






# Reviving Storage Systems Education in the 21<sup>st</sup> Century — An experience report

Animesh Trivedi\* , Matthijs Jansen\* , Krijn Doekemeijer\* , Sacheendra Talluri\* , and Nick Tehrani†<sup>§</sup>   
<https://atlarge-research.com/courses/storage-systems-vu>

\* Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands

† BlueOne Business Software LLC, USA

**Abstract**—We live in a data-centric world where many fundamental shifts in our daily lives are powered by Big Data. To meet the performance, cost, and energy demands of modern Big Data systems, there have been significant technological and engineering breakthroughs in the field of storage systems with novel hardware innovations and software architectures. Nevertheless, a typical computer science student still associates data storage solely with the technology of hard disk drives (HDD), which was invented six decades ago. One key reason for this association is the lack of courses on modern storage systems in computer science education curricula. In this paper, we make a concerted effort to summarize the state of storage systems education across universities, popular textbooks, and policies (ACM/IEEE). We make a case that the storage systems should have its own home course in educational curricula. We report on our experience of designing and offering one such course at the Vrije Universiteit, Amsterdam over the past four years. We further contribute to the educational material in this direction by making the course lectures, video recordings, assignments, and grading framework freely and openly accessible at <https://atlarge-research.com/courses/storage-systems-vu>.

**Index Terms**—Storage systems, Education, Experience report

## I. INTRODUCTION

We live in a data-driven society where a multitude of activities around us generate large quantities of data on a daily basis, collectively termed Big Data. According to one estimate, by 2030, we will produce one Yottabyte (1 Billion Petabytes) of data every year [1]. The availability of Big Data has enabled breakthroughs in various technological, societal, and economic domains such as transportation (autonomous vehicles), health-care (precision medicine, genomics), the exploration of scientific frontiers (CERN LHC and SKA experiments), democratic processes, climate science, and the AI/ML revolution [2]–[5].

The hard disk drive (HDD), which IBM invented in the late 1950s, has been the dominant technology to store data for many decades [6]. Consequently, advancements in our data storage and processing methods have been shaped by the technological trends and limitations of HDDs. For example, storage design recommendations such as avoiding random and/or small I/O from HDDs and data caching in DRAM (Jim Gray’s 5-minute caching rule [7]) have been developed to extract the maximum performance from HDD-based storage systems. The fundamental design of HDDs has not changed over the past decades, but the capacity, performance, and cost per GB have improved significantly [8]. Even today, HDD and

*“Unlike other related communities who have well-established “home courses” (e.g., operating systems, computer networks, databases), storage systems are mostly taught under the hood of other courses. [...] Many students may only associate storage systems with hard disk drives or a specific file system, which is obviously less attractive compared to, say, self-driving cars.” — Data Storage Research Vision 2025 (section 6) [24].*

HDD-driven software represent one of the success stories in the field of large-scale data storage and processing, and are taught in multiple universities (see §IV).

In the mid-2000s, the big data revolution with cloud computing created pressure on the performance of HDD-based systems, specifically the demand for low latency access to support emerging web-scale services [9]–[12]. At the same time, NAND flash-based solid-state storage devices (SSDs) emerged in mainstream commodity computing as an alternative to HDDs [13], [14]. The semiconductor-based storage design of flash cells freed them from the mechanical constraints restricting HDD’s performance improvements. NAND flash is one example of a broad class of storage technologies called Non-Volatile Memory (NVM) storage [15]–[17]. Today, NAND flash is commercially successful in the mass data storage segment, replacing HDDs in many performance-critical infrastructures. According to market projections, SSDs will represent 32% of all storage capacity shipped by 2026 [18], and its market is expected to grow to \$149.06 billion in 2026 at a compound annual growth rate (CAGR) of 31.86% [19]. In the high-performance computing (HPC) sector, 46 of the 50 world’s top storage systems use NVM/SSD-based storage [20], [21]. In cloud computing, all major vendors offer SSD-based storage services [22]. The emergence of SSDs has fundamentally reshaped every aspect of modern data storage and processing, re-writing many fundamental rules and design trade-offs [23] (see §II).

Despite its success and popularity, flash storage, and more broadly storage, has traditionally not been associated with a core computer science course (see §III). As a result, many students still associate the word “storage systems” with HDD-based systems that they study in their undergraduate education [24]. This HDD-centric view of storage continues even at the graduate level, as many top universities we surveyed lack

<sup>§</sup>Work done while the author was at the Vrije Universiteit, Amsterdam.

TABLE I: High-level comparison of HDD and SSD operational properties.

	HDD	SSD
<b>addressing</b>	sectors, 512B	also block/sector based, but byte-addressability now possible with CXL
<b>read/write</b>	symmetric performance	asymmetric, reads ( $\sim 10s \mu\text{secs}$ ) are faster than writes ( $\sim 100s \mu\text{secs}$ , w/o cache)
<b>random/sequential</b>	sequential I/O preferred	smaller gap between sequential and random performance, sequential <i>writes</i> preferred
<b>small/large I/O</b>	large I/O preferred	small difference between large and small I/O
<b>parallelism</b>	limited, a few I/O requests	very high, 100s of requests in parallel supported
<b>data placement</b>	inner vs outer tracks [25]	different dies, planes, channels, and blocks matter [26]
<b>data overwrite</b>	supported	not supported, flash cells must be erased (as a block, takes 10-100s ms) before writing
<b>maintenance</b>	limited supported needed	active support needed in the FTL for GC, wear-leveling, error corrections
<b>performance</b>	0.1-10msec, 10-100s MB/s	10-100 $\mu\text{sec}$ , 1-10s GB/s, 100-1,000k small I/O ops/sec (2-3 order-of-magnitude better)

a specialized course on *modern* storage systems<sup>1</sup> (see §IV). Storage systems-related topics are often covered under different courses such as databases, advanced operating systems, research seminars, distributed systems, computer architecture, etc., thus lacking a coherent narration and vision in education. This issue has been explicitly recognized by storage researchers in their 2018 plenary meeting to discuss “Data Storage Research Vision 2025” [24] (see the page-1 box). This gap in the education pipeline also creates challenges for the industry. For example, IBM reports that “nearly three-quarters of businesses have either already switched to NVM SSD storage or are planning to in the next year” [27]. At the same time, the lack of awareness and expertise are major hindrances and threats to adopting SSD/NVM [24], [28].

In this paper, we make a case that storage systems are a core component of computer science and require a dedicated course in university curricula. To facilitate the establishment of such a course, we report on our experience teaching a modern storage systems course at the graduate level at Vrije Universiteit (VU), Amsterdam since 2020. We provide the rationale for the course design, the selection of topics, and the motivation for assignments. We further discuss student reviews, and share our experiences running this course. Our key contributions in this article are:

- 1) We identify the major innovations in storage systems in the past decade due to the emergence of Non-Volatile Memory (NVM) Storage (§II) and make a case that *modern* storage systems (e.g., hardware, software, services, ecosystems) should have its own home course in university curricula.
- 2) We analyze how well the ACM/IEEE undergraduate computing curricula cover these innovations (§III). We then survey the top 35 universities (based on their ranking and reputation) and report how they cover modern storage systems topics in their graduate studies with a brief commentary on the textbook coverage of material (§IV). Our course survey is available at <https://zenodo.org/doi/10.5281/zenodo.10328799>.
- 3) We present the design of the (modern) Storage Systems

<sup>1</sup>**What do we consider modern?** We take a broad position (past 15 years) where topics related to post-HDD technologies specifically around NVM storage (e.g., NAND flash, Optane, Phase-change memory, Z-NAND flash) that have a direct impact on cloud/commodity systems are considered modern.

course at VU Amsterdam with its requirements and motivation for selecting topics based on their impact on systems software. We share our experience, student reviews, and current challenges facing this course (§V).

- 4) In order to facilitate the development of such a course at other institutions, we openly share the teaching material (slides, videos, assignments, automated grading infrastructure) under the Creative Commons (CC) BY 4.0 license at <https://atlarge-research.com/courses/storage-systems-vu>.

## II. FLASH STORAGE – A SELECTIVE PRIMER

Unlike the multicore revolution and emergence of advanced accelerators like GPUs for AI/ML workloads that were directly tied to the slowdown of Moore’s Law and Dennard scaling [29], the NVM storage revolution was a result of gradual economic and technological advancements [30]–[32]. In this work, due to its wide-scale impact on commodity cloud computing and HPC, we focus our discussion on non-volatile flash-based SSDs. However, we acknowledge that this is a selective view of a broader storage landscape that expands to tape [33], glass [34], DNA [35], and other emerging technologies [36], [37]. Table I shows a high-level comparison between SSDs and HDDs.

Flash storage is a type of non-volatile memory (NVM) storage technology invented by Toshiba in 1980. A flash cell uses an electronic charge on a floating gate between a control gate and the channel of a CMOS transistor to store data. NAND flash is a technique that packs flash cells together densely, which is a good fit for commercial mass storage capacities scaling up to GBs-TBs, but only allows addressing as a single unit called a *page*. A page is a unit of I/O, typically around 512B, 4–8KiB, and is always written or *programmed* sequentially at the cell level. After being programmed once, a flash page cannot be overwritten and must be erased before it can be programmed again. *Erasing* a page is an expensive operation typically done with a group of pages together, known as a *block*. Erasing a block may require relocating data from some pages to a new location. The (new) process of erasing a block and data management is known as *garbage collection* (GC). A GC process during an I/O operation leads to interference and performance degradation as both GC and the I/O operation need to access the same flash chips. The

TABLE II: Storage-related keywords (“SSD”, “storage”, “flash”, “memory”, “I/O”, “NVMe”, “Big Data”) hits and the context of SSD-related topics coverage in the curriculum. Topics such as memory (in Computer Architecture), network (Computer Network), and OS (Operating System) typically have their home courses, which storage lacks. NVMe had zero mentions.

	storage	SSD	flash	disk	I/O	memory	network	OS	remarks
CE2016	22	2	5	2	17	58	207	50	emphasis on low-level flash cell properties
CS2013	37	0	1	26	43	293	401	165	mostly covers disk-based storage concerns
CS2023	48	5	0	12	41	247	216	126	explicit mention of SSDs, but inadequate coverage of modern storage topics
IS2010	6	0	0	0	2	1	48	6	data storage concerns are discussed
IT2017	38	0	0	0	6	6	257	40	storage consumer point of view discussed
SE2014	2	0	0	0	0	1	15	8	very limited introductory discussion in OS
CSEC2017	7	0	0	4	0	4	87	5	very limited introductory discussion in OS
DS202x	30	0	1	1	3	14	16	17	disk-based) big data and cloud systems

excess data movement by the GC in response to a write from the host is known as *write amplification*. Flash cell wearing is also a concern as cells have limited program/erasure lifetime, after which they become corrupted. SSD vendors have extra capacity in their SSDs to tackle this challenge, called *over-provisioning*. Flash blocks are organized in a plane, die, and multiple parallel channels, thus supporting high parallelism for I/O operations. For an in-depth treatment of SSD-related topics we refer the reader to [14], [38]–[40]. Typically, a software layer called *Flash Translation Layer (FTL)* runs inside the SSD firmware that hides the complexity of flash media management from the host software and provides an HDD-alike fast block interface. This hiding leads to a semantic gap between the SSD and the host, thus creating “Unwritten SSD Contracts” [26].

**Impact:** The emergence of NVM storage and memories has significantly reshaped our storage architecture in domains such as, and not limited to: storage protocols like NVM Express (NVMe) [41], [42]; host interfaces [43], [44]; block layer [45]; I/O schedulers [46]–[49]; storage disaggregation [50]–[54]; programmable storage [55]; virtualization [56]–[58]; energy consumption [59], [60]; file systems [61], [62]; data structures [63]; key-value stores [64]; operating system designs [65]–[68]; I/O APIs [69], [70]; domain-specific workloads [71], [72]; distributed system [73], [74]; databases [75], [76]; security [77]; programming [78], [79]; computing architecture [80]–[83]; persistent memories [84]; and system failure patterns [85], [86].

### III. STATE OF THE COMPUTING CURRICULUM

Over the past decades, ACM and IEEE have jointly published computing curricula to shape computing education. The primary aim of this section is to provide (1) a summary of storage-related topic coverage within the seven sub-disciplines of computing curricula (see §2.3 in [87]) in §III-A and (2) a contrasting analysis between the 2013 and 2023 ACM/IEEE Computer Science curricula to report (any) revisions reflecting the emergence of NVM storage in computing in §III-B. The discussion here pertains to undergraduate (bachelors) education – we discuss graduate courses in §IV. To conduct

this analysis, we searched for storage-specific keywords in the documents: “SSD”, “storage”, “flash”, “memory”, “I/O”, “NVMe”, and “Big Data”. Based on the keywords’ presence and location, we collect insights from the policy document.

#### A. How are storage-related topics covered in various sub-disciplines of computing?

Computing curricula recommendations are split into the following sub-disciplines (as of 2023) for which ACM/IEEE provides recommendations (available at <https://www.acm.org/education/curricula-recommendations>):

- **Computer Engineering Curricula 2016 (CE2016):** The primary focus in this curriculum is on electrical properties, packaging, and signaling of storage technologies (SRAM, flash). Module CE-CAE-7 covers circuit properties of memory technologies, including flash. CE-CAO-7 describes characteristics of secondary storage, including SSDs. CE-CAO-9 includes data access topics from SSDs. Flash memory is discussed further in the context of embedded systems.
- **Computer Science Curricula 2013 (CS2013):** Considering this curriculum came out in 2013, there is only a single mention of flash memory in the “AR/Memory System Organization and Architecture” module. Much of the storage and data management-related discussion is centered around disk-based technologies (mentioned 26 times) in modules such as (core topics) “OS/Scheduling and Dispatch”, “SF/Resource Allocation and Scheduling” and (electives) “OS/Real-time and Embedded Systems”, “IM/Distributed Databases”, etc.
- **Information Systems 2010 (IS2010):** There is a limited and selective discussion of infrastructure and systems for data in “IS 2010.4 IT Infrastructure”, “IS 2010.2 Data and Information Management”, and “IS 2010.3 Enterprise Architecture”, with more recent advancements not covered. Their new competency model, IS2020, also shows similar coverage for storage-related topics.
- **Information Technology Curricula 2017 (IT2017):** Modules such as “ITE-PFT Platform Technologies”, “ITS-VSS-08 Storage”, “ITS-CCO-07 Cloud Infrastructure and

TABLE III: Storage topics coverage across universities in their graduate-level courses. “?” means that the course syllabus was unavailable, but a storage-specific course was present in the curricula.

	UW Madison	UC Santa Cruz	UC San Diego	UC Santa Barbara	KAIST	UChicago	UMichigan	Purdue University	Shanghai Jiao Tong	UC Riverside	Oxford	MIT	Stanford	ETH	CMU	Cambridge	NUS	UC Berkeley	Harvard	TU Munich	Imperial	Princeton	Tsinghua	NUS	UWashington	Cornell	Georgia Tech	UIUC	Caltech	Peking	Columbia	EPFL	UToronto	UEdinburgh	NYU
FS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Flash	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FTL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NVMe	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
KV	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PMem	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Disagg.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
API	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Dist.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Data”, “ITE-SWF Software Fundamentals”, and “ITS-ANE-06 Storage area networks” discuss storage concerns. However, the scope is from the technology-consumer point of view, hence, no recent advancements are mentioned.

- **Software Engineering Curricula 2014 (SE2014)** and **Cybersecurity Curricula 2017 (CSEC2017)**: They have a limited discussion around storage-related topics.
- **Data Science Curricula (under development) 202x (DS202x)**: This curriculum contains two explicit knowledge areas that cover data storage: “Big Data Systems (BDS)” and “Computing and Computer Fundamentals (CCF)”. In BDS, “BDS-Distributed Data Storage” mentions storage hierarchies without explicit recommendations or topics. CCF discusses architecture and operating systems fundamentals, covering topics similar to CS2013, from which it is derived.

**Summary:** Our key finding from the analysis of these curriculum documents is that the topic of storage has a scattered focus across many knowledge areas and domains, thus confirming the observation from [24]. In contrast, networking and operating systems-related concepts are treated extensively in a *single* knowledge area within their home course.

*B. How have computer science curricula recommendations changed between 2013 (the last release) and 2023 (the Gamma release, Aug 2023) from the storage perspective?*

For this comparison we analyzed the CS2023: ACM/IEEE-CS/AAAI Computer Science Curricula Gamma revision<sup>2</sup> report, released in August 2023 [88]. One of the most prominent changes in this curriculum is the inclusion of the Association for Advancement of Artificial Intelligence (AAAI) to cover AI/ML education (powered on Big Data). In the Architecture and Organization (AR) Knowledge Area, the 2023 document mentions three key updates from 2013 (page 55): (1) recent advances in memory caching and energy consumption; (2) emergence of Heterogeneous Architectures with domain-specific accelerators; and (3) Quantum computing. Updates on storage fall in the first class of changes. SSDs are mentioned in “AR-D/AR-Memory: Memory Hierarchy” and “OS-Devices:

<sup>2</sup>The final version of the curriculum is published on January 18th, 2024.

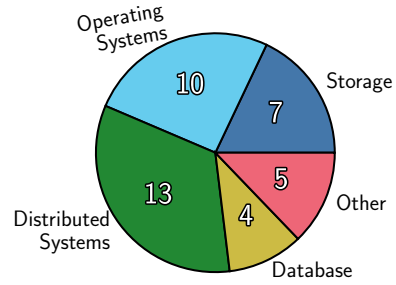


Fig. 1: Graduate-level systems courses that cover *any* storage topics. When multiple courses cover storage topics at a university, all of them are counted.

Device management”. “AR-E/AR-IO: Interfacing and Communication” has one core learning outcome: “List the advantages of magnetic disks and contrast them with the advantages of solid-state disks”. However, the 2023 curriculum also falls short of recognizing and addressing the emergence of NVM storage as one of the most fundamental changes in the past 15 years. In contrast, in other knowledge areas, such as Data Management (DM), major advancements since 2013, such as NoSQL and MapReduce, are included.

IV. STATE OF THE GRADUATE-LEVEL EDUCATION

We investigate the state of storage system education (graduate level) in the top 25 universities in the computer science category of the Times Higher Education rankings (2023). We include 10 additional highly ranked universities from the computer systems research category of csrankings.org for a total of 35 universities. We analyze the graduate-level course guides and syllabuses for the content of the selected systems courses, such as operating systems, distributed systems, databases, computer architecture, and storage-specific courses (if offered). We infer course content from the course description and the paper reading list when the syllabus is unavailable. The results of our survey are publicly available at <https://zenodo.org/doi/10.5281/zenodo.10328799>.

We report that 25 of the 35 surveyed universities have storage topics in their curricula. Storage is most often covered in

distributed systems or operating systems courses, as depicted in Figure 1. Storage-specific courses are offered by 7 out of the 35 universities. We further investigate the modern storage-specific topics covered by the different systems courses. The results are depicted in Table III. Distributed (Dist.) storage is the most popular topic, which aligns with the fact that the distributed systems course is the most common systems course referring to storage. File systems are the second most popular topic and are covered in most operating systems courses. SSD- and flash-aware file systems are covered primarily by storage-specific courses but also by a few other operating systems and general systems courses. The impact of the API and programming model features, such as immutability on storage systems, is covered by 4 courses. Only four courses cover the Flash Translation Layer (FTL). Persistent memory and disaggregation were only covered by one course each. NVMe and key-value stores are covered by storage courses, but also a few database courses. In most cases, the coverage of modern NVM-related storage topics is restricted to a few lectures.

#### A. Brief Commentary on the Text Books

Unlike Computer Networks, Operating Systems, and Computer Architecture, which have popular and established textbooks, storage-related topics are covered across various books, thus leading to a fragmentation of knowledge. This fragmentation mirrors how these topics are currently perceived in education and policy documents. We briefly comment on the coverage of modern storage topics in the following popular OS and architecture textbooks that are used in various curricula:

- **Computer Architecture: A Quantitative Approach (John L. Hennessy and David A. Patterson)** – In the 4<sup>th</sup> edition (2007) of the book, chapter 6 covers Storage Systems and topics such as disk internals, performance models, I/O scheduling, and queuing theory. However, in the 5<sup>th</sup> (2012) and 6<sup>th</sup> (2019) editions of the book, the chapter is removed and put as an appendix. The 5<sup>th</sup> edition introduces the advancements in flash memory in Chapter 2, Memory Hierarchy (chapter 2.3). The section introduces system-visible properties such as read, write, and erase operations and their performance properties, limited flash cell endurance, and packaging concerns. The 6<sup>th</sup> edition includes a brief discussion about Trends in Technology (chapter 1.4), PCM and Intel 3DXP (Optane) (chapter 2.2), and implications for networked-flash for Warehouse-Scale Computers (chapter 6.3). However, there are no subsequent discussions regarding the system software-level implications of NVM storage.
- **Modern Operating Systems (Andrew S. Tanenbaum, Herbert Bos)** – The foreword of the 5<sup>th</sup> edition (2022) explicitly mentions that the authors have given attention to flash-based SSDs. The book introduces SSD internals and persistent memory (chapter 1.3.3), flash-based file systems with the flash-translation layer (FTL) design and responsibilities (chapter 4.3.7), and the NVMe controller and PCIe-attached storage (chapter 5.4.2). Hence, the book

offers a good but introductory text about the pertinent concepts up to the file system level.

- **Operating Systems: Three Easy Pieces (Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau)** – The OSTEP book (v1.0, Dec 2023) has “Persistence” as the theme where multiple classical storage-related topics are covered extensively. The book has a dedicated chapter on flash-based SSDs (chapter 44) that comprehensively introduces multiple flash-SSD-based concepts such as flash cells, I/O operations, performance, erase, FTL designs, and garbage collection. The book also covers log-structured file systems (chapter 43) that play an important role in the design of flash-based file systems, thus offering the best coverage of introductory SSD-related topics.
- **Operating Systems: Principles and Practice (Thomas Anderson and Michael Dahlin)** – Volume IV of the 2<sup>nd</sup> edition (2014) dedicates chapter 12 to Persistent Storage. Flash storage is discussed in 12.2, where basic concepts such as flash cell design, FTL, NOR and NAND flash, erase, wear-leveling, and bad block management via over-provisioning are introduced. The chapter also briefly compares the performance of Intel 710 Series SSD (2011) with HDDs, although the performance and endurance numbers are over a decade old. In the chapter 12.3 on future directions, performance, capacity, and cost trends are identified for flash with a brief introduction to other emerging NVM technologies. No further software implications are discussed (except TRIM for file systems).
- **Operating System Concepts (Avi Silberschatz, Peter Baer Galvin, Greg Gagne)** – The 9<sup>th</sup> edition (2012) of the book briefly introduced SSDs (chapter 10.1.2), exclusively focusing on their performance properties without any internal details. Chapter 10.4.6 discusses how the high-performance and non-moving parts of SSDs need a NOOP I/O scheduler when considering I/O scheduling in the context of HDDs and SSDs. The 10<sup>th</sup> edition (2018) of the book recognizes “Nonvolatile memory secondary storage” as one of the four important trends shaping the design of operating systems. Chapters 11.1 and 14.5 cover further concepts in the design of an SSD, such as cell design, FTL, garbage collection, wear-leveling, over-provisioning, NVMe, and TRIM.

#### B. Summary

Unlike the undergraduate curriculum, the graduate courses we analyzed do cover *selected* recent advancements in storage systems. However, the details are spread over multiple courses due to the absence of a home course. Hence, it becomes challenging for an instructor to build an end-to-end narration where innovations from multiple domains can come together to “interpret storage systems in a broad way” [24]. As we approach the end of Moore’s Law, a cross-domain, across-the-courses understanding is important in delivering vertical hardware and software specialization. Various non-functional properties, such as performance, energy, or reliability gains, are delivered in an end-to-end manner where each system

component must be optimized and re-designed for a workload-specific specialization (vertical integration). One analogy here is to imagine how networking education would look if concepts from the layered reference architecture (L1/L2: media access, Ethernet; L3: IP; L4: TCP/IP) were taught in different courses. *Storage Systems education needs its own reference architecture and home course.*

## V. DESIGN OF A MODERN STORAGE SYSTEMS COURSE

We now present the design of a modern storage systems course at VU Amsterdam. It is a 6 ECTS (1 ECTS = 28 hours of study in the Netherlands) course offered in the Computer Systems and Infrastructure (CSI) track to Computer Science masters students. The course was established in 2020 and was offered for the 4<sup>th</sup> time in the 2023–2024 academic year. The course covers a 7-weeks of teaching block, with 2 lectures (1h:45mins each) and 1 lab session (1h:45mins) per week.

### A. Inclusion and Exclusion Criteria in the Course

One of the first challenges in designing a modern storage systems course is selecting topics we wish to cover. As we discussed in §II, the emergence of fast NVM storage has an impact on multiple aspects of systems building. We selected 11 lectures with a design of assignments where one or more topics could be implemented in practice. The remaining 3 lecture slots (7 weeks x 2 lectures/week) are used for project evaluation (2) and one guest lecture. We applied the following inclusion (IN) criteria when selecting topics to cover:

- (IN-1) The topic is about NVM internals/SSD architecture;
- (IN-2) The topic is about a system design and/or systems software (e.g., complexity, programming, debugging) whose advancements can be directly linked to unique properties of NVM storage;
- (IN-3) The topic is an emerging hardware/software trend that influences how storage is treated inside a computer;
- (IN-4) The topic pertains to one of the following non-functional topics: performance, efficiency, complexity, heterogeneity, or reliability linked directly to unique properties of NVM storage;
- (IN-5) The topic should be recent and relevant (2008–now, past 15 years);

We applied the following exclusion (EX) criteria:

- (EX-1) The topic pertains to the physics of NVM storage (partially due to the instructor’s lack of expertise);
- (EX-2) The topic is in domains other than cloud/data-center computing, such as embedded/sensor systems and cyber-physical systems; these domains have unique design considerations, such as a strong emphasis on reliability and data corruption;
- (EX-3) Having a unique focus on big data systems, we exclude other commonly used technologies such as high-density tapes or cost-economical disks from our discussion;
- (EX-4) System designs that only consume NVM storage because it is the most common storage technology around without any consideration for NVM unique properties;

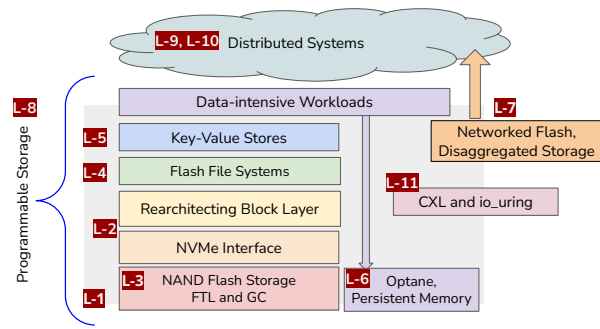


Fig. 2: Storage Systems lecture themes at VU Amsterdam.

### B. Course Structure at VU Amsterdam

The primary theme that governed our design process is to design a layered approach to study and understand the impact of NVM storage on software systems. To achieve this goal, we took inspiration from the network layered model and stacked the topics covered so they built on top of each other. Figure 2 shows the overall layout. In this section, we provide the content covered in the lectures (L) and our motivations for choosing the topic. A secondary goal is to connect students to the cutting-edge research in storage systems.

- [L-1] Introduction to NVM Storage:** We introduce NAND/NOR/XOR flash cells and chips, media-level differences between SSDs and HDDs, SSD packaging (dies, blocks, pages), and performance and endurance properties. We highlight that the storage design guidelines and trade-offs have changed with SSDs (Table I).
- [L-2] NVM Impact on Host-Interfacing:** We discuss how 2-3 orders-of-magnitude performance improvements of SSDs with high parallelism necessitated the development of a new host controller (NVM Express, NVMe) and the redesign of the Linux block layer [45], [89].
- [L-3] SSD Internals (FTL and GC):** We introduce the concept of the FTL and its responsibilities for active flash chips management (L-1). We then discuss why the host software (file systems, data stores) need to be aware of the FTL design, GC operations, and trade-offs captured as the Unwritten Contracts [26].
- [L-4] NVM Impact on File Systems:** We discuss how SSD internal properties and FTL designs (L-2/L3) prefer sequential writes, which can be generated by log-structured file systems (LFS). We analyze LFS designs, GC, their optimizations for flash SSDs (F2FS, SFS), and novel FS designs with software-defined flash [44], [90], [91].
- [L-5] NVM Impact on Key-Value Stores:** We introduce important lookup structures (B+ tree, hash table, LSM tree) and how unique properties of flash storage (asymmetrical read/write, and high parallelism) require these structures to consider their application-level read/write amplifications and space requirements (the RUM Conjecture [92]).
- [L-6] Emergence of Byte-Addressable Persistent Memories (PMEM):** The availability of PMEM like Optane [93] enable us to build novel abstractions like persistent data

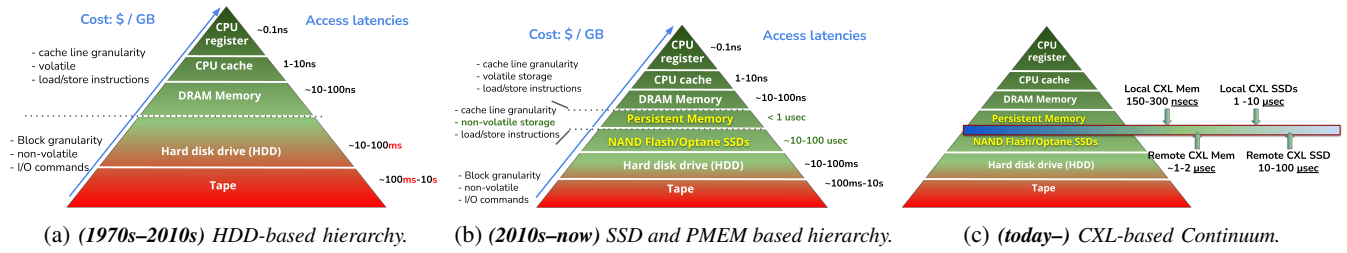


Fig. 3: Evolution of the Storage-Memory hierarchy from HDD-driven hierarchy to modern **Storage-Memory Continuum**.

structures [94], [95] and even novel OS designs [65], thus blurring the difference between a file and memory [96].

**[L-7] NVM Impact on Networked Storage:** We draw an analogy with L-1/2 and establish that fast storage requires fast networks and *co-designed* network-storage protocols as such NVMe-over-Fabrics (NVMeoF). We discuss the concept of storage/flash disaggregation [97] and various block-level, file system-level, and application-level (RDMA) access to remote storage [51].

**[L-8] NVM Impact on Computational Storage:** We introduce the problem of the data movement wall and the opportunity with SSDs that have an active programmable element, the FTL. Hence, a user can run its data processing program close to storage, inside an SSD with the FTL, thus enabling Computational or Programmable storage. We discuss its origins [98], [99], modern interpretations [55], and various hardware/software design options that offer performance and efficiency via specialization [100].

**[L-9] Impact on Data Processing:** We identify the opportunity with fast storage and networks (L-7) that help distributed data processing frameworks to efficiently manage their runtime state and data exchange operations [101]. We then discuss how data storage formats [102] become a bottleneck due to their HDD-era design assumptions, such as “*the CPU is fast and I/O is slow*” [23].

**[L-10] NVM Impact on Distributed Transactions:** We link the write-once property of flash chips with the design of a transaction system and discuss networked flash-based (L-7) Corfu [73] and Tango [74] transaction systems. Such transactional systems were considered in the past as well, however, the limited random performance of HDD-based systems prohibit such a design with parallel clients.

**[L-11] NVM Impact on Future Hardware (CXL) and Software (io\_uring):** In this lecture (new in the 2023 edition), we consider how NVM storage connects to wider hardware trends (CXL) and necessitates development of a new software I/O API (io\_uring). CXL connects all storage and memory elements in a byte-addressable, coherent manner [82], thus giving rise to a new “**Storage-Memory Continuum**” with a highly granular performance spectrum instead of the classical triangle of cache-memory-storage hierarchy (Figure 3). Later, we discuss the emergence of io\_uring, a new asynchronous high-performance I/O API in Linux and its design [69], [70]. These developments led to a re-division of labor among

hardware-software-OS in a storage system.

### C. Project Work

To facilitate “learning-by-doing”, in the assignment (A) work the students design and implement an FTL for an NVMe SSD with Zone Namespaces (ZNS) in QEMU and build a workload-specific file system with RocksDB.

- (A1) The students become familiar with the Linux development environment, framework (QEMU, NVMe, ZNS), and tools using state-of-the-practice Linux `nvme-cli` tools. An advantage of this setup is that these are commonly used systems research tools and frameworks that the students also later use in their thesis or research projects.
- (A2) The second assignment is developing a hybrid log-based FTL in the userspace (using `libnvme`) for a ZNS SSD that has a flash chips-like read, sequential-write, and reset interface [43]. The FTL converts the ZNS SSD into a conventional SSD (read/write anywhere, no reset), thus hiding its unique interface from the software.
- (A3) In the third assignment, the students develop a garbage collection algorithm where the FTL log needs cleaning as it gets full, and make decisions for the live pages stored in the log. Since the assignments are graded based on their code execution performance, many students often start with big, complex designs, which eventually get simplified as they have to debug and optimize the code (the KISS philosophy). At this stage, if a student fails to complete this assignment, they can continue the project using a file-backed A3 implementation, instead of ZNS-based.
- (A4) In this assignment, the students familiarize themselves with RocksDB, and design and implement a RocksDB-specific file system on top of their FTL-ZNS device (A3). We offer a free design space where the students can develop their own file system design. In the past, students have implemented ext2, LFS, and FAT32-style file systems.
- (A5) The last assignment is to make the whole project durable and consistent with an orderly shutdown and restart. Here, a student needs to reserve ZNS capacity to store FTL and file system metadata, and detect and recover the previously stored state from the ZNS SSD. The students can take additional bonuses to demonstrate the completeness of their code by running it on real NVMe ZNS SSDs donated by Western Digital.

The project is graded on functional completeness and execution speed using an automated grading framework that links to a student’s GitHub assignment source code repo (§VI).

“Storage system is a topic that’s not often discussed during computer science. This course is excellent in presenting the latest advancements in the storage technologies” – from 2020.

“The course introduced me to the state of the art of storage systems. Something I would’ve never thought was an interesting field.” – from 2021.

“Very interesting material. Great that it’s up to date. Great lectures. I especially liked that in the lectures you mentioned papers from the history/development of the field.” – from 2022.

“The content was very interesting and gave a good introduction/picture to this domain, even if I sometimes was overwhelmed by the content at 9:00 AM :)” – from 2023.

Fig. 4: Statements from student reviews in response to the question “What do you think was good about this course?”.

TABLE IV: Statistics on and student reviews, on a scale of 1 (strongly disagree) to 5 (strongly agree), of the Storage Systems course given at VU Amsterdam.

Statistic or question	Academic Year			
	2020	2021	2022	2023
Students started	28	29	42	50
Students passed	18	21	30	34
Passing rate (%)	64	72	71	68
Student reviews	6	9	10	14
Overall rating of the course	4.5	4.6	4.5	3.5
I learned a lot from this course	4.7	4.9	4.9	4.3
The material was clear and informative	-	4.4	4.6	4.1
The material was engaging	-	4.4	4.5	3.9
The course was well organized	4.5	4.2	4.2	4.1
The feedback on my work helped me	-	3.4	3.9	3.3

## VI. EXPERIENCE

In this section, we discuss our experience teaching the Storage Systems course four times between 2020–2023. Each year, we ask the students questions on how they perceived the course, which they could score between 1 (strongly disagree) and 5 (strongly agree), followed by open-ended questions on how the course could be improved. We present an overview of the student reviews and course statistics in Table IV. The student feedback is generally positive, with an overall rating of about 4.5 for the course between 2020–2022. The latest edition (2023) observes a lower rating (3.5), with the students pursuing highly specialized graduate programs in security remarking on the course’s lack of security-related content. Furthermore, the pool of students who typically respond is usually small, with at most 43% of passed students reviewing the course, resulting in possibly skewed reviews. Nonetheless, we are working to improve the course based on the current student feedback.

We highlight three recurring themes in student reviews here. First, the students appreciate that the project is deeply embedded in the state-of-the-practice software ecosystem, and they become familiar with multiple technologies at the end of the course: QEMU, kernel development, NVMe SSDs, RocksDB, `nvme-cli` and `libnvme`. Since the course covers many recently published papers, many students attempt advanced bonuses in the class or later pick up on research projects that build on top of their class project assignments. We have successfully published peer-reviewed conference papers and talks with students from this class [53], [103], [104].

Second, the students appreciate the project’s vertical integration approach (instead of independent assignments), where

they experience and implement a complete storage stack to support a workload. With such a vertically integrated project, the students often realize complex multi-layer dependencies of NVMe-related challenges, thus enabling us to achieve our key goal of an end-to-end understanding of the storage stack for vertical integration and optimizations. For example, students implementing a log-structured file system realized they are dealing with a log-on-log problem when their FTL also implements a log [105]. Others notice the restricted nature of the block interface that creates the semantic gap between an SSD and the host software.

Third, the students appreciate the automatic benchmarking and framework introduced in 2021. We introduced this framework for two reasons: First, it serves as a uniform platform for comparing the performance of student projects, measured as execution time over multiple iterations. We give students access to this platform to test and benchmark their code against other students in a public competition. Second, the framework helps to automate grading, keeping the grading overhead for the teaching staff manageable under increasing student numbers.

**Challenges:** Looking forward, we see three challenges with the course. First, *how do we scale to support 100+ students while keeping the course interesting?* System and storage programming is complex with a large design space, hence there is a risk that a few safe FTL and file system designs will always be popular with students. As a result, as the number of students taking this course increases ( $\sim 2\times$ , Table IV), the number of interesting project designs will not increase. Second, *how do we identify plagiarism with the rise of AI-assisted coding frameworks (GitHub co-pilot) and systems (ChatGPT)?* Though not specific to this course, these tools make it increasingly challenging to detect plagiarism. As the course matures, the number of complete project solutions in the open also increases. Lastly, *how do we cover the complete from-silicon-chips-to-cloud-services stack of storage systems instead of only the SSD FTL and file system layers we currently cover?* To cover the complete stack, we need to include many lower-level (flash cell behavior, wear-leveling) and high-level (data distribution, scalability, networking, failure management) considerations. While this approach gives students a sense of ownership of the whole storage stack, it is not feasible to cover such a gamut of topics in a single project.

We believe that to keep the project interesting and diverse, approaches such as Gamification [106] could play an important



role. With Gamification, we can offer multiple ways for students to finish and build each assignment. This approach results in unique projects, which reduce the chance of cheating and allow us to cover more topics between silicon chips and cloud services than a single student would have to implement for the project. Furthermore, we can steer the design from a pure performance-based design to other design goals, such as energy efficiency or reducing the DRAM needed to run data-heavy workloads. To combat cheating, we also have an interview-based assessment of the project where students have to explain their design choices and code implementation.

On gender diversity in our class over the last four years, the number of female students who have participated in the course remains under half a dozen per year. We aim to include this matter with other initiatives in the department to improve gender representation in systems research specifically and broadly in computer science [107].

## VII. CONCLUSION

The emergence of NVM storage in commodity computing has fundamentally impacted every aspect of modern systems building. Nevertheless, this impact is not reflected in educational curricula, textbooks, and policy documents. In this paper, we make a case that there is a need for a concerted effort to establish storage systems as one of the foundational pillars of computing in education. In this direction, we present the design of a modern storage systems course to fill this gap in the curriculum. We discuss our motivations, experience from offering the course over the past 4 years, its current status, and our future plans. In order to facilitate wider dissemination, we make the course material (slides, recordings, assignments, and grading framework) freely and openly accessible.

**Acknowledgements:** This work is supported by generous hardware donations from Western Digital arranged by Matias Bjørling, and the Dutch Research Council (NWO) grant OCENW.KLEIN.561. Many members of the AtLarge research team at the VU contributed to this work, specifically Giulia Frascaria who helped to set up the first iteration of the course. The authors thank Alexandru Iosup for his encouraging support in the preparation of this manuscript.

## REFERENCES

- [1] Huawei, "Computing 2030," [https://www-file.huawei.com/-/media/corp2020/pdf/giv/industry-reports/computing\\_2030\\_en.pdf](https://www-file.huawei.com/-/media/corp2020/pdf/giv/industry-reports/computing_2030_en.pdf), 2020, accessed: 2023-Dec-02.
- [2] Harrison and Pardo, "Data, Politics and Public Health: COVID-19 Data-Driven Decision Making in Public Discourse," *Digit. Gov.: Res. Pract.*, vol. 2, no. 1, 2020.
- [3] Demigha, "The Impact of Big Data on AI," in *IEEE CSCI*, 2020.
- [4] Akhtar et al., "The Impact of Big Data In Healthcare Analytics," in *IEEE ICOIN*, 2020.
- [5] Shrivastava and Umar, "The Impact of Big Data on Climate Change," in *IEEE ICIDCA*, 2023.
- [6] Computer History Museum, "1956: First Commercial Hard Disk Drive Shipped," <https://www.computerhistory.org/storageengine/first-commercial-hard-disk-drive-shipped/>, 2023, accessed: 2023-Dec-02.
- [7] Appuswamy et al., "The Five-Minute Rule 30 Years Later and Its Impact on the Storage Hierarchy," *Commun. ACM*, vol. 62, 2019.
- [8] Chatzieftheriou et al., "Could Cloud Storage be Disrupted in the Next Decade?" in *USENIX HotStorage*, 2020.

- [9] Ousterhout et al., "The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM," *ACM SIGOPS Oper. Syst. Rev.*, vol. 43, no. 4, 2010.
- [10] He et al., "DASH: a Recipe for a Flash-based Data Intensive Super-computer," in *ACM/IEEE SC*, 2010.
- [11] Caulfield et al., "Understanding the Impact of Emerging Non-Volatile Memories on High-Performance, IO-Intensive Computing," in *ACM/IEEE SC*, 2010.
- [12] Park and Shen, "A Performance Evaluation of Scientific I/O Workloads on Flash-based SSDs," in *IEEE CLUSTER*, 2009.
- [13] Chen et al., "Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives," in *ACM SIGMETRICS*, 2009.
- [14] Agrawal et al., "Design Tradeoffs for SSD Performance," in *USENIX ATC*, 2008.
- [15] Aswathy and Sivamangai, "Future Nonvolatile Memory Technologies: Challenges and Applications," in *IEEE ACCESS*, 2021.
- [16] Seltzer et al., "An NVM Carol: Visions of NVM Past, Present, and Future," in *IEEE ICDE*, 2018.
- [17] Liu et al., "A Survey of Non-Volatile Main Memory Technologies: State-of-the-Arts, Practices, and Future Directions," *CoRR*, vol. abs/2010.04406, 2020.
- [18] C. Mellor, "Gartner: Enterprise SSDs will hit 35% of HDD/SSD exabytes shipped by 2026," <https://blocksandfiles.com/2022/05/16/monday-gartner-hdd-ssd-numberfest/>, 2022, accessed: 2023-Dec-02.
- [19] "Global Non-Volatile Memory Express (NVMe) Market Analysis and Forecasts," <https://www.businesswire.com/news/home/20221017005645/en/Global-Non-Volatile-Memory-Express-NVMe-Market-Analysis-Forecasts-2016-2021-2021-2026F-2031F-Solid-state-Drives-SSD-s-Adapters-All-flash-Arrays-Servers-Ethernet-Fibre-Channel-InfiniB-and--ResearchAndMarkets.com>, 2022, accessed: 2023-Dec-02.
- [20] "IO500 Lists," <https://io500.org/list/sc23/full>, 2023, accessed: 2023-Dec-02.
- [21] M. Feldman, "Tracking the Worlds Top Storage Systems," <https://io500.org/list/sc23/full>, 2017, accessed: 2023-Dec-02.
- [22] "Cloud Flash Storage: SSD Options from AWS, Azure, and GCP," <https://www.computerweekly.com/feature/Cloud-flash-storage-SSD-options-from-AWS-Azure-and-GCP>, 2020, accessed: 2023-Dec-01.
- [23] Nanavati et al., "Non-Volatile Storage: Implications of the Datacenter's Shifting Center," *ACM Queue*, vol. 13, no. 9, 2015.
- [24] Amvrosiadis et al., "Data Storage Research Vision 2025: Report on NSF Visioning Workshop Held May 30-June 1, 2018," National Science Foundation, Tech. Rep., 2018.
- [25] Rasmussen et al., "TritonSort: A balanced Large-Scale sorting system," in *USENIX NSDI*, 2011.
- [26] He et al., "The Unwritten Contract of Solid State Drives," in *ACM EuroSys*, 2017.
- [27] Mesh Flinders, IBM, "SSD vs. NVMe: What's the Difference?" <https://www.ibm.com/blog/ssd-vs-nvme/>, 2023, accessed: 2023-Dec-02.
- [28] "Non-Volatile Memory Express Market Analysis," <https://markwideresearch.com/non-volatile-memory-express-nvme-market/>, 2023, accessed: 2023-Dec-02.
- [29] Hennessy and Patterson, "A New Golden Age for Computer Architecture," *Commun. ACM*, vol. 62, no. 2, 2019.
- [30] "A History of Flash Memory and Its Rise in the Enterprise," <https://www.techtarget.com/searchstorage/feature/A-history-of-flash-memory-and-its-rise-in-the-enterprise>, 2020, accessed: 2023-Dec-02.
- [31] D. Richter, *Flash Memories: Economic Principles of Performance, Cost and Reliability Optimization*. Springer Dordrecht, 2014.
- [32] Leventhal, "Flash Storage Today: Can Flash Memory Become the Foundation for a New Tier in the Storage Hierarchy?" *ACM Queue*, no. 4, 2008.
- [33] Lockwood et al., "Storage 2020: A Vision for the Future of HPC Storage," <https://www.osti.gov/biblio/1632124>, 2017.
- [34] Anderson et al., "Project Silica: Towards Sustainable Cloud Archival Storage in Glass," in *ACM SOSP*, 2023.
- [35] Ceze and Strauss, "DNA Data Storage and Near-Molecule Processing for the Yottabyte Era," (Keynote) *USENIX FAST*, 2021.
- [36] Boukhobza et al., "Emerging NVM: A Survey on Architectural Integration and Research Challenges," *ACM TODAES*, vol. 23, no. 2, 2017.
- [37] Yu and Chen, "Emerging Memory Technologies: Recent Trends and Prospects," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, 2016.

- [38] Li et al., “Fantastic SSD Internals and How to Learn and Use Them,” in *ACM SYSTOR*, 2022.
- [39] Michelsoni et al., “Inside solid state drives (ssds),” 2018.
- [40] Arpaci-Dusseau and others, *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 2018.
- [41] D. H. Walker, “A Comparison of NVMe and AHCI,” [https://sata-io.org/sites/default/files/documents/NVMe%20and%20AHCI\\_%20\\_long\\_.pdf](https://sata-io.org/sites/default/files/documents/NVMe%20and%20AHCI_%20_long_.pdf), Accessed: 2023-Dec-02.
- [42] “NVMe Specifications Overview,” <https://nvmexpress.org/specifications/>, Accessed: 2023-12-02.
- [43] Bjørling et al., “ZNS: Avoiding the Block Interface Tax for Flash-based SSDs,” in *USENIX ATC*, 2021.
- [44] Ouyang et al., “SDF: Software-Defined Flash for Web-Scale Internet Storage Systems,” *ACM ASPLOS*, 2014.
- [45] Bjørling et al., “Linux Block IO: Introducing Multi-Queue SSD Access on Multi-Core Systems,” in *ACM Syster*, 2013.
- [46] Shen and Park, “FlashFQ: A Fair Queueing I/O Scheduler for Flash-Based SSDs,” in *USENIX ATC*, 2013.
- [47] Woo et al., “D2FQ: Device-Direct Fair Queueing for NVMe SSDs,” in *USENIX FAST*, 2021.
- [48] Tavakkol et al., “FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives,” in *IEEE ISCA*, 2018.
- [49] Ren et al., “BFQ, Multiqueue-Deadline, or Kyber? Performance Characterization of Linux Storage Schedulers in the NVMe Era,” in *ACM/SPEC ICPE*, 2024.
- [50] Klimovic et al., “ReFlex: Remote Flash = Local Flash,” in *ACM ASPLOS*, 2017.
- [51] Trivedi et al., “Flashnet: Flash/network stack co-design,” in *ACM SYSTOR*, 2017.
- [52] Hwang et al., “TCP == RDMA: CPU-efficient Remote Storage Access with i10,” in *USENIX NSDI*, 2020.
- [53] Gootzen et al., “DPFS: DPU-Powered File System Virtualization,” in *ACM SYSTOR*, 2023.
- [54] Min et al., “Gimbal: Enabling multi-tenant storage disaggregation on smartnic jbofs,” in *ACM SIGCOMM*, 2021.
- [55] Do et al., “Programmable Solid-State Storage in Future Cloud Data-centers,” *Commun. ACM*, vol. 62, no. 6, 2019.
- [56] Peng et al., “MDev-NVMe: A NVMe Storage Virtualization Solution with Mediated Pass-Through,” in *USENIX ATC*, 2018.
- [57] Li et al., “LeapIO: Efficient and Portable Virtual NVMe Storage on ARM SoCs,” in *ACM ASPLOS*, 2020.
- [58] Xue et al., “Spool: Reliable Virtualized NVMe Storage Pool in Public Cloud Infrastructure,” in *USENIX ATC*, 2020.
- [59] Harris and Altıparmak, “When Poll is More Energy Efficient than Interrupt,” in *ACM HotStorage*, 2022.
- [60] —, “Ultra-Low Latency SSDs’ Impact on Overall Energy Efficiency,” in *USENIX HotStorage*, 2020.
- [61] Lee et al., “F2FS: A New File System for Flash Storage,” in *USENIX FAST*, 2015.
- [62] Tehrani et al., “A Survey on the Integration of NAND Flash Storage in the Design of File Systems and the Host Storage Software Stack,” *CoRR*, vol. abs/2307.11866, 2023.
- [63] Fevgas et al., “Indexing in Flash Storage Devices: A Survey on Challenges, Current Approaches, and Future Trends,” *The VLDB Journal*, vol. 29, 2019.
- [64] Doekemeijer and Trivedi, “Key-Value Stores on Flash Storage Devices: A Survey,” *CoRR*, vol. abs/2205.07975, 2022.
- [65] Bittman et al., “Twizzler: a Data-Centric OS for Non-Volatile memory,” in *USENIX ATC*, 2020.
- [66] Alverti et al., “DaxVM: Stressing the Limits of Memory as a File Interface,” in *IEEE/ACM MICRO*, 2022.
- [67] Bergman et al., “ZNSwap: un-Block your Swap,” in *USENIX ATC*, 2022.
- [68] Tsalapatis et al., “The Aurora Single Level Store Operating System,” in *ACM SOSP*, 2021.
- [69] Didona et al., “Understanding Modern Storage APIs: A Systematic Study of Libaio, SPDK, and io\_uring,” in *ACM SYSTOR*, 2022.
- [70] Ren and Trivedi, “Performance Characterization of Modern Storage Stacks: POSIX I/O, Libaio, SPDK, and Io\_uring,” in *CHEOPS*, 2023.
- [71] Subramanya et al., “BLAS-on-flash: An Efficient Alternative for Large Scale ML Training and Inference?” in *USENIX NSDI*, 2019.
- [72] Bae et al., “FlashNeuron: SSD-Enabled Large-Batch Training of Very Deep Neural Networks,” in *USENIX FAST*, 2021.
- [73] Balakrishnan et al., “CORFU: A Shared Log Design for Flash Clusters,” in *USENIX NSDI*, 2012.
- [74] —, “Tango: Distributed Data Structures over a Shared Log,” in *ACM SOSP*, 2013.
- [75] Lee et al., “A Case for Flash Memory SSD in Enterprise Database Applications,” in *ACM SIGMOD*, 2008.
- [76] Haas and Leis, “What Modern NVMe Storage Can Do, and How to Exploit It: High-Performance I/O for High-Performance Storage Engines,” *Proc. VLDB Endow.*, vol. 16, no. 9, 2023.
- [77] Zhang et al., “Rowhammering Storage Devices,” in *ACM HotStorage*, 2021.
- [78] Di et al., “Fast, Flexible, and Comprehensive Bug Detection for Persistent Memory Programs,” in *ACM ASPLOS*, 2021.
- [79] Renen et al., “Persistent Memory I/O Primitives,” in *DaMoN*, 2019.
- [80] Condit et al., “Better I/O through Byte-Addressable, Persistent Memory,” in *ACM SOSP*, 2009.
- [81] Li et al., “Pond: CXL-Based Memory Pooling Systems for Cloud Platforms,” in *ACM ASPLOS*, 2023.
- [82] Jung, “Hello Bytes, Bye Blocks: PCIe Storage Meets Compute Express Link for Memory Expansion (CXL-SSD),” in *ACM HotStorage*, 2022.
- [83] Yang et al., “Overcoming the Memory Wall with CXL-Enabled SSDs,” in *USENIX ATC*, 2023.
- [84] Breukelen and Trivedi, “Persistent Memory File Systems: A Survey,” *CoRR*, vol. abs/2310.02880, 2023.
- [85] Maneas et al., “A Study of SSD Reliability in Large Scale Enterprise Storage Deployments,” in *USENIX FAST*, 2020.
- [86] Lu et al., “NVMe SSD Failures in the Field: the Fail-Stop and the Fail-Slow,” in *USENIX ATC*, 2022.
- [87] CC2020 Task Force, “Computing Curricula 2020: Paradigms for Global Computing Education,” 2020.
- [88] Kumar and Raj, “A First Look at the ACM/IEEE-CS/AAAI Computer Science Curricula (CS202X),” in *SIGCSE*, 2022.
- [89] Yang et al., “When Poll Is Better than Interrupt,” in *USENIX FAST*, 2012.
- [90] Josephson et al., “DFS: A File System for Virtualized Flash Storage,” *ACM Trans. Storage*, vol. 6, no. 3, 2010.
- [91] Zhang et al., “De-Indirection for Flash-Based SSDs with Nameless Writes,” in *USENIX FAST*, 2012.
- [92] Athanassoulis et al., “Designing Access Methods: The RUM Conjecture,” in *EDBT*, 2016.
- [93] Yang et al., “An Empirical Guide to the Behavior and Use of Scalable Persistent Memory,” in *USENIX FAST*, 2020.
- [94] Coburn et al., “NV-Heaps: Making Persistent Objects Fast and Safe with Next-Generation, Non-Volatile Memories,” in *ACM ASPLOS*, 2011.
- [95] Hoseinzadeh and Swanson, “Corundum: Statically-Enforced Persistent Memory Safety,” in *ACM ASPLOS*, 2021.
- [96] Li et al., “ctFS: Replacing File Indexing with Hardware Memory Translation through Contiguous File Allocation for Persistent Memory,” in *USENIX FAST*, 2022.
- [97] Klimovic et al., “Flash Storage Disaggregation,” in *ACM EuroSys*, 2016.
- [98] Keeton et al., “A Case for Intelligent Disks (IDISks),” *SIGMOD Rec.*, vol. 27, no. 3, 1998.
- [99] Riedel et al., “Active Storage for Large-Scale Data Mining and Multimedia,” in *VLDB*, 1998.
- [100] Ruan et al., “INSIDER: Designing In-Storage Computing System for Emerging High-Performance Drive,” in *USENIX ATC*, 2019.
- [101] Stuedi et al., “Unification of Temporary Storage in the Nodekernel Architecture,” in *USENIX ATC*, 2019.
- [102] Trivedi et al., “Albis: High-Performance File Format for Big Data Systems,” in *ATC*, 2018.
- [103] Doekemeijer et al., “Performance characterization of nvme flash devices with zoned namespaces (zns),” in *IEEE CLUSTER*, 2023.
- [104] C. Lukken, “OpenCSD, simple and intuitive computational storage emulation with QEMU and eBPF,” <https://archive.fosdem.org/2023/schedule/event/csd/>, Accessed: 2023-Dec-02.
- [105] Yang et al., “Don’t Stack Your Log On My Log,” in *2nd INFLOW*, 2014.
- [106] Iosup and Epema, “An Experience Report on Using Gamification in Technical Higher Education,” in *ACM SIGCSE*, 2014.
- [107] Frachtenberg, “Underrepresentation of Women in Computer Systems Research,” *PLOS ONE*, vol. 17, no. 4, 2022.