UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Laércio Pioli Júnior

A Differentiated Proposal of Three Dimension I/O Performance Characterization Model Focusing on Storage Environments

Juiz de Fora

2020

Laércio Pioli Júnior

A Differentiated Proposal of Three Dimension I/O Performance Characterization Model Focusing on Storage Environments

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Mario Antônio Ribeiro Dantas

Coorientador: Victor Ströele de Andrade Menezes

Juiz de Fora

2020

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Pioli Júnior, Laércio.
A Differentiated Proposal of Three Dimension I/O Performance Characterization Model Focusing on Storage Environments / Laércio Pioli Júnior. – 2020.
175 f. : il.

Orientador: Mario Antônio Ribeiro Dantas
Coorientador: Victor Ströele de Andrade Menezes
Dissertação (Mestrado Acadêmico) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2020.

1. I/O Characterization. 2. I/O Performance. 3. Storage Environments. 4. High Performance Environments. 5. Storage System. I. Dantas, Mario Antônio Ribeiro, orient. II. Menezes, Victor Ströele de Andrade, coorient. III. Titulo.

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Programa de Pós-Graduação em
Ciência da Computação
www.pgcc.ufjf.br

Laércio Pioli Junior

“A Differentiated Proposal of Three Dimension I/O Performance Characterization Model Focusing on Storage Environments”

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 25 de agosto de 2020.

Dedicado à minha mãe e ao meu pai.

AGRADECIMENTOS

I would like to thank in my native language.

Primeiramente, gostaria de agradecer à minha família que sempre incentivou e apoiou meus estudos durante todo meu processo de aprendizado. Gostaria de agradecer especificamente minha mãe (Maria Eliane Ribeiro Campos Pioli) e meu pai (Laércio Pioli) por tudo que me tornei, pela base familiar e ensinamentos que pude obter em toda minha vida. Este trabalho é fruto de toda educação e apoio que me deram, obrigado por me ajudar a ser quem sou e por todo amor proporcionado.

Gostaria de agradecer também à todos of professores do DCC e do PGCC que, de alguma forma, contribuíram na minha formação tanto da graduação quanto do mestrado. Em especial, gostaria de agradecer ao meu amigo e orientador, Mario Antonio Ribeiro Dantas, não somente pela orientação nessa caminhada de pesquisa, mas também por todo ensimanento extra-acadêmico que contribuiu para que minha expansão interna como pesquisador pudesse acontecer. Não obstante, gostaria também de agradecer por todas as oportunidades de pesquisa que me foram ofertadas e principalmente, por toda a motivação e confiaça que me foi creditado ao longo das varias reuniões realizadas. Finalmente, gostaria de agradecer pela amizade que tornou o processo de aprendizado mais produtivo. Gostaria de agradecer também meu co-orientador Victor Ströele por todo ensinamento e amizade desde a graduação e também por toda a confiança de que seria capaz de efetuar nossas pesquisas na melhor qualidade possível. Agradeço também ao professor Jean-Francois Mehaut da universidade de Grenoble, França, por permitir que nossos experimentos pudessem ser executados em um ambiente computacional de alto nível e qualidade, o Grid'5000. Finalizando, gostaria de agradecer também o antigo aluno de doutorado do prof. Mario Dantas e atual Dr. pela UFSC, Eduardo Camilo Inacio, por sempre colaborar com nossas pesquisas e processo de experimentação.

Agradeço também a UFJF e todos seus funcionarios pelo suporte nesse processo.

Finalmente, agradeço ao Governo Federal Brasileiro pelo suporte e infraestrutura fornecida.

“Ein Buch muß die Axt sein für das gefrorene Meer in uns.”
(Franz Kafka)

RESUMO

O gargalo de E/S continua sendo um problema central em ambientes de alto desempenho. Os ambientes de computação em nuvem, computação de alto desempenho (HPC) e big data compartilham muitas dificuldades para fornecer dados em uma taxa de tempo desejável solicitada por aplicações de alto desempenho. Isso aumenta a possibilidade de criar gargalos em todo o processo de alimentação de aplicativos pelos dispositivos de hardware inferiores localizados na camada do sistema de armazenamento. Nos últimos anos, muitos pesquisadores propuseram soluções para melhorar a arquitetura de E/S considerando diferentes abordagens. Alguns deles aproveitam os dispositivos de hardware, enquanto outros se concentram em uma abordagem sofisticada de software. No entanto, devido à complexidade de lidar com ambientes de alto desempenho, criar soluções para melhorar o desempenho de E/S em software e hardware é um desafio e oferece aos pesquisadores muitas oportunidades. A classificação dessas melhorias em diferentes dimensões permite que os pesquisadores entendam como essas melhorias foram construídas ao longo dos anos e como elas progridem. Além disso, também permite que futuros esforços sejam direcionados para tópicos de pesquisa que se desenvolveram em menor proporção, equilibrando o processo geral de desenvolvimento. Esta pesquisa apresenta um modelo de caracterização tridimensional para classificar trabalhos de pesquisa sobre melhorias de desempenho de E/S para instalações de computação de armazenamento em larga escala. Esse modelo de classificação também pode ser usado como uma estrutura de diretrizes para resumir as pesquisas, fornecendo uma visão geral do cenário real. Também usamos o modelo proposto para realizar um mapeamento sistemático da literatura que abrangeu dez anos de pesquisa sobre melhorias no desempenho de E/S em ambientes de armazenamento. Este estudo classificou centenas de pesquisas distintas, identificando quais eram os dispositivos de hardware, software e sistemas de armazenamento que receberam mais atenção ao longo dos anos, quais foram os elementos de proposta mais pesquisados e onde esses elementos foram avaliados. Para justificar a importância desse modelo e o desenvolvimento de soluções que visam melhorias no desempenho de E/S, avaliamos um subconjunto dessas melhorias usando um ambiente de experimentação real e completo, o Grid5000. Análises em cenários diferentes usando um benchmark de E/S sintética demonstra como os parâmetros de vazão e latência se comportam ao executar diferentes operações de E/S usando tecnologias e abordagens distintas de armazenamento.

Palavras-chave: Caracterização de E/S, Desempenho de E/S, Ambiente de Armazenamento, Ambientes de Alto Desempenho, Sistema de Armazenamento.

ABSTRACT

The I/O bottleneck remains a central issue in high-performance environments. Cloud computing, high-performance computing (HPC) and big data environments share many underneath difficulties to deliver data at a desirable time rate requested by high-performance applications. This increases the possibility of creating bottlenecks throughout the application feeding process by bottom hardware devices located in the storage system layer. In the last years, many researchers have been proposed solutions to improve the I/O architecture considering different approaches. Some of them take advantage of hardware devices while others focus on a sophisticated software approach. However, due to the complexity of dealing with high-performance environments, creating solutions to improve I/O performance in both software and hardware is challenging and gives researchers many opportunities. Classifying these improvements in different dimensions allows researchers to understand how these improvements have been built over the years and how it progresses. In addition, it also allows future efforts to be directed to research topics that have developed at a lower rate, balancing the general development process. This research present a three-dimension characterization model for classifying research works on I/O performance improvements for large scale storage computing facilities. This classification model can also be used as a guideline framework to summarize researches providing an overview of the actual scenario. We also used the proposed model to perform a systematic literature mapping that covered ten years of research on I/O performance improvements in storage environments. This study classified hundreds of distinct researches identifying which were the hardware, software, and storage systems that received more attention over the years, which were the most researches proposals elements and where these elements were evaluated. In order to justify the importance of this model and the development of solutions that targets I/O performance improvements, we evaluated a subset of these improvements using a a real and complete experimentation environment, the Grid5000. Analysis over different scenarios using a synthetic I/O benchmark demonstrates how the throughput and latency parameters behaves when performing different I/O operations using distinct storage technologies and approaches.

Key-words: I/O Characterization, I/O Performance, Storage Environments, High Performance Environments, Storage System.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ADC	Areal Density Capability
ANL	Argonne National Laboratory
ASTC	Advanced Storage Technology Consortium
BDC	Big Data Computing
BGAS	Blue Gene Active Storage
BPMR	Bit-Patterned Magnetic Recording
BD	Blu-ray Disc
CARS	Contention-Aware Resource Scheduling
CCFC	Cost-aware Client-side File Caching
CD	Compact Disc
CFQ	Completely Fair Queuing
CMR	Conventional Magnetic Recording
CN	Compute Node
CPU	Central Processing Unit
DDC	Decoupled DMA Cache
DISC	Data Intensive Scalable Computing
DOE	Department of Energy
DVD	Digital Versatile Disc
DM	Drive Managed Drives
DRAM	Dynamic Random-Access Memory
DSA	Direct Storage Access
EB	Exabytes
EBSE	Evidence-based Software Engineering
ECC	Error Correction Code
EEPROM	Electrically Erasable Programmable Read Only Memory

FIO	Flexible I/O
FLOPS	FLoating-point Operations Per Second
FTL	Flash Translation Layer
GPU	Graphics Processing Unit
HA	Host Aware Drives
HDFS	Hadoop Distributed File System
HAMR	Heat Assisted Magnetic Recording
HDD	Hard Disk Drive
HDMR	Heated-Dot Magnetic Recording
HIL	Host Interface Logic
HM	Host Managed Drives
HPC	High-Performance Computing
HPDA	High Performance Data Analytics
HPE	High-Performance Environments
IC	Integrated Circuit
IMR	Interlaced Magnetic Recording
IO	Input/Output
ION	IO Node
IOPS	Input/Output per Second
IOR	Interleaved Or Random
IORE	Interleaved Or Random Extended
LBA	Logical Block Addressing
LLNL	Lawrence Livermore National Laboratory
LMR	Longitudinal Magnetic Recording
LTO	Linear Tape Open
MAMR	Microwave-Assisted Magnetic Recording

MLC	Multi-level cell
MOS	Metal Oxide Semiconductor
MSMR	Multi-Sensor Magnetic Recording
MSR	Microsoft Research Cambridge
NERSC	National Energy Research Scientific Computing Center
NICS	National Institute for Computational Sciences
NUDT	National University of Defense Technology
NVM	Non-Volatile Memory
NVRAM	Non-Volatile Random-Access Memory
ODS	Optical Data Storage
OLCF	Oak Ridge Leadership Computing Facility
OLTP	Online Transaction Processing
ONFS	On-line and Near-line File System
ORNL	Oak Ridge National Laboratory
OS	Operational System
OST	Object Storage Target
PBA	Physical Block Address
PCM	Phase Change Memory
PE	Process Element
PFS	Parallel File System
PLC	Penta-level cell
PMR	Perpendicular Magnetic Recording
POSIX	Programmable Operating System
QLC	Quad-level cell
RAID	Redundant Array of Inexpensive Drives
RAM	Random Access Memory

RPM	revolutions per minute
SCM	Storage Class Memory
SLC	Single-level cell
SLM	Systematic Literature Mapping
SLR	Systematic Literature Review
SMR	Shingled Magnetic Recording
SN	Storage Nodes
SPC	Storage Performance Council
SRAM	Static Random-Access Memory
SS	Storage System
SSD	Solid-State Drive
TDMR	Two Dimensional Magnetic Recording
TPC	Transaction Processing Performance Council
VM	Virtual Machine
VMMS	Virtual Main Memory Storage
WSC	Warehouse-Scale Computers

CONTENTS

1 INTRODUCTION

Nowadays, scientific and high-performance applications demand high computational performance. These high-performance applications generally depend on some technologies employed in the lower layers of the system, as well as on software techniques designed specifically for complex solutions. The use of mechanical storage devices, such as hard disk drives (HDDs) as a storage source for high-performance applications, has limitations when targeting maximum performance. It takes a long time to perform the tasks normally employed in the process of reading and writing information to disk. Conventional HDDs need to position the mechanical arm over the data being operated, which causes sluggish execution of the desired operation (TANENBAUM, 1995). Disk head movement from simultaneous requests made by processes or applications makes I/O performance relatively low (ZHANG; DAVIS; JIANG, 2012). Applications that perform billions of write and read operations on their storage devices may perform poorly due to this inherent device issue. This problem gets worse when data is stored in non-sequential sectors of the stored device, increasing the amount of reading head movement and thereby reducing performance.

The complexity and quantity of data have increased at a high rate, contributing to the worsening of the problem. A large amount of generated, computed and stored data decreases the longevity and the I/O general performance of computer systems. The main technological challenges encountered are related to the need to capture, analyze, model, and visualize this scientific information (HEY; TANSLEY; TOLLE, 2009). High-performance I/O is also essential for big data analysis (NAKASHIMA et al., 2017). These new scenarios, consider structured databases, images, sound, and several other types of data. Examples that can illustrate these environments are shown in (GOMES et al., 2018), (AGOSTINHO et al., 2018). As a result, NoSQL and NewSQL software technologies (e.g. MongoDB, BigTable, Redis, Cassandra, HBase, Neo4j, CouchDB) were conceived to tackle problems related to this data diversity. However, sometimes, tools designed to this perspective are not capable to cover all issues from those environments.

Data-intensive applications access multiple parts of large files through parallel access over HPC back-end storage systems distributed across clusters. I/O operations or file management calls (e.g. open, close, read, write, stat) impose some restrictions over the performance of the entire application. Depending on the characteristics of the workload, some I/O operations can affect the application performance by different proportions. A subset of these I/O operations (e.g. open and close) are more presumed to impact applications when executing over small files. It happens because the spend time to open and close the file is higher than the time to read and write a small quantity of data. On the other hand, other I/O operations (e.g. read and write) are more likely to impact applications that manage large size data. Considering that data-intensive applications manage large dataset files and perform a big quantity of I/O operations, we concentrate our research on the operations that are characterized by these reading and writing data access.

1.1 PROBLEM DEFINITION

The low I/O performance continues to affect high-performance environments and this issue is addressed in this investigation. Despite the great effort of the academic community to overcome this problem, sometimes concentrating efforts throughout a unique given solution, we identify that there is not only a single way to improve I/O performance in Storage System (SS).

The solutions seem to depend on each other for the performance of the entire storage system to be satisfactory Researchers propose solutions to improve I/O architecture through different perspectives. Some of them take advantage of hardware while others focus on software approaches. One of the most used approaches in storage systems to support the I/O layer is utilizing a range variety of memories and devices (e.g. flashboards, Solid-State Drives (SSDs), 3D XPoint, NVDIMM, Phase Change Memory (PCM), etc) as quick and auxiliary storage for HDDs.

Solutions considering heterogeneous storage systems were proposed over the years (ZHANG et al., 2019), (ZHOU et al., 2016b), (ZHOU et al., 2016a), (XIE et al., 2015) and (JU et al., 2016). Different storage devices are also used to achieve better I/O. PCM (CHOI; BAHN, 2018), (LIU; WANG; YU, 2018) and (LIU et al., 2016), SSD-M.2 (NAKASHIMA et al., 2017) are some instances. Many authors are proposing Flash Translation Layer (FTL) solutions targeting improvements on software layer (MATIVENGA et al., 2019), (PAN et al., 2019), (XIE; CHEN; ROTH, 2017) and (WANG et al., 2016).

Software solutions to improve I/O performance on storage systems are also employed. Some solutions are focused on improving I/O considering better database performance. (KIM; YEOM; SON, 2019), (WU; HUANG; CHANG, 2019), (SOULÉ; GEDIK, 2016) and (KARIM et al., 2015) while other focuses on File systems (ZHANG et al., 2019), (FAN; WANG; YE, 2018), (XIAO et al., 2018), (KIM; YEOM, 2017), (JIN; ZHU; HUANG, 2017), (LIU et al., 2017c), (HE; WANG; SUN, 2015) and (OU et al., 2015). Proposing other and efficient I/O schedulers (YAN et al., 2019), (YANG et al., 2019), (MAO; WU; DUAN, 2017), (PARK et al., 2018) and (JO; RO, 2016), or improving existential Linux I/O schedulers (e.g. Completely Fair Queuing (CFQ) (AXBOE, 2010), Noop (AXBOE,) and deadline) is an approach to increase the overall I/O performance (GUO; HU; MAO, 2015), (YEH; YANG; SUN, 2015) and (JI et al., 2016).

As presented above, there is no one way to increase I/O performance in storage environments. The improvements can be applied within different layers and each improvement belongs to a class that relates different proprieties. The challenge in this context is to create a model capable of characterizing the researcher's proposals to better understanding and development of this research field.

1.2 OBJECTIVES

Addressing the issue that affects many High-Performance Environments (HPE), the main objective of this research work is to devise a model that presents perspectives of how the I/O performance improvements are addressed to overcome low I/O performance in SS. It can be used to lead to a better understanding of the impact of different storage approaches that consider distinct storage devices. Classifying researchers' improvements in dimensions that divide software components from hardware components, allows us to understand how these improvements have been built over the years and how it progresses. The usage of this characterization model allows us to direct future efforts towards subjects that really need attention. Therefore, reducing bottlenecks that hinder the entire development structure. The overall overview of the actual efforts could be divided into a macro view of three elements. The two first elements are "software" and "hardware" and the combination of these previous elements creates a new element, a "storage system".

As an extension of the primary objective, two additional contributions are presented. The first one was a systematic literature mapping that classifies hundreds of distinct researches publications that were proposed to improve I/O performance on devices, systems, and environments using the presented model. There, we covered ten years of research on I/O improvements, presenting which were the devices, software, and storage systems that received more attention over the years, which were the most researcher's proposals elements, and where these elements were evaluated. The second task was to select a subset of these proposed I/O improvements concerning the I/O layer and evaluate them using a real testbed, the Grid5000. Analysis over different scenarios using a synthetic I/O benchmark demonstrates how the throughput and the latency parameters behave when performing different I/O operations using distinct storage technologies and approaches.

1.3 CONTRIBUTIONS

The contributions of this work are the following:

- A characterization model as a feature for classifying research works on I/O performance of storage environments.
- A systematic mapping survey that classifies hundreds of distinct researches publications that were proposed to improve I/O performance on devices, systems, and environments.

- The assessment of employ different storage devices to analyze throughput and latency ratio.
- An extensive number of experiments analysis evaluated in the Grid'5000 over different scenarios aiming to show that I/O performance might vary according to the employed parameters

1.4 METHODOLOGY

The used methodology in this research was defined considering the following phases: (*i*) a secondary literature study considering approaches in how storage systems were dealing and receiving I/O performance improvements. (*ii*) a characterization model proposal that considers I/O performance improvements considering the storage systems environments. (*iii*) validation of the proposed model through a classification study that incorporates hundreds of documents. (*iv*) construction of an experimental study that considers different approaches to data storage in order to evaluate latency and throughput.

The secondary study of the literature was conducted through a search for mappings and systematic reviews of the literature on improvements in I/O performance. The purpose of this study was to identify how the proposals were being carried out and in what context they were.

The proposal for a characterization model was developed to allow improvements in I/O performance in storage systems to be better understood. With a better understanding of these improvements, we argue that it will be possible to identify how these improvements have been built over the years. Identifying your progression allows you to target the next challenges to issues that require more attention. Therefore, bottlenecks that hinder the entire development structure can be reduced by bringing storage development closer to processing development.

The proposed model validation incorporates the elaboration of a systematic mapping of the literature that evaluates hundreds of documents that aim to improve I/O performance. We understand that by classifying these hundreds of documents we are covering a large part of the improvements and showing that the model becomes useful and promising to the scientific community.

The experimental study carried out incorporates a process that relates some elements presented in the model with some researcher's improvements to improve the I/O performance. Through this experimental process, we found that the usage of storage media in different storage approaches interferes with the performance of the overall storage system. Software layer approaches also influence performance and was considered in the latency and throughput evaluation process. The experimentation was performed in a real complex Grid system, the Grid'5000 located in France.

1.5 OUTLINE

The main concepts behind this work are the subject of Chapter 2. We discuss some related works in Chapter 4. The proposal is presented in Chapter 5. A tertiary literature study that presents results when using this model is presented in Chapter 3. Moreover, Chapter 6 shows the experimental results as well as discussions over different perspectives. Finally, Chapter 7 concludes this research, summarizing this work and outlining future works.

2 FUNDAMENTALS

The I/O bottleneck remains a central issue in HPE. The existing gap between power processing and storage latency increases this issue. The processing evolution was predicted by Moore's Law in the 1960s and it remains true until nowadays in century XXI. Its law states that the number of transistors on an integrated circuit would double every eighteen months. On the other hand, computing systems are composed not only by processing but also by memory and storage hierarchy that support this processing. Figure 1 presents the development of both storage and processing systems. Elements arranged in purple

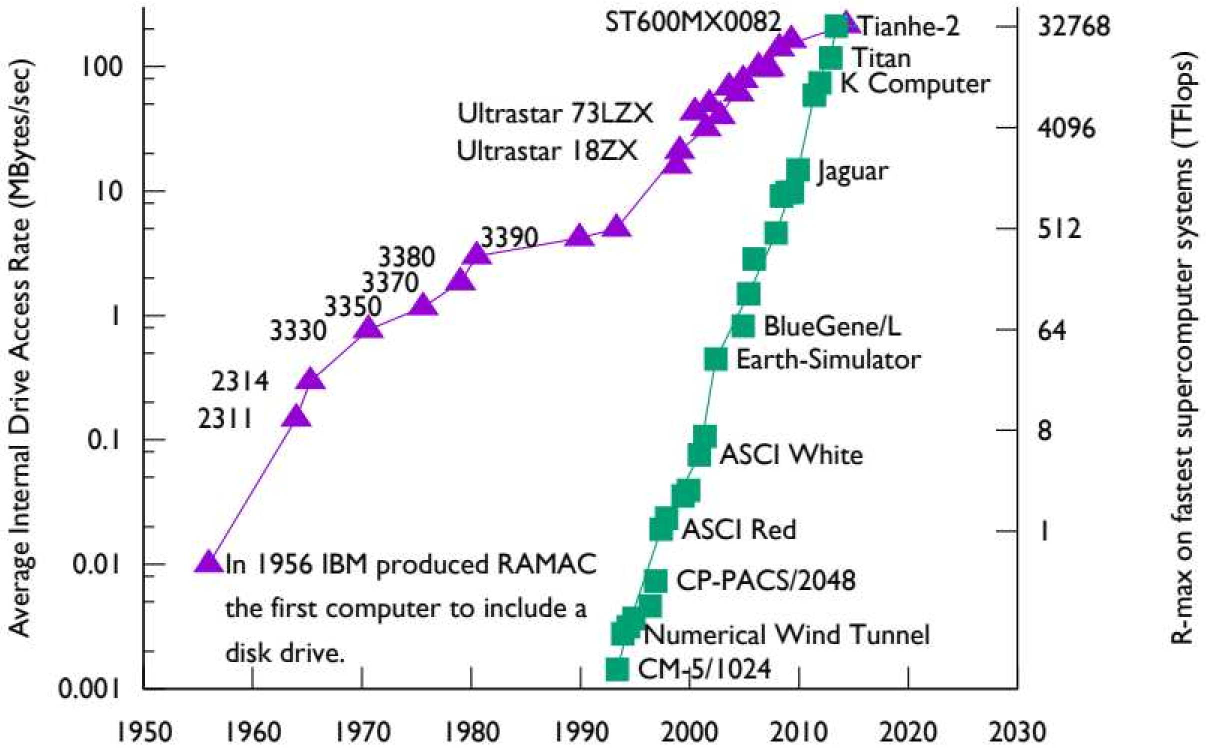


Figure 1 – CPU performance (green, 80%) vs Storage bandwidth (purple, 11%) improvement per year. (LATHAM; BAUTISTA-GOMEZ; BALAJI, 2017)

are related to the average internal drive access rate and the elements arranged in green relate to the processing development. We noticed that gap between these technologies widens over time. The slope of the line becomes much more upward for processing than for storage. Although the storage capacity increases considerably, its performance evolves at a low rate. The fact is that the storage technologies evolve at a slower speed than processing technologies and it generates the main I/O performance problem. This I/O bottleneck problem affects the application performance that has the necessity to move Exabytes (EB) of computed or generated data between Compute Node (CN) and Storage Nodes (SN). Typically, these demanding applications are executed in clustered architectures. This fact makes the I/O performance a hot area of study and researchers are proposing solutions to improve the I/O architecture by different approaches.

The storage hierarchy is classified as primary and secondary storage. The main difference between them is that the primary storage is located as much as possible closer to the processing with the goal of supporting the most accessed data, thus reducing latency in processing. This makes cache and DRAM memories ideal for supporting the processed data. The secondary storage is located furthest from the processing which increases latency compared to the previous one. Thus, it is normally used to store cold data where data access is less frequent. The place where the data is placed implies directly on the system-wide performance due to the latency data transfer. The closer the data is placed from the Processing Element (PE), the smaller will be the latency to transfer data between the system elements. Into the computation architecture elements, there are different levels of data storage commonly known as *primary* and *secondary* storage. Figure 2 presents this class and a big variety of technologies that belong to the storage ecosystem.

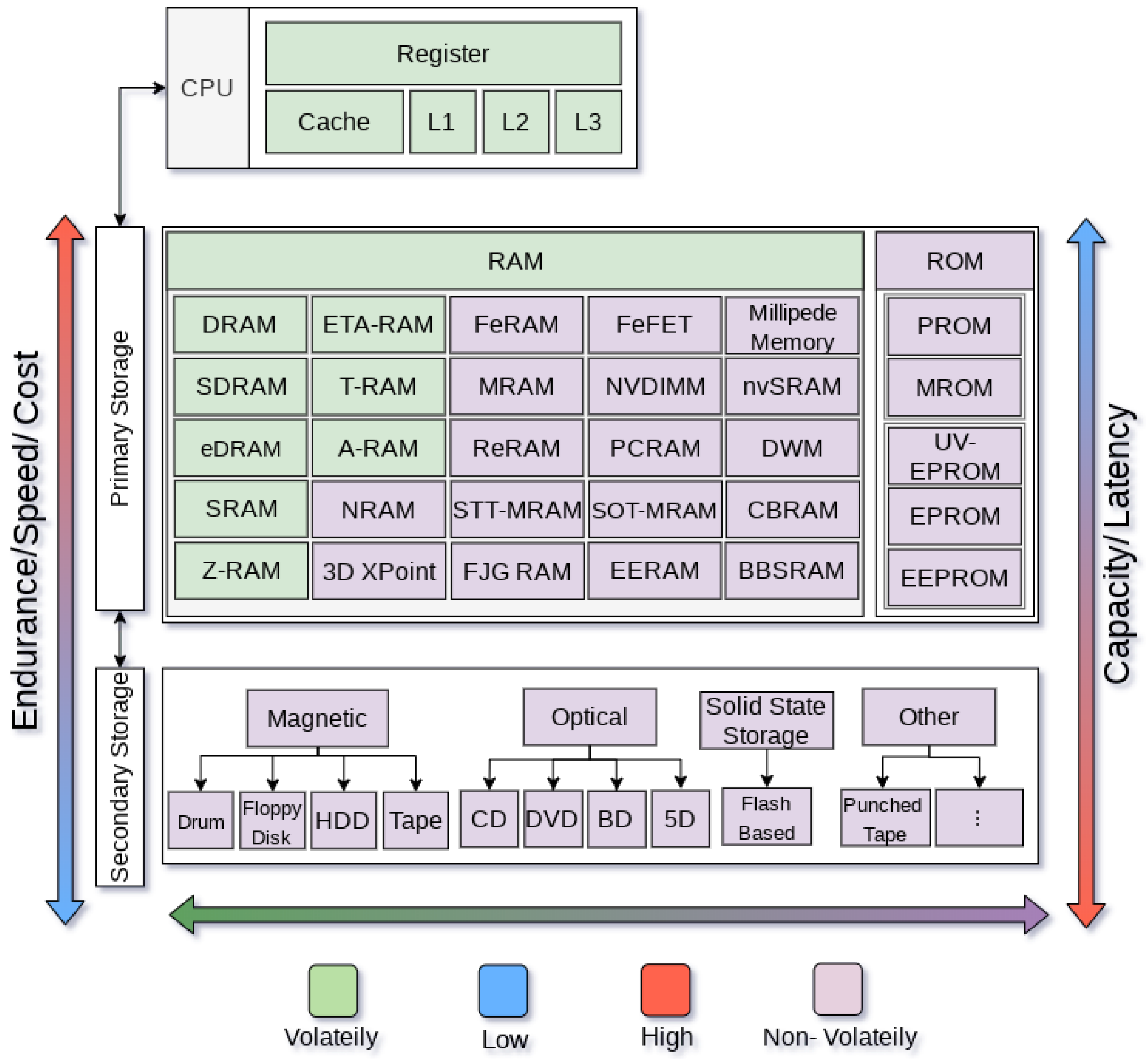


Figure 2 – Memory and Storage Technologies

Its huge variety of elements were projected to reduce the computational effort when computing and transferring data. Primary storage is usually optimized for Input/Output per Second (IOPS) and latency whereas secondary storage systems are optimized for throughput (SRINIVASAN et al., 2012).

Table 1 presents a big variety of storage technologies and some of their performance characteristics. There, we can analyze some I/O operations time (e.g. read and write) as much as energy performance (e.g. leakage Power and Dynamic Energy).

Table 1 – Memory/Storage Device Technologies Characteristics (BOUKHOBZA et al., 2017), (MEENA et al., 2014)

Technology	PCM	STT-RAM	DRAM	SRAM	NAND Flash	MRAM	FRAM	RRAM	HDD
Volatility (Yes/No)	No	No	Yes	Yes	No	No	No	No	No
Read (ns)	20-60	2-35	∼10	∼0.2-2	15000-35000	3-20	20-80	∼10	$3x10^6 - 5x10^6$
Write (ns)	20-150	3-50	∼10	∼0.2-2	200000-500000	3-20	50-75	∼50	$3x10^6 - 5x10^6$
Write Endurance	$10^8 - 10^9$	$10^{12} - 10^{15}$	$>10^{15}$	10^{16}	$10^4 - 10^5$	$>10^{12}$	$10^{14} - 10^{15}$	$10^8 - 10^{11}$	$>10^{15}$
Cell Area (F^2)	4-12	6-50	60-100	120-200	4-6	∼25	6-40	4-10	N/A
Leakage Power	Low	Low	Medium	High	Low	-	Low	Low	(Mechanical parts)
Dynamic Energy (R/W)	Medium/ High	Low/ High	Medium	Low	Low	-	Low/ High	Low/ High	(Mechanical parts)

2.1 PRIMARY STORAGE

The storage class where PEs can access directly the data is called *"Primary storage"*. The closer PE storage place to keep data are the register built into the processor, the cache memory, and the Random Access Memory (RAM) commonly called "memory".

RAM is one of the most used computer memory devices that computer systems use to share the momentary content processed by a PE. As the name suggests, in this kind of memory data can be read and assessed in any other differently from Tapes which permits only sequentially data access. It is an important computer hardware characteristic because permits applications to access and share data quickly. It is composed mainly of Metal Oxide Semiconductor (MOS) cells that is a metal oxide silicon transistor which, together with silicon metal, are the base elements to form an Integrated Circuit (IC). MOS also known as *MOS transistor*, *MOSFET*, *MOS FET* or *MOS-FET*, due to its material components, amplifies or switches electronic signals through an applied voltage controlling the flow of current. These electrical characteristics presented on it permitted less energy consumption than previous magnetic core memory which pleased everyone interested in that. The most used type of RAM is Static Random-Access Memory (SRAM) and Dynamic Random-Access Memory (DRAM).

As the name suggests, DRAM is a RAM memory where the access is performed randomly and dynamically. Materially, each cell that composes this kind of memory has a capacitor compared to the RAM. Therefore, each cell has a transistor and a capacitor, and these elements, through energy maintenance, are used to represent a bit into a cell. Consequently, due to intrinsic material characteristics of capacitors over time, it is slowly discharged by the Joule Law through the dissipation process. To prevent this effect, this variation of memory needs an auxiliary refreshment mechanism that ensures that the energy of each cell will not be dissipated and consequently the data that is represented by a set of bits, will not be lost. This refreshment mechanism is normally an IC element integrated into the RAM and it acts reading the cells and rewriting the exact content at the same place. For this reason, this kind of memory is a volatile device which means that this content is kept only if its energy is also kept. Due to his reason and other elements such as low latency and high endurance this memory is the main device used in computer systems as main memory.

SRAM is also a RAM memory with different particularities compared to the DRAM. Its *"S"* in its name extends from *"Static"* because it does not need a refreshing system as DRAM. It uses a bi-table latching circuitry, also known as (flip-flop), and is composed of six transistors destined to maintain data in addition to two transistors for control access. Its hardware architecture is mainly built in CMOS technology and the high quantity of transistors turns it more expensive than DRAM that has only one transistor and one capacitor. This kind of technology used to store data is usually coupled directly into IC and does not need to be coupled using a bus which increases latency. Therefore, it is faster than DRAM and its greatest application is for CPU cache and small embedded systems when time access is more important than size. However, SRAM keeps being a volatile memory and, despite presenting some data remanence level, it needs the power to maintain data. Nonetheless, SRAM has a complex internal structure, consumes more power, and is more expensive than DRAM.

2.2 SECONDARY STORAGE

Big data and data analytic brought the necessity to save data on a scale never seen before. The quantity of generated data is so big that it is estimated by IDC, that the amount of data in 2025 will surpass 175 zettabytes (REINSEL; GANTZ; RYDNING, 2018). The advances of big data and their scientific applications have increased this proportion making magnetic tapes and hard disks the most important cold data storage. Different from hot-storage, which contains data that always or almost always are available for assessment, cold-storage is characterized by filing data that is not often requested for processing. Nowadays, the most common cold storage system is composed of magnetic tapes and hard disks. Both of them are constituted of magnetic technology, saving the data after turning the energy off. The magnetic tape and disk store data using the same physical principle. They magnetize the surface area which means the presence or absence of information.

Improving area densities of magnetic technologies is not an easy task to be developed, however, it has been investigated by industrial companies and research academy over the years. The more is the density of a magnetic surface, the more bits are introduced into its region and consequently allow more data to be saved into its magnetic space. Figure 3 presents the trends in areal density effort, measured in $Gbits/in^2$, to increase magnetic storage densities. It considers the two most used magnetic devices, HDD, and tape storage media.

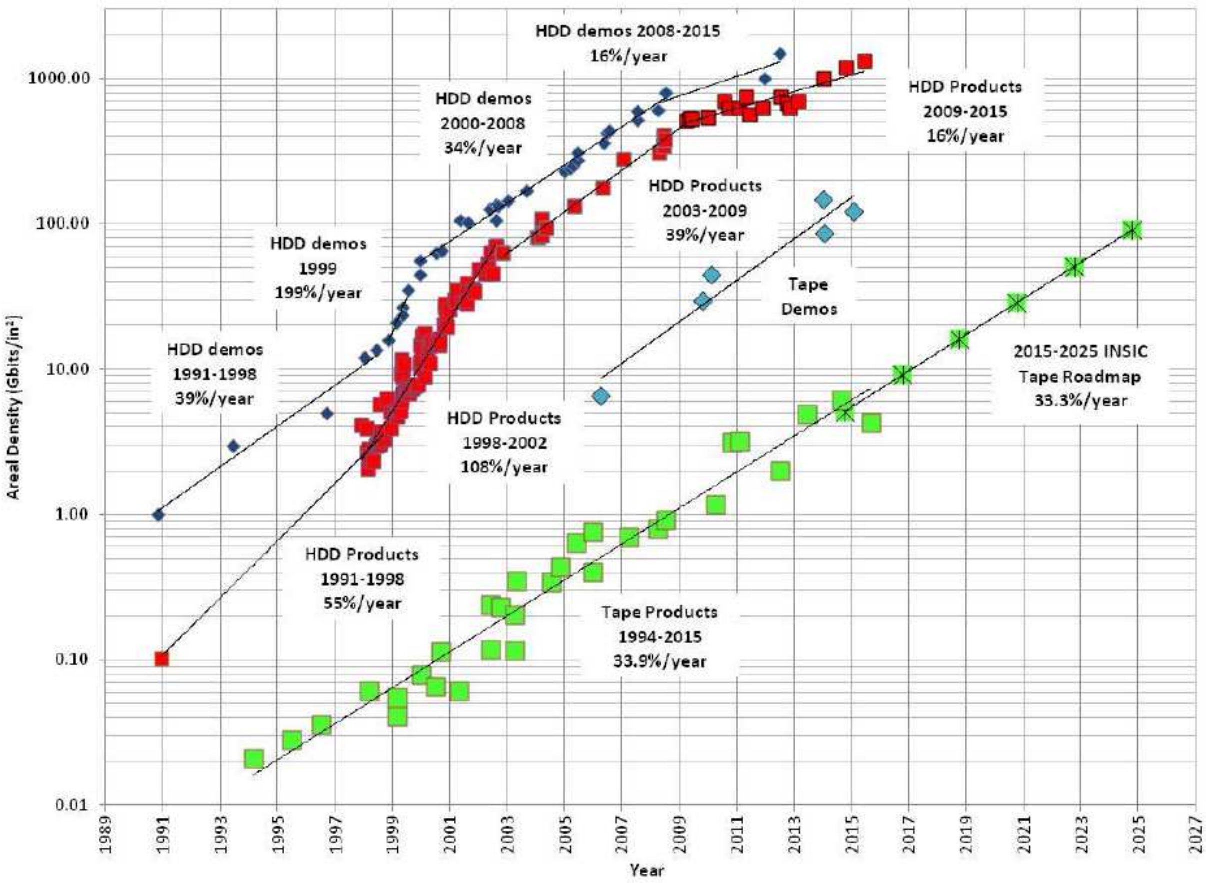


Figure 3 – Areal Density Trends. HDD and Tape, Demos and Products - Roadmap (TEAM et al., 2015).

Although today's magnetic tapes are capable of storing more data on a Linear Tape Open (LTO) cartridge than HDDs, it received more areal density recording improvements in a fixed period but it is decreasing its rate. The year 1999 seemed to be the cornerstone for magnetic HDDs. In the interval from 1991 to 1998, HDDs demos showed improvements of 39% per year on average, and HDD products reached 55% per year. In 1999, HDD demos reached a peak of almost 200% per year. It is the highest rate presented in Figure 3. In an interval of 4 years, from 1998 to 2002, HDDs products had their area density improvements on average of 108% per year. This factor can be understood with the consolidation of

HDDs media as a secondary storage source. After that period, from 2003 to 2009, HDD products received increments of 39% per year. Finally, in the last period from 2008 to 2015 and from 2009 to 2015, both HDD products and HDD demos reached 16% per year. Those small improvement numbers in areal density could indicate that HDDs are slowing down and becoming saturated.

On the other hand, magnetic tapes products seem to be constant and present a continuous improvement rate between the years 1994 to 2015, presenting an average of 33.9% in that period. Projections indicate that this rate will remain almost the same with a rate of 33.3 % between the next years of 2015 until 2025. The two most cold storage used technologies have a very different characteristic when talking about the performance of I/O operations. As we know, HDDs are composed of a number of platters that have cylinders, tracks, clusters, and sectors. These components are assessed by a read/write head when seeking data on the device. It makes those operations faster than the linear method used by magnetic tape to perform the same operations. The average data access time in tapes are around fifty to sixty seconds depending on the model that you are using (DROWNING. . . , 2018). The same operation performed within an HDD is about five to ten milliseconds. Considering the maximum range mentioned above, it means that the data time access to perform the I/O operation within a tape could be six thousand higher than the access time to perform the same operation on the HDDs.

2.2.1 Magnetic Tape

Indeed, IBM company was a great precursor in the usage of magnetic tape, or tape for short, as storage media. Magnetic tapes have been improved broadly over the years. The tape era has begun in 1951, specifically in March by the Bureau of the Census. It has used for the first time magnetic tapes to record data on a computer through *UNIVAC* system. It was the first one to handle both alphabetic and numerical data which changed the computation field (JR et al., 1951). It encouraged IBM, in 1952, to use tape as storage for the commercial computer. In 1968 IBM introduced the self-threading drive called IBM 2420. After that, was implemented and created the first robotic tape library, the IBM 3850. Moreover, in 1984 IBM also introduced thin-film head technology through IBM 3480. Furthermore, in 1989 and 1993, they introduced the *helical scan* digital data Storage and a digital linear tape respectively. In the year 2000 was introduced the LTO and in 2009 IBM introduces the Linear Tape File System. Finally, in 2017, IBM released the latest generation of LTO, the LTO-8.

Differently from social common sense, tapes remain, until nowadays, been used as the most storage media device by companies to store cold data and long term storage. It has also seemed like a high resistance and durability media device that allows us to recover data from natural disasters. Among the many advantages over HDDs and SSDs, companies choose them because they can offer more reliability if compared to other storage media. Power consumption is also an important feature provided by tapes due to the behavior of the robotic libraries after all the data has been recorded. When some tape cartridge is not in usage, they keep stored into robotic libraries which retain these cartridges without need to waste energy. Moreover, tapes are a good form of mass storage and have a lower error rate. The nature of the device itself provides a high-security power. The device must be mounted on a physical drive to be accessed. Among these previous good advantages, the low cost is the most important of them. It is the main reason why companies use tape storage. Companies have savings of six times economically to store the same amount of data compared to hard disk (DROWNING. . . , 2018).

Whereas a hard disk device can reach storage amounts in dozens of terabytes (TB) scale, a roll of tape can reach hundreds of TB. In 2017, IBM and Sony have presented a magnetic tape technology that could reach 330 (TB) in a unique tape cartridge. It measured 1098 in length(meters) and its density was 201 billion bits per sq inch. A huge stored amount of data contrasts with the assessment time on retrieving data. It is worth noticing that, the latency is very high and I/O operations take much time to be performed. As the tape technology is a sequential storage technology, which means that the data are sequentially accessed, the time to access data is bigger compared with semiconductor memories and magnetic disks. Due to the need to go through the whole tape until the part that contains the desired information, tapes present the worst I/O performance. It kept being implemented by enterprises mainly for cold storage or non frequently data access. Normally, this information is kept just to ensure historical data logging avoiding data loss.

The process to save the information on magnetic tape looks similar to the process of doing it on the magnetic disk. Physically speaking, both use magnetism principles to save data. In Tapes, the head is coated by a magnetic layer that is passed over the tape magnetizing it in write operation or detecting data when in the reading operation. Disks use a similar process, though it is performed with many read-write reads and it is explored in the next section. They do it by switching the two states of a polarity which could be true or false. True means that this tape area refers to number one and false means that area refers to the absence of the number one, in other words, number zero. This is the basic idea in using magnetism to store data, although, due to the variety and evolution of these technologies and devices, this process may vary in some way. Nonetheless, this process is implemented along the entire track and this is the way how is registered the presence of information into magnetic tapes and disks. Figure 4 presents a slice of magnetic tape with some elements.

In this schematic process, the smallest unit is a bit which is grouped into frames. Frames are grouped forming a record that is the smallest amount of data. It means that if some application or process needs to read or write data, it is infeasible for it read a

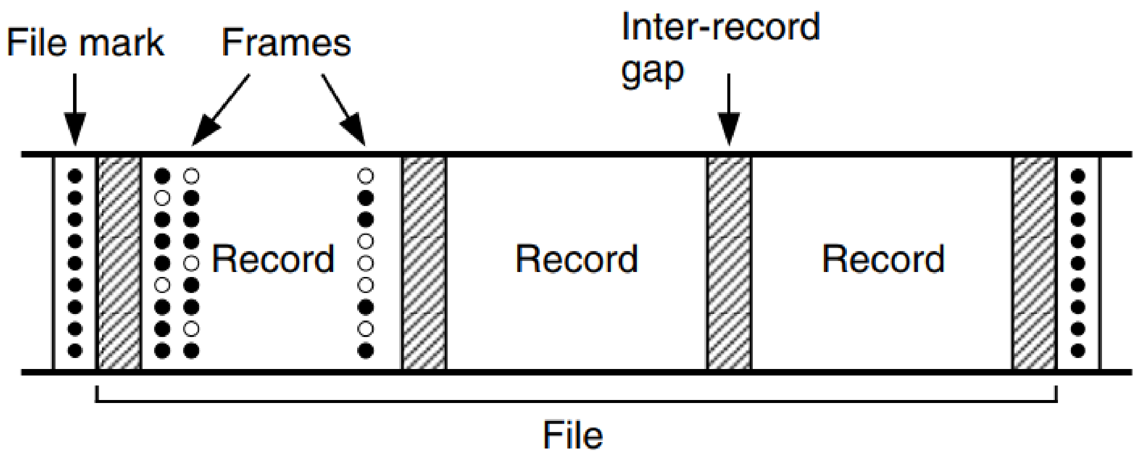


Figure 4 – Magnetic Tape Slice (MURDOCCA; HEURING, 1999)

specific frame. Instead, it will read a record that contains the necessary data. In this way, records are written into tapes. Between records, there are inter-record gaps, which are empty spaces. These spaces occur because in the writing and reading process, there is a necessary time to put the reel motor into motion on the right velocity and as long as that speed is not reached, the tape continues to pass over the read and write head. It generates a lack of information into the tape which creates these gaps. Eventually, this time is not synchronized and a gap is embedded into a record. This problem is referred to as *jitter* and may occur in tapes. Sets of records generate a file into a tape and as they are accessed sequentially, random access becomes infeasible. A file mark can also be introduced at the beginning of each record working as a flag that indicates where a record begins.

2.2.2 Magnetic Disk

Unlike tapes, where data can be easily removed as an independent dataset (tape cartridge) and moved independently of the mechanical read-write head, in magnetic disks, it is coupled into the platter, inside the HDD, where data is written. The quantity of read-write heads is also a different characteristic compared to tapes. A unique HDD can have a set of N x 2 heads, where N is the number of magnetic platters located within a HDD. It happens because each surface may receive one independent head. Therefore, some HDDs can be multi-platter with more than one platter. Figure 5 presents a conventional HDD organization with some important elements presented into it.

An HDD has a spindle that connects all platters when multi-platter, in their zero-radius point and allows them to be stacked on top of each other. This element rotates in a specified direction, synchronously, and allows all disks to rotate at the desired speed. Its rotation speed can vary from 3600 to 10000 revolutions per minute (RPM). Between them is located each read-write head which is connected to an arm that moves inreaching and outreaching a then different fraction of the surfaces. Each platter has two faces, the

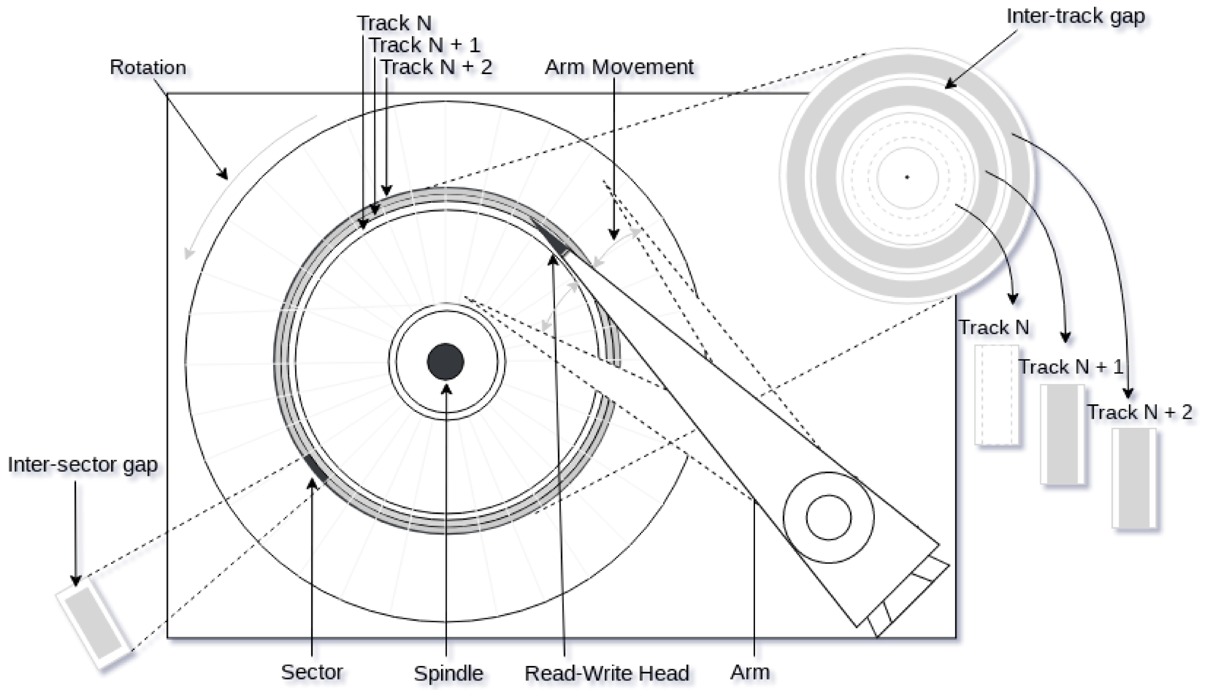


Figure 5 – HDD Components

top and the bottom which permits more data to be recorded into the platter. These platters are usually made from aluminum or glass and to allow them to become highly magnetic they are coated with small particles of iron oxide and other elements.

The smallest area is a sector, typically measured at 512 bytes. It is spaced by an inter-sector gap and is stored serially. As in tapes, each sector in platter disks is also separated by an inter-sector gap. A set of sectors compose a track that is independent of each other and has a distinct diameter, the farther from the spindle the greater the diameter will be. The tracks are also separated by inter-track gaps. These gaps make it necessary because there is a finite time to move the arm to the right sector and track the position. When a new disk begins to write data, the arm will move to the outermost edge of the platter, beginning in these outermost edge tracks, and, as more capacity is required, the arm moves toward the inner edge. A zone is composed of tracks that have the same number of sectors per track. If all tracks have different size each other, all zones will correspond to the tracks. A cylinder is a track projection along with all surfaces platters parallel to the spindle.

The complexity of these mechanical and non-mechanical elements into HDDs makes time to transfer data vary. Arms are mechanical elements and their movement across their ranges imposes an additional latency. When a data block is requested, the arm should find in which track the data is located. The time to make this arm movement is known as *seek time*. Indeed, this task is a costly process and the largest contributor to the latency of HDDs. After positioning the arm on the right track, it is necessary that the right sector, considering the disk rotation, passes through the arm head. In Figure 5 is possible to verify that, if the highlighted sector is required by some application, it is necessary to wait for almost a half-disk rotation for the sector pass through the read-write head. This process is known as *rotational latency*. Finally, after positioning the sector under the head, there is a necessary time to read the entire sector content. This process is known as *transfer time*. These elements make the read-write time increase considerably, especially in I/O intensive applications.

As our world becomes more data-driven, digital data become overloaded and complex. In HDD industry, new technologies solutions are being proposed to satisfy this high demand of more Areal Density Capability (ADC). However, a concerning fact about HDDs industry was how to increase ADC while decreasing storage device size. In Figure 6, is presented the Advanced Storage Technology Consortium (ASTC) technology roadmap with suggestions and predictions based on the evolution of the areal density recording of HDD technologies. PMR technology seems to be reached its limit in areal density which

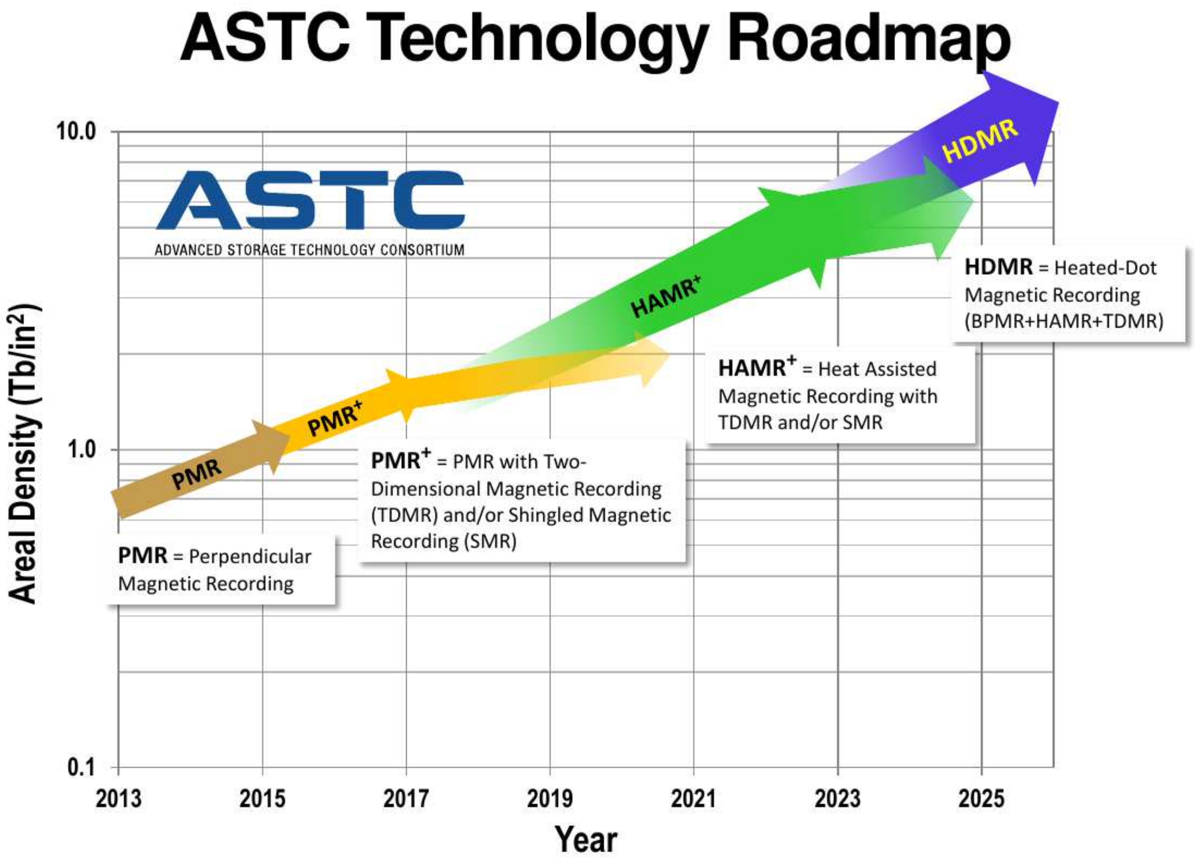


Figure 6 – HDD Areal Density Roadmap (ASTC. . . , 2016)

has the potential to store about $1Tb/in^2$ and has been supported by new technologies such as Two Dimensional Magnetic Recording (TDMR) and SMR. However, PMR increased by TDMR and SMR seems to survive until 2021 where it is expected to reach $2Tb/in^2$ or more. Meanwhile, Heat Assisted Magnetic Recording (HAMR) arises as an interesting solution and followed by TDMR or SMR seems to be the solution of the nearer future and that presents a better ADC measured in Tb/in^2. It is expected that this recording technology can reach $5Tb/in^2$ or even more until 2025. Finally, Heated-Dot Magnetic Recording (HDMR) with some Bit-Patterned Magnetic Recording (BPMR) collaboration seems to be the future in areal density reaching more than $10Tb/in^2$ after 2026.

The technology used by HDDs has changed over a 60-year process and remain to have good improvements concerning density increasing. Before PMR technology is fixed into the market in the 2000s, Longitudinal Magnetic Recording (LMR) was the conventional magnetic storage format for HDDs. Today, there are some interesting methods under the development of magnetic recording: PMR, SMR, IMR, HAMR, Microwave-Assisted Magnetic Recording (MAMR), and HDMR. Some technologies such as PMR, SMR, and IMR display track differently each other but have hardware similarities while others such as HAMR, MAMR, and HDMR are methods with different technologies, concepts, and physics. Figure 7 presents an aerial view of how tracks are recorded into the first three technologies. PMR, Figure 7a, present their tracks written in parallel, side-by-side, and with an inter-track gap separating each track. SMR, Figure 7b, present tracks closer together similar to roof shingles and IMR, Figure 7c, presents interlaced tracks with different linear densities.

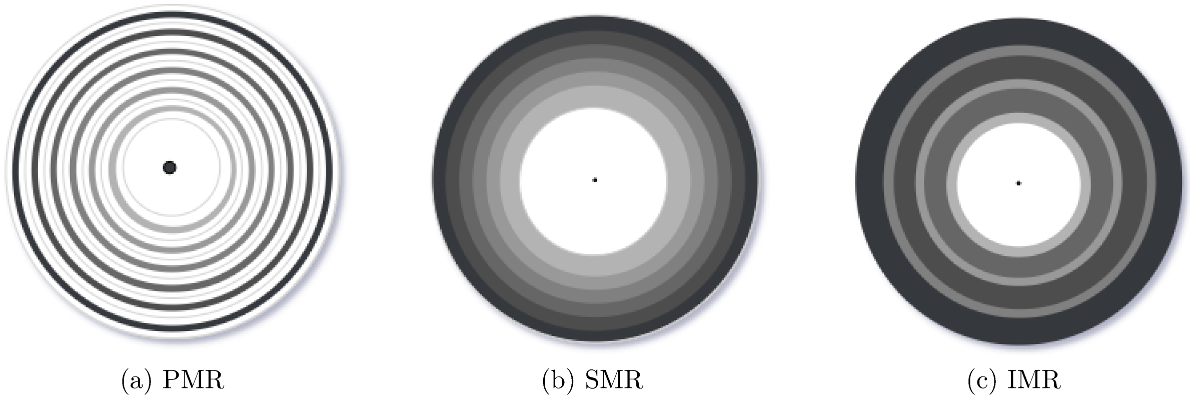

(a) PMR (b) SMR (c) IMR

Figure 7 – Magnetic Recording-based HDD Technologies

Conventional HDDs presents their tracks written in parallel, each one side-by-side and with an inter-track gap that separate each track. The tracks are independent, data are recorded at the same track that they were solicited and it can also be recorded in a random order without any previous organization. Its technology is also known as PMR or Conventional Magnetic Recording (CMR) and its track design, as well as the proportion of the writer/reader head components sizes, are presented by Figure 8. This technology was a milestone because it has begun to record data perpendicularly creating a larger field for written data, differently from LMR that used flattened area. It also turned possible to reduce the magnetic interference ratio due to this additional physical design. In that parallel track disposition, it is not possible to have an overlapped track unless a problem occurs. Nowadays, head technology uses a bigger writer-head than reader-head. Therefore, when a writer requisition is performed, the track next to it is not affected by the previous write, thus not causing loss of information or additional time to organize these tracks. As mentioned earlier, the data inside HDDs are represented magnetically and, in this technology, the pole is aligned to represent the desired data bit that can be 0 or 1. The way in which these tracks are disposed on HDDs reflect directly in the areal density of the devices which is measured concerning the number of bits per square inch. Typically, this area density is an element sought after by HDD manufacturers companies, because the higher this density, the more data can be stored in the same proportion of the material.

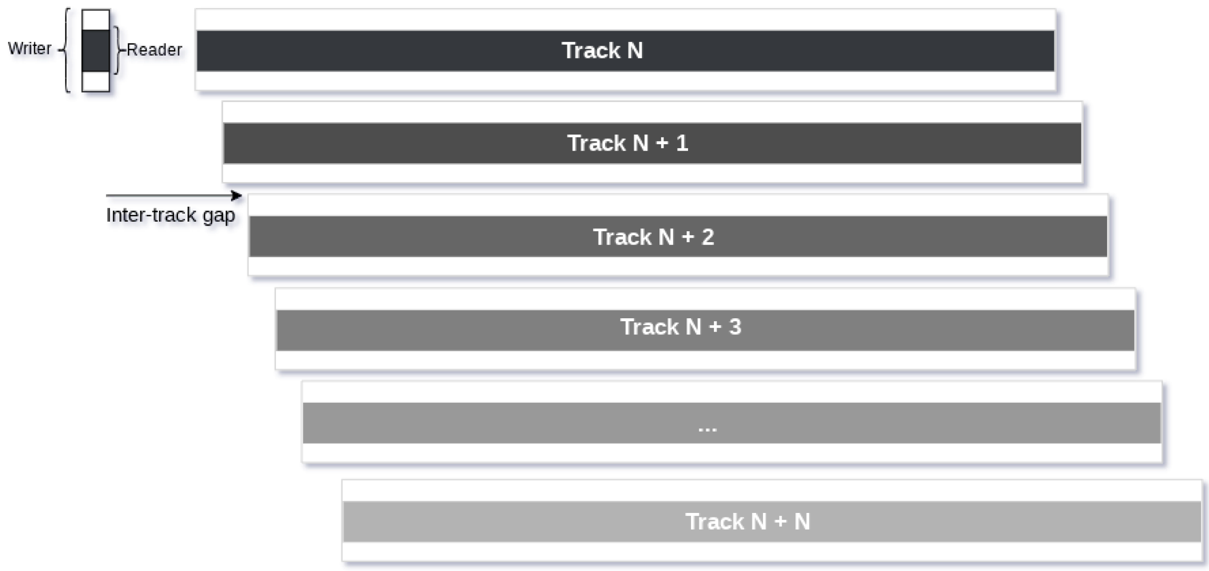


Figure 8 – PMR Tracks Design

The reader and writer physical elements which compose perpendicular magnetic recording HDDs reached their limits. Without new future hardware technologies, it is impossible to reduce even more their physical elements and tracks (SEAGATE..., 2020). Thus, other recording technologies were created considering the display in which tracks were written in HDD. SMR is another HDD technology that displays tracks differently from the conventional PMR method. It squeezes tracks closer together similar to roof shingles which leverages portions of the magnetic area that were not available to record data in the conventional method. Figure 9 presents this track design showing overlapped bands that do not have inter-track gaps as presented in PMR. This design write tracks sequentially in bands increasing density considerably and maximizing the number of tracks per inch. For instance, HDDs from Seagate Technology Company reached increments of up to 25% more space than using PMR method (SEAGATE..., 2020).

Hardware writing elements are bigger than reader elements presented on the arm head, thus, tracks are totally available to be a reader without compromising the data

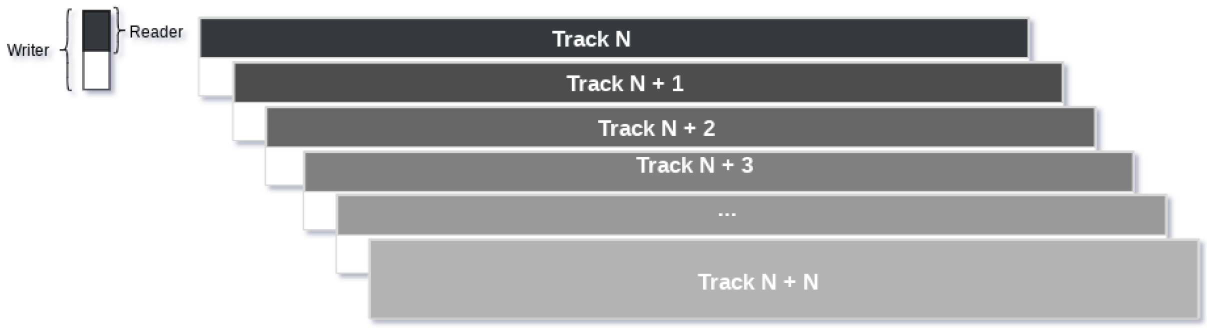


Figure 9 – SMR Tracks Design

integrity and reliability. However, this fact generates an interesting effect when the writing operation is required. When it happens, the writer's head does not overwrite the data on the same track. Instead, they are written onto the next empty track leaving the old data at the same place without changing them. Further, when HDD becomes idle, the controller collects these data from the adjacent tracks periodically and systematically into an auxiliary location. After that, it will delete the old data in the requested track, write back the content of the first write operation into their original track, and execute this process for all the adjacent tracks which should come after this initial track that had their content copied to this auxiliary place. This process is performed as a cascade system ensuring that the adjacent tracks will not lose their content. It is also possible to notice in Figure 9 that the last track receives data in its totality because there is no other track to overlap this one. Because of this necessary extra time to reorganize data, SMR drives is not recommended being used by applications that have high I/O requests.

There are generally three variants of SMR drives: the Drive Managed Drives (DM), Host Managed Drives (HM), and Host Aware Drives (HA). DM provides block interface to host applications such as File Systems and Databases through mapping the Logical Block Addressings (LBAs) to Physical Block Addresss (PBAs) layers (HE; DU, 2017).

All requests from the host are managed by the drive. It is also known to have autonomous or transparent usability from the host with a hidden SMR nature. This technology is also easy to deploy and compatible with previous drive versions. However, the background tasks present very unforeseeable performance, thus, are recommended for personal computers and not recommended for enterprise workloads. HM hosts, as the name suggests, have an imposing role in management tasks. It is also known to being restrictive where all data streams to perform I/O operations are managed by the host as zone management. This drive variation presents a predictable performance compared to the previous one and is more feasible for High-performance scale applications. Finally, HA is a combination of the technology presented by both previous versions, being a hybrid drive. This technology gives the host some control over the tasks but is not totally dependent on them. It is also known as the cooperatively managed method. It is the most used model by traditional enterprise system due to this mixing of previous features (MICROCHIP. . . , 2019).

IMR technology, which displays interlaced tracks with different linear densities, is presented (HWANG et al., 2016). Figure 10 presents its design. First, these two different linear densities, L_e, and odd linear density L_o, are pre-fixed divided into sub-zones. Odd tracks that receive lower linear density L_o, are recorded with a lower frequency, while even tracks are recorded with faster frequency receiving a higher linear density L_e. Therefore, $L_e$is greater than L_o. Odd tracks are recorded first and even tracks are recorded later into magnetic HDDs. Assume N even and M odd numbers. Based on that, tracks $N + 1$, $N + 3$ and $N + M$ will be written first into HDD with linear density L_o. After that, tracks N, $N + 2$ and the subsequent even tracks will be written after the odd tracks into HDD with linear density L_e.

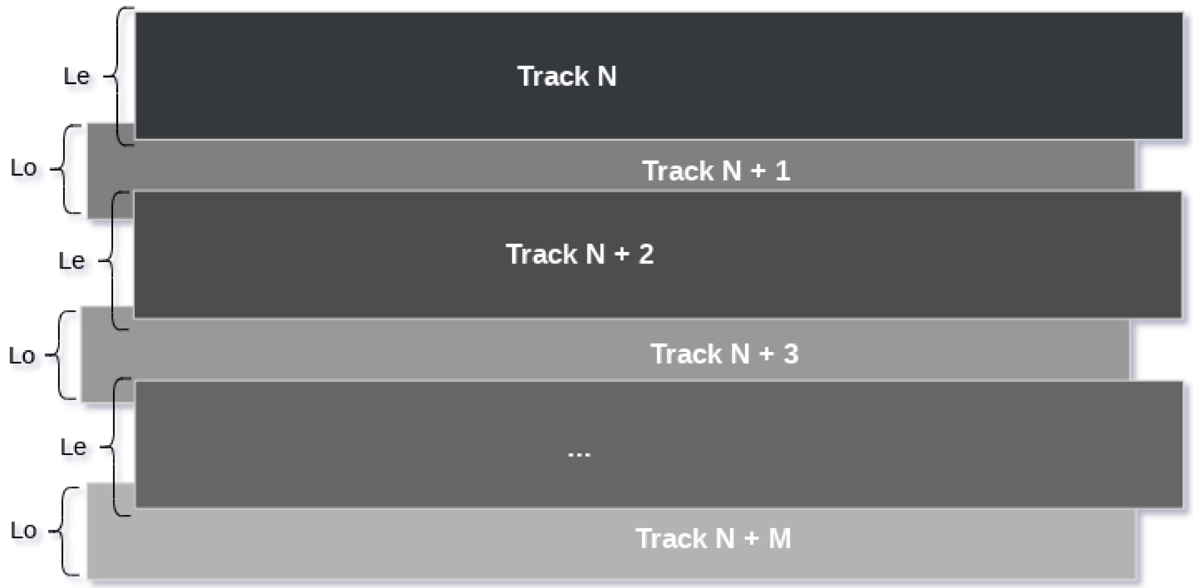


Figure 10 – IMR Tracks Design

Hybrid approaches are also interesting and possible. Mixing PMR and SMR features into a unique drive may present a good option combining time access and capacity. Although SMR technology increases HDDs capacity considerably, it also decreases performance when random writes are very requested. The design of physical shingled tracks imposes write restrictions and consequently decreases the driver performance. As mentioned earlier, SMR writes must be performed considering tracks orders because each rewrite destroys data located into the next tracks. Therefore, SMR devices present better performance for cold data presenting good results for sequentially writing, thus is not recommended for hot data.

PMR technology proved efficient performance when random writes are requested due to the individuality of each track. Differently from SMR technology, PMR increases

IOPS rate and is recommended for hot data. The idea behind this is the technique known as *short stroking*. A stroke is a set of tracks accessed by an application. As physical HDD blocks are mapped to logical addresses through LBA scheme, applications can access these LBAs. When an application accesses an entire LBA area, it uses 100% of a *stroke*. Therefore, *short stroking* technique uses a small range of LBA reducing head movements and increasing IOPS. Fusing these two technologies presented by Figure 10 seems to bring good results and its idea was presented in Google's "Disks for Data Centers" (BREWER et al., 2016). Thus, PMR space is used for hot data and SMR used for cold data. Researchers are also keeping improving and giving attention to this hybrid device, proposing new data management and features to explore such drives (WU et al., 2019). Figure 11 shows this hybrid technology.

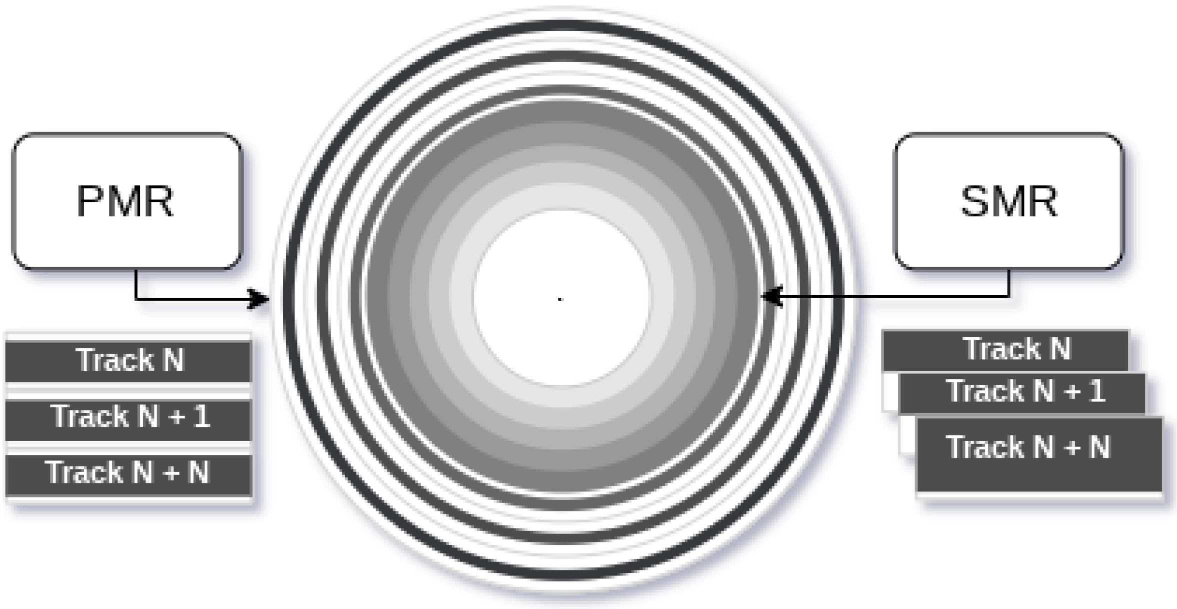


Figure 11 – Hybrid PMR-SMR Drive

New concepts are presented by the next technology generators with a big rate of area density led by HAMR, MAMR, and HDMR.

HAMR is another recording technology that targets larges ADC increases using different recording methods. It was developed using new physics, optics approaches, recording materials, and even lubricants to withstand high temperatures (KRYDER et al., 2008). It also requires the development of novel elements such as a light delivery system, a robust hard disk interface, a therm magnetic, and a cooling media. The recording head that integrates concepts from the optic and magnetic field was also modified. In essence, the superficial area that should receive data is heated by a laser reducing its coercivity and expanding its surface, after, data is written into the expanded media and finally, this heated area is cooled to reduce its size for initial size. The areal density metric ASTC presented by (GRANZ et al., 2016) is commonly used to measure HAMR technology.

In MAMR recording technology, a thin film medium is introduced to extend the recording density. It can achieve interesting results and through a perpendicular switching field. MAMR allows recording at a field which is located below the medium coercivity. Its technology also turns possible to produce an ac field through a spin momentum transfer (ZHU; ZHU; TANG, 2007). It makes use of a spin torque oscillator that generates a microwave field which is used to assist writing into the device without losing reliability. It is expected that MAMR deliver up to $4Tb/in^2$ which is four times the ADC of PMR actual technology. The successor of HAMR is already under development.

HDMR is the technology that is expected to deliver up to 100 TB of storage in the future but it is difficult to be available for the commercial audience until 2025. As the name implies, it also uses a laser to heat the medium before writing the data.

Granz et al. (GRANZ et al., 2019) presented an interesting analysis comparing CMR, SMR, and IMR combining Multi-Sensor Magnetic Recording (MSMR) with one, two, and three readers. MSMR is a read-back architecture that uses more than one reader to read the same written track. In this experiment, they used the HAMR head technology also using the areal density metric ASTC. The ADC is measured in $Tbit/in^2$ and an average of ten heads produced results presented in Figure 12. A conventional HDD with one head reader using the traditional recording technology HAMR CMR achieved 1.34 $Tbit/in^2$ whereas the best result using HIMR with MSMR and three heads achieved 1.91 $Tbit/in^2$. Their total evaluation can be visualized in Figure 12.

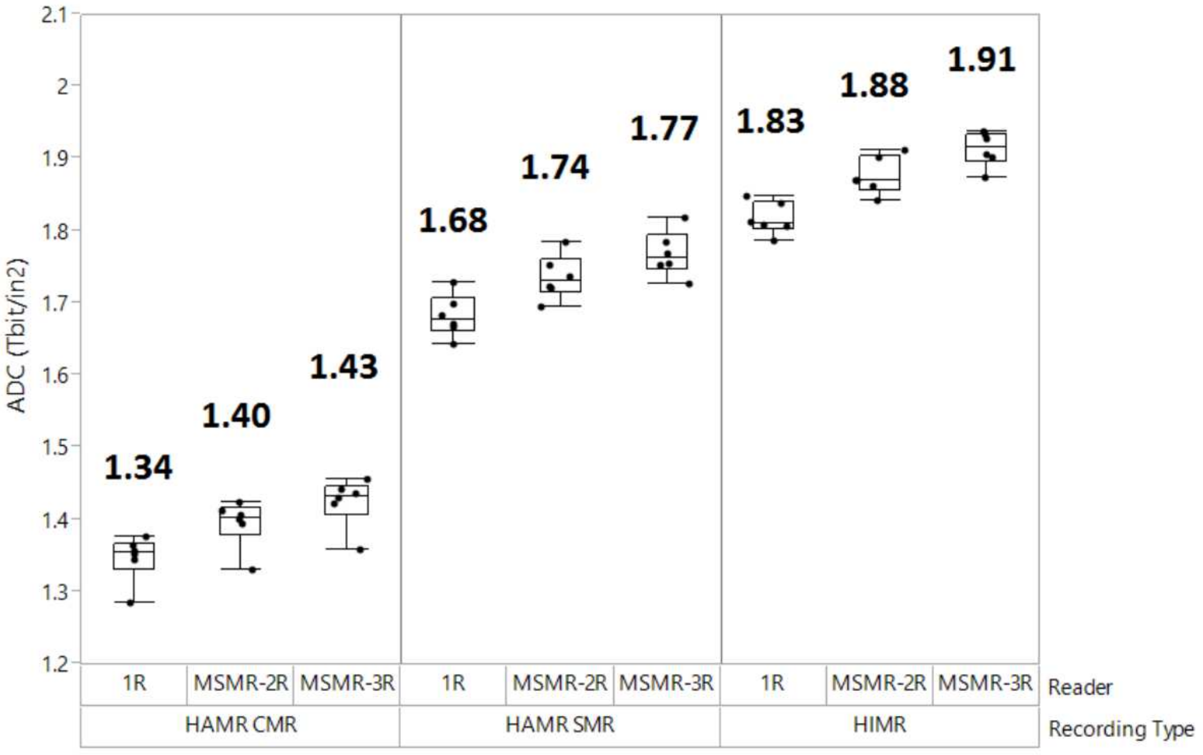


Figure 12 – Areal Density Comparison Between PMR, SMR and IMR (GRANZ et al., 2019)

2.2.3 Optical Data Storage

Storage devices such as Compact Disc (CD), DVD, and Blu-ray Disc (BD) are quite similar and use the same principle to store and retrieve information. The device's structure is similar, all of them have approximately 12 centimeters in diameter, and data is read through an optical laser. Although these visible similarities, they have different and important properties such as the number of bits, wavelength, line distance, among others. One of the factors that enable optical devices to store different amounts of information is related to the wavelength size emitted by the laser beam. To record data into a digital disc, puncturing its surface is necessary. The shorter the wavelength, the smaller the marks created and projected on the device. This factor allows the distance between marks to be reduced, allowing more marks to be applied to devices, thus resulting in greater data storage quantity on disks.

Table 2 summarizes some characteristics of these three media devices.

Table 2 – CD, DVD and BD Characteristics (COSTA, 2007)

Mídia	Depth	Vertical Distance	Wavelength	Pit Length	Capacity
	(µm)	(µm)	(µm)	(µm)	GB
CD	0,13	1,6	780	1,0	0.7
DVD	0,11	0,74	650	0,32	4.7
BD	0,07	0,32	405	0,14	25

CDs use an infrared laser and its wavelength is equal to 780 nm. DVDs use a red laser and have a wavelength equal to 650 nm. Differently from the previous devices, BDs use a violet laser and have a wavelength equal to 405 nm. The distance between the line marks in CDs is 1.6 µm. In DVDs such marks is 0.74 µma and finally BD is 0.32 µm. The depth of the holes on the surface of the disc are also different CDs is about 0.13 µm, while on DVD it's a little bit smaller around 0.11 µm. Deeper, in BD it reaches 0.07 µm. The vertical distance between lines also changes and can be up to five times between CDs and BDs. The CDs have vertical distance is 1.6 µm. On DVDs that number drops more than a half to 0.74 µm. BD has the vertical distance five times shorter than the vertical distance of CDs being 0.32 µm. This is an interesting measuring because the smaller is its distance and the vertical distance the more information the surface would be able to record. Some of these attributes can be verified in Figure 13.

Data reading in optical media is performed by reflecting the incident light through an interferometer. Due to the smooth surface of the media, the incident light is reflected being interpreted by the digit 1, however, when there is a hole as shown by the figure Figure 13, the light is not reflected being interpreted by the bit 0. Figure 13 presents the three different media devices surfaces with marks and the distance between them. The images were expanded by 20,000 times allowing the authors to measure the depth of the

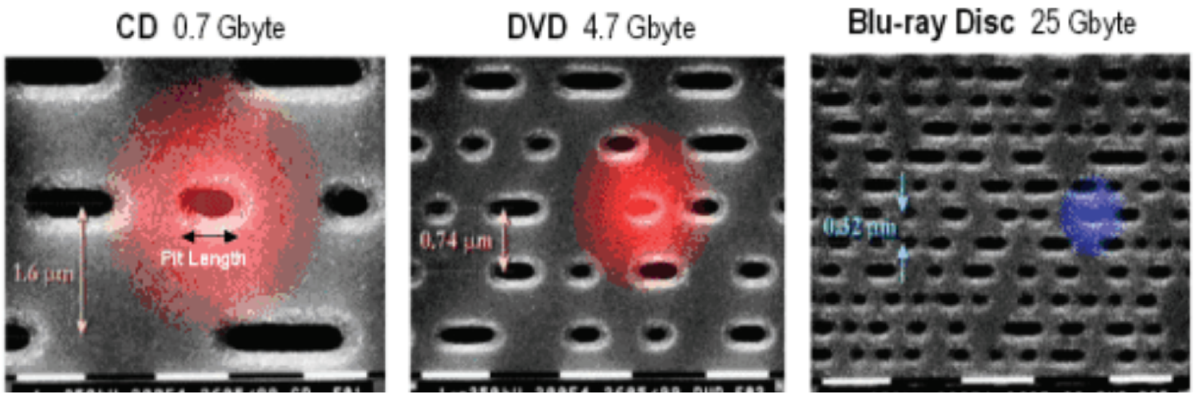


Figure 13 – Optical Storage (OPTICAL..., 2000)

fissures on the disc's surface. It is also possible to verify that the cracks are presented by the black coloring forming a horizontal line while the gray coloring represents the smooth surface of the disk where there is no data recording.

2.2.4 Solid State Storage

In the 1980s, Dr. Fujio Masuoka begun the development of a new memory design to reduce the high cost of producing applied by Electrically Erasable Programmable Read Only Memory (EEPROM). He proposed a new configuration on the number of transistors per cell that reduces memory cell area while working at Toshiba (INOUE; WONG, 2004). The main idea was reducing the quantity of two transistor cells presented on the EEPROM to one transistor cell. Consequently, some changes in design functioning occurred. One of them was that the byte erases scheme presented on the EEPROM was dismissed starting then the creation of a new simultaneous multi-byte erase scheme. Comparing with EEPROM memories, which have two transistors per cell and an area that occupied 272 sq. microns, simultaneously erasable EEPROM or Flash EEPROM has one-transistor per cell and an area that occupied 64 sq. microns.

In Flash EEPROM, erase operation is performed at the entire chip while byte-erasable EEPROM erases one byte at a time. It seems to be more economical because, in the case of semiconductor memory, the bit cost is dependent on the memory cell area per bit. The new proposed technology has received a name that became known as flash memory. With this change, the operation could erase numerous memory cells simultaneously and the price of the device decreases. The two dominant types of Flash memory are NAND-type and NOR-type. The difference between their designs is shown in Figure 14.

Flash memories achieve random access by connecting memory cells to the bit lines in parallel. Introducing memory cells in parallel to bit lines gives flash memories random access even for NAND and NOR types. However, NAND flash memories achieve a smaller area, reaching a lower price. It is composed basically of a dual gate structure, where an insulator element, normally silicon dioxide, isolate a floating gate from a control gate

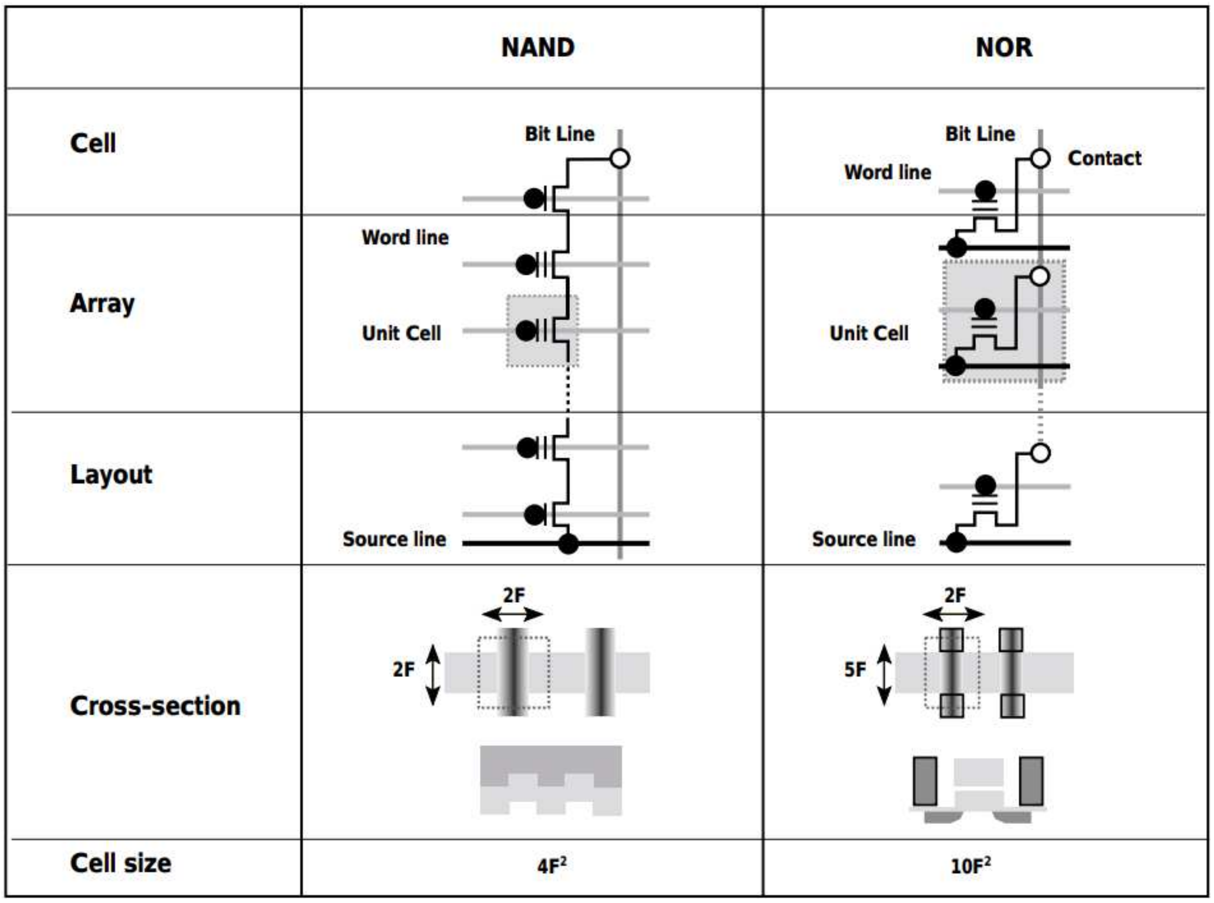


Figure 14 – NAND Flash vs. NOR Flash. (INOUE; WONG, 2004)

preventing energy from being dissipated from the floating gate. Indeed, the bit cost is the most important characteristic of semiconductor memories, thus turning NAND flash memory more consolidated in the market. However, other factors are important depending on which is the focus. NAND flash memories impose certain resistance due to their cell array design to read out the first data byte. This becomes a decisive attribute for operations where the start of an activity is critical.

Write operation time, also known as programming, together with erasing time, are also possible operations to be executed in both NAND and NOR-type memories. However, its time to be executed into NAND type is much lower than in NOR-type because it uses Fowler-Nordheim tunneling to execute them, thus allowing memory cells to be programmed simultaneously. It is also a factor that reduces power consumption into NAND type because it does not increase considerably as much as the number of memory cells increases. In NOR type, the programming operation is allowed to be performed only by a byte which is also a factor of time and power increasing. The number of cycles a block or chip can be erased and programmed, also known as endurance, is tremendously different. It is estimated that NAND type provides nearly 1,000,000 cycles while NOR-type provides about 100,000 cycles. Table 3 summarizes some important characteristics from them.

Table 3 – NAND vs NOR Comparison

	NAND	NOR
Cost	Low	High
Initialization Time	High	Low
Cell Size	Low	High
Power Usage	Low	High
Erasing/Programming Time	Low	High
Endurance	High	Low

Because of some characteristics presented in Table 3 but led by cost rate, NAND type memory is widely adopted as solid-state mass storage instead of NOR type. To produce it in large scale NAND flash-based SSDs is widely used as secondary storage for computing systems due to many advantages compared with conventional HDDs. Fast access speed, small size, shock resistance, and low consumption are examples that justify their usage and attention by researchers. However, with these advantages, new challenges and concepts which did not exist in the old magnetic disk technology came up. To better understand this technology, we present some factors about these components placed into a flash device that may influence the performance when performing applications.

We begin presenting the main elements of a modern SSD in Figure 15. It is composed of elements decomposed basically into three blocks. The first one is the most important, the *SSD Controller*. There, it is possible to verify a range of elements, each one responsible for certain hardware and software management. The Embedded cores are processors introduced into SSD controllers specifically designated to manage channel parallelism and internal bandwidth. These cores manage many operations including I/O request scheduling, wear-leveling, garbage collection, and other data management through FTL. SSDs also provides a DRAM faster memory which is faster than NAND flash memory. It acts as a buffer to reduce latency among the communication of those elements storing temporary controller data structures basically. It is connected through a DRAM controller which helps with a specifics hardware interface.

Error Correction Code (ECC) accelerators are also provided by SSD devices. It acts by providing several hardware accelerators to reduce error correction code processing or data encryption. The Host I/O interface makes the interconnection between the SSD component and the computer system. It receives I/O operations from the host CPU and distributes among the other elements presented into the SSD controller. The connection between SSD Controller and NAND flash memory chips is performed through flash interface channels that send specific commands to execute I/O operations in individual channels in parallel. It is worth noticing that Figure 15 presents basic elements from a modern NVMe SSDs (KOO et al., 2017).

The logic elements of a flash chip are presented in Figure 16. It is composed basically of four elements which are: die, plane, block, and page. These elements have specific and defined functions to store and read the data. The "die" component is the smallest unit that can independently execute commands. This element is composed normally of one or more "planes". Planes are composed of "blocks" which are the smallest unit that can be erased and might be 4-8 MB in size. Finally, each block is composed of "pages" which are the place that can be programmed. Pages have typically 8-16 KB in size and receive program operations such as read and write.

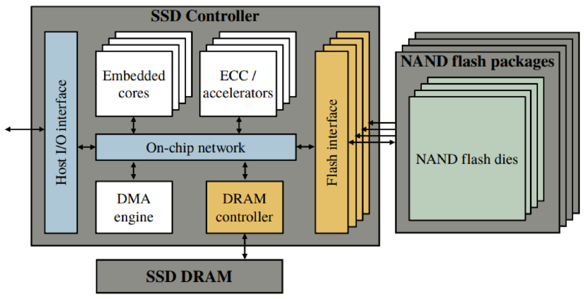


Figure 15 – Architecture modern SSD (KOO et al., 2017)

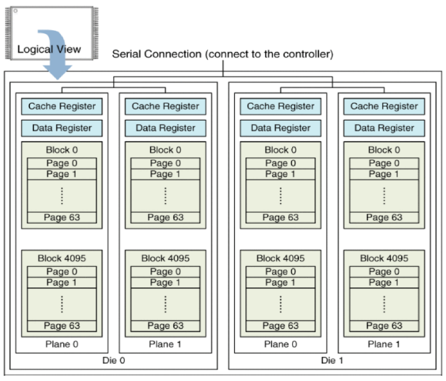


Figure 16 – Flash-memory package design (HSIEH; LIN; YANG, 2013)

After writing data on the pages and consuming it, the used page should be available again to write data at a later time. At this moment, a flash device particularity happens because there is no "update" operation in this technology and some problems come up with this aspect. To "update" data within a page, the block where the page is located should be fully erased to be rewritten. Thereby, an issue emerges because an "erase" operation clears data from all pages in the related block and some of them could have usable data that should not be erased. This is a basic hardware characteristic of flash memory and it is known as erase-before-write. To solve this, some techniques such as, copy the available data to another place or hold off from doing the erase operation on the related block, were created.

The first option copy the content from all available pages to an auxiliary block targeting erase the related block. It may be performed only in situations when this action is really necessary because this action increases the number of extra page copies. Because of that, it is not recommended to perform many erase operations on flash memory due to their limited number of erasing options. Flash chips have a limited number of program and erase options known as program erase cycle (PE cycle). The more erased the block is, the more degraded it gets reducing the NAND flash chips lifetime. The second option marks the page that has old data as invalid and writes the new data into an empty page. The element which makes this mapping between invalid pages and empty pages is an intermediate software layer called FTL. It has many functionalities that help and improve the management of data in flash devices and are discussed below.

Merging these two approaches is one way to solve the update problem consciously. To understand how these operations can be carried out jointly, let's suppose some scenarios. Figure 16 presents one flash ship with two die elements. Each of them has two planes and each with 4095 blocks being composed of sixty-four pages. In the beginning, all pages are empty and the system is able to perform program operation (i.e. write operation) on the pages. After performing it, some pages are now used and it contains data that in the future will be retrieved.

It is worth noticing that much data are sent to be written on the pages, the FTL layer coordinates the quantity of free, used, and stale page status of all blocks and tries to manage through the wear leveling the data on the blocks. In some states after performing many program operations and stamping some than with stale status, after writing the updated data to an empty page, an "update" should be necessary to set the stale pages free again. But as we know, the stale pages cannot be turned free again without the entire block first being erased. First, it is recommended to copy the used pages somewhere else and after that erase the entire block. It should be very good managed by the FTL because if there are not free pages where the used pages could be copied, a problem related to no space on the device will raise even owning stale pages.

Suppose, in Figure 16, that plane zero has sixty-three used blocks and one is half empty. Make the assumption that there is only its plane into the flash chip. At a certain moment, the quantity of used pages on a first block is greater than the number of free pages on the half empty block and its first block has an uncertain quantity of stale pages. This scenario makes the first block to be erased necessary to turn these stale pages free again to free up more space. However, it is very likely that this operation will never free up again because there are no free pages on the half block to copy the user data from the first block to then erase it. This situation should be prevented by algorithms managed by FTL because intensive I/O operations make it frequently and depend on such memory devices to manage data.

Flash Translation Layer - FTL

The FTL is an important layer that can be implemented in different ways in storage systems. One of these perspectives could be in the form of software where the FTL layer can be introduced into the SSD controller. Another way to introduce it is by implementing it as a hardware layer into the storage systems. FTL has the purpose of intermediate, rearing, and adjusting, software systems and hardware devices toward the best fit among them and have been introduced by many researchers in different ways. Figure 17 presents the overall architecture of flash memory systems with the FTL layer applied between the file system and the flash memory chip. However, FTL algorithms not only have a great effect on storage performance and lifetime but also determine hardware cost and data integrity (LEE et al., 2016).

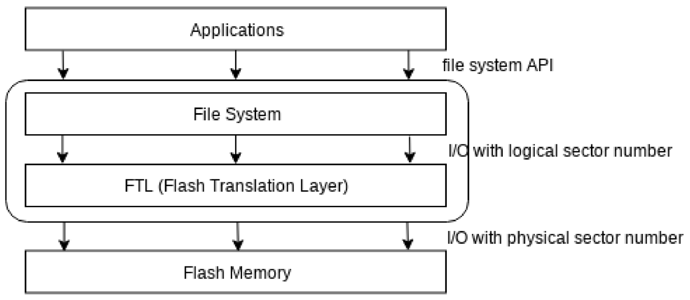

Applications
file system API
File System
I/O with logical sector number
FTL (Flash Translation Layer)
I/O with physical sector number
Flash Memory

Figure 17 – Overall architecture of flash memory system (CHUNG et al., 2009)

According to Chung et al. (CHUNG et al., 2009), an FTL runs some firmware algorithms and it should provide tree basics functionalities. Considering these three basic functions, it is possible to project-specific algorithms toward the improvement of the overall system. It is also possible to use and project FTL algorithms to increase points such as data integrity, resource requirements, I/O parallelism, and performance.

- Logical-to-physical address mapping

  According to Figure 17, the applications, through a file system API, issues a series of a requisition into the file system layer. Through the FTL algorithm, the logical addresses which come from the file system are converted to physical addresses in flash memory. This functionality is the main among others provide by an FTL algorithm.

- Power-off recovery

  FTL data structures tend to be preserved even when an abrupt power-off event occurs.

- Wear-leveling

  Another functionality that an FTL algorithm should provide is a wear-leveling function to spread the data overall blocks fairly evenly. With this functionality, the data are equally distributed between the blocks, and the erase program operation is avoided preserving the lifespan of SSD devices.

According to (LEE et al., 2016) it is possible to classify FTL schemes in two general types. As mentioned in subsection 2.2.4, flash chips are composed of planes with several blocks and pages inside each die and this classification takes into account these physical components. Page-level and block-level FTL schemes are the two kinds of generic FTL schemes mentioned. In the first one, a direct mapping can be accomplished through the logical pages from file systems and the physical NAND flash pages. Different from the first approach, block-level FTL schemes map the logical block to a physical block fixing a page offset within a block.

Hybrid FTL - (HFTL) HFTL differs from the previous version in some ways. One of them considers the division of the blocks level function in NAND flash chips. The FTL software firmware arranges the blocks in data and log blocks assigning different functions to each block. Block-level mapping is responsible for managing the blocks and represents the storage space. These blocks are named data blocks. On other hand, the blocks reserved for logging newly updated data that are managed by page-level mapping are designed as log blocks.

Although the overall number of blocks used as log blocks is quite small, the FTL algorithm manages the relation between the new and old data into pages of these blocks. As mentioned in subsection 2.2.4, each block contain a number of pages and if this block was designated by the FTL algorithm as a log block, these pages will receive only newly updated data. As the new data are being updated into log blocks pages, the quantity of free pages becomes small and scarce and the FTL algorithm has the function of deal with that fact.

The question is solved through an operation performed by the FTL called merge operation, which aims to create new free pages in log blocks. The valid pages presented in log and data blocks are merged into new data blocks, releasing free log blocks. It is described by (LEE et al., 2016) three interesting types of the merge operations called switch merge, partial merge, and full merge.

2.3 HIGH-PERFORMANCE STORAGE ENVIRONMENTS

Storage systems can be interpreted as being two different elements. It can relate to a software management system that organizes and control how data are stored, accessed, manipulated, and retrieved. Parallel File Systems (PFSs) (e.g. Lustre(BRAAM; SCHWAN, 2002), PVFS(ROSS; THAKUR et al., 2000), Ceph(WEIL et al., 2006)) and key-value (Redis, Riak, Dynamo, Voldemort, BerkleyDB), document-based (MongoDB, ChuchDB), wide-column (Cassandra, HBase, BigTable, Hypertable) and graph-based (Neo4j) databases are some examples of software that manages data into storage systems. However, storage systems can also relate to a storage environment usually composed of much storage media (e.g. magnetic disks, tapes, solid-state drives, etc). In a large-scale environment, this storage connects media devices through high-speed network connectors (e.g. Infiniband, Myrinet, Omni Path), but also in small environments its devices are connected through an Ethernet network. This environment might also provide reliability over the data employing data redundancy and distributing data over different disk media.

This feature is provided by a Redundant Array of Inexpensive Drives (RAID) and can also be introduced by a hardware or software implementation. When implemented from a software perspective, the operational system (OS) makes the RAID management through the disk controller. In such a case the necessity to employ a RAID controller is discarded and because of that is costlier than the other approach. On the other hand, the hardware implementation uses proprietaries disks layouts, that are unique and usually not compatible with each other. Because of that, hardware implementation requires a dedicated controller for this task turning its implementation expensive. As explained before, in this research we are considering the storage system as being a combination of software and hardware that works asynchronously.

Many factors such as the big disparity between processing and storage followed by the large quantity of produced datasets by large-scale applications support some author's idea that new storage models become necessary. It is worth noticing that some proposed studies advocates new storage forms considering similarities by the way HPC and Big Data applications manage their storage system (THE..., 2018). Figure 18 presents some similarities and differences considering HPC and big data environments.

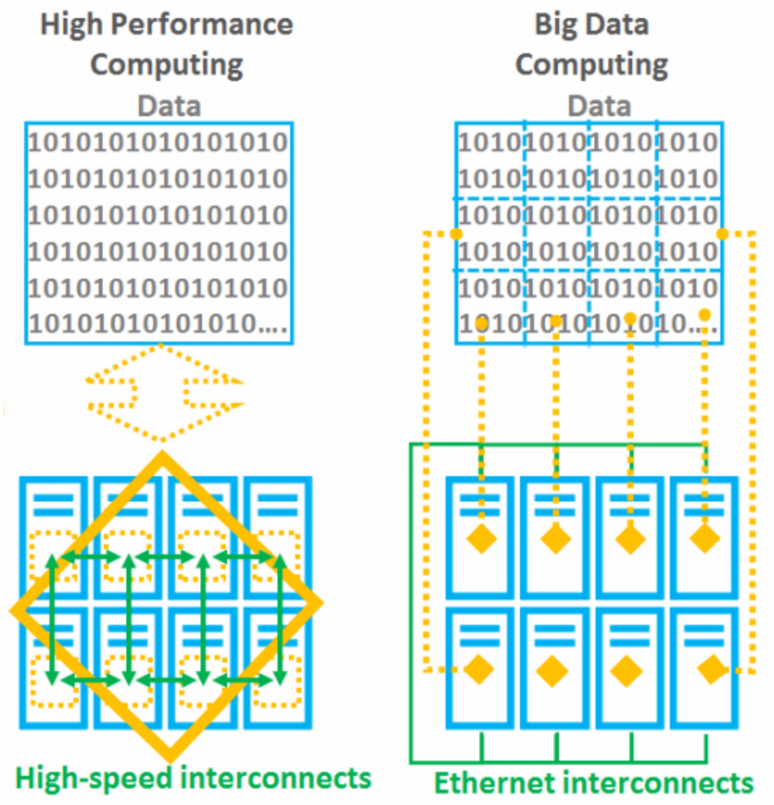


Figure 18 – HPC and Big Data Computing Architecture (THE..., 2018)

On the left, we verify a basic HPC stack composed of software and hardware elements. Usually, an HPC stack is computing-oriented and deal with interaction among parts of an entire system. By being computing-oriented, usually, this class of environment operates large sets of data in parallel using a big quantity of CN at the same time to target a desirable result. HPC application are usually iterative and closely coupled (THE..., 2018). The entire system tends to be busy running only one application or instance of that application. CNs are connected throughout a high speed interconnects (e.g. Infiniband, Myrinet, Omni Path) in which data sets are shared using message passing. Therefore, connecting several CNs throughout quickly interconnects fabrics, the parts of the systems can be visualized as a big and unique supercomputer.

On the other hand, Big data stacks present different characteristics and were designed to deal with other classes of applications and problems. Usually, this environment-class is data-intensive and uses a big quantity of data sets to generate and discover insights over the computed data. In this class, data are divided over many parts of a whole and distributed over multiple instances of the same application. Differently from HPC architectures, here is interesting to having several CNs with medium performance rather than having highly powerful nodes as required in HPC environments.

2.3.1 Big Data

The ever-increasing data production and consumption have changed how enterprises and academy are dealing with information. Engineering (e.g. molecular nanotechnology, and earthquake), business (e.g. computational finance and information retrieval), natural sciences (e.g. bioinformatics and astrophysics) are some of the many data study fields which contribute to this increasing scenario. However, these research studies do not only require a high power processing but also generate a massive volume of data. Data acquisition, storage, analysis, and visualization have an important role in this data deluge found nowadays. For instance, in the astrophysics field, data acquisition is very well-known. Data that came from the exploration of galaxies which tries to capture as much as possible propagated wave signal over the space to find patterns and discover how the galaxies are evolving are captured from huge telescopes endlessly. The analyses of these large volumes of data is another important step in this scenario which consumes much power processing from resources to generate understandable information through the visualization process for post-human analysis.

IDC predicts that, by 2025, more than 175 zettabytes (ZB) would be the total amount of generated, captured, replicated, and consumed data on a global scale (REINSEL; GANTZ; RYDNING, 2018). These data arise from three primary locations through a mapped ecosystem which are: the core, edge, and endpoints. The core is composed of elements presented in large data centers such as private and public cloud and it is responsible for the largest part of this data projection. Edge is the middle layer which is composed of elements such as branch offices and cell towers. In the border, it is possible to notice the endpoint where the data flow to the core. This layer is composed of some elements such as mobile, vehicles, and assets.

Figure 19 presents a historical and data projection over the next years until 2025 according to these three fore mentioned elements from a creating and storing perspective. A mentioned point is that the elements that have created and stored the data do not need to be the same and have different comportment through time. From a data storage perspective (dotted lines in Figure 19), it is possible to notice that the amount of data being stored in endpoints are decreasing quickly and, by the graphics, it has already crossed the quantity of stored data by the core element. The stored data in the core is growing at an accelerated rate and is expected to be the most used way to store data in the future. The edge data store seems to be increasing at a low rate compared to the core and by the projection, in 2025 it will be something about seven times smaller than the amount of data stored in the cores. By creating data perspective (solid lines, in Figure 19) the endpoints elements are also declining, the core and edge are increasing their data creation ratio and the difference between them and endpoints are truly significant.

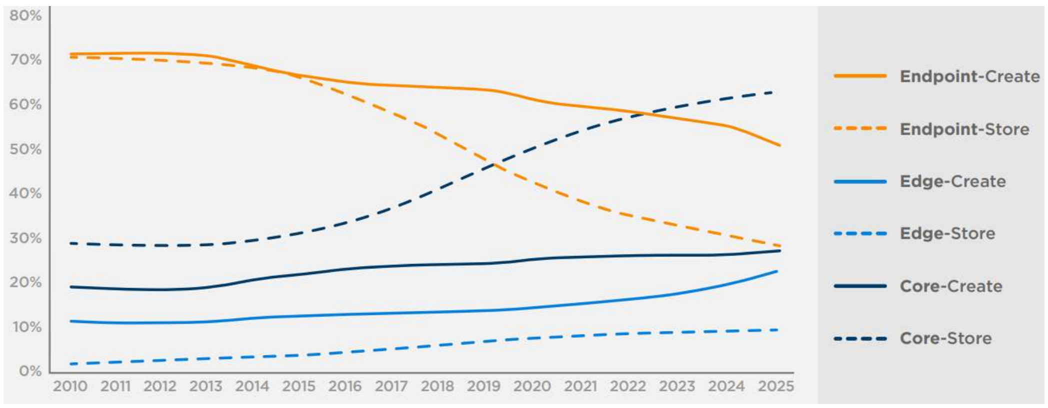


Figure 19 – IDC White Paper, Creating and Storing Data by Core/Edge/Endpoint (REINSEL; GANTZ; RYDNING, 2018)

Big Data is a paradigm, usually referred to as 5Vs, (e.g. volume, velocity, variety, value, and validity) that employs data sets management. Usually, the quantity of created data is so big that conventional software and techniques can not deal with this data without employing some bottleneck process. This refers to the volume term aforementioned. The velocity in which the data change is also concerned. It grows quickly in seconds fraction imposing even more processing efforts. The variety refers to the diversity of these data. Nowadays, data is captured over a broad range of sources (web data, business generation, mobile data, and so on) Data can be also structured or unstructured and, nowadays, the big quantity of existed data is unstructured imposing more effort to organize them before processing and analyzing. Value is concerned about adding value over the processed data to enterprises. Enterprises usually generate much data and obtain interesting business value and insights over these data are also concerned. This is usually referred to as data analytics that targets to make better decisions and improve performance. Validity relates to the validation of the data, in other words, the quality of the data.

To help in those processes, usually big data employs specific methodologies. As a result, NoSQL and New SQL software environments (e.g. MongoDB, BigTable, Redis, Cassandra, HBase, Neo4j, and CouchDB) were conceived to tackle some problems. Methodologies and frameworks designed for this processing such as a map and reduce also tries to improve the big data processing field. Collecting, organizing, and analyzing are interesting big data process phases in which big data applications and computing are submitted. When we noticed these terms description in authors' papers we classify its domain as being a big data domain.

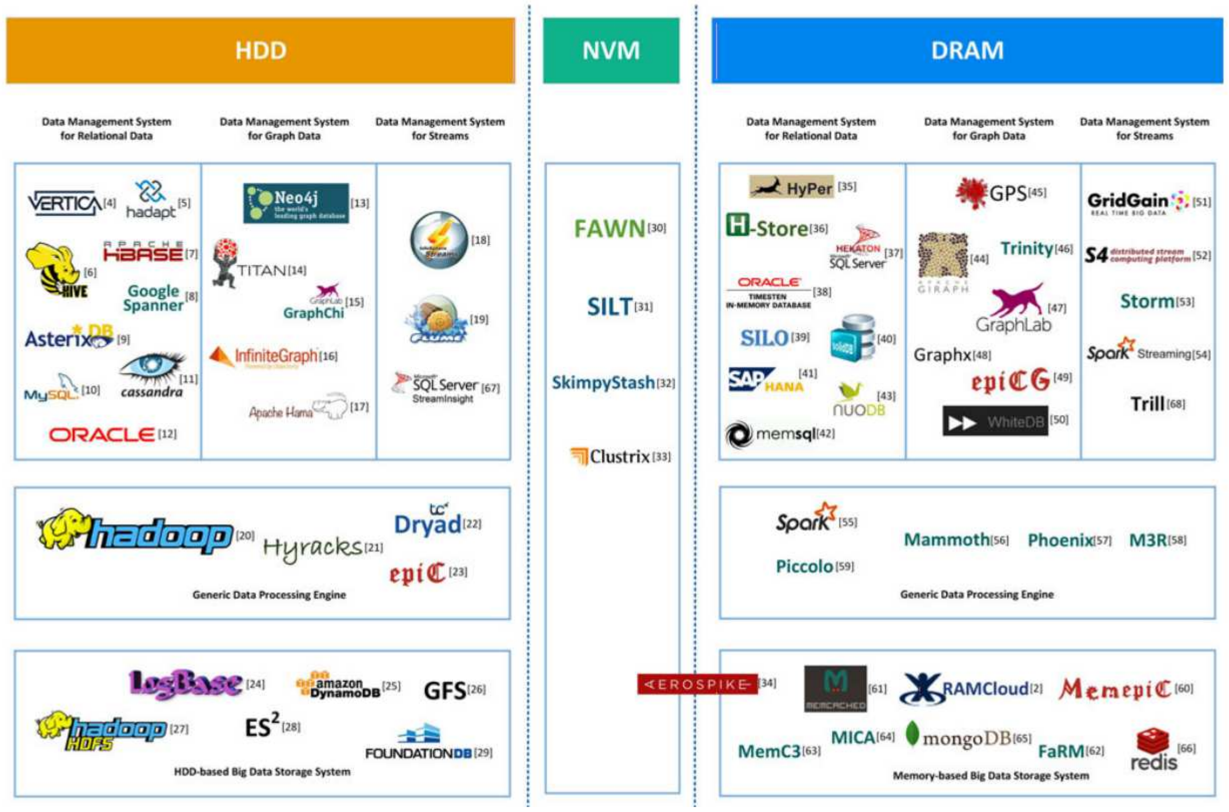


Figure 20 – Partial landscape of disk-based and in-memory data management systems (ZHANG et al., 2015a)

Figure 20 presents a landscape that relates devices and storage management systems. We noticed that many of these systems are designed and projected to use a different class of storage media. Although these are not all the systems that represent such medias devices, we can verify a broadly and distributed variety of management systems

2.3.2 High-Performance Computing (HPC)

HPC is associated with the class of compute-intensive workloads, applications, and performance-critical tasks that use a highly powerful, multilevel, hierarchically organized computing resource designed to address problems that require exhausting processing. These workloads usually refer to simulations and modeling problems commonly found in the scientific and industrial fields that are infeasible to be processed on a unique hardware capability.

Usually, HPC applications are executed in parallel using small parts of a whole system in different and distributed hardware through a synchronized process way. These applications commonly need to perform a huge quantity of computing operations to process the expected or simulated result. Usually, they are built considering different low-level parallel paradigms and programming models such as OpenMP (CHANDRA et al., 2001), MPI (SNIR et al., 1998) and PGAS. Generally, HPC storage system environment can be summarized into different sets of nodes.

CN, IO Node (ION), and SS which is composed of SN, are elements that the most high-performance computing environment presents in their architectures. This kind of environment usually employs a buffering layer(e.g. Burst Buffers (BBs) (ALI et al., 2009)), Parallel File Systems (PFSs) (e.g. Lustre(BRAAM; SCHWAN, 2002), PVFS(ROSS; THAKUR et al., 2000), Ceph(WEIL et al., 2006)) and finally a layer responsible to aggregate the storage devices (e.g. RAID Controller). When we noticed these terms described in the author papers we classify the paper domain as being an HPC domain.

Commonly called HPC applications, those applications are usually executed in parallel using small parts of a whole system in different and distributed hardware through a synchronized process way. Usually, they are built considering different low-level parallel paradigms and programming models such as OpenMP (CHANDRA et al., 2001), MPI (SNIR et al., 1998) and PGAS. However, to attach this feature, these applications should be projected using parallelism where the programmer specifies and finds the software lines code that can be improved by such a paradigm. In the running application process, it will be probably divided into sub-tasks which will be distributed among different nodes and executed by many compute units at the same time. Figure 21 presents the simplified schema for an application executed into an HPC infrastructure.

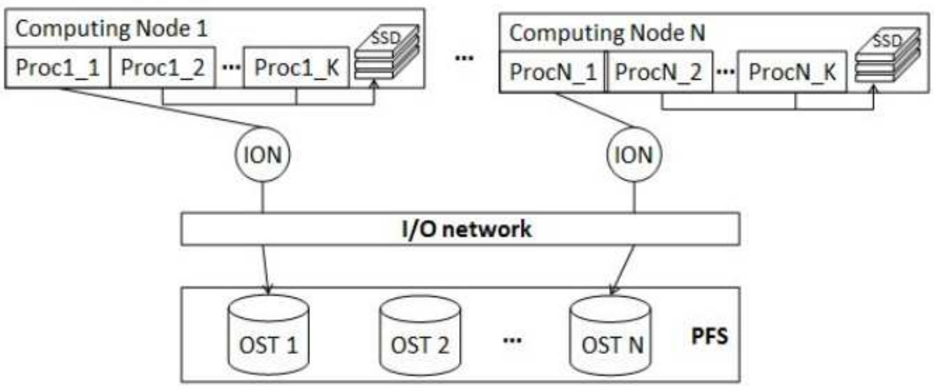


Figure 21 – The process of application execution (ZHA; SHEN, 2018)

Towards maintaining processing closer to auxiliary memory, each CN has a dedicated fast storage device used to reduce the latency when performing computational operations. The fast read and write device acts as an auxiliary memory space for the applications' data management and it is usually applied as a cache or a burst buffer layer. CN become composed of many sub-tasks which were divided by a computational task and distributed for all of them. Each sub-task can be assigned to a process and the more the number of processes on each compute node, the short the complete time of overall application due to the higher parallelism (ZHA; SHEN, 2018). However, CN are directly connected to specifics ION and then, through a high I/O network, they can send the data to be stored into the SS which is managed by a PFS. HPC applications have been developed using mainly two storage levels: the main memory and a globally-visible PFS (LATHAM; BAUTISTA-GOMEZ; BALAJI, 2017).

HPC applications are executed mainly into CN which is exclusively nodes dedicated for processing. It becomes composed of many sub-tasks that were divided by a computational task that were distributed for those dedicated machines. However, the processed data are managed by PFS (e.g. Lustre (BRAAM; SCHWAN, 2002), PVFS (ROSS; THAKUR et al., 2000), OrangeFS, Ceph (WEIL et al., 2006) that manages the flow of processed data from CN to SN.

They are composed basically of three elements. The first element is the *clients* that are executed into CN and are responsible to provide the interface to the PFS. They also claim frequently many data blocks for PFS and their requisition are striped across various data servers through parallelization increasing than the performance request on SN. The second element is the *metadata servers* that are devoted to performing management tasks such as file naming, data location, file locking among others. They have an important role in data management architecture. When some data is required, it is very difficult to access it without requiring them for these metadata servers. They have the right local for accessing each data into data servers. Moreover, they check the requester and give them proper access authorization. Thus, metadata servers are normally replicated and distributed for reducing bottlenecks. The third element is data servers which are executed into SN. Once such data access is given by metadata servers, data servers perform I/O operations (e.g. read, writes, etc) on locally stored data.

To get the best out of it, these applications need a special environment to be executed. HPC computing environment relates a specific and architecture hardware environment projected to execute those costly workloads. They were created targeting increasing the power processing computing by sharing resources and parts of individual hardware systems forming a unique and synchronized powerful system. It is synchronized over a class of nodes designated mainly for computing operations called CN. These nodes are usually multi-cores which increase the power processing and permit applications to use parts of the system in exclusive mode.

To maintain the data synchronous over the computation, the nodes communicate with each other by effected message parsing process and it requires a high-speed interconnected network placed between the nodes to reduce the latency and increase the throughput. These high-speed networks are usually constructed with InfiniBand communication standards which can be configured statically or dynamically. There are many typologies that can be used (e.g. linear array, ring, star, tree, nearest-neighbor mesh, systolic array, completely connected, 2D Grid, 3D-cube, 4D-cube, etc).

The operations executed by CN usually generate a high quantity of data which are distributed and stored on specific nodes called SN. They are mainly composed of HDD devices through a RAID vector because it provides both integrity and reliability over the data. These nodes are part of the overall HPC architecture, specifically in the last layer, and compose a complex SS. These SS are composed of a set of structured and organized storage devices (e.g. Tape, HDD, SSD) which are followed by a PFS that is the responsible software for the logic management of the stored data. Therefore, HPC architecture is an interactive and closely coupled processing environment composed of many nodes targeting achieve the highest processing power. Figure 22 presents the architecture of the main elements presented into an HPC environment.

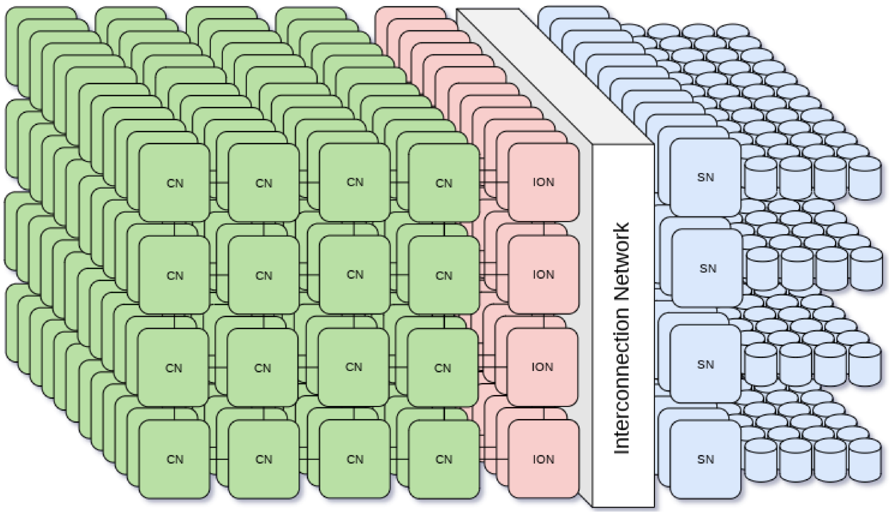


Figure 22 – HPC Storage Architecture

2.3.3 HPC Storage System

It is worth noticing that, the HPC environment can be summarized into different sets of nodes. CN, ION, and SN are elements that the most HPC environment presents in their architectures. The software that manages the data is usually the PFS. The data distribution process into SN is called file striping and is managed by the PFS. For instance, suppose a write operation request of 400 KiB solicited by an application that is running within the environment. The PFS client will receive a specified data request and divides it into fixed-size file fragments, for example, 64 KiB, to distribute them adopting some possible data distribution scheduling algorithms (e.g. round-robin, fixed priority, etc) across a set of data servers. In the future, when this data needs to be consumed by an application or resource, the PFS can be a single I/O request, serve the application request leveraging the aggregated throughput of the high-speed network interfaces and media devices to improve the data transfer rate. HPC storage stack that compose the SS is presented in Figure 23.

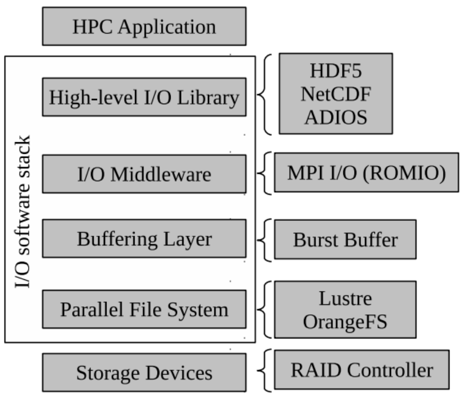


Figure 23 – Typical HPC storage stack. (HE; DAI; BAO, 2019)

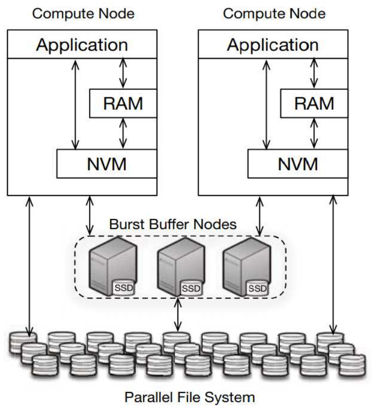


Figure 24 – Overview of Storage Hierarchy in HPC Systems. (LIANG; CHEN; AN, 2019)

The first layer is the HPC application layer which is attached to the I/O software stack. It includes high-level I/O libraries (e.g. HDF5(FOLK et al., 2011), NetCDF(LI et al., 2003), ADIOS(LOFSTEAD et al., 2008)), I/O Middleware (e.g. MPI I/O ROMIO(ROMIO. . . , 2020)), Buffering Layer (e.g. Burst Buffers(ALI et al., 2009)), PFS (e.g. Lustre(BRAAM; SCHWAN, 2002), PVFS(ROSS; THAKUR et al., 2000), Ceph(WEIL et al., 2006)) and finally a layer responsible to aggregate the storage devices (e.g. RAID Controller). These layers are composed of many parameters and have different configurations that affect directly the application's performance. Figure 24 presents the storage hierarchy commonly found in HPC environments where is possible to notice some media devices. Into the CN it worth noticing that there are two memory devices named RAM and NVM. This NVM device was already mentioned in Figure 21 as an SSD device. Of course, this NVM media device does not necessarily have to be an SSD but this device is widely used due to its price and high-throughput compared to the other NVM devices.

2.3.4 Data-Intensive Scalable Computing

Data-Intensive Scalable Computing (DISC) systems appeared due to the necessity to deal with a huge and massive amount of data. In most cases, these data need to be processed and organized through a system that could acquire, update, share, and archive datasets in an organized way. Although DISC systems came to fill requirements with an emphasis on data solution, it performs sophisticated computations over these captured data. The increased growth of the internet infrastructure led by companies such as Google, Yahoo, Facebook, and Amazon is an inspiration factor to DISC systems creation. These leader companies create new methods and technologies to solve and deal with their particular problems. These factors converge in a high-level system with goals such as scalability, fault-tolerance, availability, and cost-performance. Although these applications come from diverse scientific domains, they concern the role of data in their computation.

Bryan et al. (BRYANT, 2011) pointed out four key principles related to DISC systems, although some of them are directly related due to the hardware nature.

- Intrinsic data. The system has the duty to collect and maintain data instead of associate it with the users. As happens on most systems, it should update information processing them through background tasks. It also should implement reliability systems such as replication and error correction to ensure availability and integrity.
- High-level programming models Those systems usually employ programming models to process data consistently and independently. They are built considering parallelization and distributing concepts that do not are attached to specific hardware or machine.
- Interactive access The requirement for computing and storage should be independent and allow a variety o set up. Users also should be able to execute these programs interactively using abundant provided resources. The system should be able to return an input query quickly allowing computations in the background without losing systems performance.
- Scalable mechanisms ensuring reliability and availability. Indeed, DISC system work and consider data as the main element presented into it. Thus, these systems should provide reliability and availability mechanisms over-processed data to ensure data access when failures occur.

DISC systems also take advantage of PFS in their distributed nodes. Although the storage system is not disjoint from processing, it is usually managed by a PFS that manage the data providing high access ratio. In such a case, the storage device is attached directly to each individual server. It is also improved by external systems that ensure availability and reliability through replication over data. Concerning data processing, some big data frameworks such as Apache Spark and Hadoop are examples of tools to deal with DISC applications. These applications rely on powerful data processing operators such as Map, Reduce, Filters, etc (VALDURIEZ et al., 2018). In general, requirements such as data analysis and visualizations were not achieved by DISC systems because these environments were not designed for scientific applications.

2.3.5 DISC Storage System

Data-intensive computing facilities are projected to provide better performance and data management without losing cost-performance. The hardware infrastructure for DISC systems, although it varies depending on the objective adopted by the company or research institute, they presented common hardware aspects being labeled as WSC.

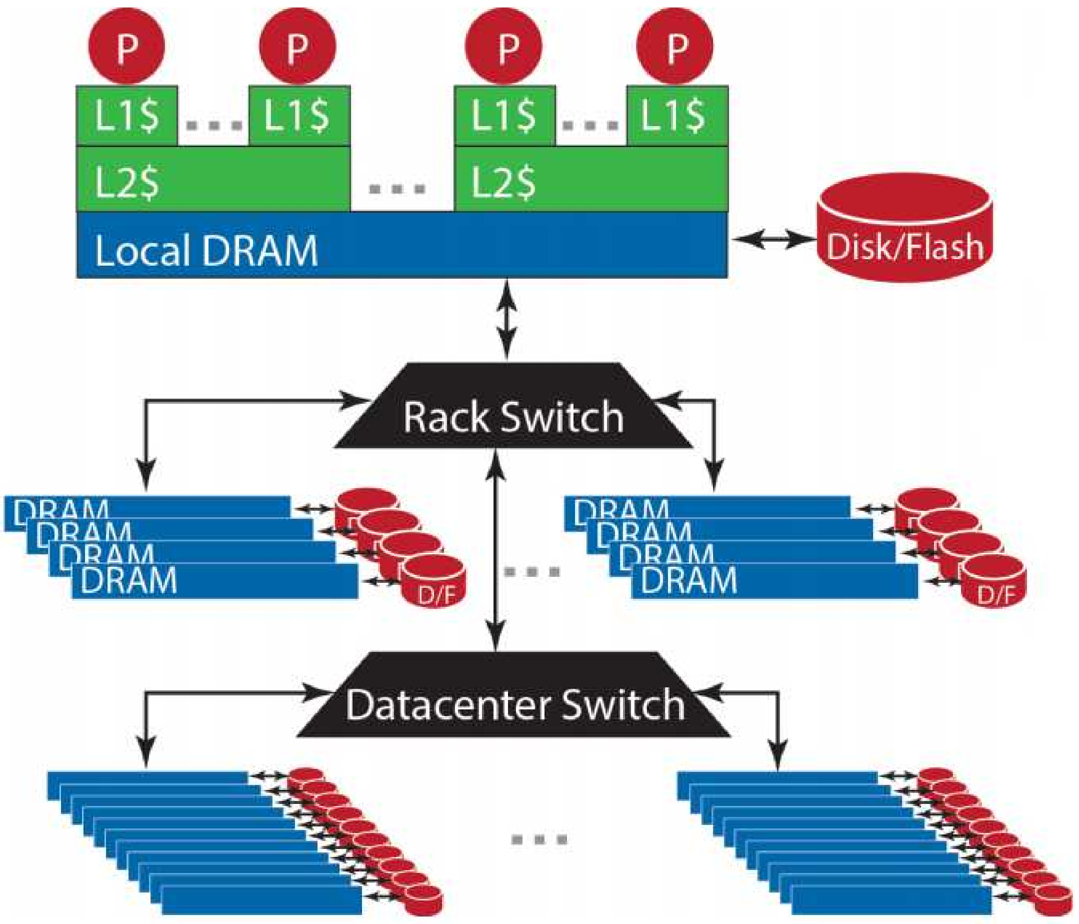


Figure 25 – Storage hierarchy of a WSC (BARROSO; CLIDARAS; HÖLZLE, 2013)

Figure 25 presents the storage hierarchy of WSC environment. It is mainly composed of servers grouped into racks creating a cluster. The layer above presents the structure of a server composed of a number of processor sockets particularly composed of microprocessors, each with its cache hierarchy and a local RAM memory distributed among the processing units. It is also attached to its magnetic devices and/or flash-based devices such as HDDs and SSDs. The connection between these servers is layered. Servers are connected within rack-level through 1-gigabit-per-second (Gbps) Ethernet switch, later, racks are connected to another cluster-level switch through 1 or 10 Gbps Ethernet switch Therefore, all servers are connected, each containing its memory hierarchy, its processing core, and its storage devices locally forming a clustered environment. The hardware present in this type of environment is not based only on performance, but on a price-performance ratio.

2.3.6 Cloud computing

Cloud computing can be classified into a model of services where computing is commoditized and delivered based on a service-level agreement between server providers and consumers. Usually, it employs a parallel and distributed system resource that allows users to access and configure many computing resources (e.g. servers, storage, processing, applications, networks, etc). This model allows users to access and use the required resources without worrying about where and how this resource was implemented. Thus, users or anyone that agree with the use terms and pay for this resource usage (e.g. business enterprises, government, etc) can access remotely the resource and applications.

Contracting a cloud resource reduces enterprises' costs compared with the in-place implementation providing easily and quickly resources to upgrade. Thus, providers such as Amazon, Google, IBM, Microsoft established data centers designated only for hosting cloud computing applications and resources ensuring high availability for data and applications access. The access to this resource is usually performed through a virtualization platform or hypervisor technologies such as Virtual Machines (VMs). This resource ensures corresponding access to the underlying hardware required isolating it from other hardware and applications. When we noticed these terms described in the author's papers we classify its domain as being a cloud domain.

2.4 FINAL CHAPTER CONSIDERATIONS

In this chapter, we presented the fundamentals necessary to the understanding of the subject that were discussing. We highlighted the existing gap between power processing and storage latency that increases the I/O bottleneck problem. Further, we presented some information about the storage hierarchy. We discussed some technologies that compose this hierarchy through a primary and secondary storage subdivision. Although the primary storage was mentioned superficially presenting some characteristics focusing on RAM, when discussing secondary storage, we provided a deep understanding of how the storage process occurs into a storage media. The media that composed the secondary storage were the magnetic tapes and discs with the addition of the optical discs and flash devices. Information about how the data is written into the chemical surface was also presented and discussed. Furthermore, we discussed high-performance storage environments and their characteristics. HPC, DISC Big data, Cloud Computing environments with the highlighting of the storage used in all of them was discussed and used as a field for further discussion.

3 SYSTEMATIC MAPPING

In this chapter, we present a tertiary study of the literature presenting how this model can help researchers and academy with interesting insights about the area development.

Systematic Literature Reviews (SLRs) and Mappings (SLMs) are secondary studies that address a research topic targeting answer related research field questions. It uses individual studies, or primary studies, as input information and aggregate results over the targeted research area for further researchers usage. To reach the desired level of understanding, SLRs try to identify, evaluate, and interpret all primary researches that are inserted on a delimited field scope, area, or phenomenon of interest. It is mainly used in medicine and health care disciplines and was adopted by other areas including psychology, economics, medicine, etc (KITCHENHAM; BUDGEN; BRERETON, 2011).

There are many advantages to performing secondary SLR and SLM studies. This type of research allows finding any possible gaps of specific topic research and direct further efforts. It is important because research gaps delay the development of a specific research field or technology reducing then the overall development ratio. Reducing these gaps might increase the research field's potential to achieve better results. It also brings the state of the art of the research field bringing and summarizing existing evidence related to the topic. Further, we also are able to verify which are the limitations presented in the research field, directing solutions to solve these limitations.

Evidence-based software engineering (EBSE) (KITCHENHAM; BUDGEN; BRERETON, 2011) targets to aggregate evidence on a specific research topic using secondary studies (e.g. SLRs and SLMs) as a methodology to aggregate empirical evidence. A systematic review is categorized as research methods aimed to identify, analyze, and interpret evidence related to a specific research question (WOHLIN et al., 2012). Mapping reviews use an equivalent methodology to identify and aggregate empirical results, differently, it does not discuss the relative merits of the presented research though. Kitchenhan et al. present an interesting research discussion about the importance of mapping studies in the software engineering field (BRERETON; KITCHENHAM, 2007). To present complete and satisfactory results, the research phases must be carried out scientifically and methodologically (WOHLIN et al., 2012). Kitchenham (KITCHENHAM, 2004) has adopted changes in these areas by formulating a systematic mapping approach for software engineering research issues. The proposed systematic mapping process consists of three phases: planning, conduction, and reporting of results (KITCHENHAM, 2004). This process methodology permits the audit of a study and improves its reliability. Therefore, this mapping was organized based on the main activities proposed by Kitchenham.

3.1 MAPPING PLANNING

There are at least five activities involved in the phase of planning (KEELE et al., 2007). However, we present here three main topics of this phase, namely: identification of the need for mapping, the specification of the research questions, and the development of a mapping protocol. Identifying the need for performing a systematic mapping is the first activity that the planning stage requests. Researchers are constantly updating their view of a subject to monitor its development. However, for new researchers and non-experts, understanding the overall development might be confused. Summarizing all this information without introducing bias is a common researcher requirement because it allows analyzing the phenomena getting general results that primary results do not present.

The research question specification is also an important activity of this planning phase. Every research seeks a solution to solve some issue, improve performance, or generate specific values. In general, the development of research might be justified by the necessity to deal with the subject. Specifying the research question is the most important part of any systematic mapping (KEELE et al., 2007). It is used as a guide that shapes the entire SLR research. Thus, research questions are the elements that identify the primary studies for further analysis.

The development of a mapping protocol in which the study will be supported can be understood as the act of putting the protocol into practice, specifies the used methods, and respecting the pre-established definitions. Among the criteria presented by Kitchenham (KITCHENHAM, 2004), for the development of the mapping protocol, we highlight some items that define the research process we considered in this mapping.

- Background.
- Research Questions Definition.
- Search Strategy for Primary Studies.
- Inclusion/Exclusion Selection Criteria Screening.
- Classifying the Papers.
- Data Extraction Strategy.
- Data synthesis.

The need for this research is based on the fact that, to date, to the best of our knowledge, no systematic mapping has been found addressing the presented topic. We can better define the objective of the present work as *"Classify and identify, systematically, the* *evidence, methods and approaches proposed by researchers that have been used to improve the I/O performance of storage environments."*

3.1.1 PICOC

Research questions should bounder the research scope. Petticrew and Roberts (PETTICREW; ROBERTS,) suggested considering an extended medical guideline, PICOC (Population, Intervention, Comparison, Outcome, Context), to construct the research questions. The elements that compose this guideline are explored below: Population refers to the group of elements that we are investigating and it is from the interest of the study. Intervention refers to the element that addresses the study. Make the question "which element is under the study?" will provide the intervention element. Comparison seeks to compare the intervention previously specified. Outcome refers to the obtained results including a practice point of view of them. Context delimits the context in which the intervention is delivered. To find and discover this information, this PICOC guideline will also be used to construct the search string. Below is presented our elements defined through the PICOC guideline.

Population (P): Storage Device and Storage System.

Intervention (I): I/O performance Improvements.

Comparison (C): -

Result (O): Strategy, Method, Approach, and Solutions.

Context (C): - HPC, Big Data, Cloud and Storage Systems.

In this research, we are interested in investigating the improvements that storage devices and systems received over the years. However, these devices and environments can be improved by many perspectives due to their development potential. We are particularly interested in investigating "I/O performance improvements". This is the element that addresses the study. In this research we are not considering other interventions to compare, thus, the comparison is null. Further, we are interested in getting all the methods, strategies, approaches, and solutions that the authors proposed. It also includes which devices were used, where they were implemented. It is possible to use and propose better I/O improvements in many research contexts. We are looking for an understanding in which context the intervention is being delivered. Included considered environments are (HPC, big data, cloud computing, and storage systems) We also are interested to understand and visualize in which environment these solutions were experimented with and evaluated.

3.1.2 Research Questions

Following the previous PICOC guidelines, we explore the bounded questions. Through the research questions, it is possible to identify the primary studies that will compose this research, data extraction, and analysis. According to Budgen and Brereton (BUDGEN; BRERETON, 2006), research questions should contain aspects that make them clear and narrow. To characterize this research, we presented a set of mapping questions that should delimit the scope studied.

3.1.3 Mapping Questions

MQ1: How many studies have been published over the years? The first question addresses the general evolution over the years of a specific research field. This question can be used to map the improvements realized by the researchers. We also verify how was its development, whether researchers are increasing or decreasing over the years. This question can be used to encourage new researchers on new topics of researches.

MQ2: Which publishing vehicles are the main targets for research production in the area? From the publication channels, it is possible to extract which researches are being carried out most frequently, thus enabling researchers to see more broadly the development of the area studied by the population and the academic community.

MQ3: Who are the most active authors in the area? We are interested in presenting the authors who contribute the most to the research area, thus offering a bibliographical analysis for new researchers who want to contribute to the presented area. This might help and encourage new researchers to interact with the researcher's study.

MQ4: Which are the author's relationship and how these authors interact with each other? MQ4 allows us to understand how authors are interacting with each other. How they network collaboration works and how they are located in their collaboration network.

MQ5: Which storage object are receiving more improvements targeting better I/O performance? Throughout this research question, we are able to see, in general, which class of improvements is being targeted by researchers. We can further analyze and verify which storage area has more necessity to be improved. This research question allows us to direct future I/O performance improvements to fill future development gaps.

MQ6: Are the authors considering energy consumption in their proposals? This question allows us to analyze and verify if authors are considering any power consumption method in their research when trying to improve I/O performance. We also provide a distribution according to each proposed characterization model. We want to analyze how power saving has been proposed by researchers. We can further analyze which class of improvements are considering energy savings at most, which of them needs to be boosted, and how that progress occurs according to the methodology method.

MQ7: Which devices were considered when proposing I/O improvements in storage environments? This research question allows us to have an overall overview of the most used devices in the authors' approaches that targets I/O performance improvements. Through this question, we can direct future efforts to improve I/O performance to specific storage devices.

MQ8: How those considered devices are distributed according to each classification model? This question allows us to visualize which devices are being used to improve I/O performance according to each perspective.

MQ9: How authors improvements are evaluated and which tool they used in their experimentation? This question allows us to visualize how authors evaluate their proposed method, which mechanism they are using, and how these mechanisms related to the specific research field were studied here.

MQ10: How hardware devices are being proposed by researchers targeting better I/O performance? (H2H-IO) MQ10 allows us to check which hardware devices are being proposed by authors to improve I/O performance. We can also verify in which class of hardware this other proposed hardware is improving.

MQ11: Which Software Class Hardware are Improving and which are this proposed hardware?(H2S) This question allows us to check which software is being boosted when an author proposes hardware to improve its performance. We also verify in which class the software beholds.

MQ12: Which kind of environment-class is receiving researchers attention when the hardware is proposed as a solution to improve I/O performance on a storage system? Which devices they are using at most? (H2SS) This question allows us to understand how the experimentation is being performed according to the environment that it is performed. It allows us to visualize how bigger is the storage environments where hardware is proposed to improve its I/O performance.

MQ13: Which software class is being improved by researchers when a software solution is proposed and where are those software improvements being implemented? (S2S-IO) This question allows us to visualize which class of software is being improved by researchers when the proposed solution by the author is a software object that targets I/O improvements. It also allows us to visualize where those proposed objects are being implemented. We verify that although some solutions improve software I/O performance, there are cases when the object is implemented in other software objects instead of the same improved software.

MQ14: Which Hardware class are receiving more software I/O improvements, where are those software solutions being implemented, and which device class researchers are using to perform the experiments? (S2H-IO). Through this research question, we show which device is receiving most authors attention when the I/O improvement is target and where they are being implemented and which class these implemented elements belong. It also allows us to understand how authors solutions are being evaluated, which environment they are using to evaluate their solution and which software is used to perform its evaluation.

MQ15: How researchers are evaluating their software solution that targets I/O improvements on storage systems, where are those software solutions being implemented, and what is the size of the user environments on the experimentation? (S2SS-IO) This question shows us where the solutions to improve I/O performance on storage systems are evaluated. As verified by previous analyzes, we also present where the author's solutions to improve I/O performance on storage systems were implemented. We are able to verify whether the evaluated environment is bigger, small, or software simulated. It also allows us to identify which large-scale environment was most used by researchers to evaluate their solution. Through this question, we verify that general storage systems and cloud storage systems environments keep getting researchers manageability. Although some solution targets storage systems, we verify that some authors evaluate their solutions in simulation software. At this point, we verify which were these software and which were this usability frequency. Finally, we also verify what are the size of small environments used by authors to evaluate their software solution. We notice that, although the proposed object has the potential to improve large-scale systems, some authors evaluate their solution in small environments.

MQ16: How are architectures proposed by researchers being built? In which context they are proposed? Where are those solutions being implemented? How is the size of the environments that evaluate architectures? This question shows us where the new architectures solutions are evaluated. We are able to verify the class of these architectures and their size. We also identify which large-scale environment was most used by researchers to evaluate their architecture solution. We verify what are the size of small, large-scale, and software-simulated environments are used by authors to evaluate their architecture solution. As happened in the storage system class, here, we verify that some authors also evaluate their solution in simulation software.

3.1.4 Inclusion/Exclusion Criteria

Inclusion and exclusion criteria target selecting research studies that fit the proposed research questions. To obtain an adjustment and a better performance in the evaluation of the studies found, the inclusion and exclusion criteria that were used to obtain the primary studies are presented below. After identifying the primary studies, the studies were subjected to the inclusion and exclusion criteria listed below:

Inclusion Criteria:

IC1: Articles dealing with solutions regarding I/O performance improvement on storage environments AND

IC2: Articles published in English AND

IC3: Articles that necessarily have a title and abstract AND

IC4: Articles published at peer-review events, such as workshop OR conference OR magazine.

IC5: Articles published from 2009 to 2019

IC6: Articles that consider some storage devices.

IC7: Articles that concerns storage devices.

Exclusion Criteria:

EC1: Articles that do not treat solutions regarding I/O performance improvement on storage environments OR

EC2: Articles that were not published in English OR

EC3: Articles that have no title OR abstract OR

EC4: Non-peer-reviewed papers OR

EC5: Articles published before 2009 and after 2019.

EC6: Articles that do not concern about storage devices.

EC7: Articles that that do not concern about storage devices.

3.1.5 Sources

The database selection process followed some criteria as described below:

- Ability to perform a search with logical characters.
- Ability to search both the body of the document and metadata such as title, keywords, and summary.
- Repositories that are within easy reach.
- Repositories containing Computer Science and Engineering content.

Six repositories were chosen that meet the above criteria.

Table 4 – Selected Sources

Sources	URL
ACM Digital Library	https://dl.acm.org/
EI Compendex	www.engineeringvillage.com
IEEExplore	ieeexplore.ieee.org
ScienceDirect	www.sciencedirect.com
Scopus	www.scopus.com
Springer	www.springer.com/

3.2 STRING SEARCH

The query string construction followed the PICOC guideline presented in item 2.1.1. However, for selecting the used terms, we presented a used methodology when choosing the string elements. Trying to find the papers that propose I/O improvements for storage devices and systems, the first element of the string is directly related to the Population presented on PICOC. The second line is a specification of possible devices that could be used to receive I/O improvements. It is worth noticing that, these device terms were carefully chosen. The search string was generated by joining the elements through AND/OR logic connectors. As presented below, some similar synonyms and terms have been added, thus aiming at greater effectiveness in returning primary searches.

Storage environments are composed mainly of devices Hard Disk Drives (HDDs), Solid-State Drives (SSDs), and Tapes. Trying to cover these storage media, the first element in the second line of the string "flash" was chosen to consider the possibility to find flash devices and flash storage systems. During manually papers searching, we verified that storage environments that employs SSDs, or flash chips are constantly referred to as flash storage systems or even flash storage (WANG; DONG; MING, 2015), (HUANG et al., 2017), (PETERSEN; BENT, 2017), (YANG et al., 2017a), (ZHAO et al., 2018) and (MAO et al., 2018) for instance.

On the other hand, improvements on devices also present the flash storage, or flash-based storage term were found (YOU et al., 2019a), (PARK et al., 2018), (KWON; KANG; EOM, 2017) and (JI et al., 2017) for instance. The closer to processing the device is, the more quickly it will be assessed. Normally, these subsets of quickly access devices receive the name cache and memory. It is also characterized to be volatile, losing all its content when power is turned off. For this reason, for covering all memory types (e.g. SRAM, DRAM, Non-Volatile Memory (NVM), etc), we chose the memory term to be inserted on the specification line of the string. We advocate that this term is capable to cover quiet types of memory technologies.

The Disk term refers also to a magnetic disk, floppy disk, and optical disk. This term was chosen because with just one term we referred to at least three device types. The last term magnetic also relates to magnetic disk and magnetic tape. The third line refers to the intervention element. I/O latency and I/O throughput are synonyms that were considered because some researchers express I/O performance correlating it with throughput and latency.

The fourth line refers to the outcomes that delimit the context in with I/O performance, in other words, the intervention, is delivered. All elements presented in the fourth line are non-specific terms such as (algorithms, specific tools, specific frameworks, or techniques). These terms are able to relate even software solutions and hardware solutions. In section 3.4 we provide more information explaining why it is important that these terms have to be generic instead of specific.

Finally, we provide the terms that refer to the context where we are looking to answer. HPC, Big Data, Cloud, and Storage Systems are usually a huge environment with high-performance computing and storage. For this reason, we selected these terms that justify our interest in huge storage.

The final search string was described as presented in Table 5

Table 5 – Used Search String on the Research

String Search

("storage device"OR"storage system")
AND("flash"OR"memory"OR"disk"OR"magnetic")
AND("I/O performance"OR"I/O latency"OR"I/O throughput")
AND("strateg*"OR"solution"OR"method"OR"approach"OR"scheme")
AND("HPC"OR"big data"OR"cloud"OR"storage system")

We evaluated the search string using specific papers that control the results. This process ensures that the search string is returning the papers that compose the basis of the study without ranging out of the scope. The desired papers appeared in the results generating evidence about the search string correctness.

3.3 MAPPING CONDUCTION

The most important step of search identification can be simplified by executing the database search string (WOHLIN et al., 2012). However, there are at least five steps related to this phase, and they are presented below: (KEELE et al., 2007):

- The research identification
- Selection of primary studies
- Study quality assessment
- Data extraction and monitoring
- Data synthesis

A factor that differs SLR from other research methods is the rigor of applying and conducting the search process. The purpose of this execution is to ensure that the return of primary studies is following the terms previously described in PICOC. To cover the entire primary search, the search string must be executed from the databases described above. Consequently, duplicate articles may return because they can be indexed by different libraries. These papers must be identified and deleted.

To assist in this process, we used parsif.al [1], an open-source tool that assists in the management and execution of the research conduction process. Through this, it was possible to detect and remove duplicate articles automatically. Besides the removal of duplicate articles, the tool helped in the classification of the articles, allowing them to be selected or not for the next steps to map. To classify these articles, we inserted all the insertion and exclusion criteria in the Parsifal tool and, according to the article, we classified the researches.

Figure 26 shows that the process was conducted throughout seven steps. After constructing the string search, we executed them into the libraries shown in section 3.1.5.

Stage one (S1) represents the quantity of returned primary studies. The total amount of returned researches represented on the S1 stage was 2613. Table 6 presents the total of returned papers according to each source library. We notice that IEEEXplorer and Springer were the libraries that returned the greatest number of articles. Together, they were responsible for covering 65.98 % of the total returned papers.

After executing the string on the libraries, we selected for further steps only the articles that were presented in peer-reviewed channels. Stage S2 represents this step. Non-peer-reviewed channels (e.g. Books Chapters, Book Reviews, Editorials, Mini review,

[1] https://parsif.al/

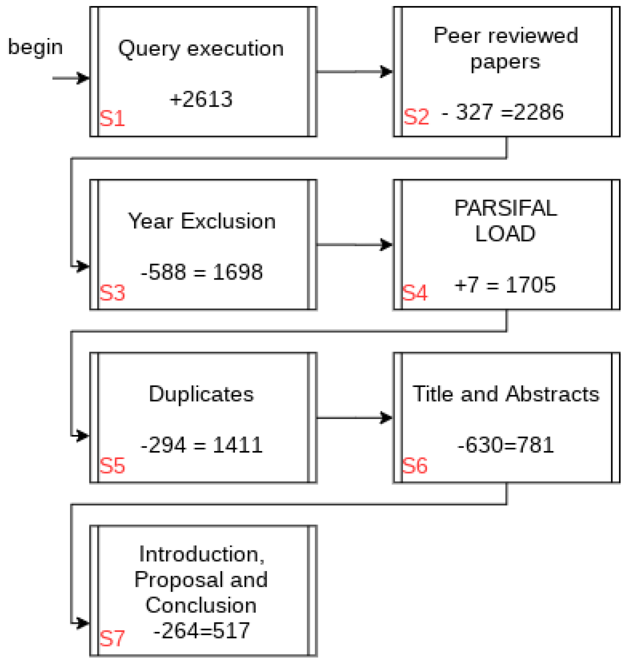


Figure 26 – Data Process Extraction

Info, Other, etc) corresponded to 327 files, and they were removed from the further steps due to the EC4. After selecting only the peer-reviewed channels papers, we delimited a range of 10 years for our analysis.

Storage devices and technologies evolve at a high speed. For instance, SSD devices that are composed of flash memories were firstly composed by Single-level cell (SLC) memory cells. It means that each cell stores one bit of data. However, the storage approach for flash memory cells evolved quickly over the years. Multi-level cell (MLC) is capable of storing two bits/cell, Triple-level cell (TLC) stores three bits/cell, Quad-level cell (QLC) stores four bits/cell, and finally Penta-level cell (PLC) stores five-bit/cell. Limiting the time to a range of ten years will provide us interesting improvements on devices while reduces the amount of paper that is not part of the scope.

S3 stage shows that 588 papers were removed because they do not belong to this time-bound EC6.

Table 6 – Articles found per source

Source	Articles Found
ACM Digital Library	176
IEEEXplorer	920
Science@Direct	431
Scopus	210
Springer	804
EI Compendex	72

In stage S4, the *BibTeX* files were inserted into the Parsifal tool. It is important noticing that, when inserting the values into the tool, we verified that 7 additional papers that were not counted when we executed the query string had arisen. These non-additional papers were duplicated papers or papers that had some string special character such as "&" for instance. These additional papers were all removed on the duplicates stage S5 together with all the duplicated papers.

In stage S5, we present the amount of total duplicated papers that resulted in 294 papers. All these 294 papers were removed.

In stage S6, we began reading all titles and abstracts of the overall quantity of papers. Here we applied the IC/EC specified in section 2.1.5. In this stage, we also characterized each paper according to the characterization model that we will discuss in section 3.4. Through the EC1, EC3 and EC7 were removed 630 papers.

Finally, in stage S7, the articles that we could not classify by reading just titles and abstracts we had to take more reading to decide whether they covered or not our scope. To verify, we read for each paper the introduction, proposal, and conclusion finalizing the overall data process extraction. We finalized the data extraction process with a total of 517 accepted papers.

3.4 I/O CHARACTERIZATION FRAMEWORK

Classifying the articles following a methodology helps us to reduce the possibility of researcher bias. For this reason, it is recommended to create a protocol to select individual studies and classify them independently. We used our characterization model presented by Figure 57 to classify all the selected papers (PIOLI; MENEZES; DANTAS, 2019). However, we present and discuss how this step was performed. In the data extraction process presented before, while reading the titles and abstracts in step six, we found the elements that satisfy this framework. When researchers propose solutions to improve I/O performance in storage environments, we verified that the proposed solutions belong to a specific domain. Each edge presented on the Figure 57 represent one characterization proposal.

3.4.1 3WPIT

After classifying all accepted papers, we created a second framework phase for further data extraction and classification. Considering our research question presented in subsection 2.1.2 and 2.1.3 we applied the 3WPIT classification.

- **W**hat - was **P**roposed
- **W**here - it was **I**mplemented
- **W**hich - **T**echnology was used.

For researcher verification and audit, all files that were part of this research can be found in GitHub [2]. There, researchers can find the "articles.xls" file that contains all classified papers and the reason why the rejected papers were not included. In this file, in the "comments" column, it is possible to verify the extracted data related to each paper. All brackets represent extracted data from the read paper that is directly related to the 3WPIT characterization. Although each classification has its proper classification elements due to the different research question, the basic elements and the meaning of its content is presented below:

[Proposed element]-[Improved/Implemented element]-[Used technology]

[Article theme]-[Power Concerning]-[Evaluation method (benchmarking)]

Depending on the domain of the paper that is receiving the improvement, the second bracket of the first line will follow the rule presented below.

- **S2S-IO** - [(Improved Class)[Implemented place]]
- **S2H-IO** - [(Improved Device)[Implemented place][Used device]]
- **S2SS-IO**- [(Domain)[Implemented place][Evaluated Environment]
- **H2H-IO** - [[Improved Place]]
- **H2S-IO** - [Improved Object Class(Improved Object)]
- **H2SS-IO**- [Improved Class[Implemented place][Evaluated Environment]]

We recommend researchers to use this classification when publishing their paper. Although we classified the paper for this research, we strongly believe that this data can be provided more accurately by the author of the research. Therefore, we recommend that as the researchers find this article, they can make the necessary correction if there is a quirk in the object classification process.

3.5 SYSTEMATIC MAPPING REPORT

In total, 2613 articles were returned whose distribution can be verified in Table 6. Of those 2613 articles found, 327 were excluded because they met the EC4 exclusion criteria. After that, we applied EC6 and exclude 588 papers which gave us a total amount of 1698 papers. Figure 27 represents the actual scenarios of stage S4 presenting the number of loaded articles according to its source extraction.

[2] https://github.com/laerciopioli/Systematic-Literature-Data.git

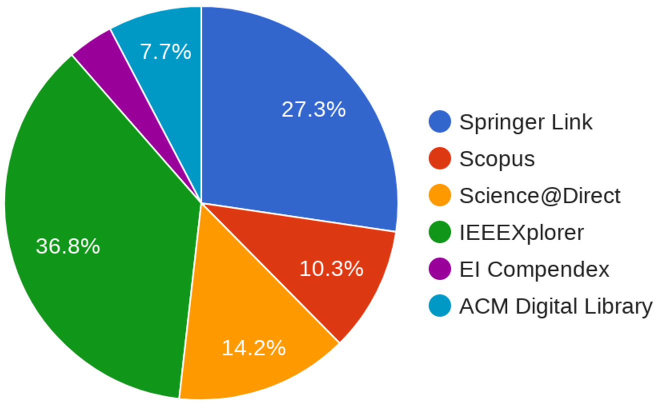


Figure 27 – Proportion of Loaded Papers in Parsifal by Source

IEEEXplorer and Springer had the highest proportion of loaded papers in the Parsifal system. Together, they imported more than half of all papers to be analyzed. From those 27,3% evaluated papers which correspond to 466 documents of springer library, 109 were accepted. 175 researches were represented by Scopus library with 10,3% papers. From those, 29 papers were accepted. Science Direct had 242 evaluated papers corresponding to 14,2% with an acceptance of 45 articles. IEEEXplorer had the highest representation with 627 papers that corresponds to 36,8%. From those, 256 were accepted. The total evaluated papers from EI Compendex were 64 documents corresponding to 3,8%. From this total, 24 were accepted. Finally, ACM Digital Library represented 7,7% which corresponds to 131 files from the evaluated papers which a total of 65 accepted papers. The total of selected papers distributed over the libraries can be visualized in Figure 28.

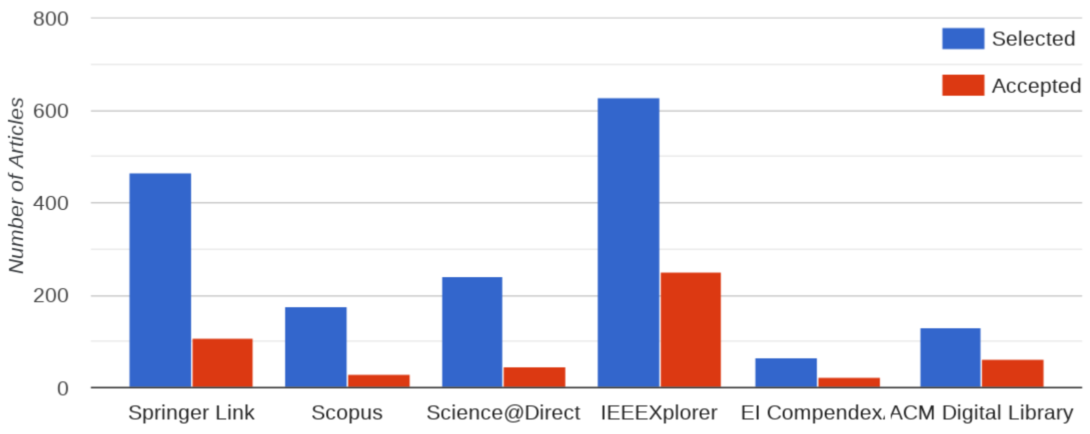


Figure 28 – Accepted and Rejected Papers by Source

3.5.1 Mapping Report

The 517 papers were analyzed and classified to answer the mapping questions (MQ). Figure 29 addresses the *MQ1(How many studies have been published over the years?)*

question, plotting the number of papers published in each year. We noticed that there was an interesting increment most of the time.

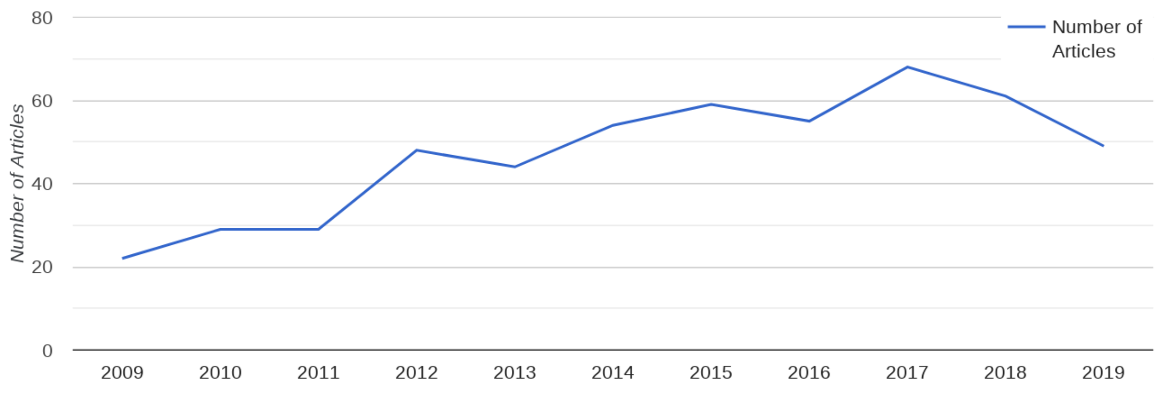


Figure 29 – Accepted Papers by Year

One fact that can justify this increment in I/O performance publications is the emerging and popularization of flash memories and SSDs. Although flash memories emerged in the 1980s, it had not the storage usage as HDDs until the 2000s. There are many ways to improve SSDs performance. To begin these possible improvements understanding, knowing their basic structure might help.

SSDs are non-mechanical storage devices, composed mainly of flash memories. Into this storage device, there are many hardware and software components designated to correct and improve its performance. Its basic storage structure is composed of NAND flash packages, each of them composed of flash dies. Each die contains blocks and each of them contains a set of pages. This hardware structure imposes some limitations such as I/O request scheduling, wear-leveling, garbage collection among others when performing I/O operations (e.g. read, writes, etc).

To overcome these issues and improve the flash structure, a SSD usually employs a software layer called FTL which runs several firmware algorithms. As shown on the other questions, we verified that most of these I/O improvements are directly related to SSD devices. We interpret this result as the subject has grown in relevance to the academic community over the last years.

MQ2 (Which publishing vehicles are the main targets for research production in the area?) is addressed by the Figure 30. We notice that the IEEEXplorer indexed a huge quantity of papers in the field studied here when performing the search string. It corresponds to a total of 48,5% of the accepted papers. IEEEXplorer is followed by springer with 20,6% of the accepted papers. Together, these two libraries correspond to almost 70% of the total accepted papers here presented. ACM, Science@Direct, Scopus, and EI Compendex complete this analysis respectively.

MQ3: Who are the most active authors in the area? This research question identifies

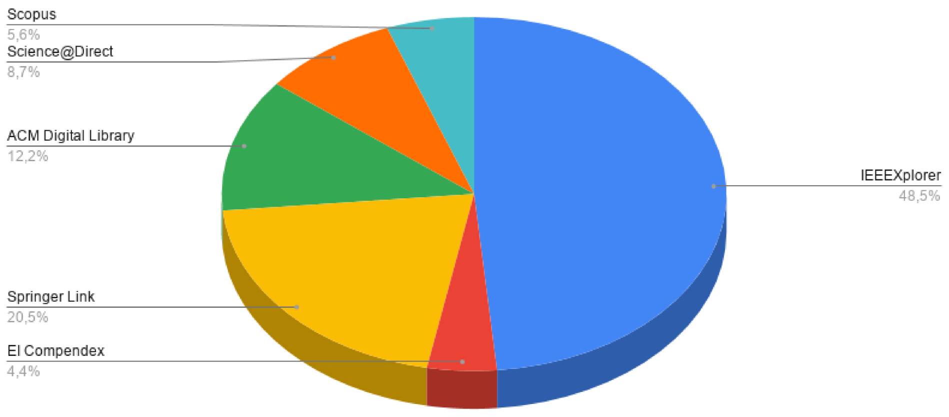


Figure 30 – Accepted Papers by Source

which researchers contributed to the development of the field of study. Figure 32 presents information about researchers publications. It presents the relation between the number of authors and their publications. The y-axis refers to the number of authors and the x-axis refers to the number of publications made by the author. In total, we identify 2235 authors that had their papers accepted. This number corresponds to the sum of the total occurrences presented in Figure 31. We verify in the data process extraction that in this research were accepted 517 papers that followed the research insertion criteria. Dividing the number of authors by the number of accepted papers we verify that the average of the authors per paper was 4.32. It means that by average, each paper would be published with four authors. We also noticed that 1390 authors published once, while 168 of them published twice and 59 authors published 3 times each. The total author's distribution can be visualized in Figure 31. We also noticed that the research publishing structure follow the power-law characteristic when analyzing the shared works. It means that the more papers the author published, the smaller the number of authors who published at the same rate. This confirmation can be analyzed in Table 7.

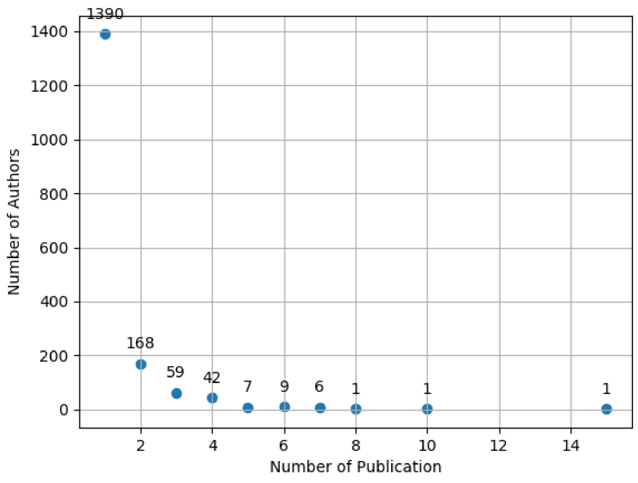


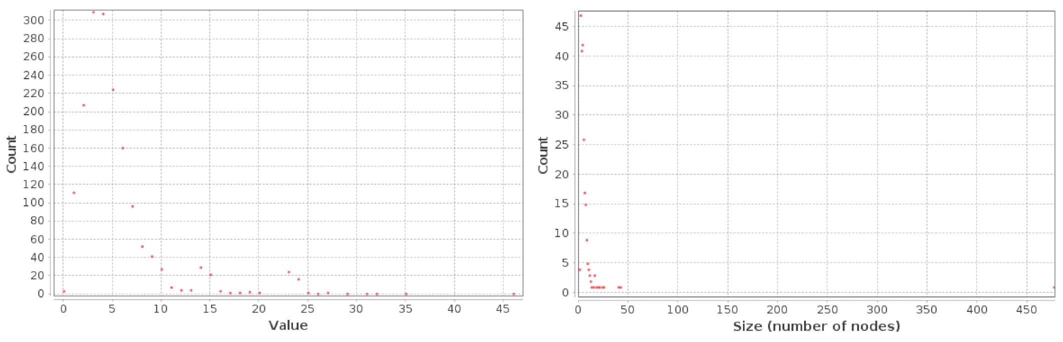
Figure 31 – Publishing Distribution

Authors	Articles Quantity
X. Liu	7
Y. Kim	7
Qin, Xiao	7
J. Lee	7
H. Kim	7
C. Wu	7
X. Zhang	8
J. Kim	10
Y. Wang	15

Table 7 – Most Published Authors

Table 7 shows the authors who had the most publications. Y. Wang was the author who published the most quantity of articles found in this research, he enrolled in 15 accepted papers. When executing the string search, the results showed that he was enrolled in 29 studies of researches. However, when applying the IC and EC, the total accepted papers were just 15 of them. He is followed by J.Kim with 10 and X. Zhang with 8 accepted publications. Initially, J. Kim was enrolled in 17 studies and X.Zhang was enrolled in 18. After these three researchers, there are more than 6 authors that published 7 papers each.

MQ4 *(Which are the author's relationship and how these authors interact with each other?)* is included in other to understand how authors are interacting with each other. The total of connected components was 228, and its distribution can be visualized in Figure 32. Figure 32a presents the degree distribution of all authors that compose the co-authorship network while Figure 32b presents the size distribution of the connected components of the network. We noticed in Figure 32a that most authors have few connections with other authors. It means that, in this network, the selected papers have few authors in common because an author is connected at least with the authors that published with him in the same work. The co-working researchers' size has a peak average of three and four researchers. After this number, as much as the number of authors connections begins to increase in a research group, the quantity of researchers begins to decrease regarded some exceptions (e.g. value 14,15,23 and 24).



(a) Degree Distribution (b) Components Distribution

Figure 32 – Networks Distribution

This indicates that some researchers publish papers with not only one group, instead, but they also publish works with many other groups. Thus, in the co-working network, these researchers will have connections with many other researchers and their degree will increase.

In Figure 32b we analyzed the size of the connected components that belong to the network. We verify that many connected components are composed by a few researchers. The number of small groups increases until 47 and then this number begins to decrease. It relates the number of groups with the number of authors that belong to its groups.

Although some points run out of the trend, we observed that as more authors belong to a group, smaller is the occurrence of these groups. This reinforces our argument that the majority of authors who published the papers in this network comprise a small research group, thus, forming then small groups. The bigger quantity of researchers is enrolled in groups with less than five researchers.

However, there is a huge component that is composed of 477 and 1667 edges. These numbers correspond to 28.34% of the total nodes and 36.95% of the total edges. Targeting understanding how this network works and how authors are exposed to it, Figure 33 presents this component. In this network, the degree of each author is expressed by the size of the node. The degree and its connection indicate the author's relation when publishing articles with other researchers.

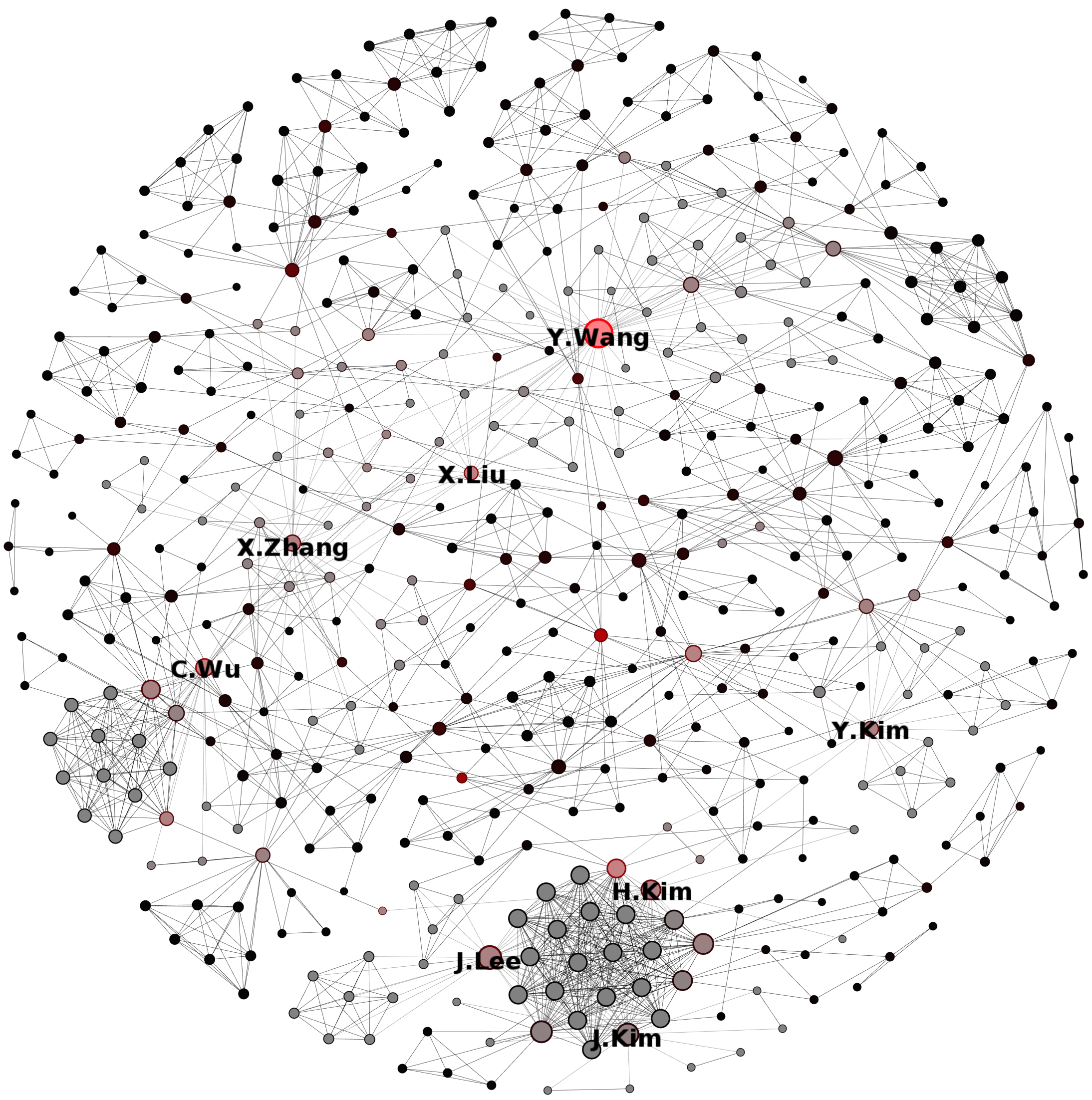


Figure 33 – Largest Connected Component Network

Of the nine authors that published more articles in Table 7, eight of them are found in this connected component. The author Qin. Xiao was identified as belonging to another component instead of the bigger one. The color scale varying from black to red symbolizes their betweenness centrality level. The more red is the node, the higher is its betweenness value and the more black is the node, the smaller is its value.

We also identified what are the betweenness of these authors. Table 8 presents the betweenness value for all authors presented on Table 7. Y. Wang was the author that published the higher quantity of paper that concerned with this research, and he also has presented the higher betweenness centrality value in this network. We verify this fact by looking for the column "overall position" presented in Table 8. This column compares the betweenness centrality value from this author with all other authors and its place is presented in this column. Although J. Kim had published 10 articles, he presented the forty-eighth position of the greatest betweenness. Though X. Zhang had published two papers less than J. Kim, his betweenness was forty positions better than J. Kim. C. Wu took the same position both in the number of published researchers and his betweenness value. These authors seem to place an interesting place in this area of study and had contributed to many studies and improvements. Qin Xiao presented the lowest betweenness shown in Table 8. This happened because when we analyzed the other connected components of the network, we confirmed that it did not take a place in the largest connected component. This is a factor that reduces the possibility of node connections and, in this case, implied in the smaller betweenness value among the authors presented in Table 8.

It is also possible to verify in Figure 33 that the authors H. Kim, J. Kim, and J. Lee seem to work together in some moment. We verify that they composed a co-authorship group at the bottom of the network. We also verify that J. Lee acted as an intermediate for another researcher group that is placed from his left.

Table 8 – Authors Betweenness

Author	Betweenness	Overall Position
X. Liu	0.009500	6
Y. Kim	0.006593	11
Qin. Xiao	0.000431	146
J. Lee	0.006222	14
H. Kim	0.006731	9
C. Wu	0.010750	4
X. Zhang	0.007108	8
J. Kim	0.002380	48
Y. Wang	0.018570	1

The next mapping questions here presented were planned to give a narrow perspective of how researchers are proposing solutions to improve I/O performance considering storage devices and environments. We presented these questions results classifying them according to our presented framework in section 3.4. This methodology was presented to cluster-specific points and elements considering hardware and software characteristics.

MQ5 *(Which storage object are receiving more improvements targeting better I/O performance?)* is addressed by Figure 34 and allows us to visualize how researchers are fixing and proposing solutions for improving I/O performance. As presented on the framework used in this research, we classified the improvements as being software and hardware, and, this solution usually improves other software, hardware, or storage systems. When a paper proposes both a software and a hardware solution for a storage system, these solutions were classified as an architecture solution just to summarize the concept of proposing both software and hardware solutions jointly.

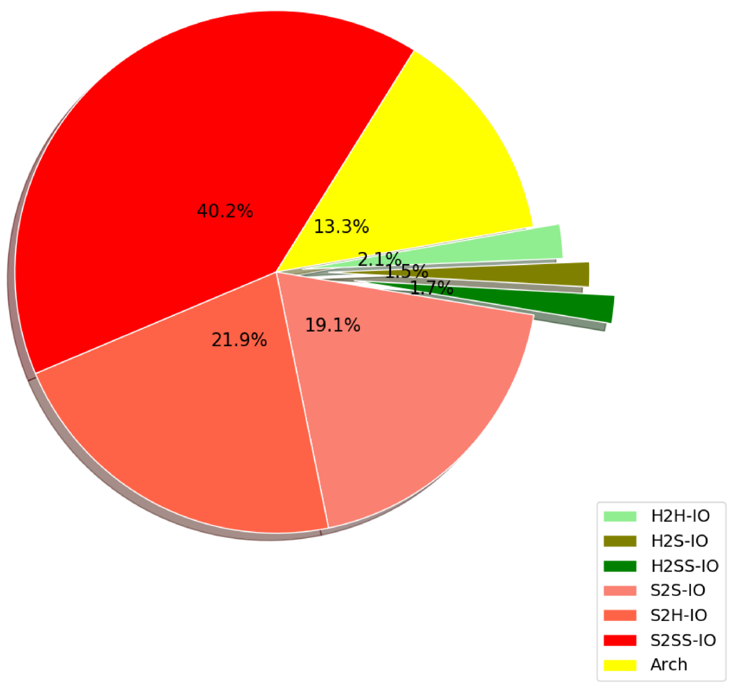


Figure 34 – Accepted Papers According to the Selection Criteria

Analyzing the researcher's proposals, we noticed that the big majority of researchers' proposals are addressed by a software solution. Software solutions correspond to 17,5 % of the overall proposals. These solutions are usually new approaches or methodologies that try to improve a software I/O performance such as new algorithms for file systems, databases, operational systems, or even applications inherent to a specific research field. We provide more details in further analysis specifying which were these improved and proposed software objects.

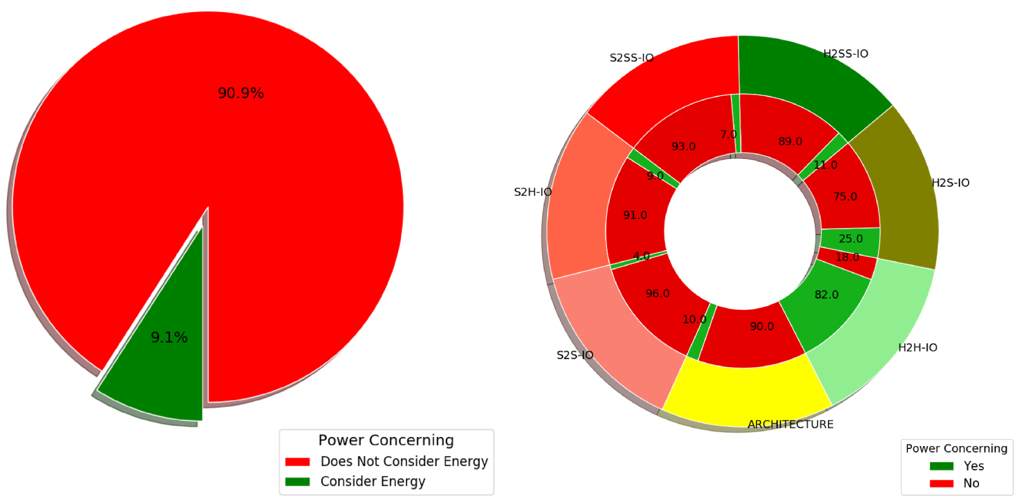
We also noticed that solutions that target better device performance represent 21,4% of the total proposed solutions. This class of improvements deals with solutions for better performance of storage devices. For instance, news or hybrids FTL algorithms for SSDs are solutions that belong to this specific group. 39,8% of the software solutions targets storage systems. We considered storage systems here any improved device that is composed of software and hardware and has many functions. For instance, when researchers propose a specific software for a huge burst buffer management or even software for an experimental small testbed. The idea behind this class is to present a subgroup that is composed of many sub-blocks of both hardware and software. Different from software improvements on software, software improvements on storage systems require the concerning of a chip, board, or even a high-performance environment.

Hardware improvements represent a small portion of the proposal's solutions. H2H-IO, H2S-IO, and H2SS-IO combined represent 5,7% of the overall improvements. Indeed, proposing hardware solutions requires a specific and propitious laboratory environment to evaluate these proposals. For instance, adding an SRAM into a specific board requires a different knowledge class due to the necessity of dealing with electrical and palpable hardware materials. Due to this reason, these proposals represent a small rate of the total proposals' evaluation.

Hardware improvements are rarer because the test phase of implementing or introducing new board chips requires the necessity to have these boards which usually are inappropriate and expensive for personal research. Different from software proposals, hardware might impact in cost to prepare the specific environment for testing and evaluations. We noticed that suggestions considering hardware improvements need to be improved. We can use this result to direct future I/O performance improvements to fill future hardware development gaps.

Although MQ6 *(Are the authors considering energy consumption in their proposals?)* is not directly related to I/O improvements, we think it is important to provide information about how the energy is being used and spend in storage environments and systems. MQ6 answer is addressed in Figure 35 that provide us information about the power concerning. Figure 35a shows whether the authors are considering power saving in their solutions while Figure 35b distributes this power concerning throughout the classification model.

We noticed from the total evaluated and accepted papers that 97,3% of them considered directly or indirectly energy saving in their research. It is interesting information because considering sustainable assessments is becoming indispensable in computing systems. We also noticed in Figure 35b the power concerning distribution by each improved area. As expected S2S-IO improved 3.0% of the analyzed papers. It makes sense because when researchers propose I/O improvements, usually they are interested in analyzing I/O throughput and latency. The access performance in such cases seems to be more recurrent than power consumption. When the proposed solution is directed



(a) Overall Power Consideration (b) Class Energy Distribution

Figure 35 – Research that Considers Energy Savings

for a hardware device this ratio increased three times than for S2S-IO. In such a case, researchers are dealing directly with storage devices and this increment is plausible from the research point of view.

Not far from the last one, software improvements that target storage systems, it is S2SS-IO, also presented similar results due to the device usability. All those results were smaller compared with solutions that considered hardware devices. In H2SS-IO 10% of the selected papers considered power in their solutions. In H2S-IO this number increased 2.5 times and in H2H-IO this number was even bigger.

Although hardware solutions presented bigger proportions, we should pay attention that the overall number of accepted papers that consider hardware were smaller compared to the software solutions. This result shows us that hardware improvements are more likely to consider energy rather than affirming that more hardware improvements consider power than software.

MQ7 *(Which devices were considered when proposing I/O improvements on storage environments?)* was included to investigate which are the devices that researchers use in their storage environment when proposing I/O solutions. To answer this question we identified which storage devices the papers were used to evaluate and propose solutions. Although computing systems are composed of a different hierarchy of storage devices, in this study we identified just the devices that researchers had some focus in their research. It is important noticing that, these devices are not improved devices, they are just the used media in the researches.

Figure 36 presents the considered devices when proposing I/O improvements in storage environments Indeed, SSDs and HDDs are storage devices that lead to this usability

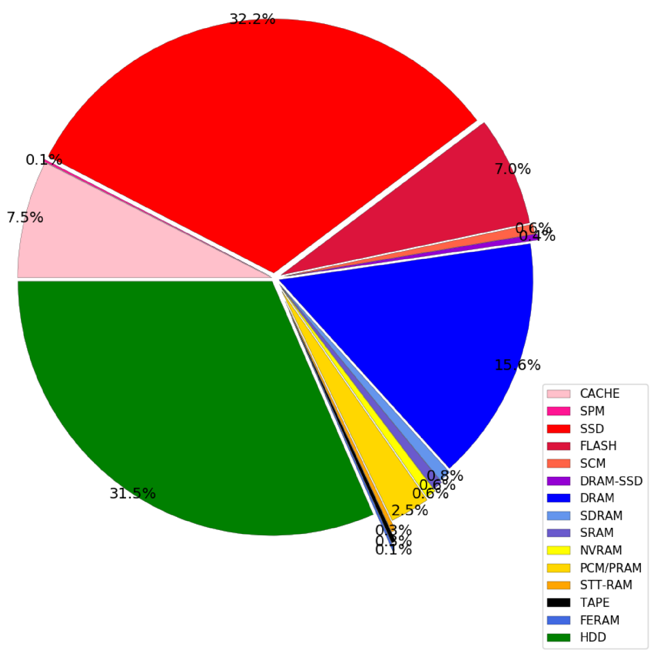


Figure 36 – Considered Devices

when I/O is targeted.

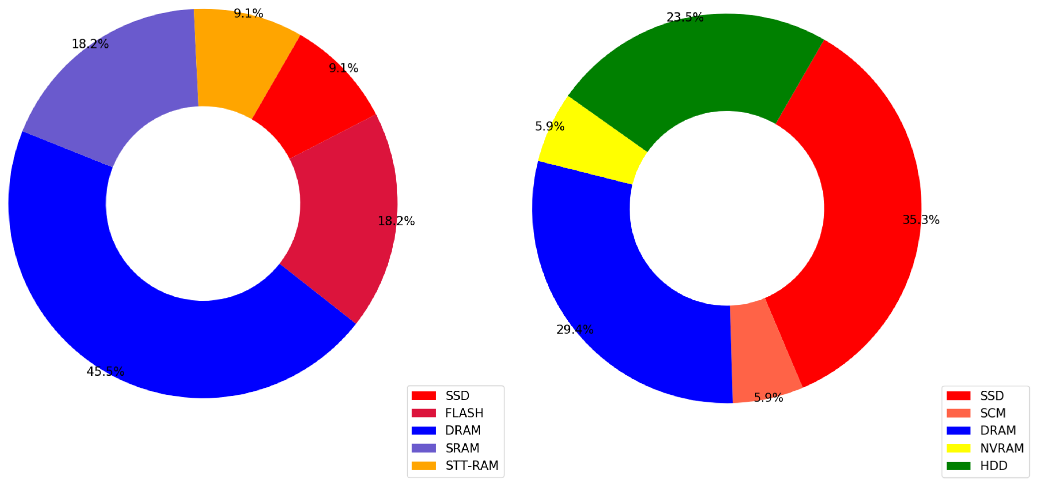
SSDs became very used both in storage systems acting in burst buffers or small testing environments. This storage device also collaborates for energy consumption since it has no mechanical components. We also noticed that some researchers propose and use flash chips in their experiments. Such experiments expose many software approaches to handle data while others implement new FTL algorithms to improve I/O throughput. However, HDDs still having their place in storage environments.

Magnetic devices are usually cheaper than flash-based devices and this is an important reason for their usage. Differently from common sense, HDDs has been evolving over the years. HDDs industry increased their research in trying to increase areal density capability (ADC) while decreasing device size.

Technologies such as PMR, SMR, IMR, HAMR, and HDMR are just some different approaches to store more bits into magnetic discs without increasing their size. As an expected result, magnetic tape is a cold storage approach that has not the goal of present higher I/O rather, this kind of storage is mainly used as secondary storage also known as cold storage.

The magnetic tape and disk store data using the same physical principle magnetizing the area of a magnetic surface meaning the presence or absence of information. Differently from social common sense, tapes remain, until nowadays, been used as the most storage media device by companies to store cold data and long term storage. However, this storage device does not target higher I/O operations due to the sequential data seek. This sequential storage technology means that the data are sequentially accessed, the time to access data is bigger compared with semiconductor memories and magnetic disks, due to the need to go through the whole tape until the part that contains the desired information. This storage device corresponds to only 0.2% of researchers' usage. Other devices media can also be verified in Figure 36 for more knowledge extracting.

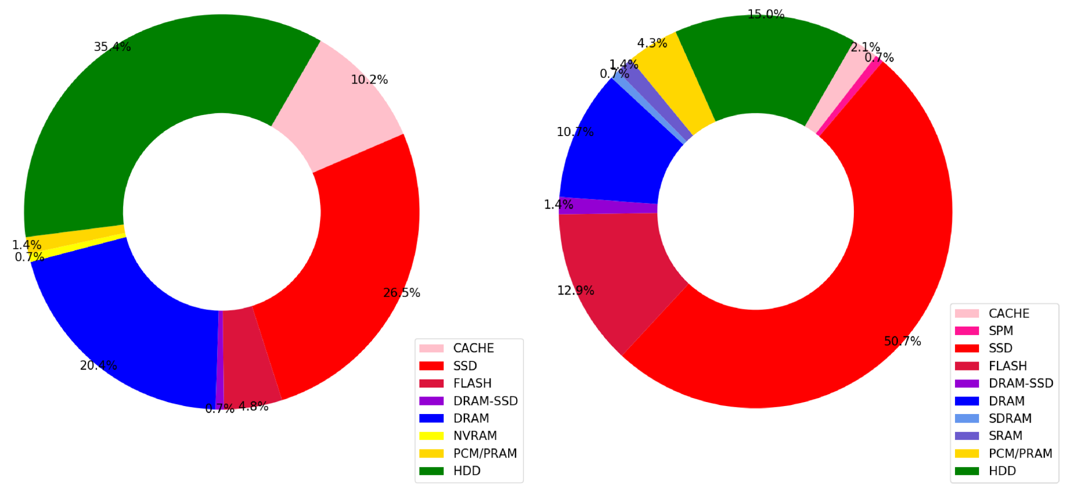
MQ8: How those considered devices are distributed according to each classification model? This question allows us to divide by subgroups which were the used devices by researchers in their experimentation when improving I/O performance. Figure 37 presents which were the devices used when hardware objects were proposed while Figure 38 presents which were the devices used when software objects were proposed.



(a) Used Devices in H2H-IO (b) Used Devices in H2S-IO

Figure 37 – Hardware Domain Used Devices

All solutions that propose hardware objects presented interesting proportions of DRAM. In Figure 37a it represented 45.5% of the total evaluated articles. In Figure 37b it almost reached 30% with 29.4%. Introducing and testing solutions that insert DRAM into boards and chips was frequently verified in this research. Also, an interesting result was the concern of researchers when trying to improve I/O on hardware proposing other hardware devices. In Figure 37a SRAM reached a considerable value that corresponds to 18.2%. This result was not presented in the other improvement class of the same group. However, all of them had bigger usability of flash memories and SSD devices. Together, DRAM and Flash memories and devices corresponded to 72,8% and 70,6% of the total device's usage in H2H-IO and H2S-IO respectively. Looking for HDD devices in Figure 37b we verified that it was used 23.5% of the time. However, in Figure 37a we do not find evidence of the usability of HDDs when improving hardware I/O performance. This result is expected due to the additional effort inherent to its characteristics when handling magnetic disks. We identified studies that use the advantages of PMR, SMR, and IMR HDDs technology to improve specific software. Because of that, we argue that the HDDs rate usage was bigger in Figure 37b than in Figure 37a.



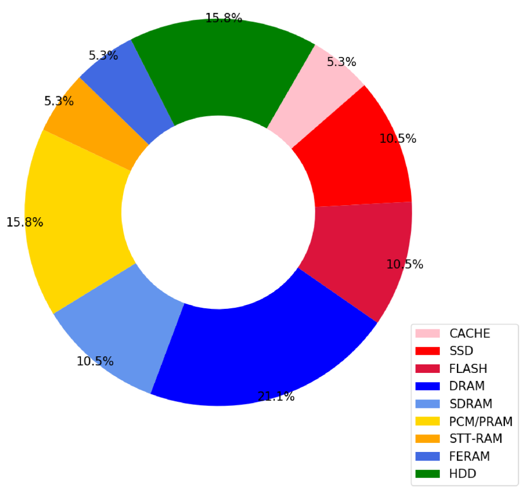
(a) Used Devices in S2S-IO (b) Used Devices in S2H-IO

Figure 38 – Software Domain Used Devices

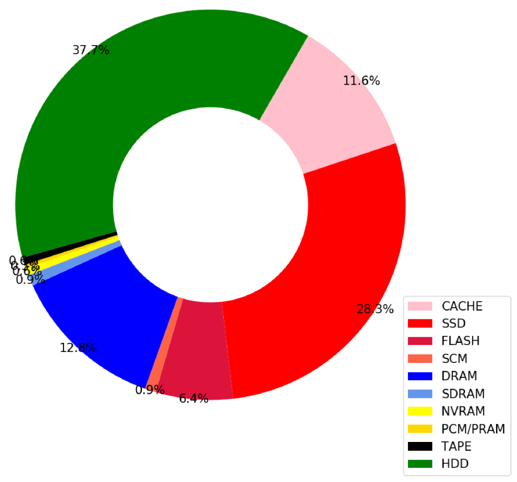
Figure 38 presents results relating to software improvements. Figure 38a presents the used devices when researchers propose software solutions to improve I/O performance on other software. We noticed that this group is characterized by the usage of HDDs, Dynamic Random-Access Memories (DRAMs), and SSDs. It makes sense because when researchers are proposing software solutions to improve I/O performance on other software, the choice of the user device on the underneath layer does not receive much attention unless the app is directly connected to the storage device. For instance, proposing improvements on Flash-File systems, usually, researchers perform its evaluation using the device that is directly related to its proposal, in such case a flash device.

Figure 38b presents the used devices when software solutions are proposed to improve hardware devices. In this case, the hardware usability is bigger compared to the subgroup presented in Figure 38b. SSD was responsible for the usability of more than 50%. This indicates that many solutions are considering and improving solid-state devices. This is because the author's proposals are directly related to the device used, which in this case, is an SSD. Differently from the devices used in Figure 38a that were not the elements that were receiving directly the improvement. The devices used in Figure 38b also indicates a big variability of the S2H-IO class. It indicates that researchers are considering and experimenting with different storage solutions.

Figure 39 presents the relationship considering storage systems. It is possible to note that, the usability rate of HDDs and SSDs presented a large proportion of the used devices. It happens because, nowadays, HDDs keep being widely used in storage systems. One of the most used approaches in storage systems is the employment of SSDs as quick



(a) Used Devices in H2SS-IO



(b) Used Devices in S2SS-IO

Figure 39 – Software and Hardware to Storage Systems Used Devices

and auxiliary storage for HDDs to support the I/O layer.

In Figure 39a we notice the big usability of PCM devices in H2SS-IO proposals. PCM device presents good I/O performance if compared with HDDs and NAND Flash memories. It is placed in between DRAM memories that present faster data access and NAND Flash that presents low latency compared to HDDs. PCM read performance is about 0.2 to 0.5 µs while NAND Flash is about 25-50 µs. The write performance is also better. It is about 0.5 to 1 µs while NAND Flash is about 100-2000 µs. Because of these good I/O rates, it is being widely used and considered in SS. Another important factor that can be observed in the group is the usability of the magnetic tape. In this case, only one study shown results that used magnetic tape. We could erroneously think that magnetic tapes are not used in storage systems. It makes sense because in this study we are analyzing specifically the improvement of I/O performance and, usually, storage systems that employ tapes are mainly interested in the scalability of storage (endurance) than its I/O ratio.

Finally, Figure 40 presents the results of the used devices in architectures.

Looking closely at the approach presented by Figure 40 we noticed that it had similar characteristics when joining approaches Figure 39a and Figure 39b. The proportion of used devices in hardware approaches to storage systems I/O (i.e. H2SS-IO) presents big usability for devices such as PCM and STT-RAM. Analyzing software approaches that aim to improve I/O performance in storage systems I/O (i.e. S2SS-IO), the relationship between HDDs, SSDs, and DRAMs is also presented in Figure 40. It was also expected because, as explained in section 3.4, when a study that proposes both hardware and software solution to improve a storage system I/O occurs, it was classified as an architecture approach. Therefore, Figure 40 presents similar characteristics if combining these two previously

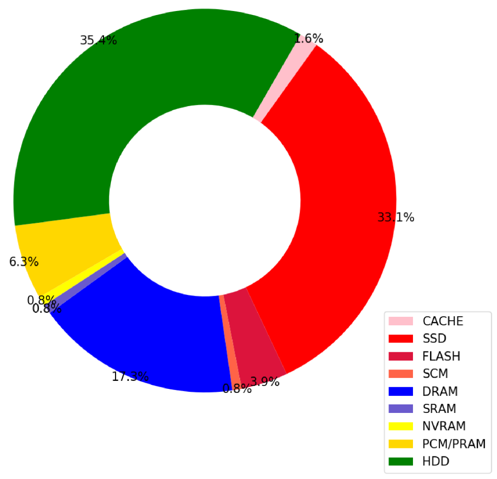


Figure 40 – Used Devices in Architectures

mentioned approaches. The proportions of HDDs, SSDs, DRAMs, and PCM devices support this hypothesis.

These research questions allow us to understand and visualize an interesting effect when hardware and software solutions are proposed. When researchers propose hardware improvements (i.e. Figure 37 and Figure 39a), we noticed that they are more likely to use DRAM and Flash memories in their evaluations process. On the other hand, when researchers propose software solutions (i.e. Figure 38 and Figure 39b), although DRAM has bigger usability, they are more likely to use HDDs and flash memories.

To answer MQ9 *(How authors improvements are evaluated and which tool they used in their experimentation?)* we aggregate all researchers methods used to evaluate their proposals. This question allows us to understand and visualize which I/O workloads generators are being used in bulk by researchers to evaluate I/O performance. We also visualize which mechanism they are using and how these mechanisms relate to the studied research field. Figure 41 summarizes the workloads that researchers used in their evaluation process. Usually, researchers use synthetic or real-world workload to perform the evaluation process. However, in some cases, the researchers evaluated their proposal using specific methods inherent to the study in question. Combined, these specific methods totaled 151 occurrences and have been omitted from the Figure 41. We also omitted workload occurrences that were used less than four times for better viewing.

In general, applications are characterized by their data movement access pattern. A different data access pattern might impose restrictions on overall environmental performance. Targeting better data access representation, researching academy and industry developed over the years many parallel I/O workload generators. These I/O workload generators, also known as I/O benchmarks, are also used to measure and evaluate the I/O performance of existing storage systems. They can be grouped into two main classes:

The first one is the application I/O kernels. As the name suggests, application I/O kernels consist of I/O generators derived from real-world applications. Usually, these applications are data-intensive and impose a huge effort both in processing and when performing I/O operations. Essentially, the processing code is removed from the application and the instructions of the application related to the I/O activity remains on the kernel. Thus, this code can be used to test and measure I/O performance over a storage configuration without imposing the processing cost.

On the other hand, synthetic I/O workload generators are not projected considering a specific application. It is constructed considering various data access patterns to represent as much as possible data workload. In this class of I/O workload generator, a set of predefined configuration drives the execution of the I/O operations e.g. (write, rewrite and read) Storage systems peak I/O performance is usually measured by this kind of evaluation tool.

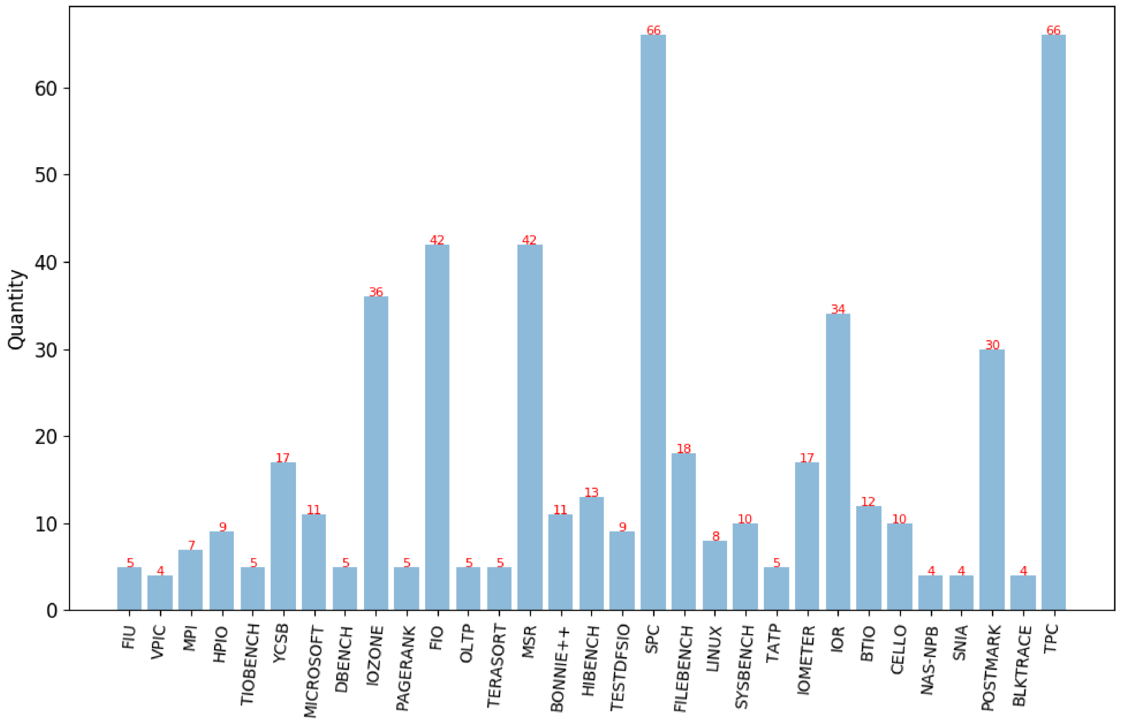


Figure 41 – Benchmarks Occurrences

As presented in Figure 41, Storage Performance Council (SPC) (COUNCIL, 2007) and Transaction Processing Performance Council (TPC) were the two most used benchmarks for researchers evaluations. Both SPC and TPC are workloads collected from Online Transaction Processing (OLTP) applications. There are many variations of these workloads, one with a specific data access pattern that characterizes a specific application. For instance, SPC is mainly known by Financial traces. As presented in Figure 42, researchers tend to use Financial1 and Finalcial2 traces.

Financial1 and Financial 2 are workloads traces collected from OLTP application

I/O that are written dominant and both are running at two large financial institution (COUNCIL, 2020). For instance, in Financial1 trace, there are 4.06 million write requests with an average write size of 4639.2 B and 1.27 million read requests with an average read size of 4125.2 B.

On the other hand, Financial2 is read dominant presenting 560 thousand write requests with an average write size of 5369.0 B and 3.14 million reads requests with an average read size of 4635.7 B. SPC also presents other very used traces workloads in researchers' experimentation. WebSearch1, WebSearch2 and WebSearch3 are search engine I/O traces that were collected from a popular search engine (COUNCIL, 2020).

TPC also received much attention from researchers when choosing their evaluation benchmark tool. There are many TPC traces available for experimentation, each one with its workload characteristic.

We verified that TPC-C was the trace that received most researchers' attention. It also is an online transaction processing (OLTP) benchmark that is composed of multiple transaction types, complex databases, and overall execution structure. It is composed of a mix of five concurrent transactions of different types and complexity (COUNCIL, 1990b). It is a decision support benchmark that consists of a suite of business-oriented ad-hoc queries and concurrent data modifications that are broad industry-wide relevant (COUNCIL, 1990d).

TPC-E is also an OLTP benchmark that is composed by a mix of twelve concurrent transactions of different types and complexity where the databases comprise third-three tables with a wide range of columns, cardinally and scaling properties (COUNCIL, 1990c).

Different from the previous ones, TPC-B is not an OLTP benchmark. Rather, it can be compared with a database stress test tool that is characterized by significant disk I/O requests, system and application execution time, and transaction integrity. It is mainly used to measure throughput in terms of how many transactions per second a system can perform (COUNCIL, 1990a).

In Figure 42 we expose which were the traces that researchers consider at most when using a benchmark tool in their experiments.

Many researchers also consider the MSR (ASSOCIATION et al., 2010) data repository to evaluate their proposals. MSR is composed of Severus traces with different access patterns. It was formed focusing on solving practical world problems with traces that represent these usual applications. There are thirty-five different traces available for MSR traces which represents 1-week block I/O traces of enterprise servers at MSR. Table 9 presents some used traces with brief data descriptions traces are presented below with a description.

Flexible I/O (FIO) benchmark (AXBOE et al., 2016), IOzone (NORCOTT, 2003)

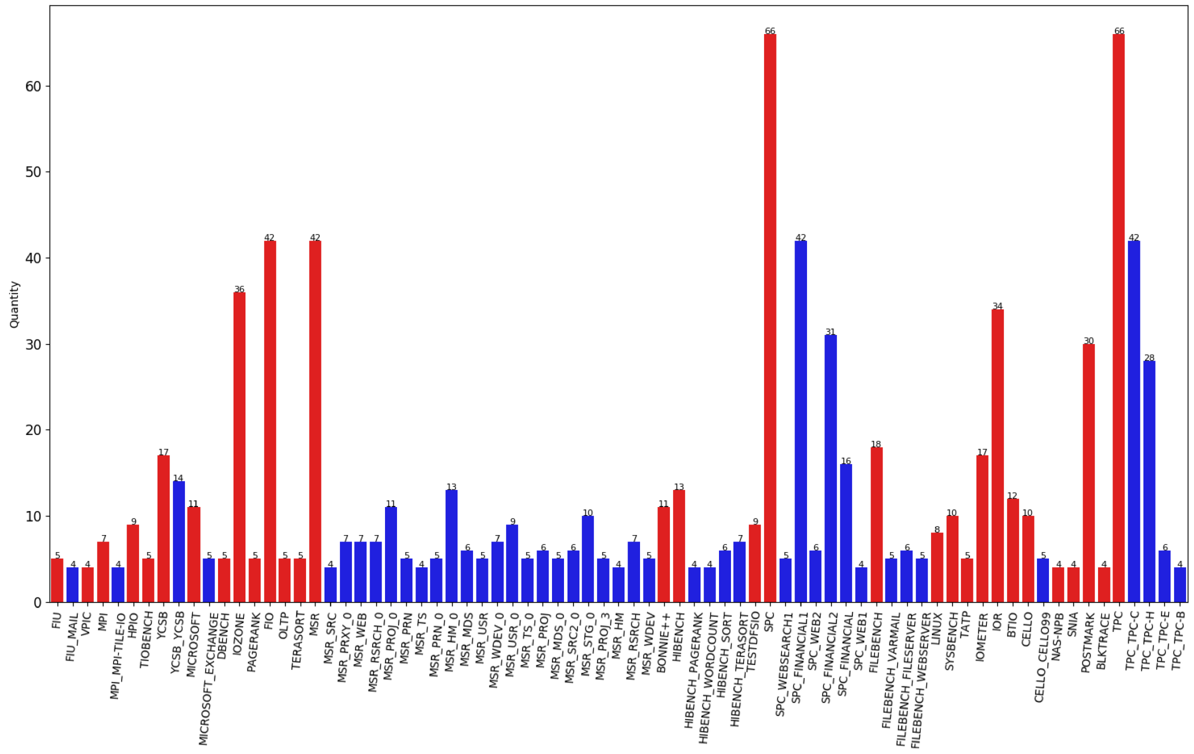


Figure 42 – Benchmarks and Traces

Table 9 – MSR Traces

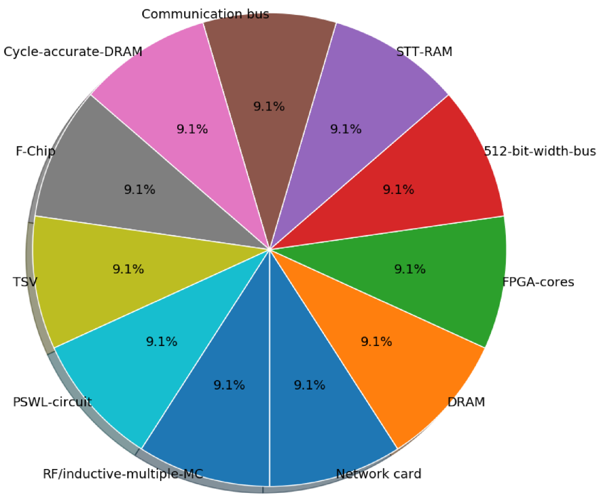
Trace	Description	Trace	Description	Trace	Description
usr	User files	mds	Media server	prn	Print server
hm	H/w monitoring	proj	Project files	wdev	Test web server
rsrch	Research projects	prxy	Firewall/web proxy	src1	Source control
src2	Source control	stg	Web staging	ts	Terminal server

and Interleaved Or Random (IOR) (SHAN; ANTYPAS; SHALF, 2008) are synthetic benchmarks widely used to evaluate I/O performance over many perspectives. FIO benchmark spawns a number o threads considering a particular type of I/O actions. It is mainly used to test random read because it can show all sorts of I/O performance information, for instance, the latency measure.

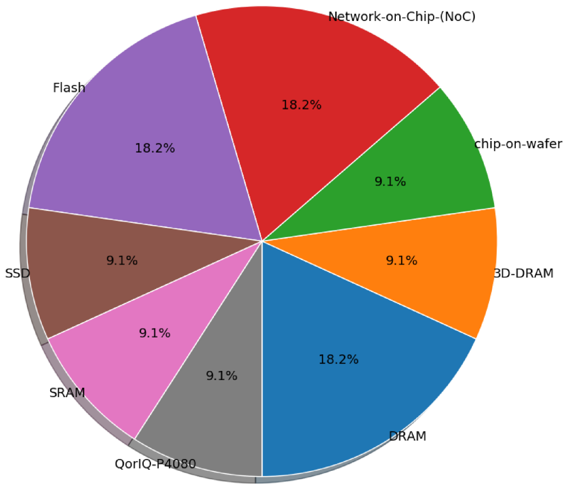
IOzone is a synthetic file system write-intensive benchmark tool. It has a sequential write pattern and is used to write many blocks over the file system is mainly used to measure file system operations and analysis measuring the streaming performance of large files increasing the number of updates to metadata and data. It can also be used to measure file cache I/O performance

IOR is a highly flexible synthetic workload generator. It is used to benchmark system performance and it is a standard in the parallel I/O research community. IOR also is used to generate a uniform random distributions workloads where users can specify the desired I/O workloads that include contiguous and non-contiguous access pattern One reason for its broad usability is also due to its compatibility with the I/O APIs found in computational science applications such as HDF5, MPI-IO, NetCDF and POSIX.

MQ10: How hardware devices are being proposed by researchers targeting better I/O performance? (H2H) MQ10 enables us to check which hardware devices are being proposed and improved by authors to improve I/O performance. Figure 43 addresses MQ10. Although Figure 43a and Figure 43b presented charts with similar colors, these colors do not represent the same meaning. Thus, these figures represent independent and individual content.



(a) Proposed Hardware to improve Hardware I/O



(b) Improved Hardware Class by Hardware Devices

Figure 43 – Hardware Solutions to Improve Hardware I/O.

Analyzing Figure 43a we verify that there are a huge variety of proposed devices and circuits that target improving I/O in other devices, chips, and boards. This chart shows us that all these proposed devices were proposed at the same proportion. This class of classification is indeed rare to happen.

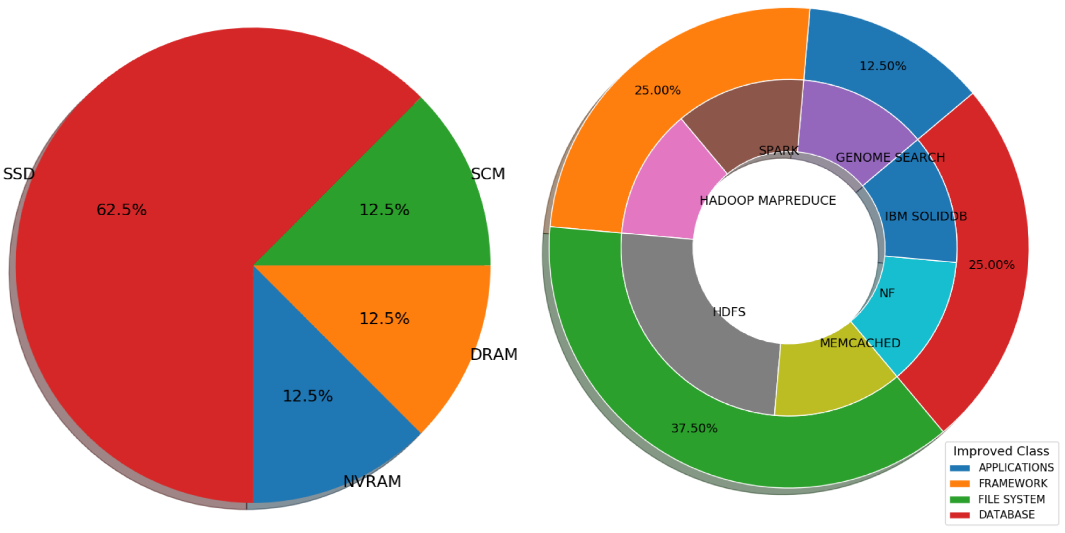
Propose hardware devices to improve other hardware devices requires a high understanding of how the underneath layers of computing behave. FPGA-cores, PSWL circuits, and communication buses are some of these researcher's proposals. These are improvements normally found in areas of study that have a lower level focus, taking into account the architecture of the device for better use of its resources, and they are easily found in electronic and electronic engineering magazines.

On the other hand, these improvements suggestions might improve other hardware devices. Figure 43b presents which were these improved hardware devices. Network on Chip (NoC), DRAM and Flash memories were the class of devices that received more contributions. It is important noticing that, in some cases, researchers' focuses were not improving only I/O but also other elements inherent to the hardware in question. All these improvements and proposals can be verified in our GitHub link [3] aforementioned.

The MQ11 *(Which Software Class Hardware are Improving and which are these proposed hardware?)* allows the reader to understand how researchers are using storage devices to improve software I/O performance. Figure 52 addresses MQ11.

Figure 44a presents the devices that were used by researchers to improve I/O performance on software. It is important to notice that it is possible to improve I/O performance from the software without making changes within your code or exercising specific code settings. For instance, if we configure a parallel file system (PFS) considering only HHDs as storage devices, we might probably receive a bigger latency result than if we use only SSDs devices. In this particular case, the seek time imposed by the HDD requires more time to find and operate over the data and metadata. However, composing high-performance storage only with SSDs devices is not recommended due to some facts such as reliability and cost.

Usually, researchers propose the usage of SSDs together with HDDs in storage environments. This storage configuration brings both velocity, integrity, cost, and reliability over the data. In this specific example, the insertion of SSDs on a specified storage system layer can improve file system throughput and latency performance. Thus, a hardware device can be proposed to improve I/O performance on software without changing file systems code lines



(a) Proposed Hardware Solutions (b) Improved Software by Hardware Devices

Figure 44 – Improved I/O Performance Software by Hardware Devices.

As expected, SSD was the device that received more researchers' attention. SSD is

[3] https://github.com/laerciopioli/Systematic-Literature-Data.git

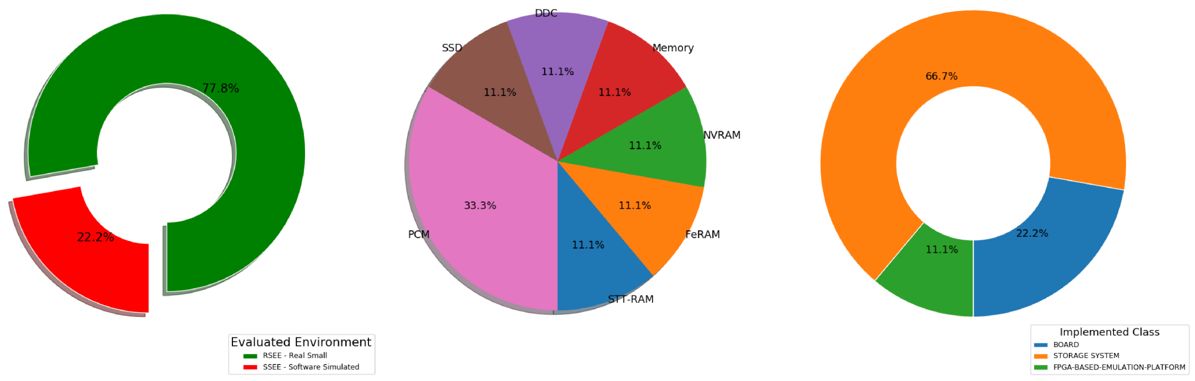
indeed a device that was employed in many storage systems, desktops, and data centers due to its many advantages. Low energy consumption, high throughput I/O and device size are some of the many advantages of employing solid-state devices in computing systems. This device corresponds to 62.5% of the total evaluated solutions proposals. DRAM, Storage Class Memory (SCM), and Non-Volatile Random-Access Memory (NVRAM) were also used to boost application performance. These are devices that present a better latency rate compared to the SSDs. However, these memory devices have a high price due to their better and quick technology. Figure 44b presents us which class of applications and which applications had their I/O performance increased. In such a case, we found that file systems, databases, frameworks, and general applications were improved by media devices. It is important noticing that the database specifications "NF" means "not found". This significance is propagated for all other applications or improvement class that we do not find the answer when reading the contents presented on the data process extractions phases presented on Figure 26.

MQ12 *(Which kind of environment-class is receiving researchers attention when the hardware is proposed as a solution to improve I/O performance on a storage system? Which devices they are using at most?)* allows us to understand how the experimentation is being executed when a hardware solution is proposed to increase I/O performance. Figure 45 addresses MQ12.

Through this question, we noticed that when the hardware is proposed to increase I/O performance, usually, researchers evaluate their experiments using small environments. It represents 77.8% of the total evaluated environments and can be verified in Figure 45a Environments such as a personal computer were verified three times and a small server was verified twice. We also verified that boards were used twice. In these cases, as addressed by Figure 45c, authors mentioned where their solution was implemented.

The first board was an Exynos 5420 Arndale octa board equipped with 2 GB Memory and 4 GB storage (XU et al., 2016) and the second board was an embedded development board constructed with NVRAM. Its usage was divided into 64 MB of SRAM and 64 MB of NAND Flash memory which was partitioned and implanted in the VFAT and NVFAT file system (DOH et al., 2009). The 64 MB of flash memory was partitioned with 32 MB used for the VFAT and NVFAT file system.

On the other hand, some experiments were carried out in simulated software. For instance, (LIU; WANG; YU, 2018) used a PCM simulation framework that simulates different caching schemes and was used to employ a PCM as a cache for a hybrid storage device. (JU et al., 2016) proposed an analytical model to indicate how to design a hybrid page cache targeting good throughput in terms of I/O per sec (IOPS) and fast cache reactivity. Their results indicate that introducing a PCM device into a page cache system improves the system performance significantly. To perform their experimentation they modified the DRAMSim2 simulator (ROSENFELD; COOPER-BALIS; JACOB, 2011) which is a consolidated memory system simulator.



(a) Evaluated Environment Class (b) Proposed Hardware to Improve Storage System I/O (c) Implemented Class

Figure 45 – Proposed Hardware and Used Environment.

Figure 45 shows us which were these proposed devices and in which environment classes these improvements were performed. We noticed that a big variety of memories were proposed by researchers to improve I/O performance on a storage system including high-speed NVRAM. PCM (WONG et al., 2010), Memristor (STRUKOV et al., 2008) and Spin-transfer Torque RAM (STT-RAM) (HOSOMI et al., 2005) are byte-addressable non-volatile memories like DRAM, but they are also non-volatile like devices used in secondary storage (e.g. HDDs and SSDs). Although PCM, Memristors, and STT-RAMs are available NVRAM media, Kannan et.al (KANNAN et al., 2011) does not propose a specific NVRAM device. Instead, they proposed the generic user-level NVRAM and evaluated their experiment with an emulated framework. We verified that NVRAM was responsible for 55.5% of the total improvements being composed by 33.3% originated from PCM, 11.1% from STT-RAMs and 11.1% from a non-specific media NVRAM device. Other media devices such as SSDs, DRAM, FeRAM, and Decoupled DMA Cache (DDC) composed the rest of the devices.

Addressing MQ13 *(Which software class is being improved by researchers when a software solution is proposed and where are those software improvements being implemented?)* allows us to visualize which software is being improved and receiving researchers attention when better I/O is the target.

Figure 46 shows us which software is receiving attention in researches when improving I/O performance. The inner border shows which was the specific improved software and the outer border is a general class to which this software belongs. As commented in the H2S-IO improvements section, it is possible to improve a software I/O performance

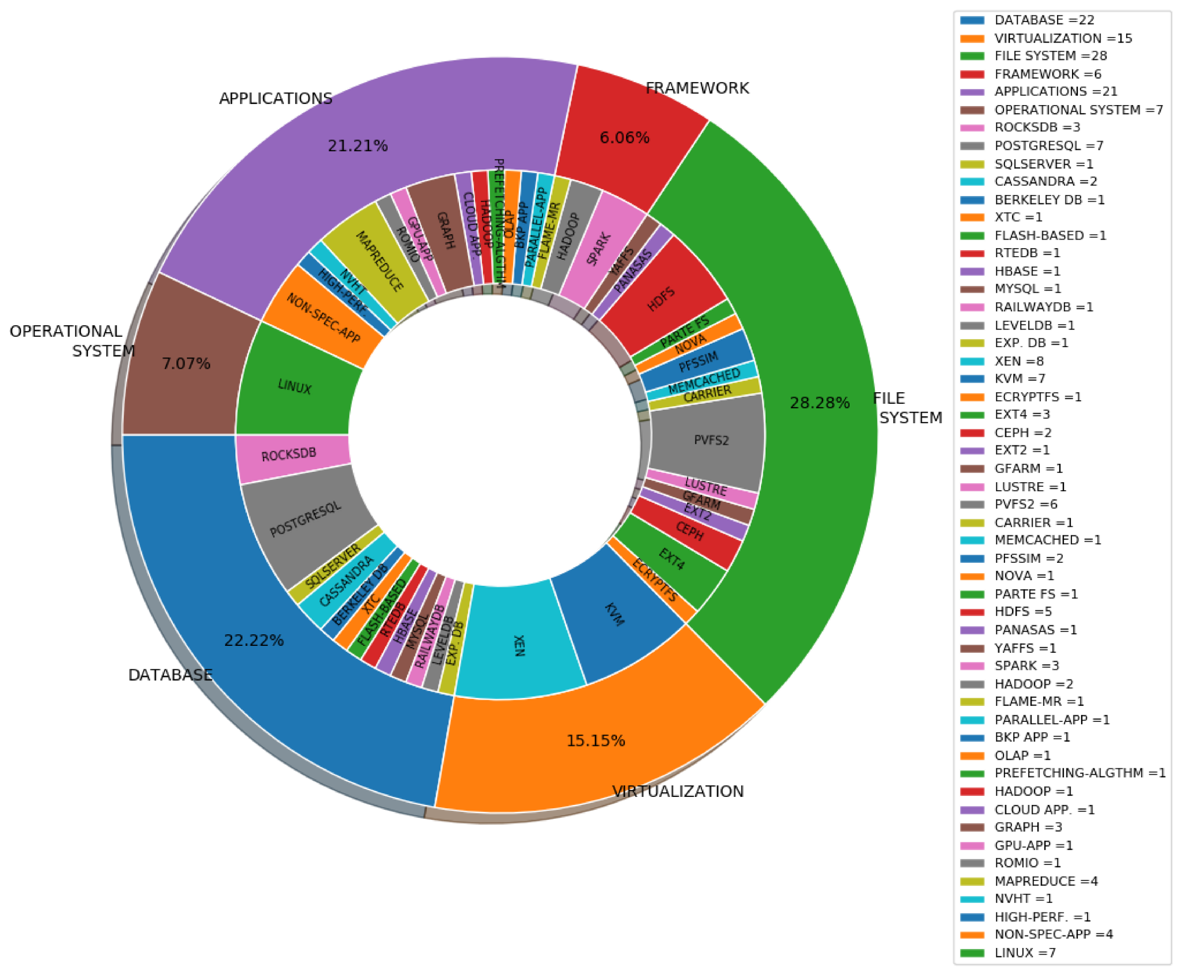


Figure 46 – Specific Improved Object over S2S-IO Improvements

without making changes within the code or exercising specific code settings.

Illustrating this thinking, file systems, databases, and overall applications almost runs over an operational system. Indeed, this is one of the operating system functions, proposes for applications and users better and user-friendly interfaces. A valid configuration over the operating system might change file system I/O performance. For instance, altering throughout CFQ, Deadline, and Noop scheduler in Linux operational system might increase or decrease the overall file systems performance. In this example, changing the Linux schedulers influences directly the above file system and application performance. Thus, the software can have its performance improved even without undergoing any changes.

We noticed in Figure 46 that researchers were engaged to find better I/O solutions mostly for file systems and databases. Together, these improvements reached more than 50% of the total improvements. We also verified that PostgreSQL was the database that received more attention. Six researchers were engaged to improve PostgreSQL I/O performance. After PostgreSQL, Rocks DB received three improvements, Cassandra received two improvements and all other databases received only one contribution. It is important to notice the existence of an "Exp.DB" in the inner border. Its name extends from "Experimental Database" that was built on top of a 16 GB Mtron MSD-SATA3025 SSD (LI et al., 2009).

Another interesting result is that improvements that target better performance on virtualization platforms such as KVM and XEN received the same quantity of improvements. Kernel Virtual Machine (KVM) (KIVITY et al., 2007) and XEN (BARHAM et al., 2003) are virtualization platforms with different characteristics.

In the KVM hypervisor, the host operational system (OS) should be necessarily Linux because it is a module of the Linux kernel. Because of that, KVM reuses many Linux kernel functions and utilities, turning it into a robust hypervisor application. In Linux, QEMU, which is a tool, that provides virtualization for applications. However, the guest OS can vary over Linux, BSD, Solaris, Windows distributions. On the other hand, XEN hypervisor supports Linux, Windows, Solaris, and BSD and was firstly part of a project hosted at the University of Cambridge.

File systems improvements focused on Parallel Virtual File System 2 (PVFS2) also known as OrangeFS, Hadoop Distributed File System (HDFS), and fourth extended file system (EXT4) with six, five, and four occurrences respectively. The other file systems received fewer improvements than those mentioned above. As expected, Linux was the operational system that received 100% of all operational systems improvements. Indeed, Linux receives many contributions for all components placed into the operational systems and is widely used in academic research.

Although these improvements are aimed at improving software I/O performance, in some cases they are implemented in other components and software. Figure 47 presents where these improvements were implemented according to its class. As expected, all improvements on the Linux operating system were implemented on the same Linux operational system. As mentioned before, Linux operational systems take an important role in computing research, and usually, it receives many researchers improvements. Because of that, the Linux operating system is robust and searched for the experimentation process, justifying improvements to better I/O.

Most virtualization had their improvements on the specific improved tool. However, some improvements were not on the evaluated tool. Instead, in some cases, researchers implemented solutions on Linux operational system. For instance, (CHIANG; UPPAL; HUANG, 2015), proposed a data prefetching method called VIO-prefetching to improve virtual I/O performance. They reached this result reaching a 43% rate improvement while running applications on a XEN virtualization system. To reach this result they have implemented a prototype VIO-prefetching in Linux operating system using XEN virtualization system. Oikawa S. (OIKAWA, 2014) proposed a storage virtualization method called Virtual Main Memory Storage (VMMS) that virtualizes NVM storage media into memories. They reached improvements from 2.13x to 5.97x for reading and

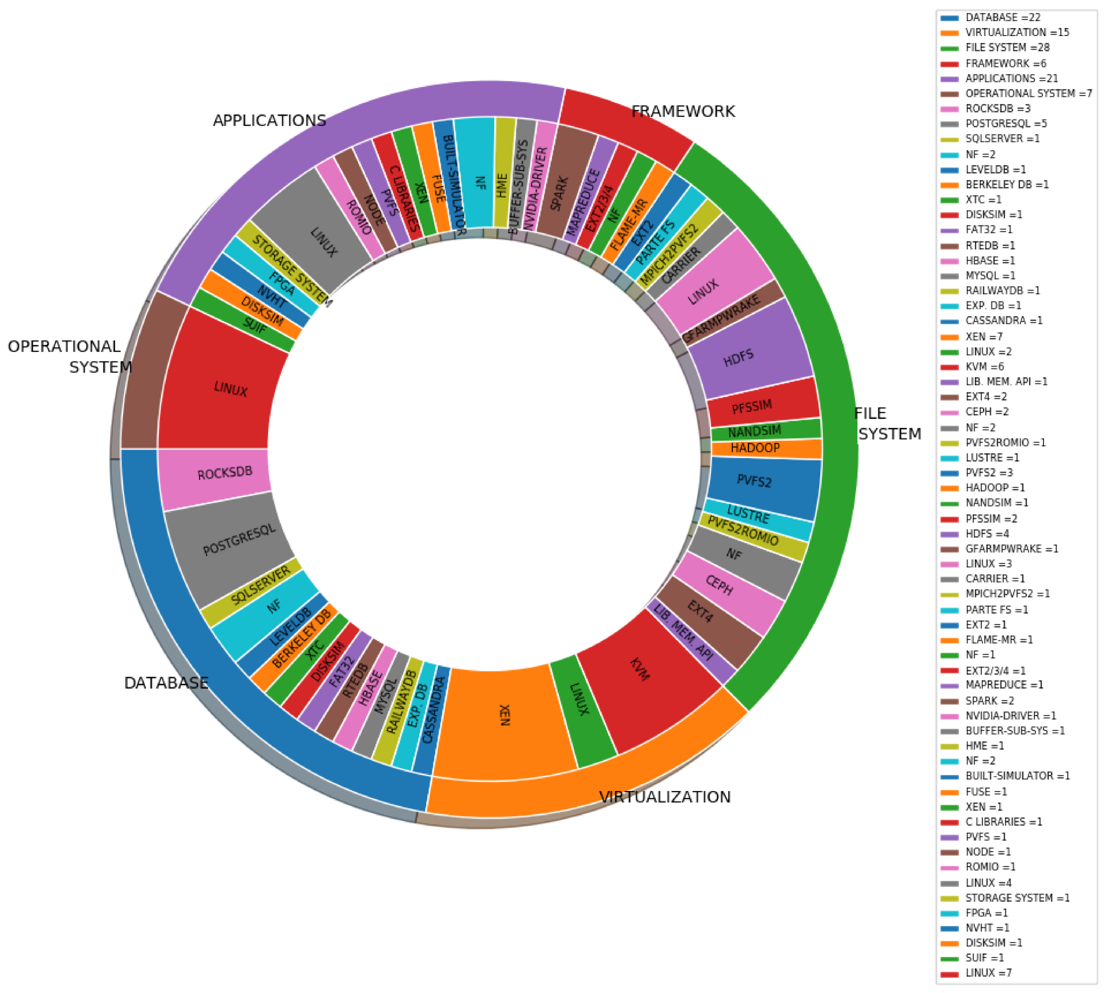


Figure 47 – Specific Implemented Object over S2S-IO Improvements

from 1.97x to 10.62x for writing, I/O operations. However, their solution was implemented in a Linux operations system that employs KVM as the virtualization platform.

Finally, analyzing the application implementations, we noticed that the "non-spec-app" which extends for "non-specific application" implemented their proposal in different software. The operational system Linux was implemented twice, the Fuse file system received one implementation and lastly, a built simulator received one implementation.

Park, J. et. al (PARK et al., 2019) presented a zero-copy read I/O scheme that minimizes the overheads of page remapping by reducing the number of remote TLB shootdown. It was based on page remapping and copy-on-write techniques and was implemented on a Linux kernel 4.12.9. Joo, Y. et. al (JOO et al., 2012) presented a SSD-aware application prefetching scheme, the Fast Application STarter (FAST). They evaluated and implemented their proposal FAST on the Linux OS. Their evaluation environment was a desktop PC with an SSD. Wang, C. et. al (WANG et al., 2012) proposed a solution (NVMalloc library) for exploiting NVM as a secondary memory partition for applications. It allowed applications to explicitly allocate and manipulate memory regions therein. They implemented their solution on the FUSE file system. Prabhakar,

R. et. al (PRABHAKAR et al., 2011a) proposed an approach that manages the storage caches providing predictable performance in multi-server storage architectures which improves the I/O throughput of applications. They implemented their solution in a built a storage cache simulator that simulates shared storage caches in the multiple storage servers.

MQ14 *(Which Hardware class are receiving more software I/O improvements, where are those software solutions being implemented and which device class researchers are using to experiment?)* allows us to visualize which device is receiving mostly researchers improvements. We also verify which device class is being used on the author's experimentation. This allows us to have a deep low-level storage understanding of the actual experimental environments.

Figure 48 presents which storage devices were improved by software solutions.

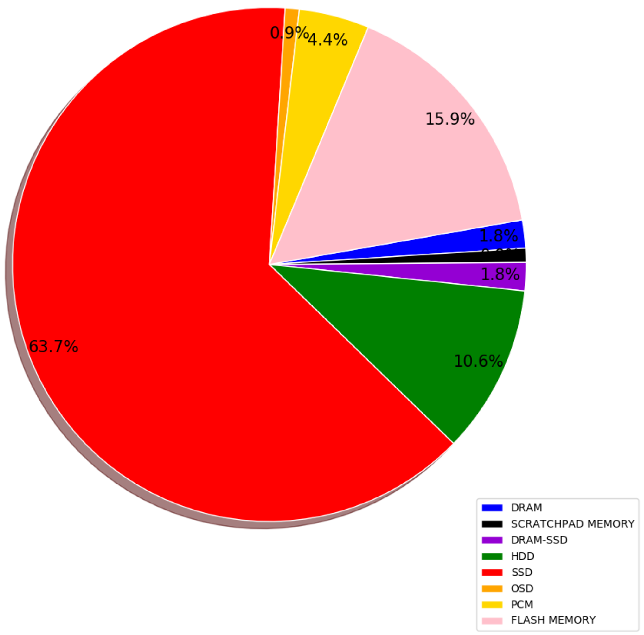


Figure 48 – Improved Storage Devices by Software Solutions

We noticed a big researcher concerning with non-mechanical devices. Together, SSDs and flash memories received more than 80% of the total evaluated papers. HDDs improvements were at least six times less than improvements on SSDs. Indeed, we verified that NAND flash-based devices employ many functions on many storage systems. However, due to its inherent flash memory architecture, it needs to perform the known erase-before write operation.

FTL is an important layer that can be implemented by different perspectives in storage devices and systems. It is composed of several firmware algorithms that perform logical-to-physical mappings, garbage collection functions, wear-leveling, and I/O interleaving among other functionalities. We verified that many of these improvements concerning with new FTL proposals (PAN et al., 2019), (WANG et al., 2016), (MATIVENGA et al., 2019), (XIE; CHEN; ROTH, 2017), (BOUKHOBZA et al., 2015) and (ZHANG; CHENG; LI, 2019). Its implementation could be in the form of software where the FTL layer can be introduced inside the SSD controller or it can be implemented as a hardware layer into the storage systems. FTL has the purpose to intermediate, repairing and adjusting, software systems and hardware devices to best fit among hardware and software functionalities. FTL layer is usually applied between the file system and the flash memory chip when introduced into a storage system. However, it is typically implemented inside the flash device on the controller. However, FTL algorithms not only have a great effect on storage performance and lifetime but also determine hardware cost and data integrity (LEE et al., 2016).

We also noticed that scheduling also has been considered among researchers. To provide better usage and access to the data, the I/O schedulers take care of the disk access requests. Usually, an existing software stack had been developed considering HDD blocks and tracks data distributions. For this reason, it is possible to get better SSD device performance by proposing and leveraging solutions that were first created considering HDDs. Therefore, researchers are to redesign new scheduler approaches considering the availability of flash memories (JI et al., 2019), (PARK et al., 2018), (YANG et al., 2019), (LI et al., 2014), (GUO; HU; MAO, 2015) and (SUN; QIN; XIE, 2014).

Figure 48 presents which storage devices were improved by software solutions while Figure 49 presents where these solutions were implemented. The inner border shows which was the improved software while the outer border shows a general classification of the improved element.

We verify that the implementations were classified into three classes (e.g. general hardware, simulator, and applications) The "NF" class was not intended to be created. However, the class identification was not clear or some time was omitted. Thus, we classify it as a NF that means "not found" and can also represent "not informed"

Inside the "Applications" class, where we condensed all implemented software, we noticed that many authors, specifically 19 studies, implemented their solutions inside the Linux operational systems. Linux was the software that received most implementations. We noticed a range of software that received modifications and implementations. For instance, Jinhua et. al (CUI et al., 2016) implemented their scheme in the host interface logic (HIL). The author argues that there the SSD specific characteristics and the data programming timestamp recorded in the FTL contributes to better scheduling decision for I/O requests. Also in applications, we do not identify which file system was used by Zhiwen et. al (JIANG et al., 2015). Thus, the general term "file system" was introduced.

After, in general, hardware class that is the class that relates implementations on hardware devices, SSD device took the second place with 11 implementations. Some of

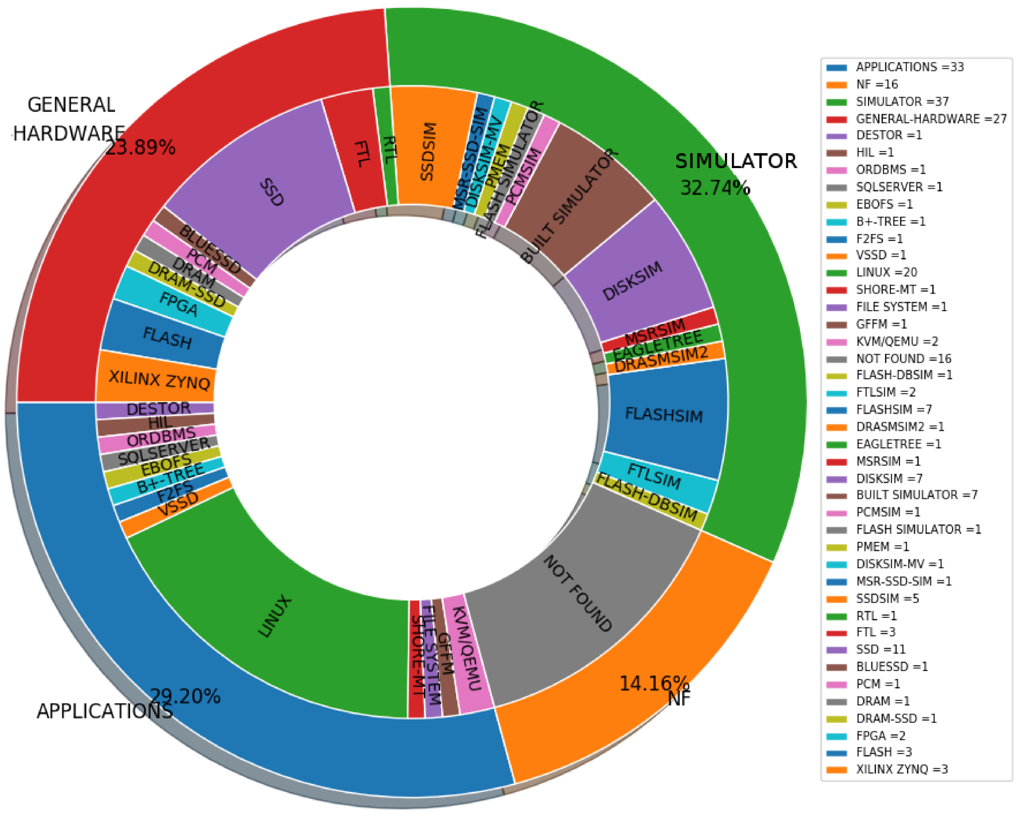


Figure 49 – Implemented Objects in S2H-IO Improvements

these implementations were specifically informed that it was on the FTL while others the authors do not mention which component of the SSD they implemented their solutions. Therefore, we create a FTL class into general-hardware even knowing that FTL, in many cases but not exclusively, can be a software layer placed between the file system and the flash memory (CHUNG et al., 2009). Finally, in the simulator class we noticed that the software that received more implementations were: Disksim (GANGER et al., 1998), Ssdsim (HU et al., 2011) and Flashsim (KIM et al., 2009) simulators. These, are very well-known simulators that allow uses to set specific parameters on their evaluations.

We also analyzed which kind of device researchers used in their experiments. It allows us to understand how authors solutions are being evaluated, which environment they are using to evaluate their solution, and which software is used to perform its evaluation Figure 50 addresses this question.

We divided these devices into two categories, the real devices, and simulated devices. This information makes interesting because we noticed that some researchers were evaluating their proposals on simulated devices.

Some of them evaluated their solutions on simulators built by themselves. We noticed a balanced use of the devices. 53.1% of the researchers were using real devices on their evaluation while 45.52% used simulated devices. It is important because we can

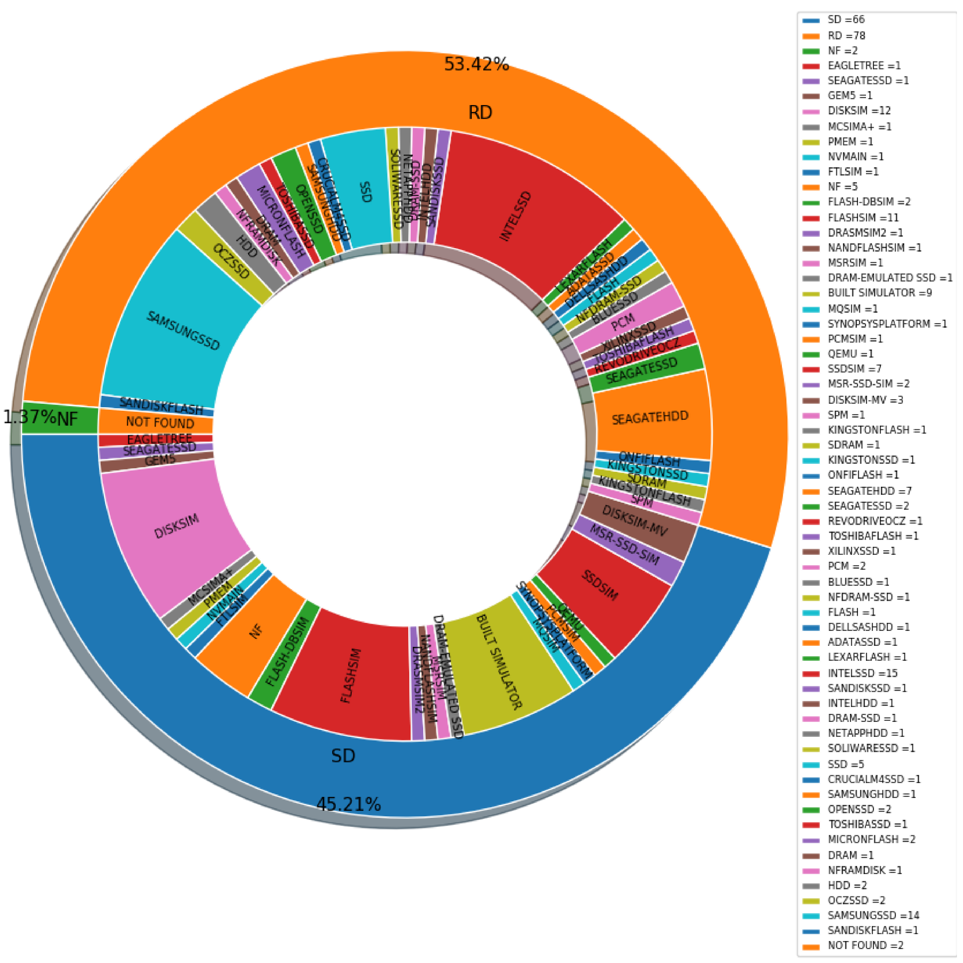


Figure 50 – Used Devices in Researchers Experimentation over S2H-IO Improvements

identify which technology is receiving more attention when researches are performed.

We also provided information about the device manufacturers. Indeed, SAMSUNG, INTEL, and SEAGATE are leaders in producing storage devices. It is important noticing that when it is presented only the device specification (e.g. HDD, SSD), it means that we could not identify the manufacturer. We were not able to identify 1.38% of the evaluated environments. They were represented by the acronym "NF" which means "not found" totaling two papers (SINGH et al., 2015) and (NI; LI, 2010). We do not found where the authors provided the information about the used devices in the experimentation.

Analyzing the simulated evaluations, we also noticed that the authors seem to be inclined to use Disksim, Ssdsim, and Flashsim simulators. Authors can measure a range of metrics and results of their experiments with these emulators. For instance, Disksim includes modules that simulate disks, intermediate controllers, buses, device drivers, request schedulers, disk block caches, and disk array data organizations (BUCY; GANGER et al., 2003). As the name suggests, SSDsim and Flashsim are flash-based emulators widely used for flash-based devices. Some authors evaluated their proposals by constructing a simulator system (built simulator) or even changing already consolidated simulator code (Disksim-MV) where "MV" stands for "modified version".

MQ15 *(MQ15: How researchers are evaluating their software solution that targets I/O improvements on storage systems, where are those software solutions being implemented, and what is the size of the user environments on the experimentation? (S2SS-IO))* presents us with a good deal of information about storage systems. First, we verify where these solutions were implemented. Secondly, it presents where the solutions to improve I/O performance on storage systems are evaluated. After, we analyze which storage environment is being used in the evaluation process. Through this question, we verify whether the evaluated environment is bigger, small, or software simulated.

Knowing the place where the proposals are implemented allows us to evaluate which devices are being modified to achieve better I / O performance results. This allows us to distribute our efforts in a targeted manner, thus solving the elements that most need attention.

Figure 51 presents where authors' solutions were implemented. Although there are a considered number of unidentified places, 15.71%, Figure 51 shows interesting results. As presented in Figure 49, this figure also distributes the authors' implementations places according to its general class. We noticed that improvements in the storage systems domain for applications increased from 28.57% in S2H-IO to 49.05% in S2SS-IO. Most of these increment in the storage system is related to the big quantity of implemented solutions in Linux operational system.

Storage systems are storage dedicated environments composed of many storage devices, in the down layer that performs many I/O operations to deliver data to applications and file systems. Usually, they are composed mainly of HDDs due to its low cost per Gigabyte (GB). However, this low-performance storage device imposes an I/O bottleneck on the overall systems due to its mechanical characteristics.

HPC, DISC, and OLTP applications usually are data-driven and demand for high-performance storage systems and I/O. Introducing SSDs to acts in the I/O caching layer is the most common approach to improving the storage system performance because it maximizes the cache hit ratio. It improves the response time when requests are supplied by the cache delivering data quickly with low latency. Because of that we noticed that many authors solutions to improve I/O performance using SSD devices were implemented on Linux (WANG et al., 2018), (AHMADIAN; SALKHORDEH; ASADI, 2019), (YANG; YANG, 2013) and (KLONATOS et al., 2011). These solutions are usually compiled jointly to the kernel before the evaluation process.

In S2S-IO analysis we inspect whether the evaluation process was executed using a real or simulated storage device. The same idea is preserved here. We investigate whether the environment used to perform the experimentation was a small, large-scale, or software

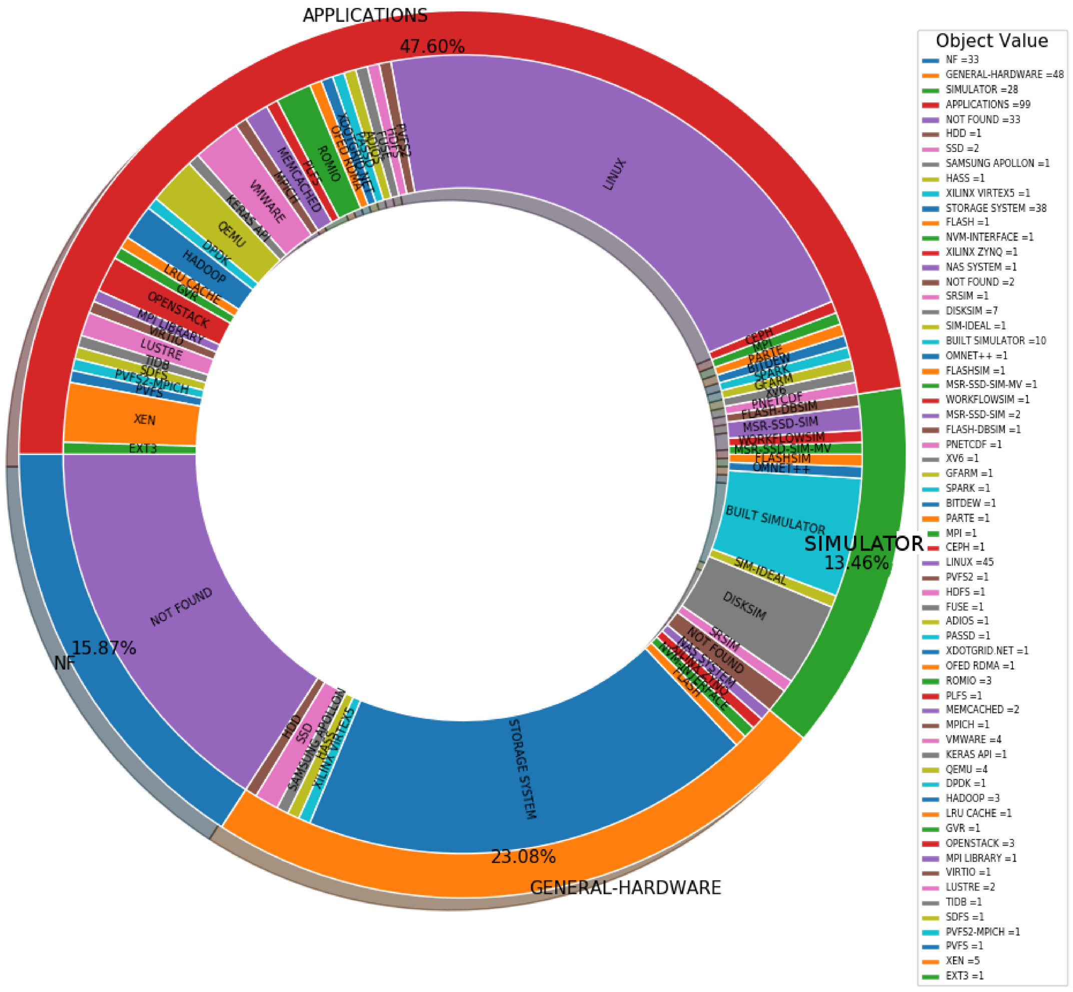
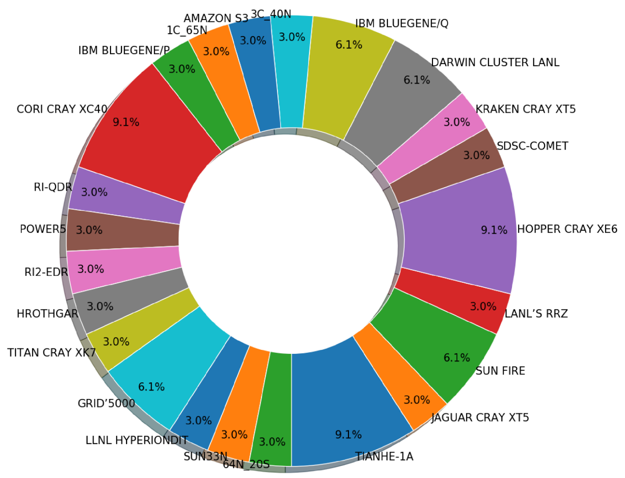


Figure 51 – Implemented Place for S2SS-IO Improvements

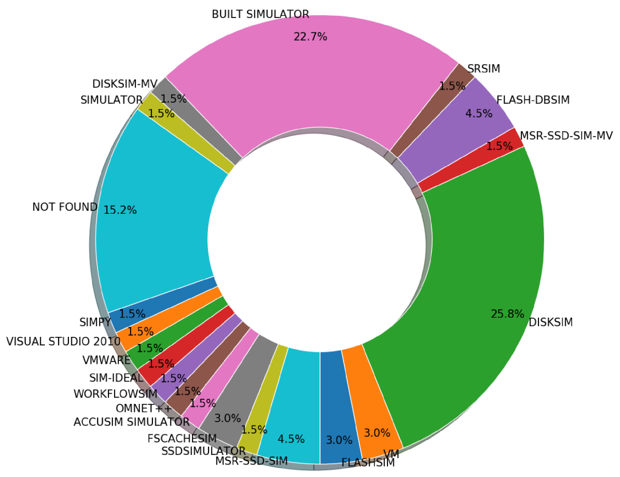
simulated environment. Below we explore these environments exposing which was the used environment and their occurrences.

- **Large-scale Evaluated Environment** - 13.0%
- **Small Evaluated Environment** - 52.9%
- **Software Simulated Evaluated Environment** - 30.8%
- **Not Found** - 3.4%

In Figure 52b we verify that some proposals were evaluated in a simulated environment instead of a real environment. Therefore, we verify which was this used software. We verify that Disksim (GANGER et al., 1998) corresponded to more than 25% of the overall used simulated software. We also noticed that researchers evaluated their proposals inbuilt simulators that correspond to almost 23%. It is important noticing that we could not identify 15.2% of the evaluated environments because in some cases it was not clear to



(a) Large-scale Used Environments



(b) Software Simulated Environments

Figure 52 – S2SS-IO Used Environments.

us to identify which were the used environment. When we analyzed S2S-IO improvements we also noticed that researchers used Disksim and built simulators as the environment to evaluate their proposals at most. However, there, authors distributed their proposal over more two more simulators: SSDsim and Flashsim. In S2SS-IO improvements, we realized that only software simulated Disksim stood out from the others.

Figure 52a presents which were the environments used by researchers in their projects. We noticed that Hopper Cray XE6, Cori Cray XC40, and Tianhe-1A was the most used large scale environments with a rate of almost 10% each. We also noticed the usage of four large-scale environments that do not present a name and were exposed as (1C_65N, 3C_40N, 64N_20S, and 54N). The letters mean (**C** - Cluster, **N** - Nodes and **S** - Server).

Among the many researcher's studies, we present some interesting improvements that belong to its characterization class. However, more details can be found in our GitHub [4] site. Zha B. et al. (ZHA; SHEN, 2018) proposed a memory aware I/O scheduling method for hybrid storage based HPC. They used a 65-node Linux-based cluster testbed. Each of the 65 nodes was connected with Gigabit Ethernet and equipped with one solid-state drive.

Midorikawa, H. et al. (MIDORIKAWA; TAN, 2016) proposed a scheme that utilizes distributed flash SSDs over cluster nodes as an extension to the main memory with a locality-aware algorithm for solving data size requirements for large-scale stencil computation. These algorithms were evaluated in different platforms and flash devices. They used 3 clusters (3C) totalizing 40 nodes (4N), being Hetero cluster with 4 nodes, Haswell cluster also having 4 nodes and TSUBAME2.5 cluster with 32 nodes.

[4] https://github.com/laerciopioli/Systematic-Literature-Data.git platform

Huang, Y. et al. (HUANG et al., 2013) proposed a method that is designed to cache the sub-requests with high I/O cost on the client end called cost-aware client-side file caching (CCFC). They argue that client-side caching is an interesting tool for addressing the performance issue of file systems when data nodes have highly unbalanced response time evaluating their proposal on 2 clusters where one of them consisted of 64 compute nodes and the other 20 file servers.

Below we discuss some environments and projects that were used in the experimentation process. We also present the basic characteristics of these used in large-scale environments.

Shankar, D. et al. (SHANKAR; LU; PANDA, 2017) proposed a high-performance key-value store with online erasure coding for big data workloads. They used three high-performance compute clusters for evaluations The first cluster was the Intel Westmere Cluster (RI-QDR) that is made up of 144 compute nodes, but they used 17 on the experiment. The second cluster was the SDSC Comet (SDSC-Comet) having 1,984 nodes but in the experiment was used 15 nodes. Finally, they used 20 nodes from Intel Broadwell Cluster (RI2-EDR) cluster.

Tianhe-1 was accomplished in 2009 where each node has two Xeon processors. The theoretical peak performance of TH-1 was 1.206 TFlops and it was composed of 6250 nodes connected by DDR Infiniband.

Tianhe-1A (TH-1A), an upgrade of the Tianhe-1 was introduced in August 2010. Tianhe-1A (TH-1A) is a supercomputer, that was developed by the National University of Defense Technology (NUDT) and located at the National Super Computer Center in Guangzhou. The theoretical peak performance of TH-1A is 4.7 petaFLOPS and it is composed of 7168 compute nodes and 1024 service nodes. It is also composed of a proprietary interconnect network and the compute nodes have two general processors (Intel Xeon X5670) and one stream processor (NVIDIA M2050 GPUs) totalizing 23,552 microprocessors. They are distributed over 140 racks where 112 are designed for computing, 8 racks for service, 6 for communication, and 14 for I/O management. The total aggregate memory is 262 TB and the disk capacity is 2 PB (YANG et al., 2011).

Tianhe-2 , also known as Milky Way-2, appeared in 2012 and is located at National Super Computer Center in Guangzhou and it is also developed by the National University of Defense Technology (NUDT). It is composed by Intel Xeon E5-2692v2 12C 2.2GHz with 4,981,760 cores available. Its aggregate memory is 2,277,376 GB and uses TH-Express-2 interconnect network. Its peak performance reached 61,444.5 TFlop/s consuming 18,482.00 kW of energy.

Tianhe-3 Today, under development, the Tianhe-3 prototype is located in Tianjin, China. It has adopted the ARM-based many-core architecture roadmap using a home built phytium and matrix processors. Its processor seems to be the Phytium FT-2000+(FTP) and MT-2000+(MTP) where FTP contains 64 ARMv8 core and MTP 128 ARMv8 cores Each FTP core can run up to 2.4 GH and each MTP core can run up to 2.0 GHz (YOU et al., 2019b).

Some projects that used Tianhe are: Yu, J. et al. proposed a memory cache system called LeCache. It acts as a locality-enhanced user-level POSIX-compliant distributed memory cache and intercepts user-level POSIX I/O calls redirecting to distributed cache (YU et al., 2017). After, in (YU et al., 2019) they proposed a workload-aware temporary cache system (WatCache) to manage the storage tier in HPC I/O hierarchy. Liu, X. et al. (LIU et al., 2017b) proposed a hierarchical hybrid storage system called on-line and near-line file system (ONFS) They build a three-level storage system in a unified namespace using DRAM and SSDs in compute nodes and HDDs in storage servers. All these experiments were executed in Tianhe systems.

Cori Cray the system is placed at the National Energy Research Scientific Computing Center (NERSC). It is a Cray XC40 that has a peak performance of about 30 petaFLOPS. Today, July-2020, it is the world's sixteenth most powerful supercomputer in the world. Cori has partitioned in Cori-Haswell and Cori-KNL over 6 rows of cabinets. Each cabinet has 3 chassis; each chassis has 16 compute blades, each compute blade has 4 nodes (CORI..., 2020). Cori-Haswell is composed of 2388 nodes where each node has two sockets Intel Xeon Processor E5-2698 v3 and the total aggregate memory is 298.5 TB. Each socket is populated with a 2.3 GHz 16-core Haswell processor. The "Haswell" processor nodes are distributed over 14 cabinets and the total peak performance that it can reach is 2.81 PFlops. Cori-"Knights Landing" (KNL) is composed of 9668 nodes where each node has a single-socket Intel Xeon Phi Processor 7250 "Knights Landing" processor with 68 cores per node @ 1.4 GHz. The total aggregate memory is 1.09 PB. The "KNL" processor nodes are distributed over 54 cabinets and the total peak performance that it can reach is 29.5 PFlops.

Some projects that used Cori are: Tang, H. et al. (TANG et al., 2018), proposed object-centric data abstractions and storage mechanisms mapping it in different levels of the storage hierarchy. The idea behind this was to take advantage of Proactive Data Containers (PDC) which is a storage hierarchy. Zang, T. et al. (WANG et al., 2018) proposes a data management service called UniviStor that provides performance optimizations and data structures tailored for distributed and hierarchical data placement. Liang, W. et al. (LIANG et al., 2019) presented a contention-aware resource scheduling (CARS) strategy for Burst buffers. It acts as a resource manager and coordinate concurrent data-intensive jobs. All these proposals have been experimented with in Cori Cray XC40.

Kraken Cray is a series of XT systems that was introduced in April 2008. Kraken XT3(April-2008), Kraken XT4(July-2008), Kraken XT5(Feb-2009), Kraken XT5(Dez-2009) and Kraken XT5(Jan-2011). Kraken XT5 (KRAKEN..., 2020) is a supercomputer at the National Institute for Computational Sciences (NICS) composed by 9,408 compute nodes. Each node has two 2.6 GHz six-core AMD Opteron processors (Istanbul) with 12 cores and 16 GB of RAM. It is composed of 112,896 compute cores and the aggregate memory reaches 147 TB. The overall disk space is 3.3PB The peak performance that Kraken can reach is 1.174 petaFLOPS.

The project that used Kraken also used a Cray and Grid5000. Dorier, M. et. al (DORIER et al., 2012) proposed a tool that leverages dedicated I/O cores multicore SMP node called Damaris It allows performing asynchronous data processing using shared memory. They evaluate Damaris on three different platforms including the Kraken Cray XT5 supercomputer using Lustre, the Grid5000 using PVFS, and in BluePrint with GPFS.

Titan was a Cray Xk7 supercomputer at the Oak Ridge National Laboratory (TITAN..., 2020). It was operated by the Oak Ridge Leadership Computing Facility (OLCF) at the US Department of Energy's (DOE's) Oak Ridge National Laboratory (ORNL). Titan was composed of 18,688 compute nodes that can reach 27 petaFLOPS of peak performance. It was composed of a 16-core AMD Opteron central processing units (CPUs) with a 2 GB memory/core. Each node had 16 cores, 32 GB of memory, and NVIDIA Kepler graphics processing units (GPUs). Titan was decommissioned on August 2, 2019.

Some projects that used Titan are: Liu, Q. et al. (LIU et al., 2017a) proposed StoreRush to resolve a contentious issue over the storage devices. StoreRush runs at the application level without requiring modification to the file and storage system. It improves write performance using a two-level messaging system to harvest idle storage via re-routing I/O requests to a less congested storage location Liu evaluated their experiment in Titan and also in the Hopper supercomputer.

Hopper was a Cray XE6 system at the National Energy Research Scientific Computing Center (NERSC) (HOPPER..., 2020). Created in September 2010, it was the first system to reach the petaFLOPS barrier. Today, Hopper evolved to Cori supercomputer fore mentioned. Hopper was decommissioned in December 2015. Composed by 153,408 processor-core 2.1-GHz AMD Magny-Cours Opteron processor, it reached a peak performance of 1.05 petaFLOPS. The total of 6,384 nodes totalizing 153,216 cores and an aggregate memory of 216,832 GB. All the NERSC systems and is system history can be verified at (NERSC..., 2020).

Some projects that used Hopper are: Liu, J. et al. (LIU et al., 2014) They developed a model-driven mechanism for selecting the data layouts that benefit the performance of different read patterns. Their model was based on the striping parameters on the Lustre file system and the block-level striping on RAID-based disks within an Object Storage Target (OST) of Lustre. Son, S. et al. (SON et al., 2017) proposed a probing tool that stripes across a selected subset of I/O nodes and enables application-level dynamic file striping to mitigate I/O variability. It allows achieving the highest I/O bandwidth available in the system. They evaluated their experiments on Hopper variants supercomputers.

IBM Bluegene/L, P and Q the family was initially built in the 1999s to advancing biomolecular simulations, computer design software, and large-scale systems (CHIU, 2013). Lawrence Livermore National Laboratory (LLNL) emphasized applications related to national security and Argonne National Laboratory (ANL) was providing massively parallel machines in general for the community. These IBM partners in relation helped IBM development-boosting hardware and software contributing to the Blue Gene family's success.

The first of the family was IBM Blue Gene*/L supercomputer (GARA et al., 2005) installed at LLNL with 596 teraFLOPS of peak performance and 104 racks totalizing 106,496 nodes. IBM Blue Gene*/L supercomputer won the TOP500 (MEUER et al., 2001) seven times between 2004 and 2007. In 2007 the IBM Blue Gene*/L evolved to IBM Blue Gene*/P (ALMASI et al., 2008) It was installed at Forschungszentrum Juelich in Germany and was composed of 72 racks with 73,728 nodes delivering 1 petaFLOPS of peak performance and becoming the first system to deliver 1 petaFLOPS in Europe. Finally, the last version of the IBM Blue Gene* family is the IBM Blue Gene*/Q The largest installation is the Sequoia and is located at LLNL reaching 20 petaFLOPS and being composed of 96 racks with 98,304 nodes. All of them were composed of Torus network being IBM Blue Gene*/L and IBM Blue Gene*/P with 3D dimension and IBM Blue Gene*/Q with 5D dimension. The provided bandwidth was 2.1 GB/s, 5.1 GB/s, and 40 GB/s for IBM Blue Gene*/L, IBM Blue Gene*/P, and IBM Blue Gene*/Q respectively.

Mira supercomputer, an IBM Blue Gene/Q supercomputer placed at Argonne National Laboratory that can reach operating speeds of 20 Peta-FLoating-point Operations Per Second (FLOPS) range. By the last TOP500 list of the most powerful supercomputers of the world evaluated in November 2019, Blue Gene/Q was placed at the 12 positions. It is composed of 49.152 CN each with 16 hyper-threaded PowerPC A2 cores (1600 MHz). The RAM presented in each CN is 16 GB of DDR3 and the network topology is used to connect the nodes in the 5D torus which gives a bandwidth of 2 Gbps per link (TESSIER et al., 2016). The CN are distributed and split in banners called *Pset* and it is composed of 128 CN. At the end of these banners, there are two bridge nodes that are regular CN connected to one ION through a network that gives 2GBps per link. The network which connects the ION to the Infiniband switch has the maximal bandwidth of 4 Gbps per link.

Some projects that used IBM Bluegene are: Schürmann, F. et al. (SCHÜRMANN et al., 2014) used SLC Flash memory through block device and direct storage access (DSA) principles integrating with IBM BlueGene/Q supercomputer at scale Blue Gene Active Storage (BGAS). The block device layer provides compatibility with common IO layers systems (POSIX, MPIO, HDF5). The DSA strategy enabled a low-overhead, byte-addressable, asynchronous, kernel by-pass access method. Eilemann et. al (EILEMANN et al., 2016) exploited a way to integrated NVM devices into supercomputers to address data-intensive processing issue. They used a scalable key-value (KV) I/O methods instead of traditional file I/O calls commonly used in HPC systems to enable higher performance. Ambareesh et. al (AMBAREESH; FATHIMA, 2016) proposed a storage middleware (HyCache+) to reduce the bi-section bandwidth of the parallel computing systems. It was based on Programmable Operating System (POSIX) and used a 2-layer scheduling approach.

Grid'5000 It is a French large-scale environment for experiment-driven research with a focus on parallel and distributed computing including Cloud Computing, High-Performance Computing, and Big Data. Many technologies compose the experiment environment through different layers. Among many other technologies, we highlighted some of them. In the computation layer, it provides CPU processors and GPUs. In the storage layer, it provides storage devices such as SSD, HDD, NVM. Finally, at the network layer, it provides Ethernet, Infiniband, and Omni-Path network interconnecting. Grid'5000 environment is composed of the connection between 8 different sites located in France. It provides many options and configurations of parallelism paradigm, solving huge problems of science.

Some projects that used Grid'5000 are: Saif, A. et. al used (SAIF; NUSSBAUM; SONG, 2018) proposed a mechanism, the IOscope tracer, for uncovering I/O patterns of storage systems workloads that perform filtering-based profiling over fine-grained criteria inside the Linux kernel. He used Grid'5000 in this experiment.

We also verify in Figure 53 the case when the proposal was evaluated within a small environment. We verify that some solutions need just some disks in their evaluations. Wang, J. et al. (WANG; CHENG, 2015) used 10 disks being 8 Disks SAS 2.5, 1 SSD Intel 32G and 1 SSD Intel 160G. Xiao, L. et al. (XIAO; YU-AN; ZHIZHUO, 2011) constructed an S-RAID 5 that includes 5 Seagate 500G Disks. Skourtis, D. et al. (SKOURTIS; KATO; BRANDT, 2012) used 4 disks in RAID 0 configuration. Park, J. K. et al. (PARK; SEO; KIM, 2019) used two devices being 1 a 32 GB HDD and 1 4 GB SSD.

We also verify authors experimenting on different boards. Lee, S. et al. (LEE et al., 2011) used a Samsung Apollon board, Zhang, J. et al. (ZHANG et al., 2015b) used a GPU-SSD board, Wang, Y. et al. (WANG et al., 2014) used a Xilinx Zynq on-chip programmable SoC architecture (CROCKETT et al., 2014). We classified as a small environment the case when the experimentation is able to be executed into university labs, personal computers (PCs), laptops, servers units, or a small number of used nodes. To classify these environments we called small environments those that used less than 40 nodes to have experimented. Therefore, it is possible to verify in Figure 53 authors

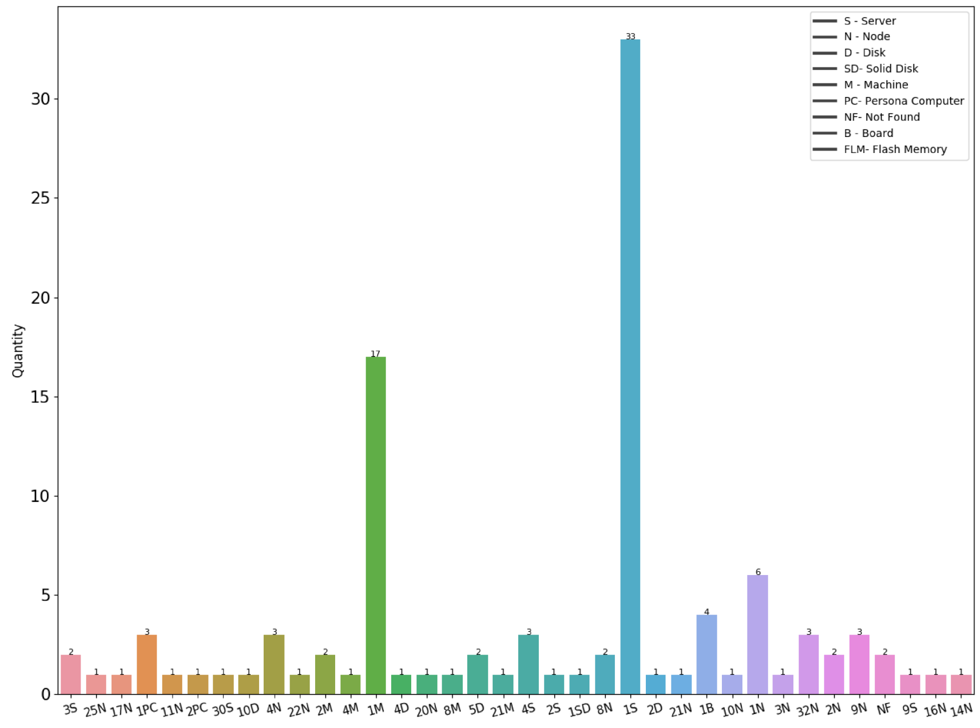


Figure 53 – S2SS-IO Small Environments Used

that used 25 nodes (LIAO; LIU; CHEN, 2013) or even 32 nodes (NICOLAE; RITEAU; KEAHEY, 2014) classified as small environments. For instance, Liao, J. et al. (LIAO; LIU; CHEN, 2013) experiment in 2 clusters totalizing 25 nodes. In his case, the metadata server was deployed on 8 storage servers of cluster 1 and the client file systems were installed on 16 nodes of cluster 2. The interconnection network was a 10GbE Ethernet. On the other hand, Nicolae, B. et al. (NICOLAE; RITEAU; KEAHEY, 2014) used 32 nodes of the Shamrock experimental testbed of the Exascale Systems group at IBM Research.

It is important noticing that, although there is a difference between the nodes and servers terms in Figure 53, we do not change the term that the researcher exposes in their research. We replicate the same written term by the author, whether the researcher writes that their experiment was performed into a number of servers we wrote the number and then the "S" letter that means in legend "server" if the researcher wrote a number of nodes we wrote the number of nodes and changed the letter to the "N" and so on. As shown, the first term of the elements always relates to the quantity of used hardware and then the letter that this number refers.

MQ16: How are architectures proposed by researchers being built? In which context they are proposed? Where are those solutions being implemented? How is the size of the environments that evaluate architectures?

This question allows us to understand how authors are evaluating their solutions, in which context these solutions are being proposed, where these solutions are being implemented, and how big is the user environment for experimentation. We are also able to verify the class of these architectures and their size if it is being evaluated into a large-scale, small, or software simulated environment. It also allows us to identify which are these large-scale environments or which small environment researchers use to evaluate their architecture proposals or which are these simulated software used.

Before starting, we will clarify what the term "architecture" means since this term does not appear on the proposed characterization model Figure 57. There, we noticed the presence of three circles (e.g. Software, Hardware, and Storage System). When an author proposes a solution to improve I/O performance, that solution must be contained in one of these two primary circles, either a hardware solution or a software solution. Later, as this solution was proposed to improve some specific element, the direction of the arrow that leaves these primary circles directs which class of the object that this solution aims to improve. Therefore, the circle that receives the direction of the arrow is the class of the object that is being improved. There, we found that it is possible to make I/O improvements from several perspectives.

First level improvements are improvements that only include the lower-level elements (e.g. software and hardware) These improvements can be hardware to hardware (H2H-IO), hardware to software (H2S-IO), software to software (S2S-IO), and software to hardware (S2H-IO). However, we note that the storage system can be classified as a second-level system because it consists of software that manages its functioning and the hardware that makes it up. Thus, we note that researchers commonly propose solutions to improve storage systems. Therefore, these improvements can either be a software improvement to improve the storage systems I/O performance (S2SS-IO) or they can be a hardware improvement to improve the storage systems I/O performance (H2SS-IO). Given this, we created a name, that is, the name architecture, to symbolize that both S2SS-IO and H2SS-IO were proposed by the author.

Usually, the creation of architectural designs for storage systems is composed of both software and hardware and because of that instead of classifying the proposals as S2SS-IO and H2SS-IO, we simply classify them as an architectural proposal. We also note that commonly, when authors propose new designs for storage systems, they propose both new hardware designs and management software that must manage that hardware. We classify all of these new storage systems proposals as storage architecture proposals.

Hybrid storage systems proposals, that is, those that are composed of storage devices of different technologies, are examples that portray these architectural proposals well. To understand in which domain the architectures are being proposed, we divide them considering 4 domains (e.g. HPC, Big Data, Cloud and Storage Systems) In addition to the motivation given by the author when describing his work, we considered the following rules related to each domain to classify the papers.

Figure 54 addresses MQ16 *(How are the architectures proposed by researchers being built? In which context they are proposed? Where are those solutions being implemented? How is the size of the environments that evaluate architecture proposals?).*

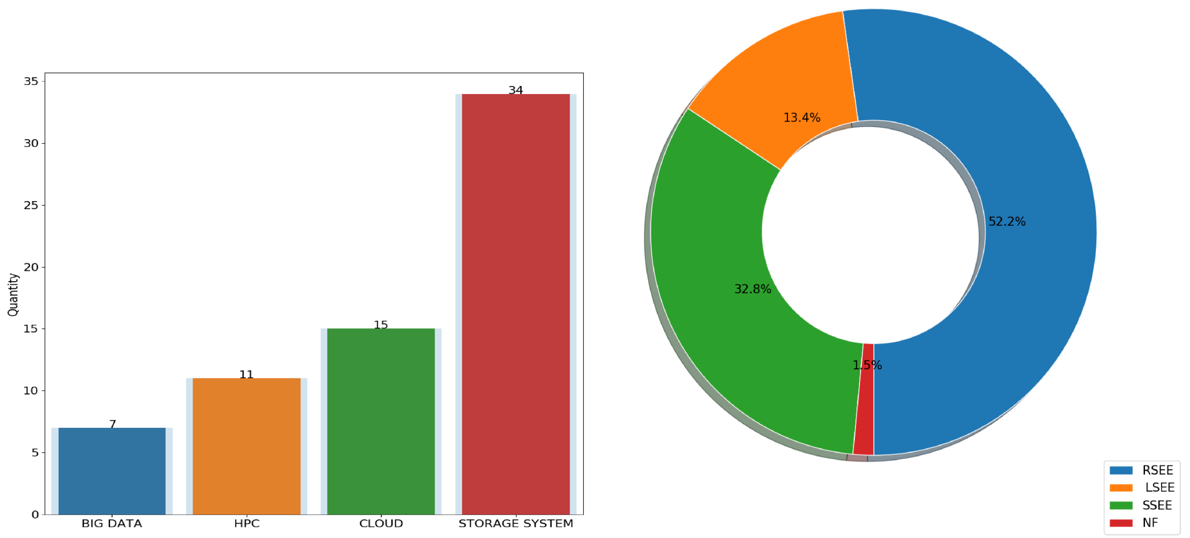
After reading and classify the architecture papers considering the terms expressed above, Figure 54a presents the domain of the author's researches. We noticed that the storage system was the domain that most papers were classified. We observed that half of these classified papers were relating and proposing hybrid, in terms of device diversity, storage systems. Below we discuss the main idea of these researches identifying the used devices and methodologies.

From the proposed architectures many of them were hybrid storage systems considering the usage of HDDs together to SSDs due to the poor random write HDDs performance (AL-WESABI; ABDULLAH; SUMARI, 2017), (WAN et al., 2012), (XIAO et al., 2012), (WANG; GUO; MENG, 2015), (STRUNK, 2012), (FENG et al., 2014), (XU; CHENG; CHEN, 2017), (YANG et al., 2013), (HUI et al., 2012), (LEE; JUNG; SONG, 2009), (CHANG; YU; CHUNG, 2018) (AL-WESABI; SUMARI; ABDULLAH, 2019). An Architecture that deals with the removal of the semantic gap through the management of cache behavior and flash memory. They also applied data deduplication to improve the usage efficiency of cache device (CHEN; CHEN; LU, 2015).

A software-defined fusion method for PCM and NAND flash memories was also proposed (LI et al., 2016). The mixing of flash memories (e.g. MLC and SLC), byte-accessible NVRAM and conventional volatile RAM as buffer caches was also considered (PARK; BAHN; KOH, 2009).

A heterogeneous storage system for backing file data in clusters considering HDDs, SSDs, and Network RAM was also proposed (MARKS; NEWHALL, 2017). A configurable cache architecture comprehensive workload characterization to find an optimal cache configuration for I/O intensive applications that use HDD, SSD and cache were also proposed (SALKHORDEH; EBRAHIMI; ASADI, 2018). The other architectures proposed were non-hybrid architectures including an only PCM-only storage system architecture (HAN et al., 2018), flash-only (LEE; KIM; MITHAL, 2014) among other solutions.

The papers domain that was classified as HPC domain also presented hybrid approaches. Petersen, T.K. and Bent, J. (PETERSEN; BENT, 2017) presented hybrid-flash arrays for HPC storage systems being an alternative to burst buffers. Wadhwa, B. et al. (WADHWA; BYNA; BUTT, 2018) proposed an object abstraction to explore storage mechanisms to enhance I/O performance. They explore how an object-based interface can facilitate the next generation of scalable computing systems. They implemented a SSD-based burst buffers that offer persistent edge storage. Through varying I/O interfaces,



(a) Architectural Domains (b) Experimental Environment

Figure 54 – Architectural Context.

file systems, types of NVM, and both current and future SSD architectures, Jung, M. et al. (JUNG et al., 2014) mapped numerous bottlenecks implicit in these various levels of the I/O stack. Prabhakar, R. et al. (PRABHAKAR et al., 2011b) constructed a dynamic data staging area for extreme-scale machines, they proposed a multi-tiered storage architecture that combines storage devices (e.g. DRAM, SSD) The proposal was evaluated into the Jaguar supercomputer at Oak Ridge National Laboratory. Other researchers focused on proposing burst buffers (BB) layers to increase I/O performance reducing burst I/O behavior (TANG et al., 2017), (WANG et al., 2014).

Big data hybrid storage systems (KRISH; ANWAR; BUTT, 2014), (OU et al., 2017) and (RUAN; CHEN, 2017). Raynaud, T. et al. (RAYNAUD; HAQUE; AÏT-KACI, 2014) develops an architecture that caches data in main memory combining the Cache-Only Memory Architecture (COMA) and the structural principle of Hadoop. It mainly transforms the main memory into an attraction memory that enables high-speed data access. Nijim, M. et al. (NIJIM; SAHA; NIJIM, 2014) developed hardware and software architectures that include parallel GPU, flash disks, and disk arrays devices.

Cloud hybrid storage systems were also proposed. Zhang, Z. et al. (ZHANG et al., 2018) proposed a log-based hybrid storage system, which comprises the PCM, flash memory-based SSD, and HDDs. Chen, H. et al. (CHEN; LEE; CHANG, 2016) proposed an I/O framework for multi-tiered storage in multi-tenant cloud-based shared storage environments. Li, D. et al. (LI et al., 2016) presented an SCM-vWrite architecture designed around SCM, to ease the performance interference of virtualized storage subsystem. Youn, Y. et al. (YOUN; YOON; KIM, 2017) presented a cloud computing burst system (CCBS) that uses NVM and composed of a unified table management module, data scoring module.

Yang, Z. et al. (YANG et al., 2017b) provide a hybrid framework of NVMe-based storage system (H-NVMe) that is composed of two VM I/O stack deployment modes "Parallel Queue Mode" and "Direct Access Mode". However, other architectures and methodologies were also found. Yin, J. et al. (YIN et al., 2017) proposed a storage scheme for object-based cloud storage systems. Ravandi, B. et al. (RAVANDI; PAPAPANAGIOTOU, 2018) proposed a framework called block software-defined storage (BSDS) that separates the data layer from control to automate the orchestration, deployment, and management of a storage system.

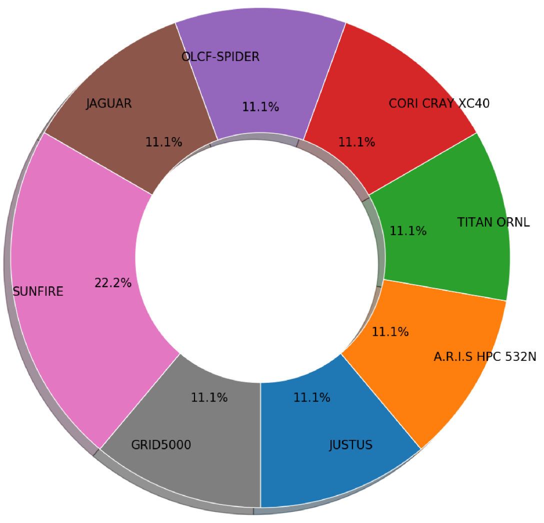
From these classified researches, the experimental environment was also collected and the general results are presented in Figure 54b. We noticed that 52.2 % of the experimentation was performed in **Real-Small Evaluated Environment (RSEE)**. Small environments are most accessible in laboratories around the world where these environments do not require much investment from the research department. However, we also noticed that much experimentation that corresponds to 34.8% of the total evaluated occurred in **Software Simulated Evaluated Environment (SSEE)**. Some experiments are easily performed using simulated environments and their execution time can be reduced due to some tools available in the configured simulation. Finally, we noticed that a small part of the researchers, more precisely 11.6%, executed their experimentation in **Large-scale Evaluated Environments (LSEE)**.

In the HPC domain, in some simulations and experiments, the use of large-scale environments is essential for the reliability of the results. In some cases, we do not find information on the author's research giving information about the user environment. Therefore, we classify 1.4% of the environments as a **Not Found (NF)**. In Figure 55 we verify which were these used environments.

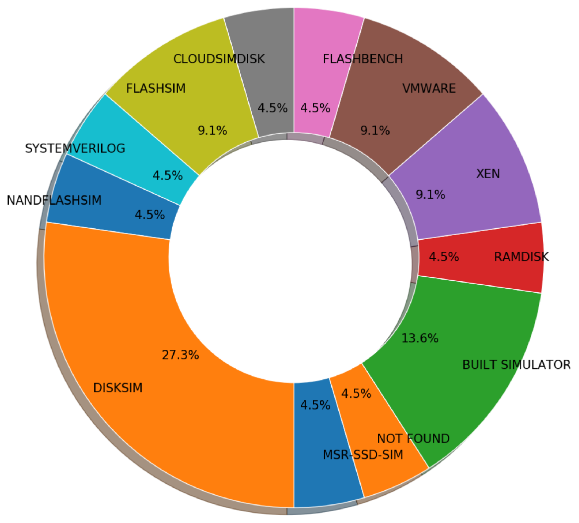
Figure 55 addresses the questions regarding the environment used by researchers when evaluating the new architectures. We noticed in Figure 55a that some large-scale environments that were used in S2SS-IO improvements were also used in architecture proposals (e.g. Jaguar (PRABHAKAR et al., 2011b), Cori xc40 (WADHWA; BYNA; BUTT, 2018), Titan (WANG et al., 2014), Sunfire (HE; SUN; FENG, 2014), (SONG; SUN; CHEN, 2011) and Grid5000(RAYNAUD; HAQUE; AÏT-KACI, 2014)).

Figure 55b presents the software used to simulate the environments. We verify that Disksim was also the simulated software that researchers had used at most in architecture proposals representing 27.3% of the total evaluations. Indeed, Disksim supports a quite of disk parameters and representations being the first option from researchers due to its efficiency and accuracy. The built simulators also represented a big portion representing 13.6% of the total evaluations.

Interestingly, we also noticed evaluations on the XEN hypervisor. Li, D. et al. (LI et al., 2016) proposed an architecture (SCM-vWrite) designed around SCM, to ease the



(a) Large-scale Used Environments for Architectures Proposals.



(b) Software Simulated Used Environments for Architectures Proposals.

Figure 55 – Used Environments.

performance interference of virtualized storage subsystem. Jo, H. et al. (JO et al., 2009) proposed a hybrid copy-on-write storage system that combines solid-state disks and hard disk drives. Their solution places a read-only template disk image on a solid-state disk, while write operations are isolated to the hard disk drive. Both authors evaluated their experiments using the XEN hypervisor.

Finally, Figure 56 addresses the small used environments. We verify that most evaluations were on servers and machines. As commented earlier, we do not classify the difference between machines and servers, instead, we replicated the author's term. Technically, a server can be a more robust environment, because, in most cases, it uses dedicated hardware modules (for example, storage, network, power, etc.)

The environment that uses the most number of nodes consists of 38 nodes. Some of these nodes have SSDs and 64 GB DDR4 memory. In this research, Enes, J. et al. (ENES et al., 2018) proposed a big data-oriented PaaS architecture with disk-as-a-resource. In this work, the scheduler offers disks as resources alongside the more common CPU and memory resources to provide a better storage solution for the user. We also identify two occurrences that have a 32-experimental environment node.

Wickberg T. et al. (WICKBERG; CAROTHERS, 2012) proposed a system that leverages high-throughput and decreases the cost of DRAM called RAMDISK Storage Accelerator (RSA). It provides an application-transparent method for pre-staging input data and commits results back to a persistent disk storage system.

Shankar, D. et al. (SHANKAR; LU; PANDA, 2016) proposed a burst-buffer that improves the performance of the I/O phase of Hadoop workloads running on HPC clusters.

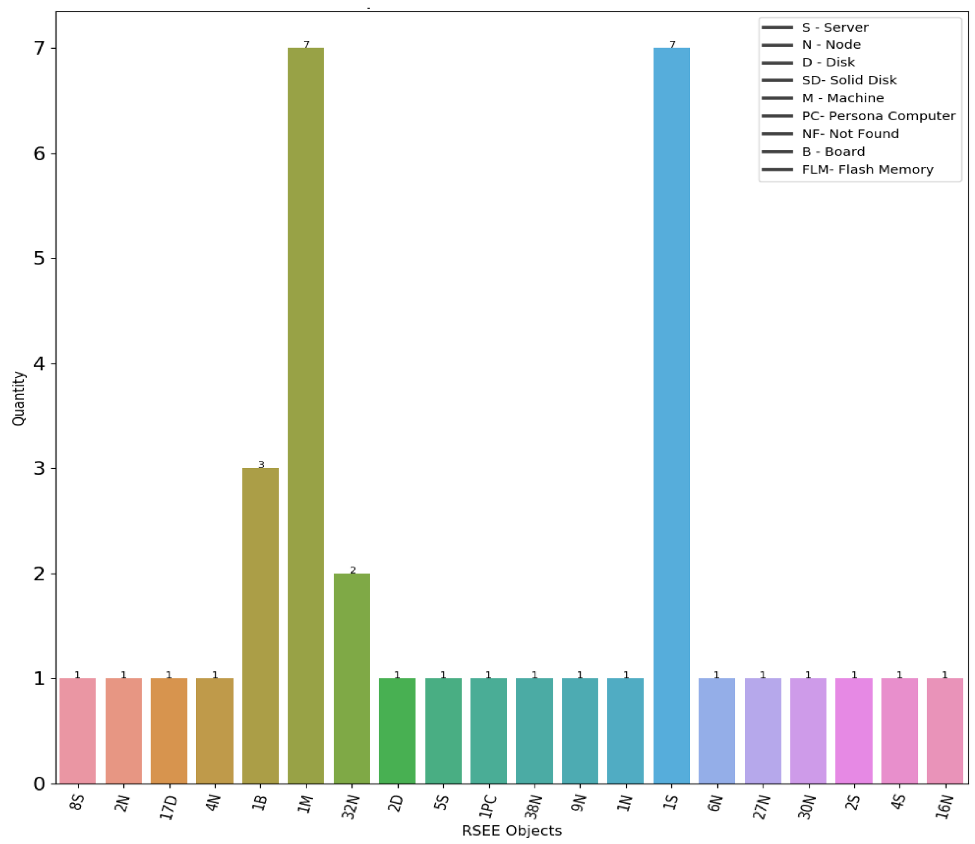


Figure 56 – Small Environments Used for Architecture I/O Improvements

We conclude by informing you that, some experiments were performed and proposed considering an FPGA board, They are:

Han, W. et al. (HAN et al., 2018) proposed a storage system composed of a PCM board that is coupled with a host platform with heterogeneous SoC (FPGA and ARM) via the FPGA mezzanine card(FMC) interface. It includes many PCM and an on-board FPGA ZedBoard platform that uses a Zynq-7000 programmable SoC as a processor.

Caulfiel, A. M. et al. (CAULFIELD et al., 2012) described storage hardware and software architecture (Moneta-D). It moves file system permission checks into hardware and uses an untrusted, userspace driver to issue requests also providing a private, virtualized interface for each process. It is composed of many elements over an FPGA board.

Ou Y. et al. (OU et al., 2017) proposed a high-performance PCIe SSD (Gemini) that was built over some hardware and software implementations. A page to block mapping FTL (PBFTL), an interface flash channel controller, a customized I/O stack, the scatter/gather DMA and the multi-queue architecture were elements that compose these hardware and software implementations

3.6 THREATS TO VALIDITY

As in any empirical experiment in software engineering, the analysis carried out and proposed in this manuscript suffers threats to validity. Some of these factors previously identified in the scope of the research are discussed and considered at this stage. As to issues that threaten the development of the study, they should be used throughout the project, to help achieve the most accurate result possible. Cook and Campbell (COOK; CAMPBELL, 1979) extent how to use the threats previously validated by Campbell in (CAMPBELL; STANLEY, 1963) for four types of occurrences, namely: *"completion, internal, construction and external"*. In this section, we investigate and discuss the main risks that this work presents.

It is important to understand that the basis of this research aims to identify studies that in some way have contributed to I/O performance improvement. I/O bottleneck storage systems are a well-known issue that affects computing systems that perform multiple IOPS and high-performance physical systems. The measurement proposed metrics found in this study that most impact computing systems refer to latency and throughput. Therefore, identifying scientific works that somehow contribute to the reduction of existing bottlenecks of these factors shapes and delimits the scope of this work. However, although the characterization model is applied to an I/O performance improvements domain, it is understandable that this model can be adapted for measurement and identification of other metrics evaluation.

Although the systematic analysis in the definition of articles was conducted with firmly established inclusion and exclusion rules (see Section 2.1.4), their characterization is subject to interpretation bias. Different ways of expressing a proposed content and different research themes sometimes turns difficult for readers to understand the real improvement object class that was proposed by the author. The use of a simple, but informative, acronym allows the reader to quickly assimilate what the research authors proposed and what it aims to improve. It could be understood as a text index, to which, through an acronym, an idea of what objects proposed and improved would be passed on.

The quantity of analyzed papers is also a factor that deserves attention. It could lead us to characterize researches differently from the real author's purpose. However, this fact does not invalidate the need for the characterizing model presented here, instead, it decreases the accuracy of the data and results presented. As mentioned in the data extraction process, as the reading aloud the article to be classified, it was immediately classified. A fact that aggravates this situation is related to the meaning of the word storage system. As mentioned earlier, a storage system can be either a data management software (e.g. file system, database, RAID system) or a physical data storage system composed of storage media (e.g. HDD, SSD, NVM, etc.). Because of that, the non-reading of the work in its entirety would open the possibility for a characterization that did not represent the researcher's real proposal. This is yet another reason why this model could be used by researchers before submitting a publication.

3.7 REPLICATION ANALYSIS

To enable the audit and encourage the usage of the classification presented above, a package with the analyzed data and its results was prepared and made available online on GitHub [5] platform.

3.8 FINAL CHAPTER CONSIDERATIONS

In this chapter, we presented a systematic literature mapping to identify researches that proposes I/O improvements in software, hardware, and storage systems. These researches were selected following a systematic literature mapping method. To increase its quality, this paper uses a framework that classifies all accepted researches according to its proposal object. During this process, 1705 articles were analyzed and a set of 517 articles were selected for further analysis. The time range covered in this work was from the last ten years of researchers' proposals to increase I/O performance. The results show a growing increment of the number of papers to solve this I/O performance issue and it may be a trend for the next years due to the increasing technological needs. Throughout 16 mapping questions. it was possible to identify the most common improved objects (e.g. software, hardware, and storage systems), which class these objects belong and in which size of environment they were evaluated.

[5] https://github.com/laerciopioli/ Systematic-Literature-Data.git

4 RELATED WORK

This chapter presents a brief review of the significant works of I/O performance. In the past, some authors have proposed results in I/O optimization approaches considering the I/O stack as a whole. Our initial search, referring to the contributions to the parallel I/O stack and workloads characteristics in huge infrastructures.

Saif et al. (SAIF; NUSSBAUM; SONG, 2018) presented an I/O tracer, called IOscope, for uncovering I/O patterns of storage management systems' workloads. Helping to achieve a better troubleshooting process, their solution contributes to having an in-depth understanding of I/O performance throughout, filtering-based profiling over fine-grained criteria inside the Linux kernel. They evaluated their proposal using two different databases, a document-based MongoDB and a wide-column Cassandra storage database. They achieved interesting results showing that the clustered MongoDB suffers from a noisy I / O pattern, regardless of the storage device used (HDDs or SSDs).

Daley et al. (DALEY et al., 2017) presented a performance characterization of scientific workflows considering the optimal usage of burst buffers. The authors analyzed the performance characteristics of burst buffers applying two scientific workflows and I/O benchmarks targeting the optimal usage of the burst buffer also identifying the peak I/O performance of the burst buffer. The used I/O benchmark was the IOR (INTERLEAVED. . . , 2016) and MDTest (MDTEST. . . ,). These I/O benchmarks are widely used in HPC and storage environments and are both MPI applications. IOR is used by many researchers to measure the peak performance of storage systems and MDTest the performance of storage system metadata operations. The two workloads used to increment their analyses were Community Access MODIS Pipeline (CAMP) and SWarp. The first one processes data obtained from MODIS (NASA. . . ,) satellite while the second uses raw images of the night sky data. Their work includes a methodology to analyze the performance of burst buffers. They analyze also considered bandwidth-sensitive and meta data-sensitive I/O workloads. Finally, they presented challenges referred to data management when introducing a burst buffer in scientific workflows.

Boito et al. (BOITO et al., 2018) shown interesting research in a five-year window on the parallel I/O for HPC environments presenting applications characterization and performance modeling. The authors focus on the parallel I/O and present the state-of-the-art in I/O optimization approaches as much as in many subjects that the I/O stack is composed. Some main presented components involved in the area, the common problems found in transmitting data through the nodes, and the techniques typically applied to achieve high performance are illustrated. Most proposals presented by the author consider solutions implemented on the software layer (e.g. caching/prefetching, I/O scheduling, Collective I/O, Requests aggregation, among others). However, the presented survey does not consider the importance of the wide variety of hardware technologies used by the storage nodes to save the data. In some cases, they just mentioned the two more common devices used to store data (e.g. HDD and SSD). Therefore, in this paper, although our proposed characterization model is broadly presented, it considers the hardware technologies used by HPC environments in storage auctions.

Calzarossa et al. (CALZAROSSA; MASSARI; TESSERA, 2016) presented a survey considering a characterization model considering the importance of the workload characterization and exploiting its importance in popular applications domains. Their focus is directed to workload from the web and with workloads associated with online social networks, video services, mobile apps, and cloud computing infrastructure. They also present and analyze a modeling technique applied for this characterization. Their proposed characterization model does not consider the three basic elements (e.g. software, hardware, and storage systems) like our presented model. They present studies in a cloud computing infrastructure, but their concern to the characteristics of cloud workloads.

Finally, Traeger et al. (TRAEGER et al., 2008) described and presented a survey considering a nine-year study examining a range of file system and storage benchmarks. They surveyed a range of 106 file-system and storage-related research papers in this study. They also described the positives and negatives qualities of both and presented a way to choose the appropriate benchmark for performance evaluation. As in the previous related works, the authors did not consider hardware characteristics and how it can influence the performance in an evaluation process of a storage system.

4.1 FINAL CHAPTER CONSIDERATIONS

In this chapter, we present researches that propose solutions regarding the development and improvement of I / O performance. Each of them presented a different perspective and solutions to deal with this issue. We have also shown that the most proposed solutions consider a specific studying regarding the workflow but all of them do not present solutions considering hardware in their solutions as we treated here.

5 PROPOSED MODEL

A characterization model for classifying research works on I/O performance improvement is presented in this chapter. Nowadays, the gap of development between CPU and storage technologies is too big, thus making the storage layer the bottleneck of the overall systems. Because of that, researchers are proposing different approaches targeting improvements in the storage layer. Some authors combine different hardware technologies to fill this issue, others develop a solution on the software layer trying to improve the algorithms and systems to better fit the applications and increase IOPS. Some authors are proposing new architectures and storage systems combining hardware devices and software applications through different approaches. Considering these observed characteristics, our model is composed of three main elements (i.e. software, hardware, and storage systems) that represent these scenarios.

When researchers propose solutions to improve I/O performance in storage environments, we verified that the proposed solutions belong to a specific domain. In this model, each circle means an element domain that is related to the author's proposal object. The software circle symbolizes an improvement made by a researcher where the object that is being proposed as a solution is any programmable solution (e.g. algorithm, method, frameworks, etc). The hardware circle symbolizes an improvement made by a researcher where the object that is being proposed as a solution is any physical component or something palpable (e.g. device, chip, accessory, etc). The third element, storage systems, only receive improvements from the previous two elements. In this model, we defined that a storage system should necessarily be composed of hardware and software elements. The first two elements on the down layer can relate to each other in both directions and both of them can suggest a solution as an object to improve the I/O performance on an entire storage systems platform.

5.1 DEFINITION

In this section, we present the necessary elements for the definition of the proposed model. First, is presented a set of axioms that were considered when creating and developing this model. Second, a graphical image that relates to this model is presented. Third, each component presented on the graphic representation individually is explored.

5.1.1 AXIOMS

1. The model is composed of two types of elements (e.g. circles and arrows).
    a) A circle represents a class.
    b) An arrow represents the relationship between different or equal classes.

c) The beginning of the arrow informs which is the class of the proposed object and the end of the arrow informs which is the class of the improved object.

2. There are at least three classes (i.e. software, hardware, and storage systems).
3. Each arrow should necessarily begin and end considering a proposed and improved object.
4. The author's solution should necessarily be proposed to improve I/O performance.
5. An author's research can propose more than one improvement relationship.
6. Storage Systems, that relates an agglomerate of devices and software, can not be proposed to improve I/O performance on any object.

5.1.2 GRAPHICAL REPRESENTATION

A graphical image that relates this model is presented in Figure 57.

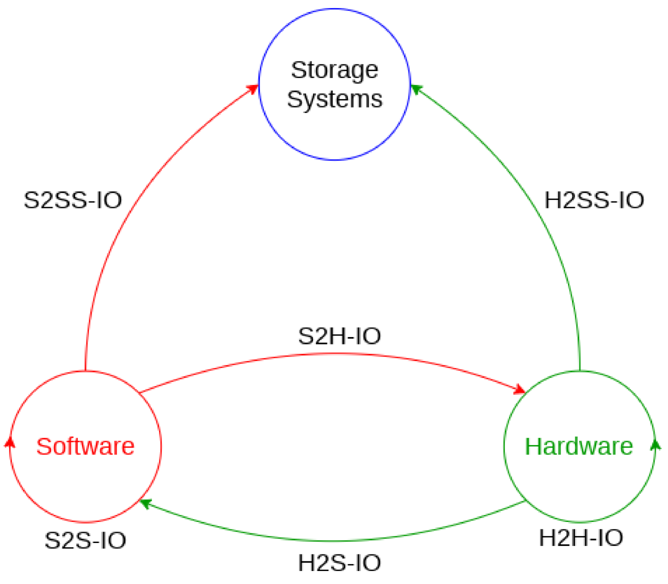


Figure 57 – Characterization of I/O Performance Improvements.

- The software circle symbolizes an improvement made by a researcher where the object that is being proposed as a solution is any programmable solution. It includes anything that can be translated into code lines, algorithms, methods, frameworks, etc. All those elements present the same characteristic and usually are introduced into other software or hardware as an improvement object. Improvement codes into PFS, databases, frameworks, operational systems, big data, and HPC applications are examples of common objects targeting I/O improvements on software. It is also possible to improve hardware devices such as a storage device, a network drive, or any board or internal or external device with a software object, for instance, firmware can be mentioned.

- Hardware circle symbolizes improvements made by a researcher where the object that is being proposed as a solution is any physical component or anything palpable. It includes devices, chips, electronic accessories, memories, storage cartridges, etc. It also could be used to improve other hardware objects or software. Usually, a new class of memories is used to improve another hardware I/O performance component. It is also possible to increase software I/O performance by providing hardware components. For instance, proposing the usage of storage devices such as SSDs, PCMs, or nvSRAM to decrease PFS latency or the use of Omni-Path or Infiniband interconnect network device into HPC clusters to increase the throughput of PFS.

- The third element, storage systems, only receive improvements from the previous two elements. It is worth noticing that, in this model, we defined that a storage system should necessarily be composed of hardware and software elements. Thus, we defined a storage system being an element that is derived from the other two elements. Because of that, it makes no sense storage systems being proposed as a solution to improve I/O performance on other objects. The first two elements on the down layer can relate to each other in both directions and both of them can suggest a solution as an object to improve the I/O performance on an entire storage systems platform.

Tables 10 and 11 present previous works targeting I/O improvements on software, hardware and storage systems. Each arrow shown in Figure 57 is related to a column shown in such tables. Emerging technologies makes studying storage technologies challenging due to low-level concepts. Throughput issues are commonly found in applications when using the I/O layer to perform I/O operations such as read and write. The used hardware might play a crucial role in the execution of these operations.

In a previous related work (PIOLI; MENEZES; DANTAS, 2019), we presented a description of other research works that were not presented in these tables but also target improvement of I/O performance on storage devices and systems. It is important noticing that, there is no additional intention in the presentation of the works exposed in Tables 10 and 11 besides showing that they have purposes in improving a similar class of objects.

These papers are part of a systematic mapping that is presented in the next chapter and targets I/O improvement as much as in storage devices, software, and storage systems in the last 10 years. These solutions were divided into groups and are directly related to each arrow presented in Figure 57.

Table 10 – Software I/O Improvements.

S2H-IO	**S2S-IO**	**S2SS-IO**
(YANG; PEI; YANG, 2019)	(WU; HUANG; CHANG, 2018)	(ZHOU; CHEN; WANG, 2018)
(JI et al., 2017)	(YANG; LIU; CHENG, 2017)	(DU et al., 2015)
(RAMASAMY; KARANTHARAJ, 2015)	(HUO et al., 2015)	(OH et al., 2012)

Table 11 – Hardware I/O Improvements.

H2H-IO	**H2S-IO**	**H2SS-IO**
(KIM et al., 2015)	(NAKASHIMA; KON; YAMAGUCHI, 2018)	(KANNAN et al., 2011)
(LEE et al., 2017)	(MOON et al., 2015)	(BU et al., 2012)
(LEE et al., 2014)	(BHATTACHARJEE et al., 2011)	(DAE-SIK; SEUNG-KOOK, 2009)

5.1.3 Software Solution to Improve I/O Performance on Hardware (S2H-IO)

The first column "S2H-IO" presented in Table 10 is an acronym for "Software solution to improve I/O performance on Hardware". These papers propose a software object to improve I/O performance on a device. Because of that, they could be characterized as a software solution to improve I/O performance on a hardware device. The arrow that represents this subject in Figure 57 leaves the red circle software and arrives in the green circle hardware.

Some papers that can be used to illustrate this class are presented in Table 10. Yang et al. (YANG; PEI; YANG, 2019) proposed a solution called WARCIP to tackle the write amplification problem which is an inherent physical property of flash memory devices. Chang et al. (JI et al., 2017) present a novel I/O scheduling scheme, called MAP+, for embedded flash storage devices. Ramasamy et al. (RAMASAMY; KARANTHARAJ, 2015) proposed an algorithm called random first flash enlargement to attack and improve the write operation of flash-memory-based SSDs.

5.1.4 Software Solution to Improve I/O Performance on Software (S2S-IO)

The second column "S2S-IO" presented in Table 10 is an acronym for "Software solution to improve I/O performance on Software". These papers consider some software object as a solution to perform its improvement. It is important noticing that, although the improvements are from software to software, they take into account the storage technologies that they are using. The arrow that represents this subject in Figure 57 leaves the red circle software and arrives in the same red circle software.

Some papers that can be used to illustrate this class are presented in Table 10. Wu et al. (WU; HUANG; CHANG, 2018) proposed a data placement method that provides databases with high I/O performance by considering an integrated mechanism and migration rule to move high-priority data between HDDs and SSDs. Yang et al. (YANG; LIU; CHENG, 2017) proposed an approach aiming at solving duplication in VM disks. Content look-aside buffer (CLB) was provided and implemented on the KVM hypervisor. The approach provides a redundancy-free virtual disk I/O and caching. Huo et al. (HUO et al., 2015) proposed a cooperative caching management algorithm to improve the performance of file systems using SSD and DRAM. The method called ACSH looks for metadata I/O file systems reducing the write traffic on SSD.

5.1.5 Software Solution to Improve I/O Performance on Storage Systems (S2SS-IO)

The third column presented in Table 10 is concerned about I/O improvements targeting storage systems. It is worth noticing that, in this characterization, we consider storage systems as a group of technologies and software which work together and asynchronously. As well as groups "S2H-IO" and "S2S-IO" this group considers some software object as a solution to perform its improvement, but unlike them, the object which is receiving the improvement is composed of software and hardware respectively, in other words, storage systems. The "S2SS-IO" acronym which means "Software solution to improve I/O performance on Storage Systems" relates improvements targeting I/O performance into a storage system through a software solution. In Figure 57 these improvements are presented as the arrow that leaves the red circle software and arrives in the blue circle storage systems.

Some papers that can be used to illustrate this class are presented in Table 10. Zhou et al. (ZHOU; CHEN; WANG, 2018) proposed an algorithm that can make better use of heterogeneous devices for storage systems and is based on consistent hashing. Du et al. (DU et al., 2015) proposed a balanced partial stripe (BPS) write scheme to improve the write performance of RAID-6 systems. Oh et al. (OH et al., 2012) proposed a schema that organizes the cache space into reads and writes, and manages these spaces according to the workload characteristics for improving the performance of hybrid storage solutions.

5.1.6 Hardware Solution to Improve I/O Performance on Hardware (H2H-IO)

The first column of Table 11 "H2H-IO" which means "Hardware solution to improve I/O performance on Hardware" is concerned about hardware improvements. Different from the previous one, the papers here propose a hardware object to improve I/O performance on another hardware device. Many contributions are daily proposed to increase hardware performance by introducing new technologies into devices. The arrow that represents this improvement subject in Figure 57 leaves the green circle hardware and arrives in the same green circle hardware through an auto relation.

Some papers that can be used to illustrate this class are presented in Table 11. Kim et al. (KIM et al., 2015) proposed the insertion of the frequency-boosting interface chip (F-Chip) into the NAND multi-chip package (MCP) including a 16-die stacked 128Gb

NAND flash. Kim et al. (LEE et al., 2017) proposed the design of which uses an on-chip access control memory (ACM) introducing any type of on-chip non-volatile memory into the micro-controller of an SSD. Lee et al. (LEE et al., 2014) proposed a stacked DRAM with a micro bump interface. They built a High-Bandwidth Memory (HBM) introducing four DRAM memories into a chip-on-wafer.

5.1.7 Hardware Solution to Improve I/O Performance on Software (H2S-IO)

The second column of Table 11 - "H2S-IO" which means "Hardware solution to improve I/O performance on Software" consider some hardware object as a solution to improve I/O performance of some software. In this case, different hardware and technologies are used as a solution to improve applications' I/O performance. In Figure 57 it is presented as the arrow that leaves the green circle hardware and arrives in the red circle software. It is worth noticing that, this approach is less frequent but still important in presenting it.

Some papers that can be used to illustrate this class are presented in Table 11. Nakashima et al. (NAKASHIMA; KON; YAMAGUCHI, 2018) improve I/O performance of a large scale DNA application using SSD device as a cache. Moon et al. (MOON et al., 2015) optimize the Hadoop MapReduce framework with high-performance storage devices. Bhattacharjee et al. (BHATTACHARJEE et al., 2011) used SSD buffer pool to enhance recovery and restart in a database engine.

5.1.8 Hardware Solution to Improve I/O Performance on Storage Systems (H2SS-IO)

Finally, the third column presented in Table 11 is concerned about I/O improvements targeting storage systems. The acronym "H2SS-IO" means "Hardware solution to improve I/O performance on Storage Systems" and considers a hardware object as a solution to improve the I/O performance of a storage system. In Figure 57 the arrow that leaves the green circle hardware and arrives in the blue circle storage systems represent these improvements.

Some papers that can be used to illustrate this class and are presented in Table 11. Kannan et al. (KANNAN et al., 2011) proposed using a nonvolatile random access memory (NVRAM) to enhance the memory capacities of computing and staging nodes. Each node has an additional Active NVRAM component. Bu et al. (BU et al., 2012) introduce a Hybrid SSD approach that combines DRAM, phase changed memory (PCM), and flash memory into a hierarchical storage system. Dae-sik et al. (DAE-SIK; SEUNG-KOOK, 2009) designed and analyzed a high-end class DRAM-based SSD storage using DDR-1 memory and PCI-e interface.

5.2 FINAL CHAPTER CONSIDERATIONS

In this chapter, we presented our characterization model for classifying research works on I/O performance improvement. We highlighted its importance presenting the existing gap of development between CPU and storage technologies with increases I/O bottleneck and applications performance. The model targets to classify researchers' proposals targeting improvements on the storage layer. The discussion about the model targeted the author's combination throughout different approaches to fill the I/O bottleneck presented on these storage systems. Some authors presented hardware solutions employed on the underneath storage layer, others develop a solution on the software layer trying to improve the algorithms and systems to better fit the applications and increase IOPS. Our model is basically composed of three main elements (i.e. software, hardware, and storage systems) that represent these scenarios.

6 EXPERIMENTAL RESULTS

Our experimentation process analyzes the throughput and latency rate when performing I/O operations (e.g. read and write) in a huge experimental environment. It considers 10 different parameters and produces results from 36 different scenarios that we expose in this document. In this empirical process, we are generating results that symbolize research works that target improvements in the I/O performance in the storage layer. This validating process intends to encourage the overall researchers to keep proposing solutions to fill this huge bottleneck that we have in those environments nowadays. The next sections present all the necessary information used in this process. Section 6.1 displays the environment and presents the hardware we used to perform it. Moreover, section 6.2 presents the factors we use on the experimentation process and relate some of these factors to the elements presented in Figure 57. In section 6.3, we show the benchmark and the parameters used to generate the results. Finally, sections 6.5 and 6.6 present throughput and latency results and discuss it finalizing the experimental evaluation overall. The instructions that relate how the information was disposed of in these two sections is presented in 6.4.

6.1 EXPERIMENTAL ENVIRONMENT

Experiments were carried out in the "dahu" cluster, from the Grid'5000 (GRID5000, 2020) testbed. Each of the 32 nodes is equipped with 2 CPUs (16 cores/CPU), 192 GiB RAM, 240 GB SSD, 480 GB SSD, and a 4 TB HDD. A CentOS 7 (kernel 3.10.0) operating system image was deployed on each node, with an ext4 file system. In these experiments, 16 nodes were used as compute nodes (CNs), generating I/O requests using the IOR-Extended (IORE) performance evaluation tool (INACIO; DANTAS, 2018) over MPICH 3.0.4, and 8 nodes as storage nodes (SNs), in which an OrangeFS 2.9.7 PFS was deployed.

6.2 EXPERIMENT FACTORS DEFINITION

In this section, Table 12 presents the factors we used to perform the experiments. We also relate some of these factors with the elements presented in Figure 57.

Table 12 – Factors Considered in this Experiment

Factor	**Values**
Storage Approach	HDD, HDD+SSD, SSD
Linux Scheduler	CFQ, noop, deadline
Number of Tasks	32, 64
Access Pattern	sequential, random

- We considered three hardware approaches to store data and metadata because we believe that the way you organize those technologies might influence the performance of your application. Firstly, we considered storing both data and metadata on HDD devices (HDD-only). After, we considered storing data on HDD and metadata on SSD (HDD+SSD). Finally, we stored both data and metadata on an SSD device (SSD-only). The approaches above presented relate the hardware storage devices used in the experiment with the green circle hardware in Figure 57.

- Many contributions are performed by researchers to improve schedulers because I/O schedulers play an essential role in those environments. We considered three Linux schedulers on this experimental phase: Complete Fairness Queuing (CFQ) (AXBOE, 2004), Deadline and Noop. Schedulers are algorithms that distribute works such as threads, processes, data flows, etc. to computational resources. They are, normally, a programmable method with some main idea to distribute these works. This factor relates schedulers to the red circle software presented in Figure 57.

- The number of tasks participating in the test was considered in this experimentation. DISC and HPC applications have a huge variability of workloads and characteristics, and we choose 32 and 64 as the number of tasks because they could influence the performance results of scientific applications.

- Finally, another two factors were considered on the experimental effort, relating to data access pattern. The data access pattern is among the most important characteristics of disk drive workloads because it is related to the disk service process (RISKA; RIEDEL, 2006). The workload access patterns used in this experimentation could be "random" or "sequential" for each scenario.

In summary, we conducted experiments with a total of 36 different scenarios where 3 came from different approaches to store data and metadata, 3 came from different I/O schedulers used for servicing requests, 2 came from the used the number of tasks, and 2 came from the data access pattern used by the benchmark. To improve the results, each experiment was performed 5 times and at the end, the average of them was calculated as the final result.

6.3 IORE BENCHMARK

IORE benchmark is a unified and flexible tool for performance evaluation of modern high-performance parallel I/O software stacks and storage systems (INACIO; DANTAS, 2018). It is based on the famous I/O benchmark IOR (INTERLEAVED..., 2016). It supports a whole experimental variety of workloads and focuses on meeting I/O research works requirements on complex and reproducible experimental workflows. It is guided by an experiment-driven execution concept. Each experiment is composed of one or more runs, which, in turn, consists of a set of configuration parameters. These parameters define the characteristics of the I/O workload to be generated on the experimentation. Offset-based workloads such as request and block size are provided by IORE. In the end, a performance statistics file is exported. The tool exports the collected performance metrics to files after experiment completion.

In each experiment we configured the scheduler we used on data servers (CFQ, Deadline, and Noop), the way we were storing data and metadata on the storage devices (data and metadata on HDD, data on HDD and metadata on SSD, data, and metadata on SSD) and the characteristics of the workload we had set into the IORE configuration file ("num_tasks" and "access_pattern"). Inside the IORE configuration file, we set up to output 4 scenarios, named as a number in a column "run_id", in which we commuted changing the parameters "num_tasks" and "access_pattern". The parameters possibilities were 32 and 64 for the num_tasks and "random" and "sequential" to the "access_pattern". The data size we use for this experiment were 32 MiB with a request size of 2 MiB for each I/O operation process.

6.4 EXPERIMENTAL RESULTS

Results from our experimental effort are presented in the next sections. Although these results could be analyzed by different approaches, they are discussed from two perspectives, which are storage and scheduler perspectives. The first group is concerned to present results from a storage perspective relating to the hardware approach defined on the classification model presented in Figure 57. Hence, researchers could increase the I/O layer considering other devices and technologies. The second group presents results considering the schedulers presented on Linux operational system. In such a way, these results could lead researchers to keep increasing the schedulers placed on systems and consequently increasing the I/O layer through a software proposed object.

Before starting, it is important to discuss how the data and information were presented in these graphics. In each figure presented, 24 outcomes for the read and write operations are shown. In each bar plot, the y-axis refers to the average, throughput in section 6.5 and latency in the section 6.6, in MiB/s, and the x-axis refers to scenarios combining different numbers of tasks and access patterns. Further, hatched bars denote read operations and non-hatched bars denote write operations. Finally, different colors were used to contrast results observed with distinct I/O schedulers.

The results were also summarized by two types of tables which allows the readers to analyze the data from different perspectives. Firstly, the tables presented after each figure represents the throughput and latency average for all scenarios presented by each previous figure. At the end of each section, the images previously presented were condensed and grouped through four additional tables. Through these final tables, we can compare all the elements side by side respecting each scenario.

6.5 THROUGHPUT ANALYSIS

This section presents the results considering the throughput parameter. Here we provide information from a storage and scheduler perspective relating to the hardware and software approaches defined on the classification model presented in Figure 57.

6.5.1 Storage Overview

In this topic we are interested in understanding how each storage approach behaves when switching the schedulers presented on Linux.

Figure 58 presents the average throughput for the I/O requests when storing both data and metadata on the HDD device switching then the I/O schedulers. It is possible to verify that, in all cases, the throughput value for the read operation is bigger than the write operation no matter which scheduler we use. Some possible factors could lead us to understand these results. Actually, reads can be faster than writes, mainly, because, they are made on a date already read into memory, rather than doing it from disk. I/O operations in HDDs are very costly due to the inherent mechanical components inside the device. The mechanical arm placed into HDDs takes a considerable range of time positioning the arm head to the desirable disk sector when seeking data. It is worth noticing that, random operation increases this seeking gap time. Factors related to the operating system and the way it handles your hardware could also straightly imply. Another factor we could suggest is the way the file system performs operations. Usually, to read a file the file system needs to traverse the directory tree until the data and then read the file loading it into memory. To write a file, the same operation through the tree should be performed and after finding the path, write the data to the desired local, but, differently from the read operation, the file system should update the metadata related to the written file.

Table 13 presents the average of throughput value for all the scenarios presented in Figure 58. We present this average because we believe that DISC and HPC applications can treat and use heterogeneous kinds of data with different access patterns and a number of tasks on the same application. It is worth noticing that, in Table 13 when storing both data and metadata on the HDD device the scheduler that presents the highest average throughput value considering read operation is the Noop and when performing the write operation is the CFQ scheduler.

These values lead us to see that schedulers might perform and present different results depending on how is the application workload characteristic. The difference between CFQ and Noop scheduler throughput for a read operation is very notorious and depending

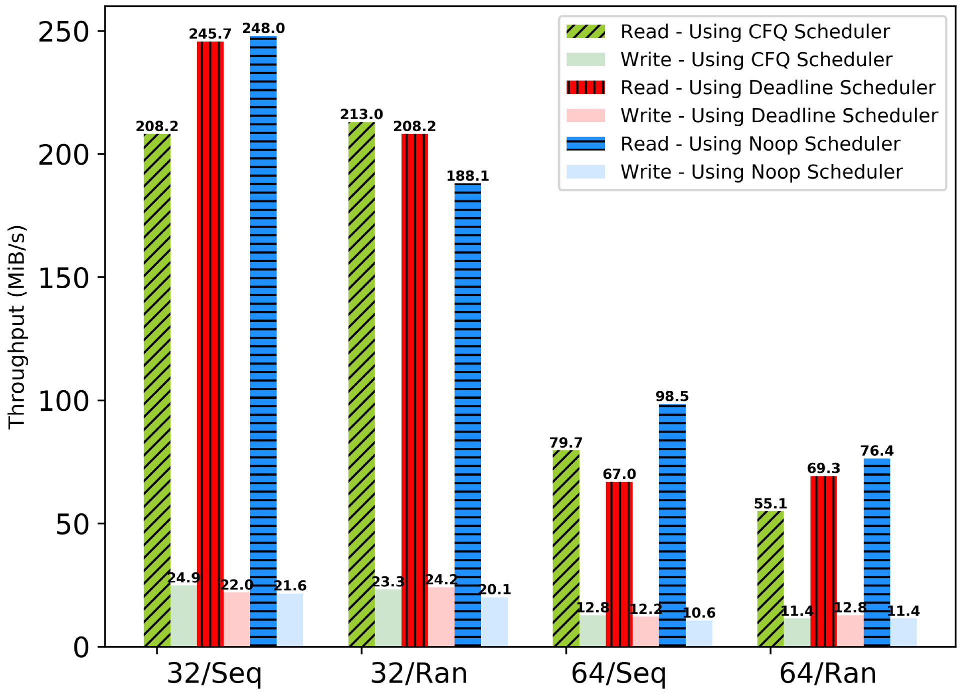


Figure 58 – Throughput Analysis Storing Data and Metadata on HDD.

Table 13 – Average in MiB/s of Throughput Presented in Figure 58 (Throughput Analysis Storing Data and Metadata on HDD).

Scheduler	Read	Write
CFQ	139	18,1
Deadline	147,55	17,8
Noop	152,75	15,93

on the application, its performance could be increased just by choosing a scheduler that fits better than the other. Although the difference is not as big as the reading operation, it is noteworthy that these schedulers also presented the highest range difference when writing files into PFS. The choice of schedulers could be justified depending on the mandatory operation of an application considering these reading and writing. If the application is characterized by one of them, these results could be used to help make the best decision. However, if the application performs both operations oftentimes, it will be necessary to make a calculation and determine which scheduler to choose to obtain the best advantage

In Figure 59 we can analyse the throughput value using a hybrid storage approach (HDD+SSD). In this approach, data were stored on HDD and metadata on an SSD device. As presented in Figure 58, in Figure 59 the I/O schedulers were also switched. Comparing Figure 58 to the Figure 59, we notice that the read operations when using the access pattern random were improved for all schedulers using the SSD device to store metadata instead of the same HDD. The device properties indicate this result. Performing random operations on HDDs is very costly due to the inherent mechanical components presented inside the device. The mechanical arm, when in random operation, spend much time moving to the right disk sector when seeking data. It consequently decreases the data throughput.

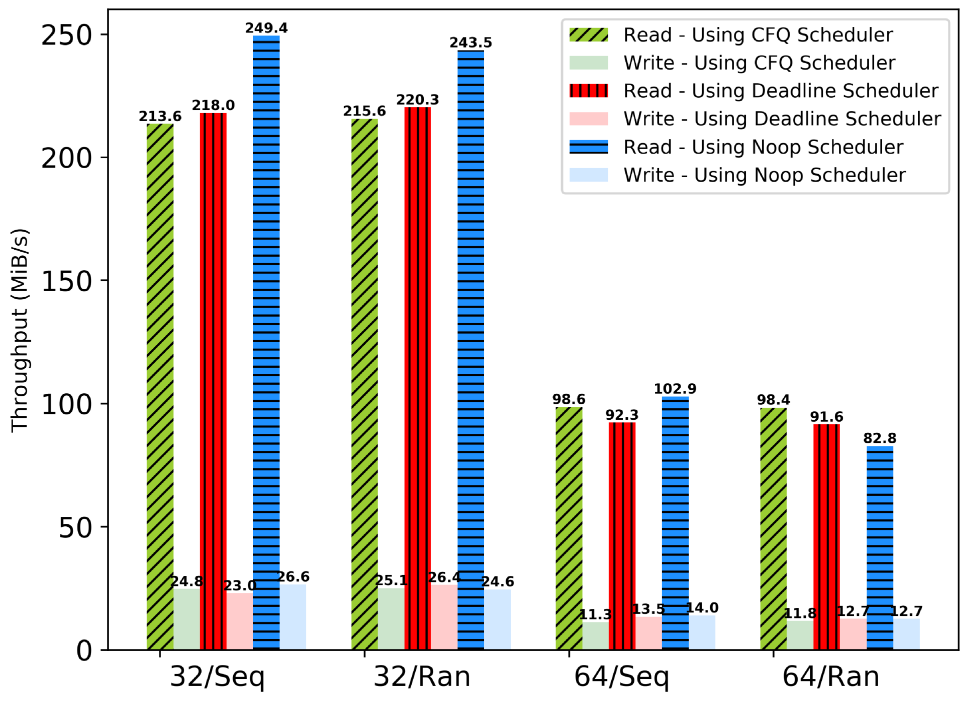


Figure 59 – Throughput Analysis Storing Data on HDD and Metadata on SSD.

Comparing with Table 13, Table 14 has the throughput when performing read and write operation for all schedulers bigger than when using the HDD-only approach. Considering write operation, except for Noop scheduler which presents a variation of 3.55 MiB/s, the other two schedulers, CFQ and Deadline, presented a closer and less than 1.5 MiB/s throughput value to the same operation. All throughput values presented in Table 14 are bigger than the values presented in Table 13 for both read and write operations. The respective difference throughput in MiB/s for reading operation are 17.55, 8, and 16.9 respectively. Through these results, we verify that read-based I/O applications throughput can be increased by many orders using this storage approach.

Table 14 – Average in MiB/s of Throughput Presented in Figure 59 (Throughput Analysis Storing Data on HDD and Metadata on SSD).

Scheduler	Read	Write
CFQ	156.55	18.25
Deadline	155.55	18.9
Noop	169.65	19.48

Depending on which operation an application applies more frequently and the access pattern used, these results could help an architect manager to choose the best choice between these two storage configurations. Considering an application that writes data through random access pattern, the selected approach does not have much influence on the throughput rate. However, if the application is read-based, perhaps the best storage configuration could be the last one, which, by our experiments, had increased the throughput rate. On the other hand, the sequential access pattern does not impose a specific scheduler to be chosen, it will depend on which is the task number that an application is based on. In general, the hybrid approach presented a bigger throughput rate for both 32 and 64 task number.

In Figure 60 was stored both data and metadata on SSD. SSDs are devices that are composed of NAND flash memories instead of magnetic disks. Depending on the quantity of Flash device that is available to the storage manager, these three presented options will be allowed to be implemented. However, as we already analyzed the two first storage configurations, we focus this analysis on the extreme case that is when using only an SSD device in your storage configuration. Comparing Figure 58 with Figure 60, we notice that,

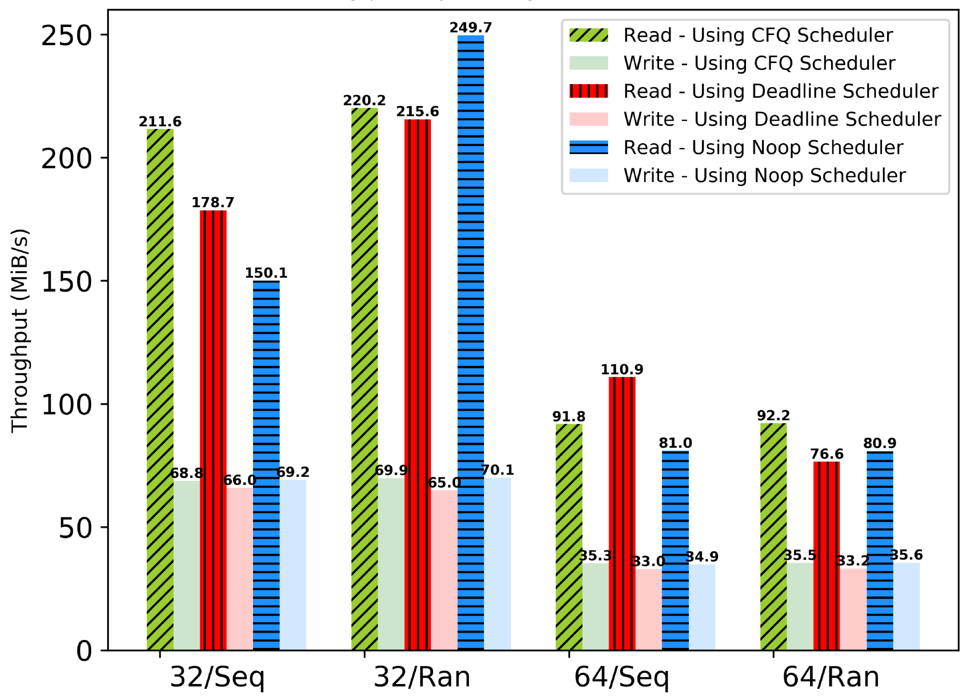


Figure 60 – Throughput Analysis Storing Data and Metadata on SSD.

in general, considering 32/seq configuration, the throughput rate when reading seems to be smaller for the flash based approach than for the HDD-only approach. The unique case where this is not true is when using the CFQ scheduler. However, if the access pattern used by the application is random, for all size tasks, the usage of flash memory seems to increase the throughput rate. It also occurred whit a small difference between the hybrid approach and SSD based approach. Nonetheless, the usage of flash memories to support 64 size tasks had also increased throughput rate, except for the noop scheduler that does not follow this pattern.

Perhaps, the most expressive results are related to the write operation. The throughput of the write operation was closer in Figure 58 and Figure 59 but the usage of only flash devices improved the write results considerably. The throughput in MiB/s when using only SSDs devices was higher compared to the previous ones. These results were expected due to the characteristics directly related to the additional mechanic component presented in HDDs. The average HDD latency is around 13ms depending on the model and the rotational speed. Although the schedulers have different methodologies to perform their operation, the results were similar for all previous cases. SSDs are composed of NAND flash memories and the time to traverse the directory tree with this technology until the right place is much smaller than when using HDDs. It is worth noticing that, an application that performs a large quantity of write operation, the usage of flash devices might increase its I/O performance considerably. The throughput rate when reading did not suffer significant variations and these results can be verified in Table 15.

Table 15 – Average in MiB/s of Throughput Presented in Figure 60 (Throughput Analysis Storing Data and Metadata on SSD).

Scheduler	Read	Write
CFQ	153.95	52.38
Deadline	145.45	49.3
Noop	140.43	52.45

Using this storage configuration, the schedulers seem to present a smaller throughput when reading. It makes sense because these tables were calculated using the arithmetic mean from all scenarios. However, the throughput when writing was increased significantly compared to the other two approaches presented earlier. The throughput value in MiB/s was increased more than twice for this operation in all configuration cases. These results could lead us to argue that if an application targets higher throughput and is write-based, perhaps consider this last storage approach would be interesting. On other hand, if the application is read-based and throughput is the target, a hybrid storage configuration can be chosen, thus saving costs instead of choosing a homogeneous flash-based storage environment.

Storage Overview Discussion

We summarized Figure 58, Figure 59 and Figure 60 conjointly, comparing how each Linux scheduler behaved in each storage approach in Tables 16, 17, 18 and 19. For instance, the values presented in Figure 58 shown the throughput using the HDD-only approach to store data and metadata. These values were distributed among the four Tables 16, 17, 18 and 19 throughout the column HDD. These expressive results were distributed respecting each task number and access pattern blocks. It allowed us to compare the Linux scheduler's throughput performance considering each storage approach side by side. The numbers within each table mean: (1) the highest throughput, (2) the intermediate throughput, and (3) the lowest throughput. It allows us to verify in which storage scenario the scheduler had its better throughput performance. As the write values are closer to each other, we concentrate on analyzing read operation.

	HDD	**HDD+SSD**	**SSD**
CFQ	3	1	2
Deadline	1	2	3
Noop	2	1	3

Table 16 – Throughput Storage Overview Discussion - Grouped Results by Number of Tasks 32/Seq

	HDD	**HDD+SSD**	**SSD**
CFQ	3	2	1
Deadline	3	1	2
Noop	3	2	1

Table 17 – Throughput Storage Overview Discussion - Grouped Results by Number of Tasks 32/Ran

Tables 16 and 17 present values considering the number of tasks equal to 32. Table 16 relates values for sequential access pattern while Table 17 relates for random access pattern. It shows that the hybrid approach presented the highest throughput value compared to the HDD-only and only-SSD approaches for the selected schedulers. The hybrid approach, in 32/Seq configuration, presented a high throughput twice represented by a number (1) and one middle value represented by a number(2). On the other hand, we verify in Table 17 that the SSD-only approach presented the highest value rate with the two highest values and one middle value. When changed the access pattern from sequential to random, the usage of the SSD-only approach increased significantly. We also noticed that when using the HDD-only approach in 32/Ran, all schedulers presented the worst throughput compared to the other storage methods. This is an interesting result because we confirm a common sense that HDDs do not present good performance when faced with random accesses.

	HDD	**HDD/SSD**	**SSD**
CFQ	3	1	2
Deadline	3	2	1
Noop	2	1	3

Table 18 – Throughput Storage Overview Discussion - Grouped Results by Number of Tasks 64/Seq

	HDD	**HDD/SSD**	**SSD**
CFQ	3	1	2
Deadline	3	1	2
Noop	3	1	2

Table 19 – Throughput Storage Overview Discussion - Grouped Results by Number of Tasks 64/Ran

Tables 18 and 19 present values considering number of tasks equal to 64. Table 18 relates to a sequential access pattern and indicates that the schedulers also presented good throughput values using the hybrid storage approach. We also noticed that the HDD-only approach presented the worst performance for almost all schedulers. The usage of the SSD-only approach presented mixed values having the deadline presenting the highest throughput while noop presented the worst throughput. Table 19 relates to random access pattern. We noticed interestingly that the schedulers presented the same behavior from Table 18 according to the storage approach. The hybrid approach, presented for all schedulers, had the best throughput value. It was followed by the SSD-only approach presenting middle results. Finally, the HDD-only approach had the worst performance compared to the previous methods. We also noticed that when the number of tasks is 64, does not matter whether the access pattern is sequential or random, the HDD-only approach had the worst throughput in 5 from 6 cases. From those 12 rows presented in Tables 16, 17, 18 and 19, we verify that the HDD-only approach represents 75% of all worst results with 9 of 12.

6.5.2 Scheduler Overview

In this topic, we are interested in understanding the performance of each Linux scheduler when changing the approach we are storing data and metadata on the devices. Figure 61, Figure 62 and Figure 63 presents the average throughput for the same I/O requests using three different Linux schedulers.

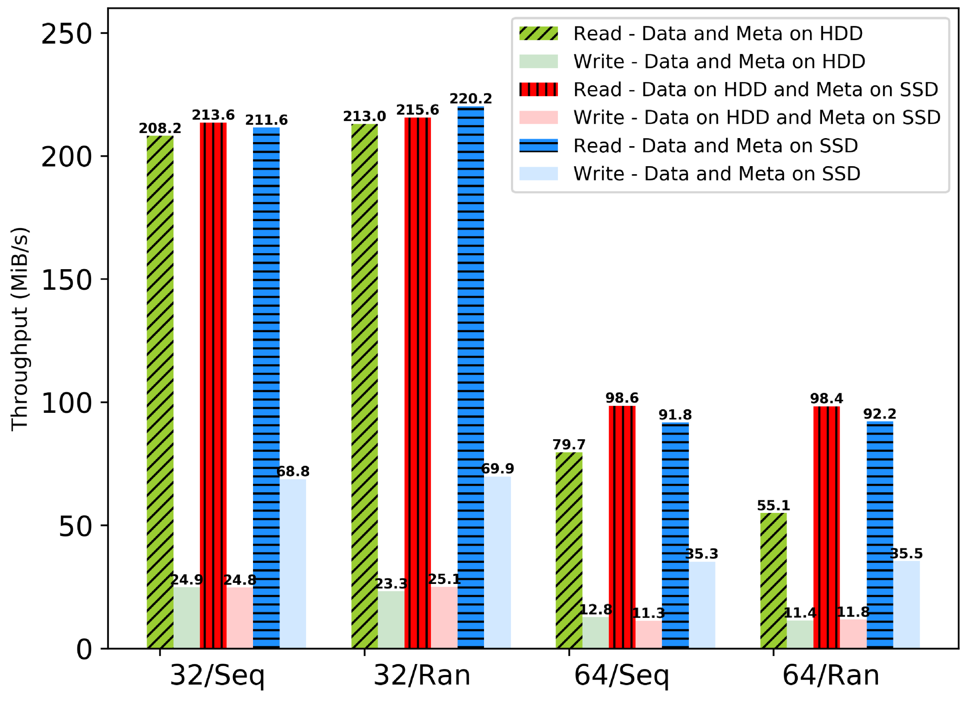


Figure 61 – Throughput Analysis Using CFQ Scheduler.

Observing Figure 61, we can verify that, the throughput seems to be regular for read operations. In general, when using the CFQ scheduler, the throughput seems to follow the same characteristics analyzed before according to the storage approaches. However, the dominant aspect is that in all configurations, when using the HDD-only approach, and for reading operation, the throughput was the lowest. We also verify that the usage of the homogeneous storage environment with flash device increased for a write operation in all cases making sense to the result presented by Figure 60.

In Table 20, verifying write values, the SSD-only approach also presented highest throughput. For reading, the hybrid approach presented the highest throughput. It is worth noticing that these values are not equal to the presented in Table 15. It happens because for calculating these arithmetic mean, we took different execution files into considerations once we executed experiments with 36 scenarios and selected the files that composed the same evaluation subset. For instance, to generate Figure 61 analysis, we used three final execution files changing the storage approach and considering only the CFQ scheduler. In Figure 58 the focus wasn't this presented here. There, we fixed HDD as the storage location for the data and metadata and switched to the same three Linux schedulers CFQ, Deadline, and Noop

Table 20 – Average in MiB/s of Throughput Presented in Figure 61 (Throughput Analysis Using CFQ Scheduler).

Storage approach	Read	Write
Data and Metadata on HDD	139	18,1
Data on HDD and Metadata on SSD	156,55	18,25
Data and Metadata on SSD	153,95	49,93

Analyzing Table 20 we verify that using the CFQ scheduler there is not only one best way to get the highest throughput. It happens because there are two best options for I/O operations which depend on the ways used to store data and metadata. Using the hybrid approach gives the best throughput for reading operation whereas using the SSD-only approach gives the highest throughput for the write operation.

Figure 62 presents throughput using Deadline scheduler. In configuration block 32/Seq, the SSD-only approach presented the smallest throughput for reading operation. For configuration 32/Ran, the throughput for the same operation performed steadily. The hybrid storage approach presented the highest throughput in configuration 32/Ran when the HDD-only approach presented the highest throughput in 32/Seq. For access pattern sequential and task number 64, the bigger throughput for both read and write operation was using only flash devices. We verify that configuration 64/Seq and 32/Seq presented asymmetrical results considering the proportions. The HDD-only had the highest value in the 32/Seq configuration whereas in 64/Seq it had the smallest one. For random configurations, in other words, 32/Ran and 64/Ran have the same rule for the storage approaches having then the hybrid with the highest throughput. Analyzing the write operation, we verify that the values were very closer for approaches that are composed of HDDs. However, the unique approach that does not use HDD presented the highest throughput for all configurations. Comparing Figure 61 with Figure 62, we notice that the deadline scheduler improved the throughput considerably for approaches that uses HDD considering 32/Seq configuration.

Comparing values from Table 20 and Table 21, the throughput were very closer for

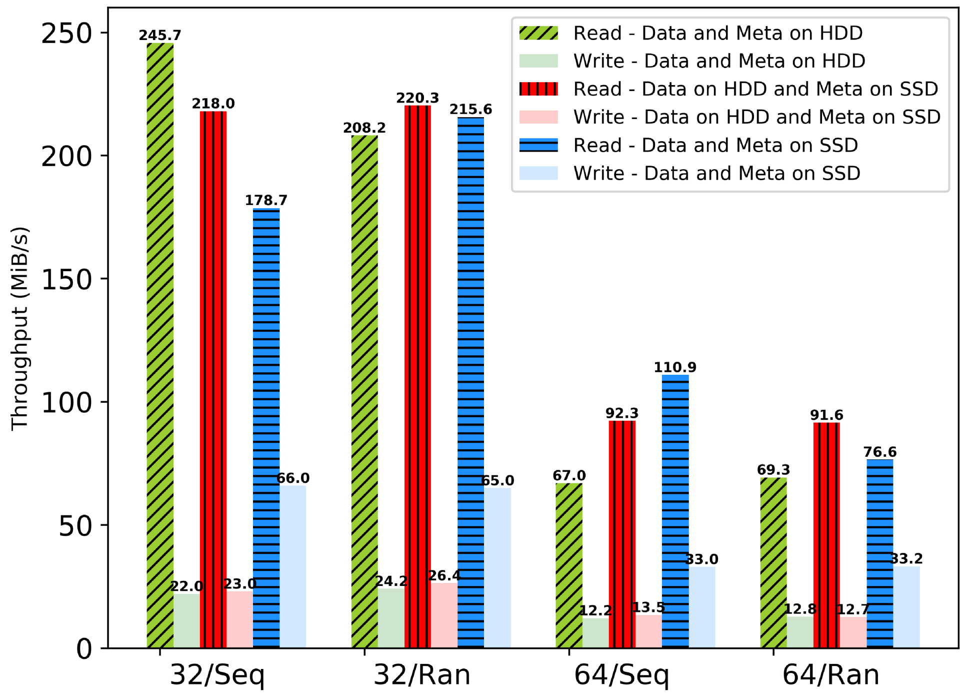


Figure 62 – Throughput Analysis Using Deadline Scheduler.

hybrid approach with a decrease of 1,0 MiB/s when using deadline scheduler. However, the HDD-only approach increase throughput by 8,55 MiB/s and the SSD-only approach decreased 8,4 MiB/s. We notice that in Table 20, the approach that presented the lowest throughput for the reading operation was the HDD-only approach. However, in Table 21, the approach that presented the lowest throughput was the SSD-only approach.

Table 21 – Average in MiB/s of Throughput presented in Figure 62 (Throughput Analysis Using Deadline Scheduler).

Storage approach	Read	Write
Data and Metadata on HDD	147,55	17,8
Data on HDD and Metadata on SSD	155,55	18,9
Data and Metadata on SSD	145,45	49,3

Figure 63 presents the throughput value for read and write operations using noop scheduler. Almost in all cases for the read operation, the hybrid approach has presented the highest throughput compared with CFQ and Deadline scheduler. Indeed, for 32/Seq and 32/Ran configurations, the hybrid storage approach presented the highest throughput. The hybrid storage approach for the noop scheduler seems to present a good throughput rate when compared to CFQ and deadline scheduler. Analyzing write operation, they were also quite similar for approaches that use HDDs in their composition. However, the SSD-only approach presented the highest throughput for almost all write operations wt the exception of the 64/Seq configuration.

Comparing Figure 63 with Figure 61 and Figure 62, we noticed that read

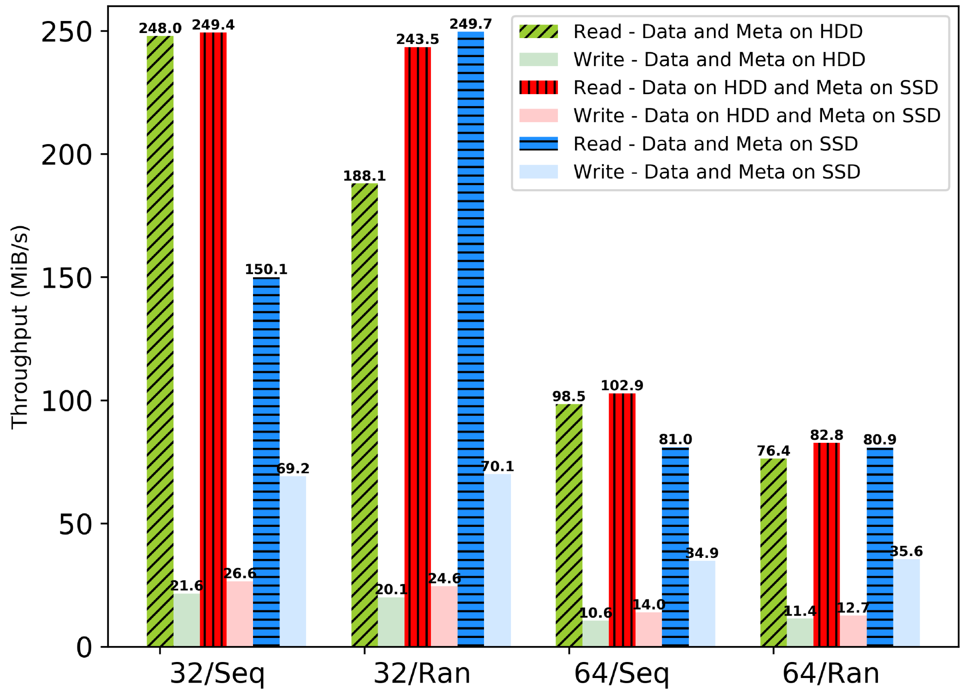


Figure 63 – Throughput Analysis Using Noop Scheduler.

operation for noop scheduler, almost in all cases, were bigger than the previous approaches for task number 32. From a total of six results shown, four scored better with noop scheduler. The throughput value presented for task number 64 was quite similar. for write operations, we noticed that in all figures they were higher than for reading operations. We believe that it happens because the write operation is more costly than the read and the factor which interferes with more weight in such a situation is the hardware used instead of the scheduler adopted.

Table 22 presents the same previous relations between the operations and the storage approaches. The values presented when using the noop scheduler seems to follow the same rule when using the deadline scheduler. It had the worst throughput rate for the SSD-only approach when reading and the highest throughput rate when writing. The Noop scheduler also presented for the HDD-only storage approach the worst throughput for writing.

Table 22 – Average in MiB/s of Throughput presented in Figure 63 (Throughput Analysis Using Noop Scheduler).

Storage approach	Read	Write
Data and Metadata on HDD	152,75	15,93
Data on HDD and Metadata on SSD	169,65	19,48
Data and Metadata on SSD	140,43	52,45

As it happened with the other two schedulers CFQ and deadline, with noop scheduler there is a big difference in throughput when performing read operations compared with write operations. One possible feature that could lead to this result could be justified because these two first approaches use a magnetic disk to perform the operations, and we have a known knowledge that magnetic disks impose some delay when trying to position the reading head of the mechanical arms inside the device. Spending time positioning the head in the disk probably decreases the throughput rate on the storage devices. Another fact that increase this idea and is presented by all schedulers analysis is that the difference between the read operation and the write operation when storing both data and metadata on SSD is smaller than the other two options. It could justify by the fact that SSD is composed of flash NAND instead of magnetic plates and the time to perform write operations in flash memory is higher than when you perform it in magnetic disks.

Scheduler Overview Discussion We also summarized Figure 61, Figure 62 and Figure 63 collectively, comparing how each storage approach behaved with each scheduler. As previously analyzed for schedulers, we also clustered in Tables 23, 24, 25 and 26 the expressive results according to the task number and access pattern, presenting how each storage approach behaved. We also concentrate on analyzing read operations in these tables. It allows us to verify how each scheduler behaved analyzing each one side by side according to the chosen storage method.

	CFQ	**Deadline**	**Noop**
HDD	3	2	1
HDD+SSD	3	2	1
SSD	1	2	3

Table 23 – Throughput Scheduler Overview Discussion - Grouped Results by Number of Tasks 32/Seq

	CFQ	**Deadline**	**Noop**
HDD	1	2	3
HDD+SSD	3	2	1
SSD	2	3	1

Table 24 – Throughput Scheduler Overview Discussion - Grouped Results by Number of Tasks 32/Ran

Tables 23 and 24 present values considering number of tasks equal to 32. Table 23 relates values for sequential access pattern. It shows that the noop scheduler presented good performance with the usage of HDD devices for both HDD-only and hybrid approach. On the other hand, the CFQ scheduler does not seem to present a good throughput value for these storage configurations, it presented the best storage configuration for the SSD-only storage approach though. Deadline scheduler placed between the two other schedulers independently of the storage approach for all cases. Table 24 presented values considering random access pattern. In this case, the noop scheduler also presented good throughput results, but this time for storage configurations that uses SSD devices. Different from Table 23 where CFQ presented the highest throughput when using SSD-only approach, in Table 24, it presented the highest throughput for the HDD-only storage approach. With this access pattern, the deadline also was placed in performance between the two other approaches. Summarizing, we verified that the noop scheduler seems to perform good values for both sequential and random access patterns.

	CFQ	**Deadline**	**Noop**
HDD	2	3	1
HDD+SSD	2	3	1
SSD	2	1	3

Table 25 – Throughput Scheduler Overview Discussion - Grouped Results by Number of Tasks 64/Seq

	CFQ	**Deadline**	**Noop**
HDD	3	2	1
HDD+SSD	1	2	3
SSD	1	3	2

Table 26 – Throughput Scheduler Overview Discussion - Grouped Results by Number of Tasks 64/Ran

Table 25 and 26 present values considering number of tasks equal to 64. Table 25 relates values for sequential access pattern. It shows that the noop scheduler also presented good performance with the usage of HDD devices for both HDD-only and hybrid approach. Actually, the results for the noop scheduler for 64/Seq configuration were equal for 32/Seq configuration. Thus, we notice that noop performs well for sequential access patterns combined with HDD devices. On the other hand, the CFQ scheduler placed between the two other schedulers independently of the storage approach in all cases. Finally, the deadline seems to perform badly in approaches that have HDDs in their composition. Table 26 presented values considering random access pattern. In this case, differently from Table 25, noop does not present the highest throughput results when using SSD devices. However, when using HDDs, it continued presenting good throughput results. We also verify in Table 26 that CFQ presented two good results, both when using SSD devices. With this access pattern, the deadline also was placed in performance between the two other approaches. Summarizing, we verified that the CFQ scheduler seems to perform well for random access patterns.

6.6 LATENCY ANALYSIS

This subsection presents the latency results obtained through the experimentation process. In all of them, we are looking for understanding how latency behaves when the benchmark performs I/O read and write operations. We also analyzed here the values considering storage and a scheduler overview.

In the first three figures, all bars with hatches present the latency value for the read operation that is related to one specific scheduler, and all bars without hatches, located after the hatched bar, present the latency value of the write operation for the same scheduler. In the last three figures, all bars with hatches present the latency value for the reading operation that is related to the way the data were stored, and all bars without hatches, located after the hatched bar, present the latency value of the write operation. The latency value for the write operation always comes after the related latency for the read operation.

6.6.1 Scheduler Overview

In Figure 64, Figure 65 and Figure 66 we present the average latency time for read and write operations switching the three presented schedulers in section 6.2 (CFQ, deadline and noop).

Figure 64 presents the latency time for the read/write operations storing data and metadata in different ways using the CFQ scheduler. It's possible to verify that in all cases the latency value for the write operation is greater than the read operation no matter which storage approach we use. Indeed, it is known that the time to read, in a normal circumstance, is smaller than the time to write. It might be related to some facts. For instance, how the hardware is handled or the characteristics of a specific operating system, or even the way the file system operates could be cited. Normally, to read some files, the file system should find the file through the directory tree and read the respective file. To write a file, however, the same operation through the tree should be performed, but, after reading, differently from the read operation, the file system has more additional functions such as updating in someplace the metadata information related to the written file. This updating place could be any place such as a standard place, a new path through the tree directory, a new device, or even a geographically distant location.

Using CFQ scheduler, it is also possible to notice, in Figure 64, that when setting the number of tasks equal to 32, the approaches presented to storage the data were very similar when relating the latency time for the read operation. However, when setting the number of tasks equal to 64, the worst approach was storing both data and metadata on HDDs while the best was using the hybrid approach (HDD+SSD). Analyzing the write operation, the results presented a huge difference when storing data and metadata using an SSD in all cases. More than three times less to perform write operations was achieved when using only SSD if compared with the hybrid approach with the access pattern equal to sequential and number of tasks equal to 64.

Table 27 presents the average of latency value for all the scenarios presented in Figure 64 when using the deadline scheduler. It is possible to see that using the hybrid approach (HDD+SSD) was the configuration storage that gave us the lowest read latency time. For the write operation, the best configuration storage was storing both data on the SSD device with almost three times less than the lowest configuration which uses HDD.

Table 27 – Average of Latency Presented in Figure 64 (Latency Analysis Using CFQ Scheduler).

Storage approach	Read	Write
Data and Meta on HDD	0,3643	2,0151
Data on HDD and Meta on SSD	0,2560	2,0871
Data and Meta on SSD	0,2815	0,6827

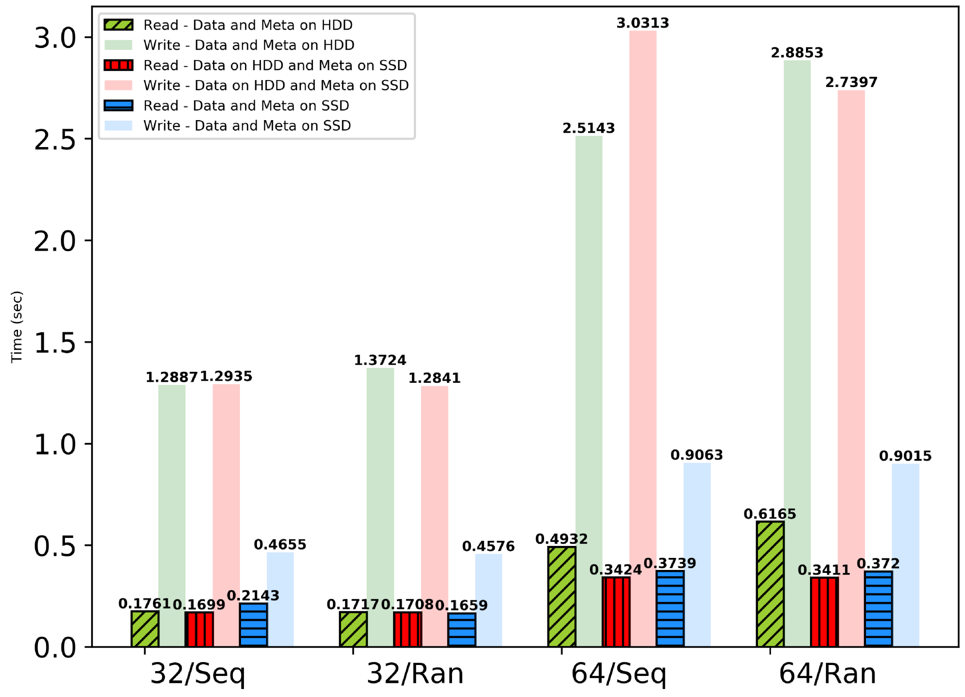


Figure 64 – Latency Analysis Using CFQ Scheduler.

Figure 65 presents the results using deadline Linux scheduler. The presented results were similar to the results presented by the CFQ scheduler. However, comparing Figure 64 with Figure 65 we can verify that when using the number of tasks equal to 64, the deadline scheduler has presented a smaller latency time to write operation when using the approach data on HDD and metadata on SSD. With the deadline scheduler, almost in all write results, storing data and metadata in an HDD was the worst approach providing the higher latency time.

Table 28 presents the average of latency value for all the scenarios presented in Figure 65 when using the deadline scheduler. As presented in Table 27, Table 28 also presented the hybrid approach (HDD + SSD) as the best results with a smaller latency time to read operation and the approach which stores both data and metadata on SSD as the best approach to decrease the latency time for the write operation.

Table 28 – Average of Latency Presented in Figure 65 9Latency Analysis Using Deadline Scheduler).

Storage approach	Read	Write
Data and Meta on HDD	0,3468	2,0183
Data on HDD and Meta on SSD	0,2708	1,9147
Data and Meta on SSD	0,2841	0,7274

Figure 66 presents the latency results when using Linux noop scheduler. It is possible to verify in this figure that the write operation latency when using HDD-only as a device to store both data and metadata was the worst storage approach. An important

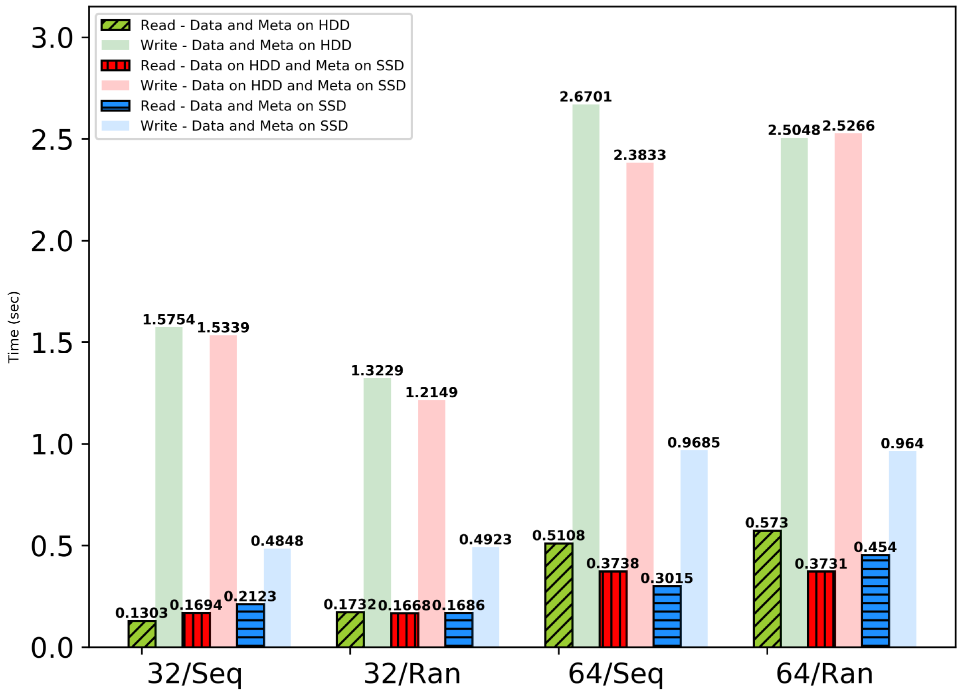


Figure 65 – Latency Analysis Using Deadline Scheduler.

fact that we noticed is that Figure 66 follows the same latency pattern presented by Figure 64 for the read operation in all cases. For instance, in Figure 64 when the storage configuration is access pattern equal to "sequential" and the number of tasks is 32, the storage approach that presented the worst latency for reading operation was when storing both data and metadata on SSD.

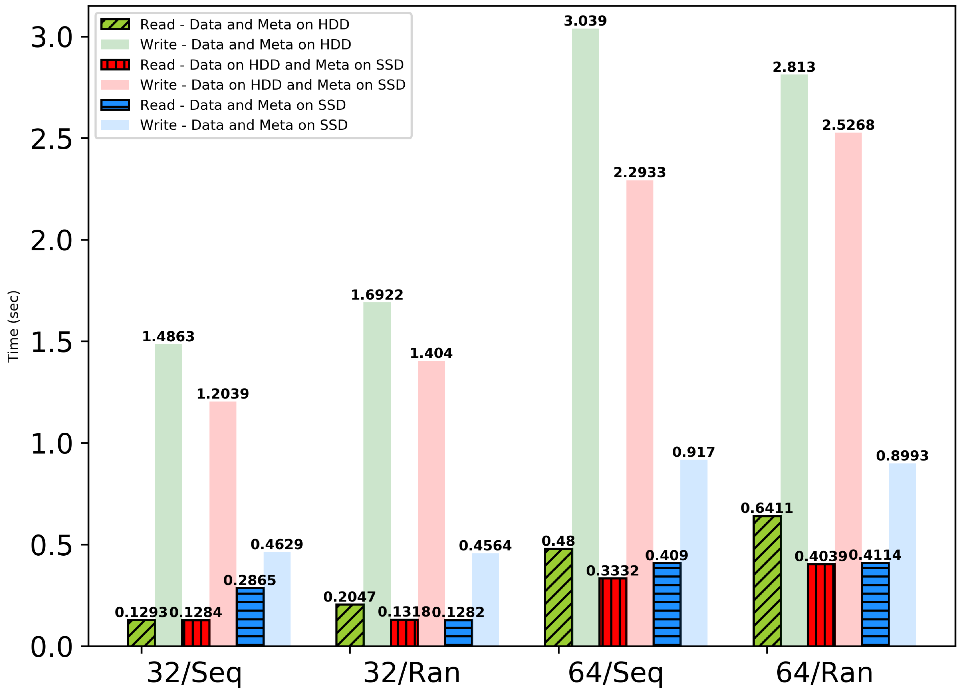


Figure 66 – Latency Analysis Using Noop Scheduler.

Table 29 presents the average of latency value for all the scenarios presented in Figure 66 when using the noop scheduler. As presented in Table 27 and Table 28, Table 29 also presented the hybrid approach (HDD + SSD) as the best results with a smaller latency time to read operation and the approach which stores both data and metadata on SSD as the best approach to decrease the latency time for write operation.

Table 29 – Average of latency presented in Figure 66 (Latency Analysis Using Noop Scheduler)

Storage approach	Read	Write
Data and Meta on HDD	0,3638	2,2576
Data on HDD and Meta on SSD	0,2493	1,8570
Data and Meta on SSD	0,3088	0,6839

It seems that regardless of the scheduler we are using, the way data storage is done greatly influences how latency will behave. We observed that in all the cases presented above when using the number of tasks equal to 64, the approach that stores data and metadata in a mechanical device such as HDD, presented the worst latency result. In general, the deadline scheduler seems to perform steadily with a low difference between the latency of each set of parameters if compared with the other two schedulers.

Scheduler Overview Discussion We also summarized Figure 64, Figure 65 and Figure 66 collectively, comparing how each storage approach behaved with each scheduler. We clustered in Tables 30, 31, 32 and 33 the expressive results according to the task number and access pattern, presenting how each storage approach behaved. Different from the throughput analysis, here we concentrate on analyzing write operation. It allows us to verify how each scheduler behaved analyzing each one side by side according to the chosen storage method.

	CFQ	**Deadline**	**Noop**
HDD	1	3	2
HDD+SSD	2	3	1
SSD	2	3	1

Table 30 – Latency Scheduler Overview Discussion - Grouped Results by Number of Tasks 32/Seq

	CFQ	**Deadline**	**Noop**
HDD	2	1	3
HDD+SSD	2	1	3
SSD	2	3	1

Table 31 – Latency Scheduler Overview Discussion - Grouped Results by Number of Tasks 32/Ran

Tables 30 and 31 present values considering number of tasks equal to 32. Table 30 relates values for sequential access pattern. It shows that noop scheduler presented good performance when SSDs devices were introduced on the storage layer. On the other hand, the deadline scheduler does not seem to present good latency values. CFQ scheduler presented an intermediate position between the other two schedulers presenting good results for HDD-only storage approach.

Table 31 presented values considering random access pattern. In this case, the deadline scheduler presented good latency results. Differently from Table 31 where deadline presented the worst latency values for all storage scheme, in Table 31, it presented the highest latency for the storage approaches that uses HDD in their storage layer. With this access pattern, CFQ also presented an intermediate position between the other two schedulers. Noop seems to perform poorly when these parameters are confronted with HDD devices.

	CFQ	**Deadline**	**Noop**
HDD	1	2	3
HDD+SSD	3	2	1
SSD	1	3	2

Table 32 – Latency Scheduler Overview Discussion - Grouped Results by Number of Tasks 64/Seq

	CFQ	**Deadline**	**Noop**
HDD	3	1	2
HDD+SSD	3	1	2
SSD	2	3	1

Table 33 – Latency Scheduler Overview Discussion - Grouped Results by Number of Tasks 64/Ran

Tables 32 and 33 present values considering the number of tasks equal to 64. Table 32 relates values for sequential access pattern. It shows mixed results. Noop scheduler presented all the different performances according to the storage approach. Its better result was using the hybrid approach and its worst was using HDD-only approach. The deadline scheduler presented the worst result for latency. Finally, CFQ presented good latency when using homogeneous storage devices in its storage system layer.

Table 33 presented values considering random access pattern. In this case, differently from Table 32, deadline presented two of the three possible good latency performance. Both these results were when using HDD devices. We also verify in Table 33 that CFQ scheduler does not present good results. It presented two of the three worst results. Noop seems to present intermediate performance when using HDD devices in its storage layer. On the other hand, it was the scheduler that presented the best latency performance when employing SSD-only storage scheme.

The results presented in both tables seem to follow a pattern. In the tables that employ a sequential access pattern, the deadline scheduler did not present good latency performance. On the other hand, when the access pattern was random, the deadline presented good results when using HDD devices in its scheme. This fact happened regardless of the size of the task.

6.6.2 Storage Overview

In the next Figure 67, Figure 68 and Figure 69 we present the average latency time for read and write operations switching the three storage approaches presented in section 6.1 (*HDD-only*, *HDD+SSD* and *SSD-only*)

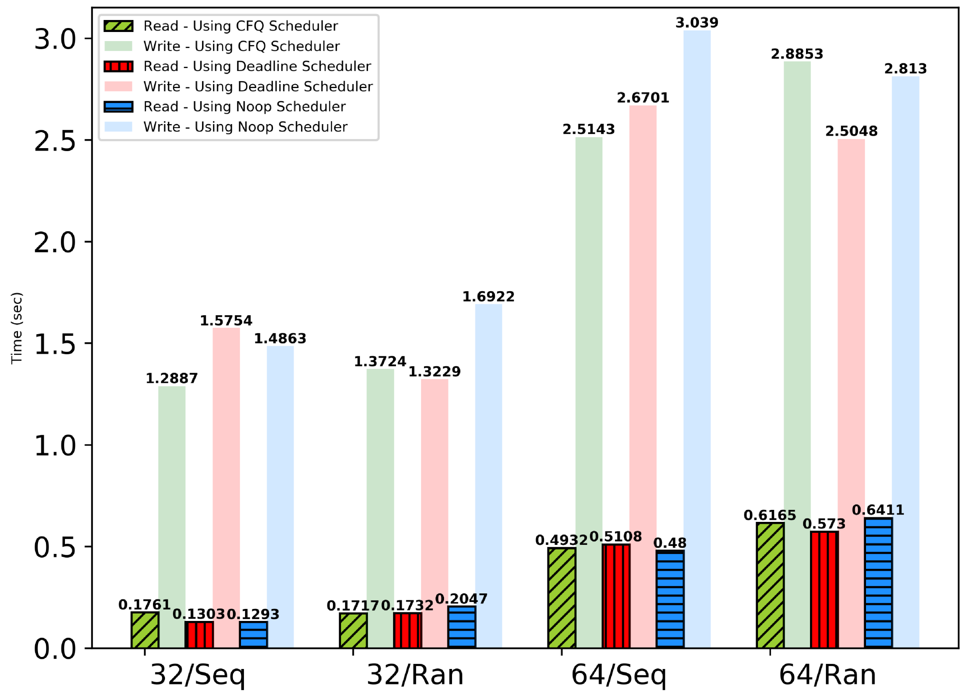


Figure 67 – Latency Analysis Storing Data And Metadata on HDD.

In Figure 67 we analyze the latency time for the operations when storing both data and metadata on the HDD device switching then the I/O schedulers. Analyzing it we can verify that when the parameter number of tasks is equal to 64, the read latency increases significantly regardless of the chosen access pattern for the three schedulers. Each storage configuration presented has a scheduler that performs better latency time according to the parameters selected. For instance, if the number of tasks is equal to 32 and the access pattern is equal to sequential, the scheduler which presented the lowest latency for the reading operation was the noop scheduler and for the write, the operation was the CFQ scheduler. Analyzing the same number of tasks equal to 32, but changing the access pattern to random, the schedulers which presented the lowest latency time for reading operation was the CFQ scheduler and for the write, the operation was the deadline scheduler.

Table 34 we present the average latency time for all the scenarios presented in Figure 67. We present this average because we believe that DISC and HPC applications can treat and use heterogeneous kinds of data with different access patterns and the number of tasks on the same application. Considering this, it's possible to see in Table 34 that when we are storing both data and metadata on the HDD device the scheduler that presents the highest average latency time to perform read operation is the *CFQ* and to perform the write operation is the *Noop* scheduler.

Figure 68 we analyze the latency time for the operations when using the hybrid approach (HDD+SSD) switching then the I/O schedulers. Just as it happened in Figure 67,

Table 34 – Average of Latency Presented in Figure 67 (Latency Analysis Storing Data And Metadata on HDD).

Scheduler	Read	Write
CFQ	0.3643	2.0151
Deadline	0.3468	2.0183
Noop	0.3637	2.2576

in Figure 68 all latency values to perform the write operation are greater than the value to perform the read operation. However, we can verify that the latency rate when storing by this way was lower in almost all read operation cases if compared to the values presented in Figure 67. The unique configuration where the read latency was higher than the presented in Figure 67 was for the deadline scheduler with the number of tasks equal to 32 and access pattern equal to random. In general, the latency when performing write operation has decreased if compared with the first storage approach. The cases where it hasn't happened were the twice for the CFQ scheduler with access pattern equal to sequential and the deadline scheduler with the number of tasks equal to 64 and access pattern equal to random. All the other write values have the latency time decreased when changing the way to store data and metadata.

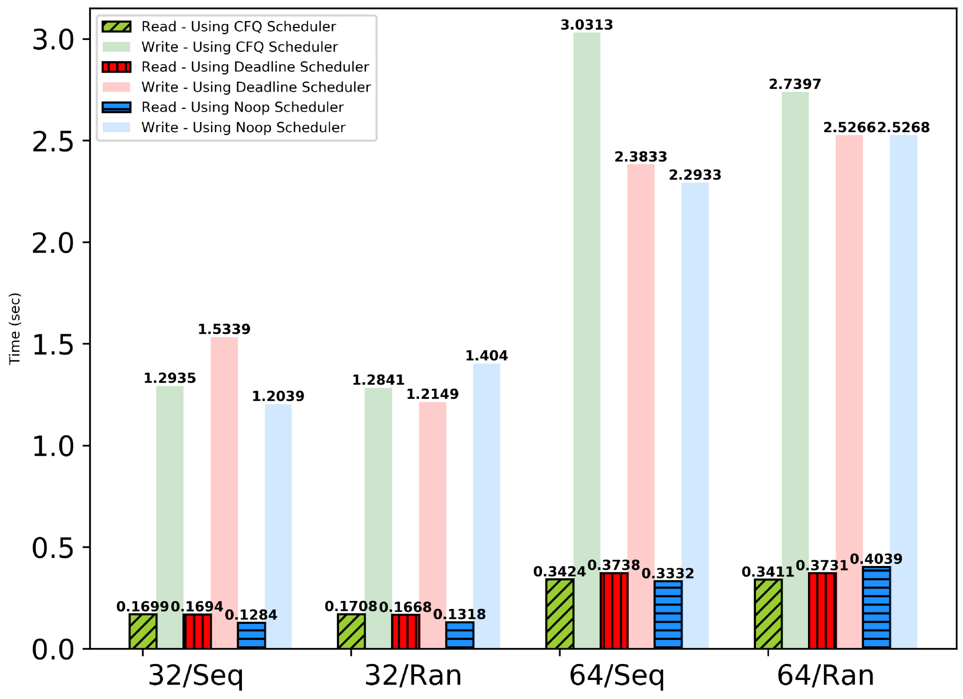


Figure 68 – Latency Analysis Storing Data on HDD and Metadata on SSD.

It's possible to see in Table 35 that the latency average to execute the read operation for all schedulers is smaller than when storing data on HDD presented in Table 34. It means that for all schedulers, all corresponding latency time for the read operation was decreased when the use of an SSD was applied. However, for the write operation, unlike the values presented by Table 34, Table 35 shows that when using an SSD device the CFQ does not perform with the same lowest latency as before. In fact, it was the unique value that had the worst latency value if compared with the previous storage approach. It is also possible to notice that when we are using the hybrid approach (HDD + SSD) the scheduler that presents the lowest average latency time to perform read and write operation is the *Noop* scheduler.

Table 35 – Average of Latency Presented in Figure 68 (Latency Analysis Storing Data on HDD and Metadata on SSD).

Scheduler	Read	Write
CFQ	0.2560	2.0871
Deadline	0.2707	1.9146
Noop	0.2493	1.8570

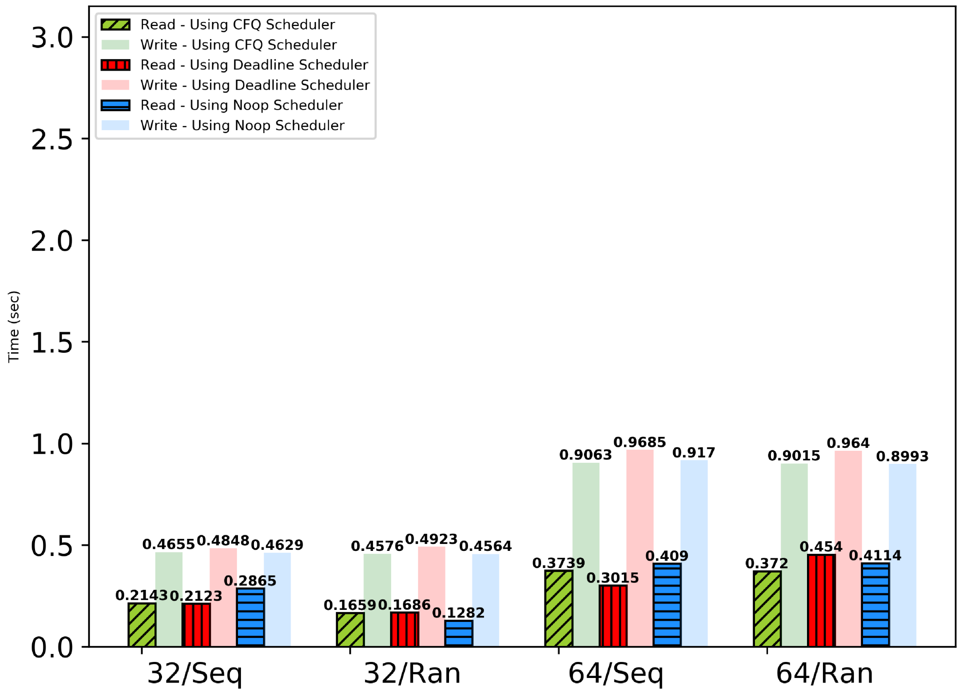


Figure 69 – Latency Analysis Storing Data and Metadata on SSD.

In Figure 69 we store even the data and metadata on SSD which is a device that does not have mechanical components switching then the same I/O schedulers. Comparing the latency time for the read operation with the values presented in Figure 68, we can notice that almost no significant improvement has been made, only three read latency values were improved using this storage approach. A curious result happened when the number of tasks is equal to 32 and the access pattern is sequential. Comparing the read latency time with the values presented by the Figure 67 we verify that when using the SSD-only approach, all schedulers presented a higher latency time. These same pattern results we can verify if compared to the Figure 68. By these interesting results, we can argue that the use of an SSD device to store both data and metadata has increased the latency time to read operations when the number of tasks is 32 and the access pattern is sequential. However, when the number of tasks is equal to 32, the use of an SSD to store only metadata, presented in Figure 68, presented five better results from six if compared to the Figure 67.

Despite no significant improvement in latency when performing a read operation, perhaps the most expressive results are related to the write operation. All values were less than one in all scenarios. It is possible to verify that the latency for the write operation in this storage configuration was significantly decreased if compared to the two previous storage approaches. Although the storage solution provided an important performance improvement when using the number of tasks equal to 32, perhaps the improvement presented for the number of tasks equal to 64 were more expressive compared to the previous results.

It is possible to notice in Table 36, that when performing the read operation, the latency ratio was higher for all schedulers if compared to Table 35. It enforces our analysis that this storage approach didn't present significant improvement as the previous results when performing read operations. However, if compared to Table 34, it presented a smaller and better overall latency time. Analyzing the write value presented in Table 36 we verify that the results for all schedulers were better than for the previous storage approaches. In this case, the scheduler which presented the better read and writes latency time was the CFQ scheduler.

Table 36 – Average of Latency Presented in Figure 69 (Latency Analysis Storing Data and Metadata on SSD).

Scheduler	Read	Write
CFQ	0.2815	0.6827
Deadline	0.2841	0.7274
Noop	0.3087	0.6839

These results could lead us to think that if the device that you are storing the data is an SSD device, it's very likely that the latency time will be decreased and thereby improve the performance of write operations.

Storage Overview Discussion

We also summarized Figure 67, Figure 68 and Figure 69 conjointly, comparing how each Linux scheduler behaved when changing the storage approach. Tables 37, 38, 40 and 40 present these results considering each scenario.

Before starting, we can see that the writing latency was very small for all environments that consider SSDs in the storage layer. In all these cases, the latency was the smallest of all when using non-mechanical storage devices to support writing. These

	HDD	HDD+SSD	SSD
CFQ	2	3	1
Deadline	3	2	1
Noop	3	2	1

Table 37 – Latency Storage Overview Discussion - Grouped Results by Number of Tasks 32/Seq

	HDD	HDD+SSD	SSD
CFQ	3	2	1
Deadline	3	2	1
Noop	3	2	1

Table 38 – Latency Storage Overview Discussion - Grouped Results by Number of Tasks 32/Ran

results were already expected and justify the big usability and quickly growth of the media device in HPE. As presented in Table. 36, the write latency when employing SSDs in the storage layer were at least three times lower on average than the other approaches. Although the values here presented are quite similar, we noticed that the latency behaved the same for both 32 and 64 task numbers in sequential access patterns.

	HDD	HDD/SSD	SSD
CFQ	2	3	1
Deadline	3	2	1
Noop	3	2	1

Table 39 – Latency Storage Overview Discussion - Grouped Results by Number of Tasks 64/Seq

	HDD	HDD/SSD	SSD
CFQ	3	2	1
Deadline	2	3	1
Noop	3	2	1

Table 40 – Latency Storage Overview Discussion - Grouped Results by Number of Tasks 64/Ran

6.7 FINAL CHAPTER CONSIDERATIONS

In this chapter, we presented an experimentation analysis in with we investigated how the throughput and latency behaved when performing I/O operations (e.g. read and write) within a high-performance experimental environment. The analysis considered 10 different parameters and produced results over 36 different scenarios. In this empirical process, we generated results that symbolized research works that targeted improvements in the I/O performance in the storage layer. We also presented information about the user environment and presented the hardware we used to perform our evaluation Moreover, we presented and explained the factors we used in the experimentation process and related these factors to the elements presented in our characterization model. Furthermore, we presented the throughput and latency results and also presented a discussion section highlighting the most expressive results we obtained.

7 CONCLUSIONS AND FUTURE WORK

Low I/O performance continues to affect high-performance environments. In the last years, many researchers have been proposed solutions to improve the I/O architecture considering different approaches. Some of them take advantage of hardware devices while others focus on a sophisticated software approach. Classifying these improvements in different dimensions allows researchers to understand how these improvements have been built over the years and how it progresses.

In addition, it also allows future efforts to be directed to research topics that have developed at a lower rate, promoting the general development process. This research presented a three-dimension characterization model for classifying research works on I/O performance improvements for large scale storage computing facilities.

We used the proposed model to perform a secondary study that covered ten years of research on I/O performance improvements. This model can also be used as a guideline framework to summarize researches providing an overview of the actual scenario. It classified hundreds of distinct researches identifying which were the devices, software, and storage systems that received more attention over the years.

We also evaluated a subset of researchers' I/O improvements using a real and complete experimentation environment, the Grid5000 to justify the importance of this model. Analysis of different storage and schedulers perspectives demonstrates how the throughput and latency parameters behaved when performing different I/O operations.

7.1 FUTURE WORK

For future work, we intend to explore this model and consider its utilization to classify not only researches related to the storage environments but also improvements that are related to other research topics such as processing and network. We are also planning to perform experiments in a large cloud environment considering different storage technologies and software approaches to also validate it using cloud concepts. Finally, we also aim to present guidelines to use this model in the software engineering field. This plan aims to adapt and translate the information presented here within the software engineering field.

REFERENCES

AGOSTINHO, Bruno Machado; ROTTA, Giovanni; PLENTZ, Patricia Della Mea; DANTAS, Mario A. R. Smart comm: A smart home middleware supporting information exchange. **IECON 2018 - 44th Annual Conf. of the IEEE Industrial Electronics Society, Washington, DC, USA, October 21-23, 2018**, p. 4678–4684, 2018.

AHMADIAN, Saba; SALKHORDEH, Reza; ASADI, Hossein. Lbica: A load balancer for i/o cache architectures. In: IEEE. **2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)**. 2019. p. 1196–1201.

AL-WESABI, Ola A; ABDULLAH, Nibras; SUMARI, Putra. On the design of video on demand server-based hybrid storage system. In: SPRINGER. **International Conference of Reliable Information and Communication Technology**. 2017. p. 306–315.

AL-WESABI, Ola Ahmed; SUMARI, Putra; ABDULLAH, Nibras. Data stream management system for video on demand hybrid storage server. **International Journal of Intelligent Systems Technologies and Applications**, Inderscience Publishers (IEL), v. 18, n. 5, p. 470–493, 2019.

ALI, Nawab; CARNS, Philip; ISKRA, Kamil; KIMPE, Dries; LANG, Samuel; LATHAM, Robert; ROSS, Robert; WARD, Lee; SADAYAPPAN, Ponnuswamy. Scalable i/o forwarding framework for high-performance computing systems. In: IEEE. **2009 IEEE Int. Conf. on Cluster Computing and Workshops**. 2009. p. 1–10.

ALMASI, Gheorghe; ASAAD, Sameh; BELLOFATTO, Ralph E; BICKFORD, H Randall; BLUMRICH, Matthias A; BREZZO, Bernard; BRIGHT, Arthur A; BRUNHEROTO, Jose R; CASTANOS, Jose G; CHEN, Dong et al. Overview of the ibm blue gene/p project. **IBM Journal of Research and Development**, IBM CORP 1 NEW ORCHARD ROAD, ARMONK, NY 10504 USA, v. 52, n. 1-2, p. 199–220, 2008.

AMBAREESH, S; FATHIMA, Jesna. A cached middleware to improve i/o performance using posix interface. **Procedia Computer Science**, Elsevier, v. 85, p. 125–132, 2016.

ASSOCIATION, Storage Networking Industry et al. Msr cambridge traces. **http://iotta. snia. org/traces/388**, 2010.

ASTC Technology Roadmap, Accessed on May 06, 2020. 2016. Available from Internet: <<"http://idema.org/?page_id=5868">>.

AXBOE, J. **Noop scheduler, 2010**. [s.d.].

AXBOE, Jens. Linux block io—present and future. In: **Ottawa Linux Symp**. 2004. p. 51–61.

AXBOE, Jens. **Completely Fair Queueing (CFQ) Scheduler**. 2010.

AXBOE, Jens et al. Flexible i/o tester. **Online] https://github. com/axboe/fio**, 2016.

BARHAM, Paul; DRAGOVIC, Boris; FRASER, Keir; HAND, Steven; HARRIS, Tim; HO, Alex; NEUGEBAUER, Rolf; PRATT, Ian; WARFIELD, Andrew. Xen and the art of virtualization. **ACM SIGOPS operating systems review**, ACM New York, NY, USA, v. 37, n. 5, p. 164–177, 2003.

BARROSO, Luiz André; CLIDARAS, Jimmy; HÖLZLE, Urs. The datacenter as a computer: An introduction to the design of warehouse-scale machines. **Synthesis lectures on computer architecture**, Morgan & Claypool Publishers, v. 8, n. 3, p. 1–154, 2013.

BHATTACHARJEE, Bishwaranjan; ROSS, Kenneth A; LANG, Christian; MIHAILA, George A; BANIKAZEMI, Mohammad. Enhancing recovery using an ssd buffer pool extension. In: ACM. **Proceedings of the Seventh Int. Workshop on Data Management on New Hardware**. 2011. p. 10–16.

BOITO, Francieli Zanon; INACIO, Eduardo C; BEZ, Jean Luca; NAVAUX, Philippe OA; DANTAS, Mario AR; DENNEULIN, Yves. A checkpoint of research on parallel i/o for high-performance computing. **ACM Computing Surveys**, ACM New York, NY, USA, v. 51, n. 2, p. 1–35, 2018.

BOUKHOBZA, Jalil; OLIVIER, Pierre; RUBINI, Stéphane; LEMARCHAND, Laurent; HADJADJ-AOUL, Yassine; LAGA, Arezki. Macach: An adaptive cache-aware hybrid ftl mapping scheme using feedback control for efficient page-mapped space management. **Journal of Systems Architecture**, Elsevier, v. 61, n. 3-4, p. 157–171, 2015.

BOUKHOBZA, Jalil; RUBINI, Stéphane; CHEN, Renhai; SHAO, Zili. Emerging nvm: A survey on architectural integration and research challenges. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, ACM New York, NY, USA, v. 23, n. 2, p. 1–32, 2017.

BRAAM, Peter J; SCHWAN, Philip. Lustre: The intergalactic file system. In: **Ottawa Linux Symp.** 2002. p. 50.

BRERETON, OP; KITCHENHAM, BA. The scope of epic case studies. **EPIC technical Report**, 2007.

BREWER, Eric; YING, Lawrence; GREENFIELD, Lawrence; CYPHER, Robert; T'SO, Theodore. Disks for data centers. 2016.

BRYANT, Randal E. Data-intensive scalable computing for scientific applications. **Computing in Science & Engineering**, IEEE Computer Society, v. 13, n. 6, p. 25–33, 2011.

BU, Kai; WANG, Meng; NIE, Hongshan; HUANG, Wei; LI, Bo. The optimization of the hierarchical storage system based on the hybrid ssd technology. In: IEEE. **Int. Conf. on Intel. Syst. Design and Eng. App.** 2012. p. 1323–1326.

BUCY, John S; GANGER, Gregory R et al. **The DiskSim simulation environment version 3.0 reference manual**. : School of Computer Science, Carnegie Mellon University, 2003.

BUDGEN, David; BRERETON, Pearl. Performing systematic literature reviews in software engineering. In: ACM. **Proceedings of the 28th international conference on Software engineering**. 2006. p. 1051–1052.

CALZAROSSA, Maria Carla; MASSARI, Luisa; TESSERA, Daniele. Workload characterization: A survey revisited. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 48, n. 3, p. 1–43, 2016.

CAMPBELL, Donald T; STANLEY, Julian C. Experimental and quasi-experimental designs for research. **Handbook of research on teaching**, Rand McNally Chicago, IL, p. 171–246, 1963.

CAULFIELD, Adrian M; MOLLOV, Todor I; EISNER, Louis Alex; DE, Arup; COBURN, Joel; SWANSON, Steven. Providing safe, user space access to fast, solid state disks. **ACM SIGPLAN Notices**, ACM New York, NY, USA, v. 47, n. 4, p. 387–400, 2012.

CHANDRA, Rohit; DAGUM, Leo; KOHR, David; MENON, Ramesh; MAYDAN, Dror; MCDONALD, Jeff. **Parallel programming in OpenMP**. : Morgan kaufmann, 2001.

CHANG, Hsung-Pin; YU, Yu-Cheng; CHUNG, Pei-Yao. Design and implementation of a shared multi-tiered storage system. In: IEEE. **2018 3rd International Conference on Computer and Communication Systems (ICCCS)**. 2018. p. 94–98.

CHEN, Hsin-Ya; LEE, Pei-Yu; CHANG, Hsung-Pin. A multi-tiered storage structure for cloud computing. In: IEEE. **2016 International Computer Symposium (ICS)**. 2016. p. 636–639.

CHEN, Xian; CHEN, Wenzhi; LU, Zhongyong. Fusion-cache: A refactored content-aware host-side ssd cache. In: SPRINGER. **International Conference on Algorithms and Architectures for Parallel Processing**. 2015. p. 297–314.

CHIANG, Ron C; UPPAL, Ahsen J; HUANG, H Howie. An adaptive io prefetching approach for virtualized data centers. **IEEE Transactions on Services Computing**, IEEE, v. 10, n. 3, p. 328–340, 2015.

CHIU, George. The ibm blue gene project. **IBM Journal of Research and Development**, IBM Corp. Riverton, NJ, USA, v. 57, n. 1, p. 1–6, 2013.

CHOI, Hyunkyoung; BAHN, Hyokyung. Accelerating storage system performances with nvram cache by considering storage access characteristics. In: IEEE. **2018 5th International Conference on Information Science and Control Engineering (ICISCE)**. 2018. p. 107–111.

CHUNG, Tae-Sun; PARK, Dong-Joo; PARK, Sangwon; LEE, Dong-Ho; LEE, Sang-Won; SONG, Ha-Joo. A survey of flash translation layer. **Journal of Systems Architecture**, Elsevier, v. 55, n. 5-6, p. 332–343, 2009.

COOK, Thomas D; CAMPBELL, Donald Thomas. **Quasi-experimentation: Design and analysis for field settings**. : Rand McNally Chicago, 1979.

CORI, accessed Jun-29-2020. 2020. Available from Internet: <<"https://www.nersc.gov/systems/cori/">>.

COSTA, Ivan Ferreira da. Pense e responda! qual o comprimento e a profundidade de bits em cd, dvd e bd? **Caderno Brasileiro de Ensino de Física**, v. 24, n. 3, p. 333–337, 2007.

COUNCIL, Storage Performance. **Storage Performance Council**. 2007.

COUNCIL, Storage Performance. **OLTP Application I/O and Search Engine I/O, UMass Trace Repository**. 2020. Available from Internet: <<http://traces.cs.umass.edu/index.php/Storage/Storage>>.

COUNCIL, Transaction Processing Performance. **TPC benchmark B standard specification**. 1990. Available from Internet: <<http://www.tpc.org/tpcb/default.asp>>.

COUNCIL, Transaction Processing Performance. **TPC benchmark C standard specification**. 1990. Available from Internet: <<http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf>>.

COUNCIL, Transaction Processing Performance. **TPC benchmark E standard specification**. 1990. Available from Internet: <<http://www.tpc.org/tpce/default.asp>>.

COUNCIL, Transaction Processing Performance. **TPC benchmark H standard specification**. 1990. Available from Internet: <<http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.18.0.pdf>>.

CROCKETT, Louise H; ELLIOT, Ross A; ENDERWITZ, Martin A; STEWART, Robert W. **The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc**. : Strathclyde Academic Media, 2014.

CUI, Jinhua; WU, Weiguo; ZHANG, Xingjun; HUANG, Jianhang; WANG, Yinfeng. Exploiting latency variation for access conflict reduction of nand flash memory. In: IEEE. **2016 32nd Symposium on Mass Storage Systems and Technologies (MSST)**. 2016. p. 1–7.

DAE-SIK, Ko; SEUNG-KOOK, Chung. A design of ddr-1 solid state drive using pci-e interface. In: IEEE. **2009 15th Asia-Pacific Conf. on Communications**. 2009. p. 889–891.

DALEY, Christopher S; GHOSHAL, Devarshi; LOCKWOOD, Glenn K; DOSANJH, Sudip; RAMAKRISHNAN, Lavanya; WRIGHT, Nicholas J. Performance characterization of scientific workflows for the optimal use of burst buffers. **Future Generation Computer Systems**, Elsevier, 2017.

DOH, In Hwan; LEE, Hyo J; MOON, Young Je; KIM, Eunsam; CHOI, Jongmoo; LEE, Donghee; NOH, Sam H. Impact of nvram write cache for file system metadata on i/o performance in embedded systems. In: **Proceedings of the 2009 ACM symposium on Applied Computing**. 2009. p. 1658–1663.

DORIER, Matthieu; ANTONIU, Gabriel; CAPPELLO, Franck; SNIR, Marc; ORF, Leigh. Damaris: How to efficiently leverage multicore parallelism to achieve scalable, jitter-free i/o. In: IEEE. **2012 IEEE International Conference on Cluster Computing**. 2012. p. 155–163.

DROWNING in data. **IEEE Spectrum magazine**, IEEE, p. 32–37, September 2018.

DU, Congjin; WU, Chentao; LI, Jie; GUO, Minyi; HE, Xubin. Bps: A balanced partial stripe write scheme to improve the write performance of raid-6. In: IEEE. **IEEE Int. Conf. on Cluster Computing**. 2015. p. 204–213.

EILEMANN, Stefan; DELALONDRE, Fabien; BERNARD, Jon; PLANAS, Judit; SCHUERMANN, Felix; BIDDISCOMBE, John; BEKAS, Costas; CURIONI, Alessandro; METZLER, Bernard; KALTSTEIN, Peter et al. Key/value-enabled flash memory for complex scientific workflows with on-line analysis and visualization. In: IEEE. **2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)**. 2016. p. 608–617.

ENES, Jonatan; CACHEIRO, Javier López; EXPÓSITO, Roberto R; TOURINO, Juan. Big data-oriented paas architecture with disk-as-a-resource capability and container-based virtualization. **Journal of Grid Computing**, Springer, v. 16, n. 4, p. 587–605, 2018.

FAN, Ya; WANG, Yong; YE, Miao. An improved small file storage strategy in ceph file system. In: IEEE. **2018 14th International Conference on Computational Intelligence and Security (CIS)**. 2018. p. 488–491.

FENG, Zhijie; FENG, Zhiyong; WANG, Xin; RAO, Guozheng; WEI, Yazhou; LI, Zhiyuan. Hdstore: An ssd/hdd hybrid distributed storage scheme for large-scale data. In: SPRINGER. **International Conference on Web-Age Information Management**. 2014. p. 209–220.

FOLK, Mike; HEBER, Gerd; KOZIOL, Quincey; POURMAL, Elena; ROBINSON, Dana. An overview of the hdf5 technology suite and its applications. In: **Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases**. 2011. p. 36–47.

GANGER, Gregory R; WORTHINGTON, Bruce L; PATT, Yale N; BUCY, J. The disksim simulation environment. **University of Michigan, EECS, Technical Report CSE-TR-358-98**, 1998.

GARA, Alan; BLUMRICH, Matthias A; CHEN, Dong; CHIU, GL-T; COTEUS, Paul; GIAMPAPA, Mark E; HARING, Ruud A; HEIDELBERGER, Philip; HOENICKE, Dirk; KOPCSAY, Gerard V et al. Overview of the blue gene/l system architecture. **IBM Journal of research and development**, IBM, v. 49, n. 2.3, p. 195–212, 2005.

GOMES, Eliza; UMILIO, Franco; DANTAS, Mario A. R.; PLENTZ, Patricia Della Mea. An ambient assisted living research approach targeting real-time challenges. **IECON 2018 - 44th Annual Conf. of the IEEE Industrial Electronics Society, Washington, DC, USA, October 21-23, 2018**, p. 3079–3083, 2018.

GRANZ, Steven; CONOVER, Michael; GUZMAN, Javier; CROSS, William; HARLLEE, Pete; RAUSCH, Tim. Perpendicular interlaced magnetic recording. **IEEE Transactions on Magnetics**, IEEE, v. 55, n. 12, p. 1–5, 2019.

GRANZ, Steven D; NGO, Tue; RAUSCH, Tim; BROCKIE, Richard; WOOD, Roger; BERTERO, Gerardo; GAGE, Edward C. Definition of an areal density metric for magnetic recording systems. **IEEE Transactions on Magnetics**, IEEE, v. 53, n. 2, p. 1–4, 2016.

GRID5000. 2020. Available from Internet: <<https://www.grid5000.fr>>.

GUO, Jiayang; HU, Yimin; MAO, Bo. Enhancing i/o scheduler performance by exploiting internal parallelism of ssds. In: SPRINGER. **International Conference on Algorithms and Architectures for Parallel Processing**. 2015. p. 118–130.

HAN, Wen-bing; CHEN, Xiao-gang; LI, Shun-Fen; LI, Ge-zi; SONG, Zhi-tang; LI, Da-Gang; CHEN, Shi-Yan. A novel non-volatile memory storage system for i/o-intensive applications. **Frontiers of Information Technology & Electronic Engineering**, Springer, v. 19, n. 10, p. 1291–1302, 2018.

HE, Shuibing; SUN, Xian-He; FENG, Bo. S4d-cache: Smart selective ssd cache for parallel i/o systems. In: IEEE. **2014 IEEE 34th International Conference on Distributed Computing Systems**. 2014. p. 514–523.

HE, Shuibing; WANG, Yang; SUN, Xian-He. Boosting parallel file system performance via heterogeneity-aware selective data layout. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 27, n. 9, p. 2492–2505, 2015.

HE, Weiping; DU, David HC. Smart: An approach to shingled magnetic recording translation. In: **15th {USENIX} Conference on File and Storage Technologies ({FAST} 17)**. 2017. p. 121–134.

HE, Youbiao; DAI, Dong; BAO, Forrest Sheng. Modeling hpc storage performance using long short-term memory networks. In: IEEE. **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)**. 2019. p. 1107–1114.

HEY, Tony; TANSLEY, Stewart; TOLLE, Kristin. The fourth paradigm. **Microsoft Research**, v. 722, 2009.

HOPPER, accessed Jun-29-2020. 2020. Available from Internet: <<"https://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2010/nersc-s-hopper-breaks-petaflops-barrier-ranks-5th-in-the-world/">>.

HOSOMI, M; YAMAGISHI, H; YAMAMOTO, T; BESSHO, K; HIGO, Y; YAMANE, K; YAMADA, H; SHOJI, M; HACHINO, H; FUKUMOTO, C et al. A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-ram. In: IEEE. **IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest.** 2005. p. 459–462.

HSIEH, Jen-Wei; LIN, Han-Yi; YANG, Dong-Lin. Multi-channel architecture-based ftl for reliable and high-performance ssd. **IEEE Transactions on Computers**, IEEE, v. 63, n. 12, p. 3079–3091, 2013.

HU, Yang; JIANG, Hong; FENG, Dan; TIAN, Lei; LUO, Hao; ZHANG, Shuping. Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity. In: **Proceedings of the international conference on Supercomputing**. 2011. p. 96–107.

HUANG, Min; XU, Ben; LIU, Zhaoqing; XU, Yishen; WU, Di. Implicit programming: a fast programming strategy for nand flash memory storage systems adopting redundancy methods. **IEEE Embedded Systems Letters**, IEEE, v. 9, n. 2, p. 37–40, 2017.

HUANG, Yaning; JIN, Hai; SHI, Xuanhua; WU, Song; CHEN, Yong. Cost-aware client-side file caching for data-intensive applications. In: IEEE. **2013 IEEE 5th International Conference on Cloud Computing Technology and Science**. 2013. v. 2, p. 248–251.

HUI, Jiao; GE, Xiongzi; HUANG, Xiaoxia; LIU, Yi; RAN, Qiangjun. E-hash: An energy-efficient hybrid storage system composed of one ssd and multiple hdds. In: SPRINGER. **International Conference in Swarm Intelligence**. 2012. p. 527–534.

HUO, Zhisheng; XIAO, Limin; ZHONG, Qiaoling; LI, Shupan; LI, Ang; RUAN, Li; WANG, Shouxin; FU, Lihong. A metadata cooperative caching architecture based on ssd and dram for file systems. In: SPRINGER. **Int. Conf. on Algorithms and Architectures for Parallel Processing**. 2015. p. 31–51.

HWANG, Euiseok; PARK, Jongseung; RAUSCHMAYER, Richard; WILSON, Bruce. Interlaced magnetic recording. **IEEE Transactions on Magnetics**, IEEE, v. 53, n. 4, p. 1–7, 2016.

INACIO, Eduardo C; DANTAS, Mario AR. Iore: A flexible and distributed i/o performance evaluation tool for hyperscale storage systems. In: IEEE. **IEEE Symp. on Comp. and Comm.** 2018. p. 01026–01031.

INOUE, Atsushi; WONG, Doug. Nand flash applications design guide. **Toshiba America Electronic Components Inc**, 2004.

INTERLEAVED Or Random (IOR). 2016. Available from Internet: <<https://github.com/LLNL/ior>>.

JI, Cheng; CHANG, Li-Pin; WU, Chao; SHI, Liang; XUE, Chun Jason. An i/o scheduling strategy for embedded flash storage devices with mapping cache. **IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 37, n. 4, p. 756–769, 2017.

JI, Cheng; WANG, Lun; LI, Qiao; GAO, Congming; SHI, Liang; YANG, Chia-Lin; XUE, Chun Jason. Fair down to the device: A gc-aware fair scheduler for ssd. In: IEEE. **2019 IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA)**. 2019. p. 1–6.

JI, Cheng; WU, Chao; CHANG, Li-Pin; SHI, Liang; XUE, Chun Jason. I/o scheduling with mapping cache awareness for flash based storage systems. In: **Proceedings of the 13th International Conference on Embedded Software**. 2016. p. 1–10.

JIANG, Zhiwen; ZHANG, Yong; WANG, Jin; XING, Chunxiao. A cost-aware buffer management policy for flash-based storage devices. In: SPRINGER. **International Conference on Database Systems for Advanced Applications**. 2015. p. 175–190.

JIN, Weitong; ZHU, Yanmin; HUANG, Linpeng. Accelerating traditional file systems on non-volatile main memory. In: IEEE. **2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)**. 2017. p. 453–460.

JO, Heeseung; KWON, Youngjin; KIM, Hwanju; SEO, Euiseong; LEE, Joonwon; MAENG, Seungryoul. Ssd-hdd-hybrid virtual disk in consolidated environments. In: SPRINGER. **European Conference on Parallel Processing**. 2009. p. 375–384.

JO, Myung Hyun; RO, Won Woo. Dynamic load balancing of dispatch scheduling for solid state disks. **IEEE Transactions on Computers**, IEEE, v. 66, n. 6, p. 1034–1047, 2016.

JOO, Yongsoo; RYU, Junhee; PARK, Sangsoo; SHIN, Kang G. Improving application launch performance on solid state drives. **Journal of Computer Science and Technology**, Springer, v. 27, n. 4, p. 727–743, 2012.

JR, J Presper Eckert; WEINER, James R; WELSH, H Frazer; MITCHELL, Herbert F. The univac system. In: **Papers and discussions presented at the Dec. 10-12, 1951, joint AIEE-IRE computer conference: Review of electronic digital computers**. 1951. p. 6–16.

JU, Gaoying; LI, Yongkun; XU, Yinlong; CHEN, Jiqiang; LUI, John CS. Stochastic modeling of hybrid cache systems. In: IEEE. **2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)**. 2016. p. 69–78.

JUNG, Myoungsoo; III, Ellis H Wilson; CHOI, Wonil; SHALF, John; AKTULGA, Hasan Metin; YANG, Chao; SAULE, Erik; CATALYUREK, Umit V; KANDEMIR, Mahmut. Exploring the future of out-of-core computing with compute-local non-volatile memory. **Scientific Programming**, IOS Press, v. 22, n. 2, p. 125–139, 2014.

KANNAN, Sudarsun; GAVRILOVSKA, Ada; SCHWAN, Karsten; MILOJICIC, Dejan; TALWAR, Vanish. Using active nvram for i/o staging. In: ACM. **Int. Work. on Petascale Data Analytics: challenges and opportunities**. 2011. p. 15–22.

KARIM, Mohd Bazli Ab; YUAN, Luke Jing; MING-TAT, Wong; ONG, Hong. Improving performance of database appliances on distributed object storage. In: IEEE. **2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)**. 2015. p. 45–52.

KEELE, Staffs et al. **Guidelines for performing systematic literature reviews in software engineering**. 2007.

KIM, Hwajung; YEOM, Heonyoung. Improving small file i/o performance for massive digital archives. In: IEEE. **2017 IEEE 13th International Conference on e-Science (e-Science)**. 2017. p. 256–265.

KIM, Hwajung; YEOM, Heon Young; SON, Yongseok. An i/o isolation scheme for key-value store on multiple solid-state drives. In: IEEE. **2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS* W)**. 2019. p. 170–175.

KIM, Hyun-Jin; LIM, Jeong-Don; LEE, Jang-Woo; NA, Dae-Hoon; SHIN, Joon-Ho; KIM, Chae-Hoon; YU, Seung-Woo; SHIN, Ji-Yeon; LEE, Seon-Kyoo; RAJAGOPAL, Devraj et al. 7.6 1gb/s 2tb nand flash multi-chip package with frequency-boosting interface chip. In: IEEE. **IEEE Int. Solid-State Circuits Conf.** 2015. p. 1–3.

KIM, Youngjae; TAURAS, Brendan; GUPTA, Aayush; URGAONKAR, Bhuvan. Flashsim: A simulator for nand flash-based solid-state drives. In: IEEE. **2009 First International Conference on Advances in System Simulation**. 2009. p. 125–131.

KITCHENHAM, Barbara. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004.

KITCHENHAM, Barbara A; BUDGEN, David; BRERETON, O Pearl. Using mapping studies as the basis for further research–a participant-observer case study. **Information and Software Technology**, Elsevier, v. 53, n. 6, p. 638–651, 2011.

KIVITY, A; KAMAY, Yaniv; LAOR, Dor; LUBLIN, U; LIGUORI, A. **kvm: the linux virtual machine monitor In: Proceedings of the Linux Symposium, 225–230**. : Canada, 2007.

KLONATOS, Yannis; MAKATOS, Thanos; MARAZAKIS, Manolis; FLOURIS, Michail D; BILAS, Angelos. Azor: Using two-level block selection to improve ssd-based i/o caches. In: IEEE. **2011 IEEE Sixth International Conference on Networking, Architecture, and Storage**. 2011. p. 309–318.

KOO, Gunjae; MATAM, Kiran Kumar; NARRA, HV; LI, Jing; TSENG, Hung-Wei; SWANSON, Steven; ANNAVARAM, Murali et al. Summarizer: trading communication with computing near storage. In: ACM. **Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture**. 2017. p. 219–231.

KRAKEN, accessed Jun-29-2020. 2020. Available from Internet: <<"https://www.nics.tennessee.edu/kraken/specifications">>.

KRISH, KR; ANWAR, Ali; BUTT, Ali R. hats: A heterogeneity-aware tiered storage for hadoop. In: IEEE. **2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing**. 2014. p. 502–511.

KRYDER, Mark H; GAGE, Edward C; MCDANIEL, Terry W; CHALLENER, William A; ROTTMAYER, Robert E; JU, Ganping; HSIA, Yiao-Tee; ERDEN, M Fatih. Heat assisted magnetic recording. **Proceedings of the IEEE**, IEEE, v. 96, n. 11, p. 1810–1835, 2008.

KWON, Kirock; KANG, Dong Hyun; EOM, Young Ik. An advanced slc-buffering for tlc nand flash-based storage. **IEEE Transactions on Consumer Electronics**, IEEE, v. 63, n. 4, p. 459–466, 2017.

LATHAM, Rob; BAUTISTA-GOMEZ, Leonardo; BALAJI, Pavan. Portable topology-aware mpi-i/o. In: IEEE. **2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)**. 2017. p. 710–719.

LEE, Dong Uk; KIM, Kyung Whan; KIM, Kwan Weon; LEE, Kang Seol; BYEON, Sang Jin; KIM, Jae Hwan; CHO, Jin Hee; LEE, Jaejin; CHUN, Jun Hyun. A 1.2 v 8 gb 8-channel 128 gb/s high-bandwidth memory (hbm) stacked dram with effective i/o test circuits. **IEEE Journal of Solid-State Circuits**, IEEE, v. 50, n. 1, p. 191–203, 2014.

LEE, Junghee; GANESH, Kalidas; LEE, Hyuk-Jun; KIM, Youngjae. Fessd: A fast encrypted ssd employing on-chip access-control memory. **IEEE Computer Architecture Letters**, IEEE, v. 16, n. 2, p. 115–118, 2017.

LEE, S; HYUN, S; KOH, K; BAHN, H. Efficient i/o processing scheme for flash memory storage systems. **Electronics letters**, IET, v. 47, n. 7, p. 436–437, 2011.

LEE, Sungjin; KIM, Jihong; MITHAL, Arvind. Refactored design of i/o architecture for flash storage. **IEEE Computer Architecture Letters**, IEEE, v. 14, n. 1, p. 70–74, 2014.

LEE, Sungjin; SHIN, Dongkun; KIM, Youngjin; KIM, Jihong. Exploiting sequential and temporal localities to improve performance of nand flash-based ssds. **ACM Transactions on Storage (TOS)**, ACM, v. 12, n. 3, p. 15, 2016.

LEE, Yangsup; JUNG, Sanghyuk; SONG, Yong Ho. Fra: A flash-aware redundancy array of flash storage devices. In: **Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis**. 2009. p. 163–172.

LI, Dingding; LIAO, Xiaofei; JIN, Hai; TANG, Yong; ZHAO, Gansen. Writeback throttling in a virtualized system with scm. **Frontiers of Computer Science**, Springer, v. 10, n. 1, p. 82–95, 2016.

LI, Hong-yan; XIONG, Nai-xue; HUANG, Ping; GUI, Chao. Pass: a simple, efficient parallelism-aware solid state drive i/o scheduler. **Journal of Zhejiang University SCIENCE C**, Springer, v. 15, n. 5, p. 321–336, 2014.

LI, Jianwei; LIAO, Wei-keng; CHOUDHARY, Alok; ROSS, Robert; THAKUR, Rajeev; GROPP, William; LATHAM, Robert; SIEGEL, Andrew; GALLAGHER, Brad; ZINGALE, Michael. Parallel netcdf: A high-performance scientific i/o interface. In: IEEE. **SC'03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing**. 2003. p. 39–39.

LI, Yu; ON, Sai Tung; XU, Jianliang; CHOI, Byron; HU, Haibo. Digestjoin: Exploiting fast random reads for flash-based joins. In: IEEE. **2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware**. 2009. p. 152–161.

LI, Zheng; WANG, Fang; LIU, Jingning; FENG, Dan; HUA, Yu; TONG, Wei; ZHANG, Shuangwu. A user-visible solid-state storage system with software-defined fusion methods for pcm and nand flash. **Journal of Systems Architecture**, Elsevier, v. 71, p. 44–61, 2016.

LIANG, Weihao; CHEN, Yong; AN, Hong. Interference-aware i/o scheduling for data-intensive applications on hierarchical hpc storage systems. In: IEEE. **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)**. 2019. p. 654–661.

LIANG, Weihao; CHEN, Yong; LIU, Jialin; AN, Hong. Cars: A contention-aware scheduler for efficient resource management of hpc storage systems. **Parallel Computing**, Elsevier, v. 87, p. 25–34, 2019.

LIAO, Jianwei; LIU, Xiaoyan; CHEN, Yingshen. Dynamical re-striping data on storage servers in parallel file systems. In: IEEE. **2013 IEEE 37th Annual Computer Software and Applications Conference**. 2013. p. 65–73.

LIU, Duo; ZHONG, Kan; WANG, Tianzheng; WANG, Yi; SHAO, Zili; SHA, Edwin Hsing-Mean; XUE, Jingling. Durable address translation in pcm-based flash storage systems. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 28, n. 2, p. 475–490, 2016.

LIU, Jialin; BYNA, Surendra; DONG, Bin; WU, Kesheng; CHEN, Yong. Model-driven data layout selection for improving read performance. In: IEEE. **IEEE Int. Par. & Distrib. Proc. Symp. Workshops**. 2014. p. 1708–1716.

LIU, Qing; PODHORSZKI, Norbert; CHOI, Jong; LOGAN, Jeremy; WOLF, Matt; KLASKY, Scott; KURC, Tahsin; HE, Xubin. Storerush: An application-level approach to harvesting idle storage in a best effort environment. **Procedia Computer Science**, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States). Oak Ridge . . . , v. 108, n. C, 2017.

LIU, Xin; LU, Yu-tong; YU, Jie; WANG, Peng-fei; WU, Jie-ting; LU, Ying. Onfs: a hierarchical hybrid file system based on memory, ssd, and hdd for high performance computers. **Frontiers of Information Technology & Electronic Engineering**, Springer, v. 18, n. 12, p. 1940–1971, 2017.

LIU, Yan; HUANG, Xin; HUANG, Yizi; GENG, Shaofeng; PENG, Xin; LI, Renfa. A variable-sized stripe level data layout strategy for hdd/ssd hybrid parallel file systems. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 29, n. 20, p. e4039, 2017.

LIU, Zhuo; WANG, Bin; YU, Weikuan. Halo: a fast and durable disk write cache using phase change memory. **Cluster Computing**, Springer, v. 21, n. 2, p. 1275–1287, 2018.

LOFSTEAD, Jay F; KLASKY, Scott; SCHWAN, Karsten; PODHORSZKI, Norbert; JIN, Chen. Flexible io and integration for scientific codes through the adaptable io system (adios). In: **Proceedings of the 6th international workshop on Challenges of large applications in distributed environments**. 2008. p. 15–24.

MAO, Bo; WU, Suzhen; DUAN, Lide. Improving the ssd performance by exploiting request characteristics and internal parallelism. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 37, n. 2, p. 472–484, 2017.

MAO, Bo; WU, Suzhen; JIANG, Hong; YANG, Yaodong; XI, Zaifa. Edc: Improving the performance and space efficiency of flash-based storage systems with elastic data compression. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 29, n. 6, p. 1261–1274, 2018.

MARKS, Benjamin; NEWHALL, Tia. Transparent heterogeneous backing store for file systems. In: IEEE. **2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. 2017. p. 30–41.

MATIVENGA, Ronnie; PAIK, Joon-Young; KIM, Youngjae; LEE, Junghee; CHUNG, Tae-Sun. Rftl: improving performance of selective caching-based page-level ftl through replication. **Cluster Computing**, Springer, v. 22, n. 1, p. 25–41, 2019.

MDTEST, last access 06-2020. [s.d.]. Available from Internet: <<https://github.com/MDTEST-LANL/mdtest>>.

MEENA, Jagan Singh; SZE, Simon Min; CHAND, Umesh; TSENG, Tseung-Yuen. Overview of emerging nonvolatile memory technologies. **Nanoscale research letters**, Springer, v. 9, n. 1, p. 526, 2014.

MEUER, Hans; STROHMAIER, Erich; DONGARRA, Jack; SIMON, Horst. **Top500 supercomputer sites**. 2001.

MICROCHIP - Introduction to SMR (Shingled Magnetic Recording) Drives, Accessed on May 03, 2020. 2019. Available from Internet: <<"https://ask.adaptec.com/app/answers/detail/a_id/17472/~/introduction-to-smr-%28shingled-magnetic-recording%29-drives">>.

MIDORIKAWA, Hiroko; TAN, Hideyuki. Evaluation of flash-based out-of-core stencil computation algorithms for ssd-equipped clusters. In: IEEE. **2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)**. 2016. p. 1031–1040.

MOON, Sangwhan; LEE, Jaehwan; SUN, Xiling; KEE, Yang-suk. Optimizing the hadoop mapreduce framework with high-performance storage devices. **The Journal of Supercomputing**, Springer, v. 71, n. 9, p. 3525–3548, 2015.

MURDOCCA, MJ; HEURING, VP. Principles of computer architecture: Class test edition. **USA: New Jersy**, p. 266–279, 1999.

NAKASHIMA, Kenji; KON, Joichiro; YAMAGUCHI, Saneyasu. I/o performance improvement of secure big data analyses with application support on ssd cache. In: ACM. **Int. Conf. on Ubiq. Inf. Management and Comm.** 2018. p. 90.

NAKASHIMA, Kenji; KON, Joichiro; YAMAGUCHI, Saneyasu; LEE, Gil Jae; FORTES, José. 1a study on big data i/o performance with modern storage systems. In: IEEE. **2017 IEEE International Conference on Big Data (Big Data)**. 2017. p. 4798–4799.

NASA MODIS, last access 06-2020. [s.d.]. Available from Internet: <<http://modis.gsfc.nasa.gov/>>.

NERSC Systems, accessed in Jun-2020. 2020. Available from Internet: <<"https://www.nersc.gov/about/nersc-history/history-of-systems/">>.

NI, Yunzhu; LI, Zhishu. Minimizing the response time of a striped disk array. In: IEEE. **2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics**. 2010. v. 2, p. 200–204.

NICOLAE, Bogdan; RITEAU, Pierre; KEAHEY, Kate. Transparent throughput elasticity for iaas cloud storage using guest-side block-level caching. In: IEEE. **2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing**. 2014. p. 186–195.

NIJIM, Mais; SAHA, Soumya; NIJIM, Yousef. Central and distributed gpu based parallel disk systems for data intensive applications. **Procedia Computer Science**, Elsevier, v. 34, p. 338–343, 2014.

NORCOTT, WilliamD. Iozone filesystem benchmark. **http://www. iozone. org/**, 2003.

OH, Yongseok; CHOI, Jongmoo; LEE, Donghee; NOH, Sam H. Caching less for better performance: balancing cache size and update cost of flash memory cache in hybrid storage systems. In: **FAST**. 2012. v. 12.

OIKAWA, Shuichi. Virtualizing storage as memory for high performance storage access. In: IEEE. **2014 IEEE International Symposium on Parallel and Distributed Processing with Applications**. 2014. p. 18–25.

OPTICAL Storage - Photos courtesy of Philips Research. 2000. Available from Internet: <<http://www.extra.research.philips.com/pressmedia/pictures/passw2.html>>.

OU, Yang; WU, Xiaoquan; XIAO, Nong; LIU, Fang; CHEN, Wei. Nis: a new index scheme for flash file system. In: IEEE. **2015 Third International Conference on Advanced Cloud and Big Data**. 2015. p. 44–51.

OU, Yang; XIAO, Nong; LIU, Fang; CHEN, Zhiguang; CHEN, Wei; WU, Lizhou. Gemini: a novel hardware and software implementation of high-performance pcie ssd. **International Journal of Parallel Programming**, Springer, v. 45, n. 4, p. 923–945, 2017.

PAN, Yubiao; LI, Yongkun; ZHANG, Huizhen; XU, Yinlong. Lifetime-aware ftl to improve the lifetime and performance of solid-state drives. **Future Generation Computer Systems**, Elsevier, v. 93, p. 58–67, 2019.

PARK, Daekyu; KANG, Dong Hyun; AHN, Seung Min; EOM, Young Ik. The minimal-effort write i/o scheduler for flash-based storage devices. In: IEEE. **2018 IEEE International Conference on Consumer Electronics (ICCE)**. 2018. p. 1–3.

PARK, Junseok; BAHN, Hyokyung; KOH, Kern. Buffer cache management for combined mlc and slc flash memories using both volatile and nonvolatile rams. In: IEEE. **2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications**. 2009. p. 228–235.

PARK, Jiwoong; MIN, Cheolgi; YEOM, Heon Young; SON, Yongseok. z-read: Towards efficient and transparent zero-copy read. In: IEEE. **2019 IEEE 12th International Conference on Cloud Computing (CLOUD)**. 2019. p. 367–371.

PARK, Jung Kyu; SEO, Yunjung; KIM, Jaeho. A flash-based ssd cache management scheme for high performance home cloud storage. **IEEE Transactions on Consumer Electronics**, IEEE, v. 65, n. 3, p. 418–425, 2019.

PETERSEN, Torben Kling; BENT, John. Hybrid flash arrays for hpc storage systems: An alternative to burst buffers. In: IEEE. **2017 IEEE High Performance Extreme Computing Conference (HPEC)**. 2017. p. 1–7.

PETTICREW, M; ROBERTS, H. Systematic reviews in the social sciences: a practical guide. 2006. **Malden USA: Blackwell Publishing CrossRef Google Scholar**, [s.d.].

PIOLI, Laércio; MENEZES, Victor Ströele de Andrade; DANTAS, Mario Antonio Ribeiro. Research characterization on i/o improvements of storage environments. In: SPRINGER. **Int. Conf. on P2P, Parallel, Grid, Cloud and Internet Computing**. 2019. p. 287–298.

PRABHAKAR, Ramya; SRIKANTAIAH, Shekhar; GARG, Rajat; KANDEMIR, Mahmut. Adaptive qos decomposition and control for storage cache management in multi-server environments. In: IEEE. **2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing**. 2011. p. 402–413.

PRABHAKAR, Ramya; VAZHKUDAI, Sudharshan S; KIM, Youngjae; BUTT, Ali R; LI, Min; KANDEMIR, Mahmut. Provisioning a multi-tiered data staging area for extreme-scale machines. In: IEEE. **2011 31st International Conference on Distributed Computing Systems**. 2011. p. 1–12.

RAMASAMY, Arul Selvan; KARANTHARAJ, Porkumaran. Rffe: A buffer cache management algorithm for flash-memory-based ssd to improve write performance. **Canadian Journal of Electrical and Computer Engineering**, IEEE, v. 38, n. 3, p. 219–231, 2015.

RAVANDI, Babak; PAPAPANAGIOTOU, Ioannis. A self-organized resource provisioning for cloud block storage. **Future Generation Computer Systems**, Elsevier, v. 89, p. 765–776, 2018.

RAYNAUD, Tanguy; HAQUE, Rafiqul; AÏT-KACI, Hassan. Cedcom: A high-performance architecture for big data applications. In: IEEE. **2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)**. 2014. p. 621–632.

REINSEL, David; GANTZ, John; RYDNING, John. Data age 2025: the digitization of the world from edge to core. **Seagate, https://www. seagate. com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper. pdf**, 2018.

RISKA, Alma; RIEDEL, Erik. Disk drive level workload characterization. In: **USENIX Annual Technical Conf., General Track**. 2006. v. 2006, p. 97–102.

ROMIO: A High-Performance, Portable MPI-IO Implementation. 2020. Available from Internet: <<http://www.mcs.anl.gov/research/projects/romio/index-archived.html>>.

ROSENFELD, Paul; COOPER-BALIS, Elliott; JACOB, Bruce. Dramsim2: A cycle accurate memory system simulator. **IEEE computer architecture letters**, IEEE, v. 10, n. 1, p. 16–19, 2011.

ROSS, Robert B; THAKUR, Rajeev et al. Pvfs: A parallel file system for linux clusters. In: **Proceedings of the 4th annual Linux showcase and Conf.** 2000. p. 391–430.

RUAN, Xiaojun; CHEN, Haiquan. Improving shuffle i/o performance for big data processing using hybrid storage. In: IEEE. **2017 International Conference on Computing, Networking and Communications (ICNC)**. 2017. p. 476–480.

SAIF, Abdulqawi; NUSSBAUM, Lucas; SONG, Ye-Qiong. Ioscope: A flexible i/o tracer for workloads' i/o pattern characterization. In: SPRINGER. **International Conference on High Performance Computing**. 2018. p. 103–116.

SALKHORDEH, Reza; EBRAHIMI, Shahriar; ASADI, Hossein. Reca: An efficient reconfigurable cache architecture for storage systems with online workload characterization. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 29, n. 7, p. 1605–1620, 2018.

SCHÜRMANN, Felix; DELALONDRE, Fabien; KUMBHAR, Pramod S; BIDDISCOMBE, John; GILA, Miguel; TACCHELLA, Davide; CURIONI, Alessandro; METZLER, Bernard; MORJAN, Peter; FENKES, Joachim et al. Rebasing i/o for scientific computing: Leveraging storage class memory in an ibm bluegene/q supercomputer. In: SPRINGER. **International Supercomputing Conference**. 2014. p. 331–347.

SEAGATE SMR- Increase Drive Capacity, Accessed on May 03, 2020. 2020. Available from Internet: <<"hhttps://www.seagate.com/br/pt/tech-insights/breaking-areal-density-barriers-with-seagate-smr-master-ti/">>.

SHAN, Hongzhang; ANTYPAS, Katie; SHALF, John. Characterizing and predicting the i/o performance of hpc applications using a parameterized synthetic benchmark. In: IEEE. **SC'08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing**. 2008. p. 1–12.

SHANKAR, Dipti; LU, Xiaoyi; PANDA, Dhabaleswar K. High-performance and resilient key-value store with online erasure coding for big data workloads. In: IEEE. **2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)**. 2017. p. 527–537.

SHANKAR, Dipti; LU, Xiaoyi; PANDA, Dhabaleswar K DK. Boldio: A hybrid and resilient burst-buffer over lustre for accelerating big data i/o. In: IEEE. **2016 IEEE International Conference on Big Data (Big Data)**. 2016. p. 404–409.

SINGH, Karishma; RASTOGI, Divya; SINGH, Dayashankar et al. Optimized two head disk scheduling algorithm (othdsa). In: IEEE. **2015 Fifth International Conference on Advanced Computing & Communication Technologies**. 2015. p. 234–240.

SKOURTIS, Dimitris; KATO, Shinpei; BRANDT, Scott. Qbox: guaranteeing i/o performance on black box storage systems. In: **Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing**. 2012. p. 73–84.

SNIR, Marc; GROPP, William; OTTO, Steve; HUSS-LEDERMAN, Steven; DONGARRA, Jack; WALKER, David. **MPI–the Complete Reference: the MPI core**. : MIT press, 1998.

SON, Seung Woo; SEHRISH, Saba; LIAO, Wei-keng; OLDFIELD, Ron; CHOUDHARY, Alok. Reducing i/o variability using dynamic i/o path characterization in petascale storage systems. **The Journal of Supercomputing**, Springer, v. 73, n. 5, p. 2069–2097, 2017.

SONG, Huaiming; SUN, Xian-He; CHEN, Yong. A hybrid shared-nothing/shared-data storage scheme for large-scale data processing. In: IEEE. **2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications**. 2011. p. 161–166.

SOULÉ, Robert; GEDIK, Buğra. Railwaydb: adaptive storage of interaction graphs. **The VLDB Journal**, Springer, v. 25, n. 2, p. 151–169, 2016.

SRINIVASAN, Kiran; BISSON, Timothy; GOODSON, Garth R; VORUGANTI, Kaladhar. idedup: latency-aware, inline data deduplication for primary storage. In: **Fast**. 2012. v. 12, p. 1–14.

STRUKOV, Dmitri B; SNIDER, Gregory S; STEWART, Duncan R; WILLIAMS, R Stanley. The missing memristor found. **nature**, Nature Publishing Group, v. 453, n. 7191, p. 80–83, 2008.

STRUNK, John D. Hybrid aggregates: Combining ssds and hdds in a single storage pool. **ACM SIGOPS Operating Systems Review**, ACM New York, NY, USA, v. 46, n. 3, p. 50–56, 2012.

SUN, Hui; QIN, Xiao; XIE, Chang-sheng. Exploring optimal combination of a file system and an i/o scheduler for underlying solid state disks. **Journal of Zhejiang University SCIENCE C**, Springer, v. 15, n. 8, p. 607–621, 2014.

TANENBAUM, Andrew S. **Distributed operating systems**. : Pearson Education India, 1995.

TANG, Houjun; BYNA, Suren; TESSIER, François; WANG, Teng; DONG, Bin; MU, Jingqing; KOZIOL, Quincey; SOUMAGNE, Jerome; VISHWANATH, Venkatram; LIU, Jialin et al. Toward scalable and asynchronous object-centric data management for hpc. In: IEEE. **2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)**. 2018. p. 113–122.

TANG, Kun; HUANG, Ping; HE, Xubin; LU, Tao; VAZHKUDAI, Sudharshan S; TIWARI, Devesh. Toward managing hpc burst buffers effectively: Draining strategy to regulate bursty i/o behavior. In: IEEE. **2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)**. 2017. p. 87–98.

TEAM, L.; ALSTRIN, A.; GOKER, T.; LANTZ, M.; MCALLISTER, J.; SPRATT, G.; ENTERPRISE, H. P. "2.0 technology roadmap 2.0 participants". n. December, p. 1–21, 2015.

TESSIER, François; MALAKAR, Preeti; VISHWANATH, Venkatram; JEANNOT, Emmanuel; ISAILA, Florin. Topology-aware data aggregation for intensive i/o on large-scale supercomputers. In: IEEE PRESS. **Proceedings of the First Workshop on Optimization of Communication in HPC**. 2016. p. 73–81.

THE technology stacks of HPC and Big Data Computing. 2018. Available from Internet: <<http://www.bdva.eu/node/1150>>.

TITAN, accessed Jun-29-2020. 2020. Available from Internet: <<"https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/">>.

TRAEGER, Avishay; ZADOK, Erez; JOUKOV, Nikolai; WRIGHT, Charles P. A nine year study of file system and storage benchmarking. **ACM Transactions on Storage (TOS)**, ACM New York, NY, USA, v. 4, n. 2, p. 1–56, 2008.

VALDURIEZ, Patrick; MATTOSO, Marta; AKBARINIA, Reza; BORGES, Heraldo; CAMATA, José; COUTINHO, Alvaro; GASPAR, Daniel; LEMUS, Noel; LIU, Ji; LUSTOSA, Hermano et al. Scientific data analysis using data-intensive scalable computing: The scidisc project. In: . 2018.

WADHWA, Bharti; BYNA, Suren; BUTT, Ali R. Toward transparent data management in multi-layer storage hierarchy of hpc systems. In: IEEE. **2018 IEEE International Conference on Cloud Engineering (IC2E)**. 2018. p. 211–217.

WAN, Jiguang; ZHAO, Nannan; ZHU, Yifeng; WANG, Jibin; MAO, Yu; CHEN, Peng; XIE, Changsheng. High performance and high capacity hybrid shingled-recording disk system. In: IEEE. **2012 IEEE International Conference on Cluster Computing**. 2012. p. 173–181.

WANG, Chao; VAZHKUDAI, Sudharshan S; MA, Xiaosong; MENG, Fei; KIM, Youngjae; ENGELMANN, Christian. Nvmalloc: Exposing an aggregate ssd store as a memory partition in extreme-scale machines. In: IEEE. **2012 IEEE 26th International Parallel and Distributed Processing Symposium**. 2012. p. 957–968.

WANG, Haitao; LI, Zhanhuai; ZHANG, Xiao; ZHAO, Xiaonan; ZHAO, Xingsheng; LI, Weijun; JIANG, Song. Oc-cache: An open-channel ssd based cache for multi-tenant systems. In: IEEE. **2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)**. 2018. p. 1–6.

WANG, Jianzong; CHENG, Lianglun. qsds: A qos-aware i/o scheduling framework towards software defined storage. In: IEEE. **2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)**. 2015. p. 195–196.

WANG, Jiangtao; GUO, Zhiliang; MENG, Xiaofeng. Sass: A high-performance key-value store design for massive hybrid storage. In: SPRINGER. **International Conference on Database Systems for Advanced Applications**. 2015. p. 145–159.

WANG, Teng; BYNA, Suren; DONG, Bin; TANG, Houjun. Univistor: Integrated hierarchical and distributed storage for hpc. In: IEEE. **2018 IEEE International Conference on Cluster Computing (CLUSTER)**. 2018. p. 134–144.

WANG, Teng; ORAL, Sarp; WANG, Yandong; SETTLEMYER, Brad; ATCHLEY, Scott; YU, Weikuan. Burstmem: A high-performance burst buffer system for scientific applications. In: IEEE. **2014 IEEE International Conference on Big Data (Big Data)**. 2014. p. 71–79.

WANG, Yi; DONG, Lisha; MING, Zhong. Optimizing emerging storage primitives with virtualization for flash memory storage systems. In: IEEE. **2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems**. 2015. p. 672–677.

WANG, Yi; HUANG, Min; SHAO, Zili; CHAN, Henry CB; BATHEN, Luis Angel D; DUTT, Nikil D. A reliability-aware address mapping strategy for nand flash memory storage systems. **IEEE Transactions on computer-aided design of integrated circuits and systems**, IEEE, v. 33, n. 11, p. 1623–1631, 2014.

WANG, Yi; QIN, Zhiwei; CHEN, Renhai; SHAO, Zili; WANG, Qixin; LI, Shuai; YANG, Laurence T. A real-time flash translation layer for nand flash memory storage systems. **IEEE Transactions on Multi-Scale Computing Systems**, IEEE, v. 2, n. 1, p. 17–29, 2016.

WEIL, Sage A; BRANDT, Scott A; MILLER, Ethan L; LONG, Darrell DE; MALTZAHN, Carlos. Ceph: A scalable, high-performance distributed file system. In: USENIX ASSOCIATION. **Proceedings of the 7th Symp. on Operating systems design and implementation**. 2006. p. 307–320.

WICKBERG, Tim; CAROTHERS, Christopher. The ramdisk storage accelerator: a method of accelerating i/o performance on hpc systems using ramdisks. In: **Proceedings of the 2nd International Workshop on Runtime and Operating Systems for Supercomputers**. 2012. p. 1–8.

WOHLIN, Claes; RUNESON, Per; HÖST, Martin; OHLSSON, Magnus C; REGNELL, Björn; WESSLÉN, Anders. **Experimentation in software engineering**. : Springer Science & Business Media, 2012.

WONG, H-S Philip; RAOUX, Simone; KIM, SangBum; LIANG, Jiale; REIFENBERG, John P; RAJENDRAN, Bipin; ASHEGHI, Mehdi; GOODSON, Kenneth E. Phase change memory. **Proceedings of the IEEE**, IEEE, v. 98, n. 12, p. 2201–2227, 2010.

WU, Chin-Hsien; HUANG, Cheng-Wei; CHANG, Chen-Yu. A priority-based data placement method for databases using solid-state drives. In: ACM. **Conf. on Research in Adaptive and Convergent Syst.** 2018. p. 175–182.

WU, Chin-Hsien; HUANG, Cheng-Wei; CHANG, Chen-Yu. A data management method for databases using hybrid storage systems. **ACM SIGAPP Applied Computing Review**, ACM New York, NY, USA, v. 19, n. 1, p. 34–47, 2019.

WU, Fenggang; LI, Bingzhe; CAO, Zhichao; ZHANG, Baoquan; YANG, Ming-Hong; WEN, Hao; DU, David HC. Zonealloy: Elastic data and space management for hybrid {SMR} drives. In: **11th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 19)**. 2019.

XIAO, Chunhua; ZHANG, Lei; LIU, Weichen; CHENG, Linfeng; LI, Pengda; PAN, Yanyue; BERGMANN, Neil. Nv-ecryptfs: Accelerating enterprise-level cryptographic file system with non-volatile memory. **IEEE Transactions on Computers**, IEEE, v. 68, n. 9, p. 1338–1352, 2018.

XIAO, Li; YU-AN, Tan; ZHIZHUO, Sun. Semi-raid: A reliable energy-aware raid data layout for sequential data access. In: IEEE. **2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)**. 2011. p. 1–11.

XIAO, Weijun; LEI, Xiaoqiang; LI, Ruixuan; PARK, Nohhyun; LILJA, David J. Pass: a hybrid storage system for performance-synchronization tradeoffs using ssds. In: IEEE. **2012 IEEE 10th international symposium on parallel and distributed processing with applications**. 2012. p. 403–410.

XIE, Wei; CHEN, Yong; ROTH, Philip C. Asa-ftl: An adaptive separation aware flash translation layer for solid state drives. **Parallel Computing**, Elsevier, v. 61, p. 3–17, 2017.

XIE, Wei; ZHOU, Jiang; REYES, Mark; NOBLE, Jason; CHEN, Yong. Two-mode data distribution scheme for heterogeneous storage in data centers. In: IEEE. **IEEE Int. Conf. on Big Data**. 2015. p. 327–332.

XU, Qi; CHENG, Yaodong; CHEN, Gang. Design and evaluation of a hybrid storage system in hep environment. **JPhCS**, v. 898, n. 6, p. 062034, 2017.

XU, Yuan-Chao; WAN, Hu; QIU, Ke-Ni; LI, Tao; ZHANG, Wei-Gong. Reducing synchronization cost for single-level store in mobile systems. **Journal of Computer Science and Technology**, Springer, v. 31, n. 4, p. 836–848, 2016.

YAN, Guohua; CHEN, Renhai; GUAN, Qiming; FENG, Zhiyong. Dls: A delay-life-aware i/o scheduler to improve the load balancing of ssd-based raid-5 arrays. In: IEEE. **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)**. 2019. p. 2445–2450.

YANG, Chun; LIU, Xianhua; CHENG, Xu. Content look-aside buffer for redundancy-free virtual disk i/o and caching. In: ACM. **ACM SIGPLAN Notices**. 2017. v. 52, n. 7, p. 214–227.

YANG, Jing; PEI, Shuyi; YANG, Qing. Warcip: write amplification reduction by clustering i/o pages. In: ACM. **ACM Int. Conf. on Systems and Storage**. 2019. p. 155–166.

YANG, Jing; YANG, Qing. A new metadata update method for fast recovery of ssd cache. In: IEEE. **2013 IEEE Eighth International Conference on Networking, Architecture and Storage**. 2013. p. 60–67.

YANG, Puyuan; JIN, Peiquan; WAN, Shouhong; YUE, Lihua. Hb-storage: Optimizing ssds with a hdd write buffer. In: SPRINGER. **International Conference on Web-Age Information Management**. 2013. p. 28–39.

YANG, Tianming; HUANG, Ping; ZHANG, Weiying; WU, Haitao; LIN, Longxin. Cars: A multi-layer conflict-aware request scheduler for nvme ssds. In: IEEE. **2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)**. 2019. p. 1293–1296.

YANG, Xue-Jun; LIAO, Xiang-Ke; LU, Kai; HU, Qing-Feng; SONG, Jun-Qiang; SU, Jin-Shu. The tianhe-1a supercomputer: its hardware and software. **Journal of computer science and technology**, Springer, v. 26, n. 3, p. 344–351, 2011.

YANG, Zhengyu; HOSEINZADEH, Morteza; ANDREWS, Allen; MAYERS, Clay; EVANS, David Thomas; BOLT, Rory Thomas; BHIMANI, Janki; MI, Ningfang; SWANSON, Steven. Autotiering: automatic data placement manager in multi-tier all-flash datacenter. In: IEEE. **2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)**. 2017. p. 1–8.

YANG, Zhengyu; HOSEINZADEH, Morteza; WONG, Ping; ARTOUX, John; MAYERS, Clay; EVANS, David Thomas; BOLT, Rory Thomas; BHIMANI, Janki; MI, Ningfang; SWANSON, Steven. H-nvme: a hybrid framework of nvme-based storage system in cloud computing environment. In: IEEE. **2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)**. 2017. p. 1–8.

YEH, Tsozen; YANG, Shuwen; SUN, Yifeng. Improving the program performance through prioritized memory management and disk operation. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 27, n. 13, p. 3345–3361, 2015.

YIN, Jianwei; TANG, Yan; DENG, Shuiguang; LI, Ying; LO, Wei; DONG, Kexiong; ZOMAYA, Albert Y; PU, Calton. Asser: an efficient, reliable, and cost-effective storage scheme for object-based cloud storage systems. **IEEE Transactions on Computers**, IEEE, v. 66, n. 8, p. 1326–1340, 2017.

YOU, Taehee; HAN, Sangwoo; PARK, Young Min; LEE, Hyuk-Jun; CHUNG, Eui-Young. Multi-token based power management for nand flash storage devices. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, 2019.

YOU, Xin; YANG, Hailong; LUAN, Zhongzhi; LIU, Yi; QIAN, Depei. Performance evaluation and analysis of linear algebra kernels in the prototype tianhe-3 cluster. In: SPRINGER. **Asian Conference on Supercomputing Frontiers**. 2019. p. 86–105.

YOUN, Young-Sun; YOON, Su-Kyung; KIM, Shin-Dug. Cloud computing burst system (ccbs): for exa-scale computing system. **The Journal of Supercomputing**, Springer, v. 73, n. 9, p. 4020–4041, 2017.

YU, Jie; LIU, Guangming; DONG, Wenrui; LI, Xiaoyong. Using locality-enhanced distributed memory cache to accelerate applications on high performance computers. In: IEEE. **2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)**. 2017. p. 160–166.

YU, Jie; LIU, Guangming; DONG, Wenrui; LI, Xiaoyong. Watcache: a workload-aware temporary cache on the compute side of hpc systems. **The Journal of Supercomputing**, Springer, v. 75, n. 2, p. 554–586, 2019.

ZHA, Benbo; SHEN, Hong. Memory hierarchy aware i/o scheduling under contention for hybrid storage based hpc. In: IEEE. **2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)**. 2018. p. 69–73.

ZHANG, Hao; CHEN, Gang; OOI, Beng Chin; TAN, Kian-Lee; ZHANG, Meihui. In-memory big data management and processing: A survey. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 27, n. 7, p. 1920–1948, 2015.

ZHANG, Jie; DONOFRIO, David; SHALF, John; KANDEMIR, Mahmut T; JUNG, Myoungsoo. Nvmmu: A non-volatile memory management unit for heterogeneous gpu-ssd architectures. In: IEEE. **2015 International Conference on Parallel Architecture and Compilation (PACT)**. 2015. p. 13–24.

ZHANG, Jingchao; MENG, Fankuo; QIAO, Liyan; ZHU, Kaihui. Design and implementation of optical fiber ssd exploiting fpga accelerated nvme. **IEEE Access**, IEEE, v. 7, p. 152944–152952, 2019.

ZHANG, Tong; CHENG, Ze; LI, Jing. Reinforcement learning-driven address mapping and caching for flash-based remote sensing image processing. **Journal of Systems Architecture**, Elsevier, v. 98, p. 374–387, 2019.

ZHANG, Xuechen; DAVIS, Kei; JIANG, Song. itransformer: Using ssd to improve disk scheduling for high-performance i/o. In: IEEE. **Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International**. 2012. p. 715–726.

ZHANG, Xiao; WANG, Yanqiu; WANG, Qing; ZHAO, Xiaonan. A new approach to double i/o performance for ceph distributed file system in cloud computing. In: IEEE. **2019 2nd International Conference on Data Intelligence and Security (ICDIS)**. 2019. p. 68–75.

ZHANG, Zheng; FENG, Dan; TAN, Zhipeng; YANG, Laurence T; ZHENG, Jiayang. A light-weight log-based hybrid storage system. **Journal of Parallel and Distributed Computing**, Elsevier, v. 118, p. 307–315, 2018.

ZHAO, Nannan; ANWARE, Ali; CHENG, Yue; SALMAN, Mohammed; LI, Daping; WAN, Jiguang; XIE, Changsheng; HE, Xubin; WANG, Feiyi; BUTT, Ali. Chameleon: An adaptive wear balancer for flash clusters. In: IEEE. **2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)**. 2018. p. 1163–1172.

ZHOU, Jiang; CHEN, Yong; WANG, Weiping. Atributed consistent hashing for heterogeneous storage systems. In: **PACT**. 2018. p. 23–1.

ZHOU, Jiang; XIE, Wei; GU, Qiang; CHEN, Yong. Hierarchical consistent hashing for heterogeneous object-based storage. In: IEEE. **2016 IEEE Trustcom/BigDataSE/ISPA**. 2016. p. 1597–1604.

ZHOU, Jiang; XIE, Wei; NOBLE, Jason; ECHO, Kace; CHEN, Yong. Suora: A scalable and uniform data distribution algorithm for heterogeneous storage systems. In: IEEE. **IEEE Int. Conf. on Net., Arch. and Storage**. 2016. p. 1–10.

ZHU, Jian-Gang; ZHU, Xiaochun; TANG, Yuhui. Microwave assisted magnetic recording. **IEEE Transactions on Magnetics**, IEEE, v. 44, n. 1, p. 125–131, 2007.

APPENDIX A - PUBLICATIONS

Title - Research Characterization on I/O Improvements of Storage Environments
Abstract: Nowadays, it has being verified some interesting improvements in I/O architectures. This is an essential point to complex and data intensive scalable applications. In the scientific and industrial fields, the storage component is a key element, because usually those applications employ a huge amount of data. Therefore, the performance of these applications commonly depends on some factors related to time spent in execution of the I/O operations. In this paper we present a characterization research on I/O improvements related to the storage targeting HPC and DISC applications. We also evaluated some of these improvements in order to justify their concerns with the I/O layer. Our experiments were processed in the Grid5000, an interesting testbed distributed environment, suitable for better understanding challenges related to HPC and DISC applications. Results on synthetic I/O benchmarks, demonstrate how to improve the performance of the latency parameter for I/O operations.
Conference: 3PGCIC-2019
Qualis: B1
Status: Published
Author: - Corresponding author

Title - An Effort to Characterize I/O Enhancements of Storage Environments
Abstract: Improvements in I/O architectures are becoming increasingly needed nowadays. This is an essential point to complex applications such as High-performance computing (HPC) and data-intensive scalable computing (DISC). In the scientific and industrial fields, the storage component is a key element, because usually those applications employ a huge amount of data. Therefore, the performance of these applications commonly depends on some factors related to time spent in execution of the I/O operations. In this scenario, researchers are proposing several approaches to deal and solve such issue. Many of than are concerned in mixing hardware devices whereas others are based on the improvement of software located in an up layer. All these improvements have the same goal which is increase the IOPS ratio of overall system. A characterization model for classifying research works on I/O performance improvements for storage environments are presented in this paper. We also evaluated some of these improvements in order to justify their concerns with the I/O layer. Our experiments were performed in the Grid'5000. Results over 36 different scenarios demonstrate how to improve the performance of the latency parameter for I/O operations. **Journal:** IJGUC
Qualis: A2
Status: Accepted
Author: - Corresponding author

Title - Characterization Research on I/O Improvements Targeting DISC and HPC

Applications.
Abstract: Improvements in I/O architectures are becoming increasingly required nowadays. This is an essential point to complex and data intensive scalable applications. Data-Intensive Scalable Computing (DISC) and High-Performance Computing (HPC) applications frequently need to transfer data between storage resources. In the scientific and industrial fields, the storage component is a key element, because usually those applications employ a huge amount of data. Therefore, the performance of these applications commonly depends on some factors related to time spent in execution of the I/O operations. However, researchers, through their works, are proposing different approaches targeting improvements on the storage layer, thus, reducing the gap between processing and storage. Some solutions combine different hardware technologies to achieve high performance, while others develop solutions on the software layer. This paper aims to present a characterization model for classifying research works on I/O performance improvements for large scale computing facilities. Analysis over 36 different scenarios using a synthetic I/O benchmark demonstrates how the latency parameter behaves when performing different I/O operations using distinct storage technologies and approaches.
Conference: IECON-2020
Qualis: B1
Status: Accepted
Author: - Corresponding author

Title - An Approach to Support the Design and the Dependability Analysis of High Performance I/O Intensive Distributed Systems
Abstract: Frequent service down times and poor system performance can affect aspects such as the availability, quality of experience and generate millions of dollars in lost revenue. High Performance Computing (HPC) environments are often required to comply with extra-functional performance and dependability requirements. The CHESS Toolset provides support for the design and the evaluation of dependability and performance system attributes. In this paper we propose an approach to support the design and the analysis of dependability attributes of HPC environments using CHESS. The approach was employed in the Grid'5000, a highly distributed and I/O intensive HPC environment. The approach was successfully applied and provided key information for demonstrating dependability, deriving new project decisions, agreeing on new design choices and resource allocation strategies.
Conference-Year: 3PGCIC-2020
Qualis: B1
Status: Accepted
Author: - co-author

Title - An Implementation Science Effort in a Heterogeneous Edge Computing Platform to Support a Case Study of a Virtual Scenario Application
Abstract: IoT devices are pillars for the Industry 4.0 software applications. However, compositions of these edge nodes are interesting open challenges in several dimensions, as the integration of diverse hardware and software packages. As an example, different type of industrial cameras and a supercomputer node to support 3D reconstructions is not a trivial approach, especially considering aspects of the IoT software engineering. In this paper, we present a research, which could be classified as an implementation science effort, focusing on a heterogeneous edge computing platform, utilized to support a case study of a real electrical engineering application. This application is characterized by a 3D virtual reconstruction paradigm for the hydro-power project. Our results indicate interesting aspects related to implementation science and challenges found in the composition and operationalization of this heterogeneous edge platform.
Journal: 3PGCIC-2020
Qualis: B1
Status: Accepted
Author: - Co-author

Title - A Survey of I/O Improvements on Storage Device and Systems: A Systematic Mapping of the Literature.
Abstract: The I/O bottleneck remains a central issue in high-performance storage environments. Cloud computing, high-performance computing (HPC) and big data environments share many difficulties in the storage layer to deliver data to applications. In the last years, many researchers have been proposed solutions to improve the I/O architecture through different approaches. Classifying these improvements in a three-dimension perspective brings many benefits. This study presents a systematic literature mapping that classifies I/O performance improvements considering the storage environment layer. The results show which were the devices, software, and storage systems that received attention over the years.
Journal: ACM CSUR-2020
Qualis: A1
Status: Submitted
Author: - Corresponding author

Title - A Three Dimension I/O Performance Characterization Model for Storage Environments
Abstract: Improvements in I/O architectures are becoming increasingly required nowadays. Data-Intensive Scalable Computing (DISC) and High-Performance Computing (HPC) applications transfer a huge quantity of data between the overall environment

using and requiring even higher performance from processing and storage environments. Processing technologies have been developed over the past years quickly than storage devices, thus collaborating for the arising of a well-known I/O bottleneck issue in storage environments. The performance of storage environments as well as the interconnection network is essential to ensure high throughput and satisfy application requests. Given this, researchers are proposing different approaches to reduce the ever-increasing gap between processing and storage combining different solutions on the storage layer. Some solutions combine different hardware technologies, while other develop software solutions to achieve such performance. Many of these software solutions are intended to improve the relationship between applications and hardware technologies targeting I/O rate per second (IOPS) increasing. This paper aims to present a three-dimension characterization model for classifying research works on I/O performance improvements for large scale computing facilities. In order to justify some I/O improvements concerning the I/O layer, we evaluated a subset of these improvements using a real testbed, the Grid5000. Analysis over 36 different scenarios using a synthetic I/O benchmark demonstrates how the throughput parameter behaves when performing different I/O operations using distinct storage technologies and approaches.
Conference: Information Systems - 2020
Qualis: A2
Status: Submitted
Author: - Corresponding author