits (528-bit packet boundaries)
SDXI [103,104]	Acceleration and memory-to-memory data mover	Standardization of memory structures, functions setup, and control, allowing forward compatibility (interconnect agnostic)
Gen-Z [105,106]	Data sharing across racks and applications	Reusage of PCIe and Ethernet PHYs, but defining its own MAC and transport layers packet structures
OpenCAPI [107,108]	Acceleration and memory expansion	Provision of a new open standard, called Open Memory Interface (OMI)
CCIX [109,110]	Acceleration and memory expansion	Reusage of PCIe Gen-4 PHY and data link layers with support for additional speeds

the same storage media [77]. Torabzadehkashi et al. integrated the second version of the Oracle Cluster File System (OCFS2) to resolve this problem [97].

Ruan et al. introduced INSIDER that is based on virtual file abstractions that act as the host-side programming model to be accessed via APIs to create file existence illusion [43]. These virtual file mappings are stored in a hidden file created by INSIDER at system startup to be used during runtime to know the mapped real file position via the filefrag tool to be transferred to the drive.

Schmid et al. introduced a near-storage-compute-aware filesystem implementation to utilize storage resources without directly exposing locations of physical storage on an FPGA [98]. By integrating their proposed framework into Linux as a filesystem driver and repurposing Unix Pipes as a well-known OS primitive, they provide developers with operators that facilitate functional units' definition and allow application development for heterogeneous hardware systems.

5. Key interfaces to CSDs

With emerging workloads, such as high-performance computing, machine learning, Artificial Intelligence (AI), and deep learning, more potential from CSDs can be unleashed, if further development is made to already existing standards, such as NVMe, or if new interconnect standards are developed. It may facilitate data movement between the different elements and ensure data coherency and resilience through protocols running over these interconnects. Such interconnects could be crucial for better performance and energy efficiency. We provide an overview of some of these interconnects in the current section however, more exploration is still needed to incorporate them in the best possible way. Table 4 summarizes the different interfaces.

5.1. NVMe-oF

NVMe has been the industry standard for SSDs, providing lower latency, more scalability for SSDs over legacy interfaces and allowing full exploitation and capture of parallelism, as well as increasing the all-flash storage performance [99].

SNIA and NVMe are collaborating to introduce a programming interface that is compatible with different CSDs, which may involve extending the interface protocol to add new commands [111,112] without being vendor specific. That would allow applications to discover and use any computational storage resource attached to the computer.

With the scaling in connectivity to hundreds of nodes and creation of storage array networks, the need for having fabric interfaces has been further pushed with NVMe introducing the NVMe-oF that allows the drives to be remote from the servers and may even be connected over standard ethernet networking over TCP/IP [113].

Different fabric transports for NVMe are being investigated and developed, which include NVMe-oF using Remote Direct Memory Access (RDMA) that comprises InfiniBand [114], RoCE [115] and iWARP [116]. Also NVMe-oF using Fiber Channel (FC-NVMe) and using TCP/IP network. Fig. 10 demonstrates the different options. Examples of RDMA-based architectures in the ISP scope are Caribou [39] and BlueDBM [34].

5.2. CXL

Compute Express Link (CXL) is an open industry-standard interconnect [100,101] designed to meet the progressing and increasing demands of workloads that involve high-performance computation. CXL provides connection to more devices through switching for fan-out in addition to more efficient memory utilization and rack-scale efficiency through pooling. CXL reuses PCIe Gen-5 PHY and adds 528-bit packet boundaries called flits. Heterogeneous processing through deployment of different architectures that span matrix, vector, scalar, and spatial processors, which include CPUs, smart NICs, GPUs, and others is supported. CXL technology also considers asymmetric complexity and offers memory semantics, where it leverages PCIe with three mix and match protocols (CXL.io, CXL.cache and CXL.memory) to create a common memory space among the host and all devices. CXL provides data coherency using a high bandwidth low-latency connectivity. It provides three device types, as shown in Fig. 11, each used for a specific purpose. To further elaborate, CXL.io may be used for link up, initialization, device discovery and enumeration. CXL.cache is mainly associated with providing host-device interaction efficiently caching the host memory through low latency. Finally, CXL.mem allows the host processor to access device-attached memories through load and store commands.

Intel Corporation presented through SNIA how CXL resolves challenges faced by persistent memories, where it has allowed their movement off the DRAM memory bus in addition to discussing the key points and changes CXL 2.0 introduced for providing such support [102].

5.3. SDXI

SNIA TWG aims to provide a path for better data movement within a memory system through standardizing an acceleration interface [103]. SNIA published the first standard for the Smart Data Accelerator Interface (SDXI) to leverage architectural stability, modularity and provide a virtualization-friendly programming interface. It also aims to provide support for both kernel and user modes and offer new capabilities to accelerate virtualized workloads. According to the standard the main aim for the SDXI platform is to provide the following [104]:

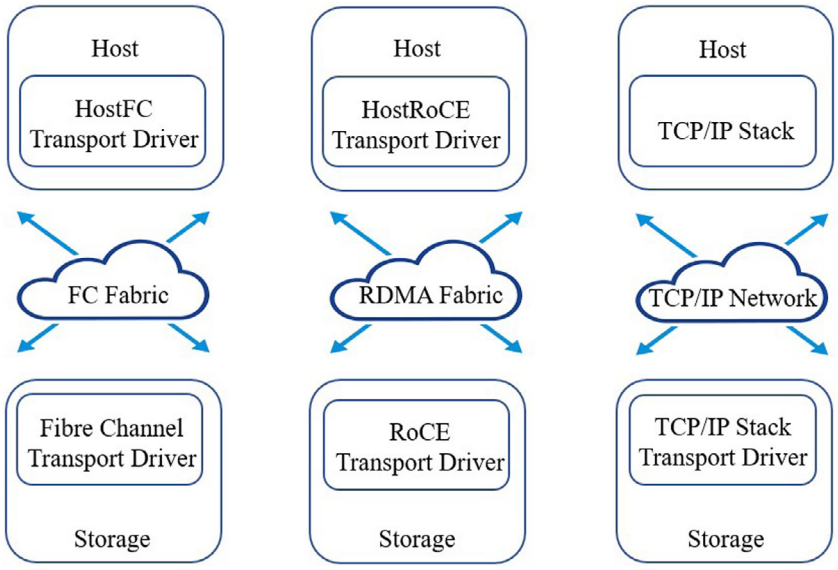


Fig. 10. Different transports for NVMe-oF.

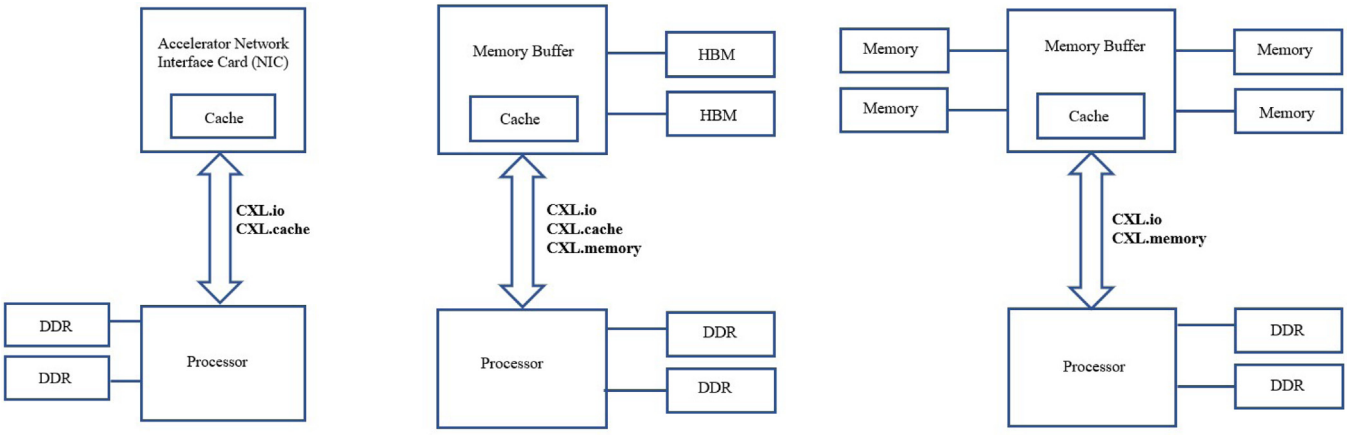


Fig. 11. CXL-associated device types [102].

- A forward-compatible interface that is both implementation-independent regarding data movement and independent of underlying I/O interconnect technology, where existing Direct Memory Accesses (DMAs) are all vendor-specific.
- A standard interface for user-mode address-space to address-space data movement without software intervention once establishing a connection in addition to providing data movement between multiple address spaces under privileged software control.
- An interface that can provide workloads or virtual machines compatibility across different servers through abstraction or virtualization by privileged software.
- Live workload or virtual machine migration allowance between servers through suspension and resumption of the architectural state of a per-address-space data mover.
- A forward compatible interface that ensures future hardware implementations can be operated on using pre-existing software drivers including user-mode drivers without modification.
- Ability for additional offloads' integration without alteration to the architectural interface.

A number of smart-data accelerators constitute the basic SDXI architecture, where they are enumerated as one or more SDXI functions to enhance performance and improve direct user-mode access. Fig. 12 [104] demonstrates the basic architecture.

5.4. Gen-Z

Using serial protocols to overcome challenges, where Printed Circuit Board (PCB) is becoming saturated and increasing the number of DIMM slots on the motherboard is no longer a scalable resolution, has certainly alleviated some problems however, serial communication brings its own throughput and latency challenges. Gen-Z is a new data access technology that can significantly enhance memory solutions built with existing or emerging memory technologies by enabling network attached memories that can be outside chassis or distributed across racks. It also surpasses other fabric protocols in having lower latency than other fabric protocols in addition to being effective in transferring reduced data volumes at cache line granularity [105]. Gen-Z has re-evaluated the layer model for serial communication to re-define some layers, such as the data-link and transport layers and

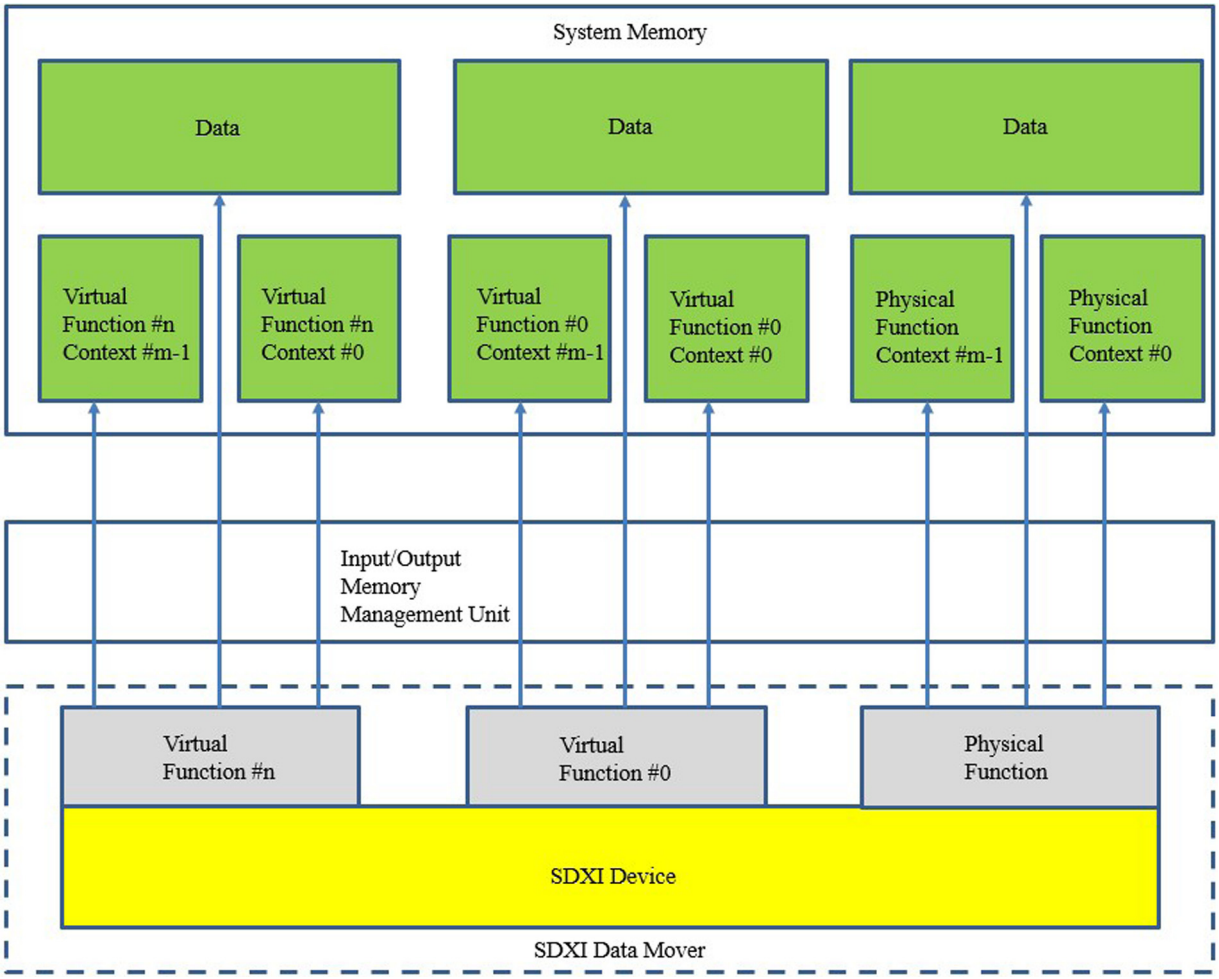


Fig. 12. SDXI architecture [104].

maintain others, such as the physical layer. For the physical layer, as in other serial protocols, same network cables and physical connectors are utilized by inheriting from the IEEE 802.3 standard and PCIe protocols where it reuses PCIe and Ethernet PHY [106]. Gen-Z abstracts network stack overhead, like UDP framework or TCP/IP. Unlike alternative fabric protocols, such as NVMe-over-Fabric or RDMA, Gen-Z defines its own packet formats, command structures, and handshake techniques enhancing the bandwidth utilization for load/store memory semantics [106]. This load/store model is enabled through Gen-Z decoupling compute from media allowing compute entities in a system to be media agnostic and disaggregated. Fig. 13 shows how Gen-Z enables rightful placement of media-specific functionality, where it belongs, allowing all devices to communicate through this load/store mechanism as peers to one another.

5.5. OpenCAPI

The Coherent Accelerator Processor Interface (OpenCAPI) addresses two main challenges: providing coherency to user-level accelerators and I/O devices in addition to making it available as open-source [107, 108]. An exemplary utilization for OpenCAPI was presented, where Schmid et al. implemented Metal FS over the CAPI Snap framework [98]. OpenCAPI is designed to support a variety of devices including coherent caching accelerators, network controllers, emerging memory technologies, in addition to storage controllers. Through the Open Memory Interface (OMI) which is an extension of OpenCAPI, standard memories that support low-latency operations can also be supported, as shown in Fig. 14.

Currently in an attempt to unify the memory interconnect standards and have a single organization driving forward, all OpenCAPI Consortium assets, as well as those of Gen-Z, will be assigned to the CXL Consortium [117,118].

5.6. CCIX

Cache Coherent Interconnect for Accelerators (CCIX) [109] from the CCIX Consortium [119] is a chip-to-chip interconnect, which allows data sharing between two or more devices in a cache coherent manner that in turn increases performance and lowers latency. The CCIX architecture expands on the base PCIe architecture using PCIe Gen-4 PHY and data link layers to enable a different number of topologies. The maximum link speed is also increased hence, increasing the raw bandwidth of the link. Figs. 15 and 16 show the layered architecture and different topology sets [109]. Xilinx presented details for memory expansion over CCIX in addition to utilizing it for storage with compute offload [110].

6. Future directions

In this section, we discuss the fundamentals needed for adoption of NDP. Future directions are suggested as follows:

- Introducing programming paradigms and prevalent software framework, which decreases the integration barrier of NDP elements to optimize data placement and movement within the system is needed [120].

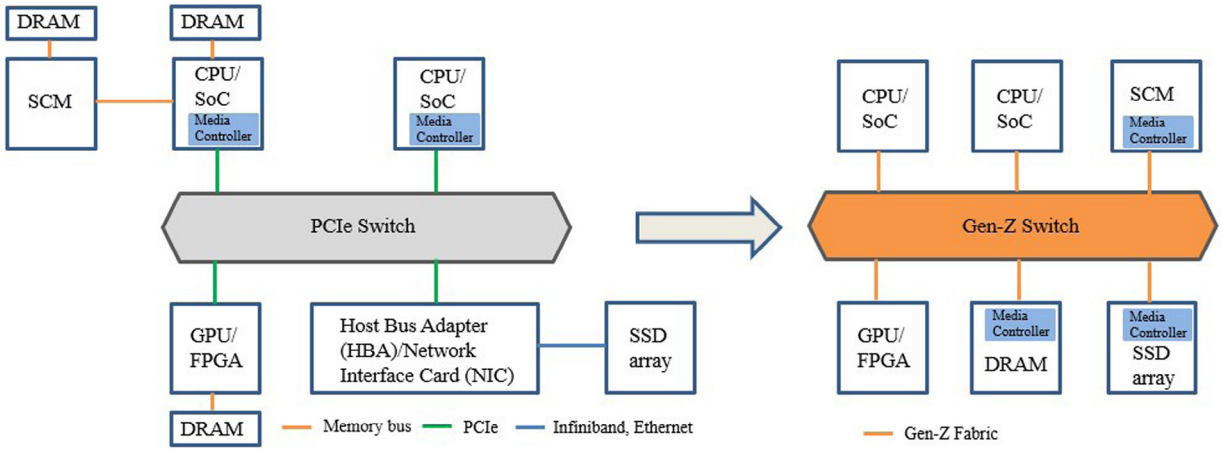


Fig. 13. Gen-Z system architecture.

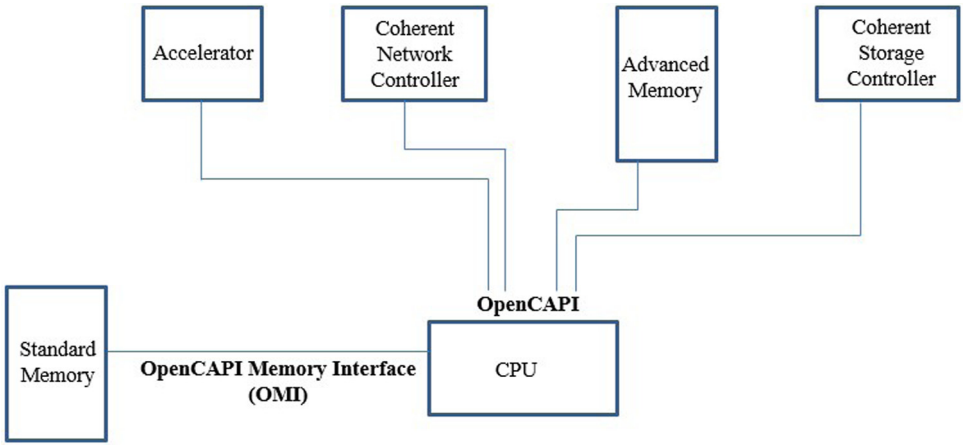


Fig. 14. OpenCAPI system architecture.

- Considering applying in-memory and in-storage computing simultaneously as a hybrid integrated architecture to exploit resources and achieve best possible processing speed and energy efficiency [121].
- Providing denser storage with higher energy-efficiency throughout the memory stack, while sustaining same access speeds of state-of-the art solutions or attempting for better [120].
- Resolving latency bottlenecks and memory bandwidth constraints through dense 3D integration between the memory hierarchy and different processing units [122–124].
- Integrating High-level synthesis (HLS) [125] tools into the CSD design flow to shorten implementation time for FPGA-based accelerators in ISP engines.
- Optimizing offloading decisions and providing task scheduling for better resource utilization as well as co-ordination between host and storage.
- Providing strategies for different benchmarks. Benchmarks depend on applications, such as the Purdue Map-Reduce (PUMA) Benchmarks Suite for Hadoop applications [126] and TPC-H for data analytics [127]. What is really challenging however, is the tools and baseline architectures against which ISP is compared. Lerner et al. discussed the different classes under which different tools fall: simulation, prototyping and hybrid [128].
- Investigating emerging NVM technologies' features, such as byte addressability for boosting ISP effectiveness [49,129].
- Exploring the emerging interconnects to provide better performance.

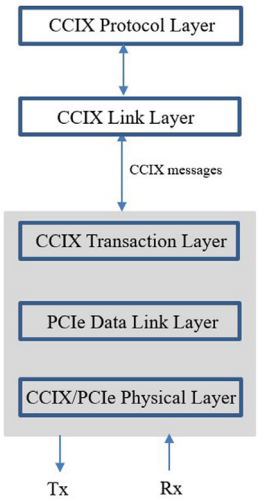
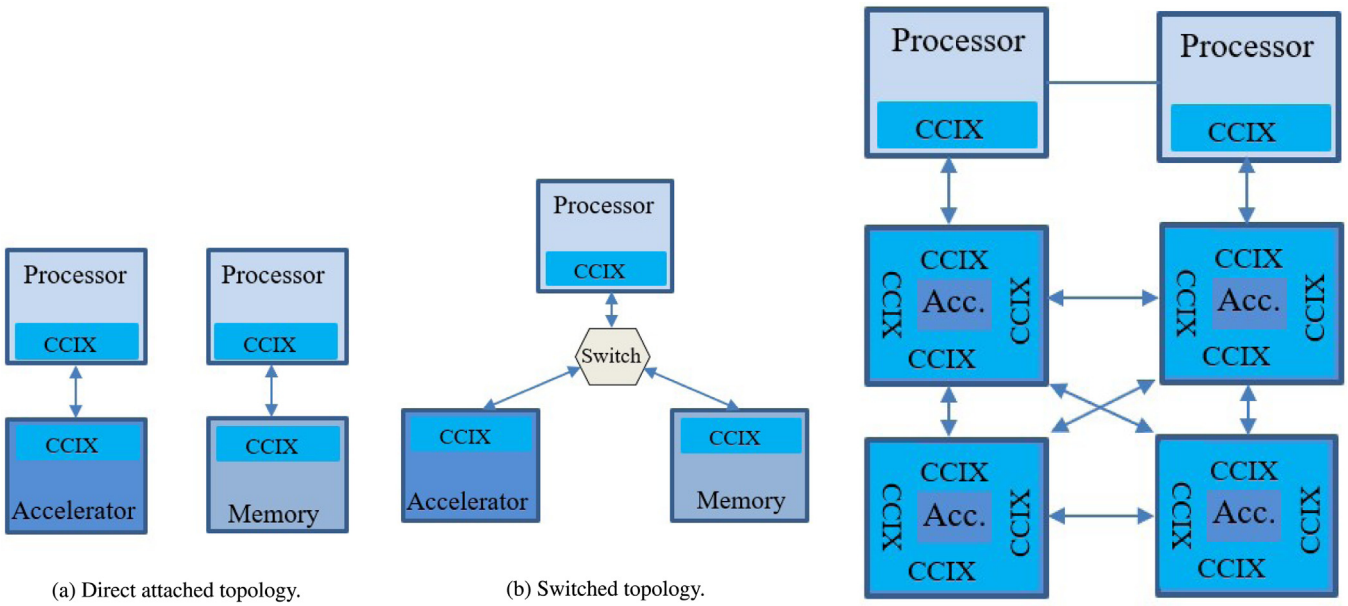


Fig. 15. CCIX architecture [109].

As part of future work, we propose adding an Over-The-Air (OTA) processor as part of the SSD controller to benefit from the CSD in performing cloud computation rather than computation on edge. A common use case for this would be in the automotive domain for black box recording, flashing over the air updates, intrusion/anomaly



(a) Direct attached topology.

(b) Switched topology.

(c) Hybrid daisy chain topology.

Fig. 16. CCIX topologies [109].

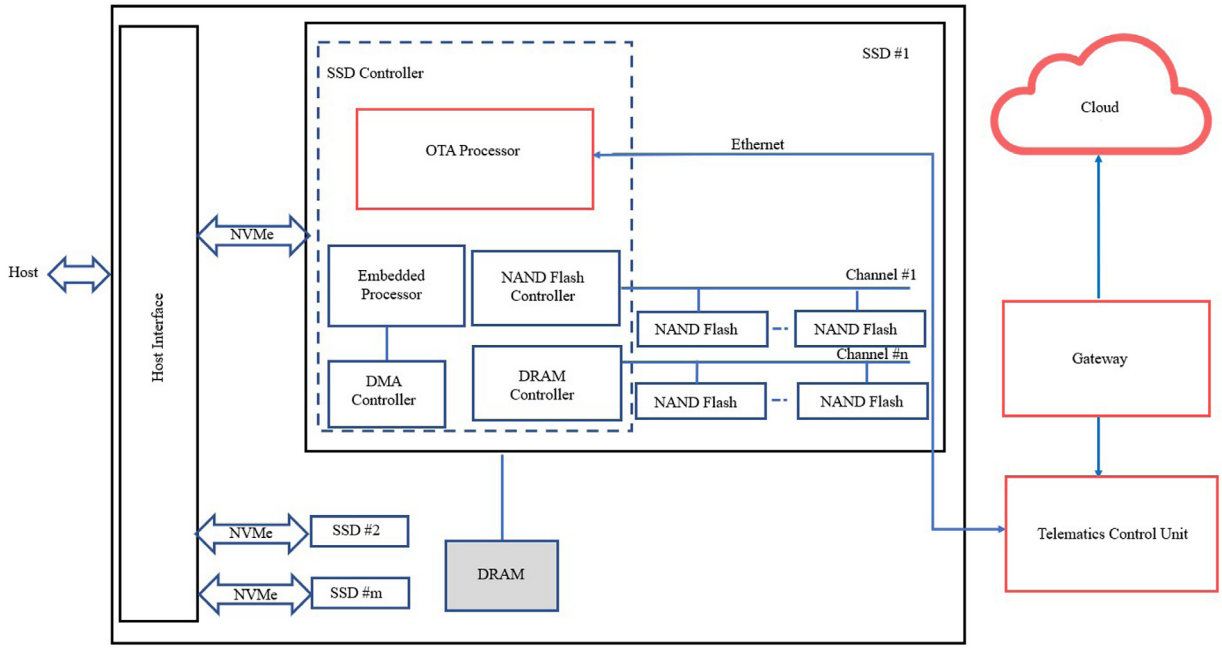


Fig. 17. Cloud-connected CSD.

detection and performing AI-on-cloud services extending fixed and programmable storage services. Fig. 17 shows the proposed architecture.

7. Conclusion

The exploding data volumes that require processing and enormous amounts of transportation from the memory system to the processor have led to challenges that triggered investigation of ISP opportunities. We presented the different architectures proposed in literature, demonstrating the coupling between a processing unit and the services provided (fixed/programmable) to elevate the performance and energy efficiency of data-intensive workloads through ISP integration and identified the key enablers for unleashing more potential from them. We also pointed some key challenges that require to be addressed and resolved in the future and identified how data movement can be facilitated through emerging interfaces.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] M.M. Angelic, J. Kaur, S. Das, Near data processing and its applications, in: R. Kumar, C.W. Ahn, T.K. Sharma, O.P. Verma, A. Agarwal (Eds.), Soft Computing: Theories and Applications - 6th International Conference, SoCTA 2021, in: Lecture Notes in Networks and Systems (LNNS), vol. 425, Springer Nature Singapore, Singapore, 2022, pp. 729–739.

[2] M. Mishra, A.K. Somani, On-disk data processing: Issues and future directions, 2017, Computing Research Repository (CoRR). arXiv:1709.02718.

[3] C. McCue, Future trends, in: Data Mining and Predictive Analysis: Intelligence Gathering and Crime Analysis, second ed., in: Methods and Applications in Data Mining, Butterworth-Heinemann, Oxford, UK, 2015, pp. 367–384.

[4] R. Awati, Processing in memory (PIM), 2023, TechTarget Business Intelligence Technology. URL https://www.techtarget.com/searchbusinessanalytics/definition/processing-in-memory-PIM. (Accessed 8 May 2023).

[5] D. Huang, Z. Qin, Q. Liu, N. Podhorszki, S. Klasky, Identifying challenges and opportunities of in-memory computing on large HPC systems, J. Parallel Distrib. Comput. 164 (2022) 106–122.

[6] What is computational storage? 2023, SNIA. URL https://www.snia.org/education/what-is-computational-storage. (Accessed 8 May 2023).

[7] J. Molgaard, Computational storage: A new way to boost performance, 2023, ARM. URL https://www.arm.com/blogs/blueprint/computational-storage. (Accessed 8 May 2023).

[8] S. Ghose, A. Boroumand, J.S. Kim, J. Gómez-Luna, O. Mutlu, Processing-in-memory: A workload-driven perspective, IBM J. Res. Dev. 63 (6) (2019) 3.

[9] O. Mutlu, S. Ghose, J. Gómez-Luna, R. Ausavarungnirun, A modern primer on processing in memory, 2022, Computing Research Repository (CoRR), arXiv:2012.03112.

[10] Hybrid Memory Cube specification 2.1, 2023, Micron. URL https://www.nuvation.com/sites/default/files/Nuvation-Engineering-Images/Articles/FPGAs-and-HMC/HMC-30G-VSR_HMCC_Specification.pdf. (Accessed 8 May 2023).

[11] High Bandwidth Memory (HBM) DRAM Standard No. JESD235D, 2023, JEDEC. URL https://www.jedec.org/standards-documents/docs/jesd235a. (Accessed 8 May 2023).

[12] K. Hsieh, S. Khan, N. Vijaykumar, K.K. Chang, A. Boroumand, S. Ghose, O. Mutlu, Accelerating pointer chasing in 3D-stacked memory: Challenges, mechanisms, evaluation, in: Proceedings of the 2016 IEEE 34th International Conference on Computer Design, ICCD, Scottsdale, AZ, USA, 2016.

[13] M. Gao, J. Pu, X. Yang, M. Horowitz, C. Kozyrakis, TETRIS: Scalable and efficient neural network acceleration with 3D memory, Comput. Archit. News 45 (1) (2017) 751–764.

[14] R. Nair, Evolution of memory architecture, Proc. IEEE 103 (8) (2015) 1331–1345.

[15] G. Singh, L. Chelini, S. Corda, A.J. Awan, S. Stuijk, R. Jordans, H. Corporaal, A.J. Boonstra, Near-memory computing: Past, present, and future, Microprocess. Microsyst. 71 (C) (2019) 102868.

[16] P. Siegl, R. Buchty, M. Berekovic, Data-centric computing frontiers: A survey on Processing-In-Memory, in: Proceedings of the Second International Symposium on Memory Systems, MEMSYS '16, Alexandria, VA, USA, 2016, pp. 295–308.

[17] S. Ghose, K. Hsieh, A. Boroumand, R. Ausavarungnirun, O. Mutlu, Enabling the adoption of Processing-in-Memory: Challenges, mechanisms, future research directions, 2018, Computing Research Repository (CoRR), arXiv:1802.00320.

[18] The storage networking industry association, 2023, SNIA. URL https://www.snia.org/. (Accessed 8 May 2023).

[19] Computational Storage Architecture and Programming Model, 2023, SNIA. URL https://www.snia.org/sites/default/files/technical-work/computational/release/SNIA-Computational-Storage-Architecture-and-Programming-Model-1.0.pdf. (Accessed 8 May 2023).

[20] M. Torabzadehkashi, A. Heydarigorji, S. Rezaei, H. Bobarshad, V. Alves, N. Bagherzadeh, Accelerating HPC applications using Computational Storage Devices, in: Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems, HPCC/SmartCity/DSS, Zhangjiajie, China, 2019, pp. 1878–1885.

[21] K. Neshatpour, M. Malik, M.A. Ghodrat, A. Sasan, H. Homayoun, Energy-efficient acceleration of big data analytics applications using FPGAs, in: Proceedings of the 2015 IEEE International Conference on Big Data, Big Data, Santa Clara, CA, USA, 2015, pp. 115–123.

[22] S. Rezaei, K. Kim, E. Bozorgzadeh, Scalable multi-queue data transfer scheme for FPGA-based multi-accelerators, in: Proceedings of the 2018 IEEE 36th International Conference on Computer Design, ICCD, Orlando, FL, USA, 2018, pp. 374–380.

[23] J. Bowen, Xilinx, Computational storage, 2023, SNIA. https://www.snia.org/sites/default/files/SDCEMEA/2020/4%20-%20Jamon%20Bowen%20Xilinx%20-%20Computational%20storage%20.pdf. (Accessed 8 May 2023).

[24] B. Gu, A.S. Yoon, D.-H. Bae, I. Jo, J. Lee, J. Yoon, J.-U. Kang, M. Kwon, C. Yoon, S. Cho, J. Jeong, D. Chang, Biscuit: A framework for near-data processing of big data workloads, in: Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture, ISCA, Seoul, Korea, 2016, pp. 153–165.

[25] S. Kim, H. Oh, C. Park, S. Cho, S.-W. Lee, B. Moon, In-storage processing of database scans and joins, Inform. Sci. 327 (C) (2016) 183–200.

[26] S. Kim, H. Oh, C. Park, S. Cho, S.W. Lee, Fast, energy efficient scan inside flash memory SSDs, in: Proceedings of the Second International Workshop on Accelerating Data Management Systems (ADMS 2011) During the 37th International Conference on Very Large Data Bases, VLDB 2011, Seattle, WA, USA, 2011.

[27] I. Jo, D. Bae, A.S. Yoon, J. Kang, S. Cho, D.D. Lee, J. Jeong, YourSQL: A high-performance database system leveraging in-storage computing, Proc. VLDB Endow. 9 (12) (2016) 924–935.

[28] K. Park, Y. Kee, J.M. Patel, J. Do, C. Park, D. Dewitt, Query processing on smart SSDs: Opportunities and challenges, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD '13, New York, NY, USA, 2013, pp. 1221–1230.

[29] R. Cheerla, Xilinx, Computational SSDs, 2023, SNIA. URL https://www.snia.org/sites/default/files/SDCEMEA/2019/Presentations/Computational_SSDs_Final.pdf. (Accessed 8 May 2023).

[30] Z. Ruan, T. He, J. Cong, Analyzing and modeling in-storage computing workloads on EISC–An FPGA-based system-level emulation platform, in: Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design, ICCAD, Westminster, CO, USA, 2019.

[31] N. Farahpour, Z. Fang, G. Reinman, FPGA-based near data processing platform selection using fast performance modeling (WiP Paper), in: Proceedings of the 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems, LCTES '20, London, UK, 2020, pp. 151–155.

[32] Berkeley Packet Filter, 2023, URL https://ebpf.io/. (Accessed 8 May 2023).

[33] S. Cho, C. Park, H. Oh, S. Kim, Y. Yi, G.R. Ganger, Active disk meets flash: A case for intelligent SSDs, in: Proceedings of the 27th International ACM Conference on International Conference on Supercomputing, ICS '13, Eugene, OR, USA, 2013, pp. 91–102.

[34] S.-W. Jun, M. Liu, S. Lee, J. Hicks, J. Ankcorn, M. King, S. Xu, Arvind, BlueDBM: An appliance for big data analytics, in: Proceedings of the 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture, ISCA, Portland, OR, USA, 2015.

[35] Y. Kang, Y.-S. Kee, E. Miller, C. Park, Enabling cost effective data processing with smart SSD, in: Proceedings of the 2013 IEEE 29th Symposium on Mass Storage Systems and Technologies, MSST, Long Beach, CA, USA, 2013.

[36] B.Y. Cho, W. Jeong, D. Oh, W. Ro, XSD: Accelerating MapReduce by harnessing the GPU inside an SSD, in: Proceedings of the 1st Workshop on Near-Data Processing (WoNDP) in Conjunction with the 46th IEEE/ACM International Symposium on Microarchitecture, MICRO-46, Davis, CA, USA, 2013.

[37] S. Seshadri, M. Gahagan, S. Bhaskaran, T. Bunker, A. De, Y. Jin, Y. Liu, S. Swanson, Willow: A User-programmable SSD, in: Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, OSDI '14, Broomfield, CO, USA, 2014, pp. 67–80.

[38] G. Koo, K.K. Matam, Te I, H.V.K.G. Narra, J. Li, H.-W. Tseng, S. Swanson, M. Annavaram, Summarizer: Trading communication with computing near storage, in: Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-50, Cambridge, MA, USA, 2017, pp. 219–231.

[39] Z. István, D. Sidler, G. Alonso, Caribou: Intelligent distributed storage, Proc. VLDB Endow. 10 (11) (2017) 1202–1213.

[40] M. Torabzadehkashi, S. Rezaei, V. Alves, N. Bagherzadeh, CompStor: An in-storage computation platform for scalable distributed processing, in: Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW, Vancouver, BC, Canada, 2018, pp. 1260–1267.

[41] X. Song, T. Xie, W. Pan, RISP: A reconfigurable in-storage processing framework with energy-awareness, in: Proceedings of the 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID, Washington, DC, USA, 2018, pp. 193–202.

[42] R. Kaplan, L. Yavits, R. Ginosar, PRINS: Processing-in-storage acceleration of machine learning, IEEE Trans. Nanotechnol. 17 (5) (2018) 889–896.

[43] Z. Ruan, T. He, J. Cong, INSIDER: Designing in-storage computing system for emerging high-performance drive, in: Proceedings of the USENIX Annual Technical Conference, USENIX ATC '19, Boston, MA, USA, 2019, pp. 379–394.

[44] M. Torabzadehkashi, S. Rezaei, A. Heydarigorji, H. Bobarshad, V. Alves, N. Bagherzadeh, Catalina: In-storage processing acceleration for scalable big data analytics, in: Proceedings of the 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP, Pavia, Italy, 2019, pp. 430–437.

[45] V.S. Mailthody, Z. Qureshi, W. Liang, Z. Feng, S.G. de Gonzalo, Y. Li, H. Franke, J. Xiong, J. Huang, W. Hwu, DeepStore: In-storage acceleration for intelligent queries, in: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-52, Columbus, OH, USA, 2019, pp. 224–238.

[46] S. Pei, J. Yang, Q. Yang, REGISTOR: A platform for unstructured data processing inside SSD storage, ACM Trans. Storage 15 (1) (2019) 13–25, 7.
[47] S. Gupta, J. Morris, M. Imani, R. Ramkumar, J. Yu, A. Tiwari, B. Aksanli, T.Š. Rosing, THRIFTY: Training with hyperdimensional computing across flash hierarchy, in: Proceedings of the 2020 IEEE/ACM International Conference on Computer Aided Design, ICCAD, San Diego, CA, USA, 2020.
[48] W. Cao, Y. Liu, Z. Cheng, N. Zheng, W. Li, W. Wu, L. Ouyang, P. Wang, Y. Wang, R. Kuan, Z. Liu, F. Zhu, T. Zhang, POLARDB meets computational storage: Efficiently support analytical workloads in cloud-native relational database, in: Proceedings of the 18th USENIX Conference on File and Storage Technologies, FAST '20, Santa Clara, CA, USA, 2020, pp. 29–41.
[49] S. Salamat, A. H.Aboutalebi, B. Khaleghi, J. H.Lee, Y. Ki, T. Rosing, NASCENT: Near-storage acceleration of database sort on SmartSSD, in: Proceedings of the 2021 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA '21, 2021, pp. 262–272.
[50] J.P. Morrison, Flow-Based Programming: A New Approach to Application Development, second ed., CreateSpace Independent Publishing Platform, 2010.
[51] Y.-S. Lee, L.C. Quero, Y. Lee, J.-S. Kim, S. Maeng, Accelerating external sorting via on-the-fly data merge in active SSDs, in: Proceedings of the 6th USENIX Conference on Hot Topics in Storage and File Systems, HotStorage '14, Philadelphia, PA, USA, 2014.
[52] MariaDB, 2023, URL https://mariadb.org/documentation/. (Accessed 8 May 2023).
[53] MapReduce, 2023, IBM. URL https://www.ibm.com/topics/mapreduce. (Accessed 8 May 2023).
[54] D. Park, Y.-S. Kee, J. Wang, In-storage computing for Hadoop MapReduce framework: Challenges and possibilities, IEEE Trans. Comput. (2016).
[55] D. Lea, A memory allocator, 2023, Datacenter Dynamics. URL https://gee.cs.oswego.edu/dl/html/malloc.html. (Accessed 8 May 2023).
[56] S. Lee, J. Kim, A. Mithal, Refactored design of I/O architecture for flash storage, IEEE Comput. Archit. Lett. 14 (1) (2015) 70–74.
[57] J. Lee, H. Kim, S. Yoo, K. Choi, H.P. Hofstee, G.-J. Nam, M.R. Nutter, D. Jamsek, Extrav: Boosting graph processing near storage with a coherent accelerator, Proc. VLDB Endow. 10 (12) (2017) 1706–1717.
[58] S. Jun, A. Wright, S. Zhang, S. Xu, Arvind, GraFBoost: Accelerated flash storage for external graph analytics, in: Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture, ISCA, Los Angeles, CA, USA, 2018, pp. 411–424.
[59] K.K. Matam, G. Koo, H. Zha, H. Tseng, M. Annavaram, GraphSSD: Graph semantics aware SSD, in: Proceedings of the 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture, ISCA, Phoenix, AZ, USA, 2019, pp. 116–128.
[60] A.D. M., Gokhale, R. Gupta, S. Swanson, Minerva: Accelerating data analysis in next-generation SSDs, in: Proceedings of the 2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM, Seattle, WA, USA, 2013, pp. 9–16.
[61] T. Vincon, A. Bernhardt, I. Petrov, L. Weber, A. Koch, nKV: Near-data processing with KV-stores on native computational storage, in: Proceedings of the 16th International Workshop on Data Management on New Hardware, DaMoN '20, Portland, OR, USA, 2020.
[62] T. Vincon, S. Hardock, C. Riegger, A. Koch, I. Petrov, nativeNDP: Processing big data analytics on native storage nodes, in: Proceedings of the 23rd European Conference on Advances in Databases and Information Systems, ADBIS 2019, Bled, Slovenia, 2019, pp. 139–150.
[63] S. Xu, S. Lee, S.-W. Jun, M. Liu, J. Hicks, Arvind, Bluecache: A scalable distributed flash-based key-value store, Proc. VLDB Endow. 10 (4) (2016) 301–312.
[64] M. Torabzadehkashi, A. HeydariGorji, S. Rezaei, H. Bobarshad, V. Alves, P.H. Chou, In-storage processing of I/O intensive applications on computational storage drives, in: Proceedings of the 23rd International Symposium on Quality Electronic Design, ISQED '22, San Francisco, CA, USA, 2021.
[65] Y. Lee, J. Chung, M. Rhu, SmartSAGE: Training large-scale graph neural networks using in-storage processing architectures, in: Proceedings of the 2022 ACM/IEEE 49th Annual International Symposium on Computer Architecture, ISCA, New York, NY, USA, 2022, pp. 932–945.
[66] M. Soltaniyeh, V.L.M.D. Reis, M. Bryson, X. Yao, R.P. Martin, S. Nagarakatte, Near-storage processing for solid state drive based recommendation inference with SmartSSDs®, in: Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering, ICPE '22, Beijing, China, 2022, pp. 177–186.
[67] Neon, 2023, ARM. URL https://developer.arm.com/Architectures/Neon. (Accessed 8 May 2023).
[68] N. Agrawal, V. Prabhakaran, T. Wobber, J.D. Davis, M. Manasse, R. Panigrahy, Design tradeoffs for SSD performance, in: Proceedings of the USENIX 2008 Annual Technical Conference, USENIX ATC'08, Boston, MA, USA, 2008, pp. 57–70.
[69] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, T. Krishna, SCALE-Sim: Systolic CNN accelerator, 2019, Computing Research Repository (CoRR). arXiv: 1811.02883.
[70] W.-S. Jeong, C. Lee, K. Kim, M.K. Yoon, W. Jeon, M. Jung, W.W. Ro, REACT: Scalable and high-performance regular expression pattern matching accelerator for in-storage processing, IEEE Trans. Parallel Distrib. Syst. 31 (5) (2020) 1137–1151.
[71] P. Kanerva, Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors, Cogn. Comput. 1 (2009) 139–159.
[72] W. Cao, Z. Liu, P. Wang, S. Chen, C. Zhu, S. Zheng, Y. Wang, G. Ma, PolarFS: An ultra-low latency and failure resilient distributed file system for shared storage cloud database, Proc. VLDB Endow. 11 (12) (2018) 1849–1862.
[73] S.G. Akl, Bitonic sort, in: D. Padua (Ed.), Encyclopedia of Parallel Computing, Springer, Boston, MA, USA, 2011, pp. 139–146.
[74] K.E. Batcher, Sorting networks and their applications, in: Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference, AFIPS '68 (Spring), Atlantic City, NJ, USA, 1968, pp. 307–314.
[75] W. Qiao, J. Oh, L. Guo, M.-C.F. Chang, J. Cong, FANS: FPGA-accelerated near-storage sorting, in: Proceedings of the 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM, Orlando, FL, US, 2021, pp. 106–114.
[76] P.C. Santos, B.E. Forlin, L. Carro, Providing plug N' play for processing-in-memory accelerators, in: Proceedings of the 2021 26th Asia and South Pacific Design Automation Conference, ASP-DAC, Tokyo, Japan, 2021, pp. 651–656.
[77] M. Torabzadehkashi, S. Rezaei, A. HeydariGorji, H. Bobarshad, V. Alves, N. Bagherzadeh, Computational storage: An efficient and scalable platform for big data and HPC applications, J. Big Data 6 (2019) 100.
[78] S. Bavikadi, P.R. Sutradhar, K.N. Khasawneh, A. Ganguly, S.M.P. Dinakarrao, A review of in-memory computing architectures for machine learning applications, in: Proceedings of the 2020 on Great Lakes Symposium on VLSI, GLSVLSI '20, 2020, pp. 89–94.
[79] H. Bao, Z. Houji, J. Li, H. Pei, J. Tian, L. Yang, S. Ren, S. Tong, Y. Li, Y. He, J. Chen, Y. Cai, H. Wu, Q. Liu, Q. Wan, X.S. Miao, Toward memristive in-memory computing: Principles and applications, Front. Optoelectron. 15 (2022) 23.
[80] P.C. Santos, J.P.C. de Lima, R.F. de Moura, M.A.Z. Alves, A.C.S. Beck, L. Carro, Enabling near-data accelerators adoption by through investigation of datapath solutions, Int. J. Parallel Program. 49 (2) (2021) 237–252.
[81] J. Ahn, S. Yoo, O. Mutlu, K. Choi, PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture, in: Proceedings of the 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA), Portland, OR, USA, 2015, pp. 336–348.
[82] K. Hsieh, E. Ebrahimi, G. Kim, N. Chatterjee, M. O'Connor, N. Vijaykumar, O. Mutlu, S. Keckler, Transparent offloading and mapping (TOM): Enabling programmer-transparent near-data processing in GPU systems, in: Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture, ISCA, Seoul, Korea, 2016, pp. 204–216.
[83] A. Boroumand, S. Ghose, B. Lucia, K. Hsieh, K. Malladi, H. Zheng, O. Mutlu, LazyPIM: An efficient cache coherence mechanism for processing-in-memory, IEEE Comput. Archit. Lett. 16 (1) (2017) 46–50.
[84] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, R. Ausavarungnirun, K. Hsieh, N. Hajinazar, K.T. Malladi, H. Zheng, O. Mutlu, CoNDA: Efficient cache coherence support for near-data accelerators, in: Proceedings of the 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture, ISCA, Phoenix, AZ, USA, 2019, pp. 629–642.
[85] J. Liu, H. Zhao, M.A. Ogleari, D. Li, J. Zhao, Processing-in-memory for energy-efficient neural network training: A heterogeneous approach, in: Proceedings of the 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-51, Fukuoka, Japan, 2018, pp. 655–668.
[86] M. Drumond, A. Daglis, N. Mirzadeh, D. Ustiugov, J. Picorel, B. Falsafi, B. Grot, D. Pnevmatikatos, The Mondrian data engine, in: Proceedings of the 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture, ISCA, Toronto, ON, Canada, 2017, pp. 639–651.
[87] R. Nair, S.F. Antao, C. Bertolli, P. Bose, J.R. Brunheroto, T. Chen, C.-Y. Cher, C.H.A. Costa, J. Doi, C. Evangelinos, B.M. Fleischer, T.W. Fox, D.S. Gallo, L. Grinberg, J.A. Gunnels, A.C. Jacob, P. Jacob, H.M. Jacobson, T. Karkhanis, C. Kim, J.H. Moreno, J.K. O'Brien, M. Ohmacht, Y. Park, D.A. Prener, B.S. Rosenburg, K.D. Ryu, O. Sallenave, M.J. Serrano, P.D.M. Siegl, K. Sugavanam, Z. Sura, Active Memory Cube: A processing-in-memory architecture for exascale systems, IBM J. Res. Dev. 59 (2/3) (2015) 17.
[88] D. Zhang, N. Jayasena, A. Lyashevsky, J.L. Greathouse, L. Xu, M. Ignatowski, TOP-PIM: Throughput-oriented programmable processing in memory, in: Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing, HPDC '14, Vancouver, BC, Canada, 2014, pp. 85–98.
[89] S. Jain, A. Ranjan, K. Roy, A. Raghunathan, Computing in memory with spin-transfer torque magnetic RAM, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 26 (3) (2018) 470–483.
[90] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.

[91] W.J. Lee, C.H. Kim, Y. Paik, J. Park, I. Park, S.W. Kim, Design of processing-"inside"-memory optimized for DRAM behaviors, IEEE Access 7 (2019) 82633–82648.
[92] Computational storage API, 2023, SNIA. https://www.snia.org/sites/default/files/technical-work/computational/draft/Computational%20Storage%20API%20v0.8r0%202022-06-29.pdf. (Accessed 8 May 2023).
[93] L. Kang, Y. Xue, W. Jia, X. Wang, J. Kim, C. Youn, M.J. Kang, H.J. Lim, B. Jacob, J. Huang, IceClave: A trusted execution environment for in-storage computing, in: Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-54, 2021, pp. 199–211.
[94] V. Costan, S. Devadas, Intel SGX explained, IACR Cryptol. EPrint Arch. (2016) 086.
[95] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, D. Song, Keystone: An open framework for architecting trusted execution environments, in: Proceedings of the Fifteenth European Conference on Computer Systems, EuroSys '20, Heraklion, Greece, 2020.
[96] V. Costan, I. Lebedev, S. Devadas, Sanctum: Minimal hardware extensions for strong software isolation, in: Proceedings of the 25th USENIX Conference on Security Symposium, USENIX Security '16, Austin, TX, USA, 2016, pp. 857–874.
[97] Oracle Cluster File System Version 2, 2023, URL https://docs.oracle.com/en/operating-systems/oracle-linux/6/admin/ol_ocfs2.html. (Accessed 8 May 2023).
[98] R. Schmid, M. Plauth, L. Wenzel, F. Eberhardt, A. Polze, Accessible near-storage computing with FPGAs, in: Proceedings of the Fifteenth European Conference on Computer Systems, EuroSys '20, Heraklion, Greece, 2020.
[99] NVM express (NVMe), 2023, URL https://nvmexpress.org/. (Accessed 8 May 2023).
[100] D.D. Sharma, Compute Express Link (CXL): Enabling heterogeneous data-centric computing with heterogeneous memory hierarchy, IEEE Micro 43 (2) (2023) 99–109.
[101] CXL™ 2.0: A high-speed interconnect for persistent memory challenges, 2023, SNIA. URL https://www.snia.org/educational-library/cxl%E2%84%A2-20-high-speed-interconnect-persistent-memory-challenges-2021. (Accessed 8 May 2023).
[102] CXL 2.0 - architecture and benefits for computational storage, 2023, SNIA. URL https://www.snia.org/educational-library/cxl-20-architecture-and-benefits-computational-storage-2021. (Accessed 8 May 2023).
[103] A new path to better data movement within system memory, computational memory with SDXI, 2023, SNIA. URL https://www.snia.org/educational-library/new-path-better-data-movement-within-system-memory-computational-memory-sdxi. (Accessed 8 May 2023).
[104] Smart Data Accelerator Interface ("SDXI") specification, 2023, SNIA. URL https://www.snia.org/sites/default/files/technical-work/sdxi/release/SNIA-SDXI-Specification-v1.0.pdf. (Accessed 8 May 2023).
[105] M. Krause, M. Witkowski, Gen-Z DRAM and persistent memory theory of operation, 2023, Gen-Z Consortium White Paper. URL https://genzconsortium.flywheelsites.com/wp-content/uploads/2019/03/Gen-Z-DRAM-PM-Theory-of-Operation-WP.pdf. (Accessed 8 May 2023).
[106] Gen-Z emerging technology for memory intensive applications, 2023, SNIA. URL https://www.snia.org/educational-library/gen-z-emerging-technology-memory-intensive-applications-2020. (Accessed 8 May 2023).
[107] J. Stuecheli, W.J. Starke, J.D. Irish, L.B. Arimilli, D. Dreps, B. Blaner, C. Wollbrink, B. Allison, IBM POWER9 opens up a new era of acceleration enablement: OpenCAPI, IBM J. Res. Dev. 62 (4/5) (2018) 8.
[108] OpenCAPI, 2023, URL https://opencapi.org/about/. (Accessed 8 May 2023).
[109] An introduction to CCIX, 2023, CCIX Consortium White Paper. URL https://www.ccixconsortium.com/wp-content/uploads/2019/11/CCIX-White-Paper-Rev111219.pdf. (Accessed 8 May 2023).
[110] M. Mittal, Xilinx, Memory expansion and storage acceleration with CCIX technology, 2023, SNIA. URL https://www.snia.org/sites/default/files/SDC/2019/presentations/Computational/Mittal_Millind_Storage_Acceleration_with_CCIX.pdf. (Accessed 8 May 2023).
[111] NVMe computational storage: A new hope for accelerators and DPUs, 2023, SNIA. URL https://www.snia.org/educational-library/nvme-computational-storage-new-hope-accelerators-and-dpus-2021. (Accessed 8 May 2023).
[112] A guide to computational storage on ARM, 2023, ARM White Paper. URL https://www.arm.com/resources/white-paper/computational-storage. (Accessed 8 May 2023).
[113] NVMe over fabrics overview, 2023, URL https://nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf. (Accessed 8 May 2023).
[114] InfiniBand Trade Association (IBTA), 2023, URL https://www.infinibandta.org/. (Accessed 8 May 2023).
[115] RoCE: RDMA over Converged Ethernet, 2023, The RoCE Initiative. URL https://www.roceinitiative.org/. (Accessed 8 May 2023).
[116] iWARP RDMA here and now, 2023, Intel. URL https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/iwarp-rdma-here-and-now-technology-brief.pdf. (Accessed 8 May 2023).
[117] S. Moss, Gen-Z Consortium closes after CXL Consortium merger, 2023, Datacenter Dynamics. URL https://www.datacenterdynamics.com/en/news/gen-z-consortium-closes-after-cxl-consortium-merger/. (Accessed 8 May 2023).
[118] A. Shah, OpenCAPI to be folded into CXL, 2023, HPCwire. URL https://www.hpcwire.com/2022/08/01/opencapi-to-be-folded-into-cxl/. (Accessed 8 May 2023).
[119] CCIX Consortium, 2023, URL https://www.ccixconsortium.com/. (Accessed 8 May 2023).
[120] The decadal plan for semiconductors, 2023, URL https://www.src.org/about/decadal-plan/. (Accessed 8 May 2023).
[121] M. Kim, S.-H. Kim, H.-J. Lee, C.-E. Rhee, Case study on integrated architecture for in-memory and in-storage computing, Electronics 10 (15) (2021) 1750.
[122] C. Liu, I. Ganusov, M. Burtscher, S. Tiwari, Bridging the processor-memory performance gap with 3D IC technology, IEEE Des. Test Comput. 22 (6) (2005) 556–564.
[123] N. Ghiasi, M. Sadrosadati, G. Oliveira, K. Kanellopoulos, R. Ausavarungnirun, J. Gómez-Luna, A. Manglik, J. Ferreira, J. Kim, C. Gianoula, N. Vijaykumar, J. Park, O. Mutlu, RevaMp3D: Architecting the processor core and cache hierarchy for systems with monolithically-integrated logic and memory, 2022, Computing Research Repository (CoRR), arXiv:2210.08508.
[124] C. Qian, L. Huang, P. Xie, N. Xiao, Z. Wang, A study on non-volatile 3D stacked memory for big data applications, in: G. Wang, A. Zomaya, G. Martinez, K. Li (Eds.), Algorithms and Architectures for Parallel Processing - 15th International Conference ICA3PP 2015, in: Lecture Notes in Computer Science (LNCS), vol. 9528, Springer, Cham, Switzerland, 2015, pp. 103–118.
[125] M.C. McFarland, A.C. Parker, R. Camposano, The high-level synthesis of digital systems, Proc. IEEE 78 (2) (1990) 301–318.
[126] F. Ahmad, S. Lee, M. Thottethodi, T.N. Vijaykumar, PUMA: Purdue MapReduce benchmarks suite, 2012, Purdue e-Pubs. URL https://engineering.purdue.edu/~puma/puma.pdf. (Accessed 8 May 2023).
[127] TPC-H benchmarks, 2023, URL https://www.tpc.org/tpch/. (Accessed 8 May 2023).
[128] A. Lerner, P. Bonnet, Not your Grandpa's SSD: The era of co-designed storage devices, in: Proceedings of the 2021 International Conference on Management of Data, SIGMOD '21, 2021, pp. 2852–2858.
[129] D.-H. Bae, I. Jo, Y.A. Choi, J.-Y. Hwang, S. Cho, D.-G. Lee, J. Jeong, 2B-SSD: The case for dual, byte- and block-addressable solid-state drives, in: Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture, ISCA, Los Angeles, CA, USA, 2018, pp. 425–438.

Dinal Fakhry received her B.Sc. from Ain-Shams University, Cairo, Egypt in 2018. She joined Mentor Graphics 2018–2023 working on hardware emulation targets, Memory Softmodels. She is currently working at Qualcomm as an emulation engineer for the secure processor. Her research interests include computer-aided design and verification, and Integrated Circuit design.

Mohamed Abdelsalam received his B.Sc. and M.S Degree from Ain-Shams University, Cairo, Egypt, and Doctor of Information Science and Technology from Osaka University, Osaka, Japan. He joined Mentor Graphics 1998-2002 working in development of circuit simulation and IC layout tools in CSD, and development of FPGA Advantage/HDS tool in DCS, and again in 2008 to 2021, in Global R&D Egypt MED solutions as Principal Engineer, working on hardware emulation targets, Memory Softmodels, Virtual Device Solutions and recently in 2022 as Software Engineering Director for solutions targeting new vertical market segments for Pre-Silicon Autonomous Verification Environment (PAVE360 Solutions), Cloud Connectivity and ML/AI applications. Mohamed AbdElSalam invented new techniques in the area of functional verification including Hybrid In-Circuit Emulation and Virtual Devices as Verification Components in SoC Verification Platforms. He has many publications in the area of real-time systems, HW/SW Co-design and System Level Modeling.

M. Watheq El-Kharashi received the B.Sc. (Hons.) and M.Sc. degrees in computer engineering from Ain Shams University, Cairo, Egypt, in 1992 and 1996, respectively, and the Ph.D. degree in computer engineering from the University of Victoria, Victoria, BC, Canada, in 2002. He is currently a Professor of computer organization and a chair of the Department of Computer and Systems Engineering. Ain Shams University; and also an Adjunct Professor with the Department of Electrical and Computer Engineering, University of Victoria. He has published 170

articles in refereed international journals and conferences and authored two books and eight book chapters. His current research interests include advanced system architectures, especially networks-on-chip (NoC), systems-on-chip (SoC), and secure hardware. More specific interests include hardware architectures for networking (network processing units) and security; advanced microprocessor design, simulation, performance evaluation, and testability; and computer architecture and computer networks education.

Mona Safar received the Ph.D. degree (2011), M.Sc. degree (2007), and B.Sc. degree (first class honors) (2004) in computer engineering from Ain Shams University, Cairo, Egypt. She is currently an Associate Professor in the Department of Computer and Systems Engineering, Ain Shams University. Her research interests include reconfigurable computing, application specific architectures, embedded systems, computer-aided design and verification, and electronic system level design.