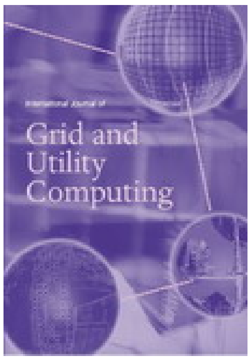


**International Journal of Grid and Utility Computing**

ISSN online: 1741-8488 - ISSN print: 1741-847X
https://www.inderscience.com/ijguc

# An effort to characterise enhancements I/O of storage environments

Laercio Pioli, Victor Ströele, Mario A.R. Dantas

**DOI:** 10.1504/IJGUC.2023.10054824

**Article History:**

| | |
|---|---|
| Received: | 11 November 2020 |
| Accepted: | 14 May 2021 |
| Published online: | 21 March 2023 |

# An effort to characterise enhancements I/O of storage environments

Laercio Pioli*, Victor Ströele and
Mario A.R. Dantas

Department of Computer Science,
Federal University of Juiz de Fora - UFJF,
Juiz de Fora, MG, Brazil
Email: laerciopioli@ice.ufjf.br
Email: victor.stroele@ice.ufjf.br
Email: mario.dantas@ice.ufjf.br
*Corresponding author

**Abstract:** Data management and storage are becoming challenging nowadays due to the huge amount of created, processed, and stored data. The growing gap between power processing and storage latency increases this performance disparity. Targeting reducing I/O bottleneck in storage environments, researchers are proposing interesting improvements in I/O architectures. High-Performance Computing (HPC) and Data-Intensive Scalable Computing (DISC) applications are kinds of such systems that are faced with data challenges due to the need to deal with many parameters when managing data. This study describes our characterisation model for classifying research works on I/O performance improvements for storage systems and devices that improve HPC and DISC overall application performance. We present a set of experiments using a synthetic I/O benchmark performed inside the Grid'5000. We show that the latency when performing I/O operations can undergo many variations if we take into account the presented factors evaluated in the experiments.

**Keywords:** I/O characterisation; I/O performance; I/O improvement; I/O model; HPC; DISC; Big Data; GRID5000; storage system; storage environments.

**Reference** to this paper should be made as follows: Pioli, L., Ströele, V. and Dantas, M.A.R. (2023) 'An effort to characterise enhancements I/O of storage environments', *Int. J. Grid and Utility Computing*, Vol. 14, No. 1, pp.51–61.

**Biographical notes:** Laercio Pioli is currently a PhD student of the computer science course at the Federal University of Santa Catarina (UFSC). He received his master's degree (2020) and bachelor's degree (2017) also in computer science from the Federal University of Juiz de Fora (UFJF). His interesting researches areas are distributed systems, high-performance computing, storage environments, IoT, and software engineering.

Victor Ströele received the BSc degree in computer science from the Federal University of Juiz de Fora, in 2005, and the master's and PhD degrees from the Systems Engineering and Computer Science Program, Federal University of Rio de Janeiro, 2007 and 2012, respectively. He is currently an Associate Professor with the Federal University of Juiz de Fora. He has experience in computer science, with emphases on data mining and complex networks, working mainly on the following themes: e-Learning, recommender systems, clustering algorithms, social network analysis, streaming data, and Informatics in Education.

Mario A.R. Dantas, is a Professor in the Department of Computer Science (DCC) at the Exact Sciences Institute (ICE) at the Federal University of Juiz de Fora (UFJF) and in the Graduate Program in Computer Science (PPGCC), at the Technology Centre (CTC), Federal University of Santa Catarina (UFSC), with a PhD in Computer Science from the University of Southampton (UK), Visiting Professor at the University of Western Ontario (Canada, 2012) and Senior Visiting Researcher in Riken (Japan, 2017–2018). He is the author of hundreds of scientific articles, dozens of chapters in books, and three books. He has advised numerous undergraduate, specialisation, master, and doctorate research works.

*This article is a revised and expanded version of a paper entitled 'Research Characterisation on I/O Improvements of Storage Environments' presented at the '14th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)', Antwerp, Belgium, 7–9 November 2019.*

## 1 Introduction

The ever-increasing data production and consumption have changed how enterprises and academies 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 not only require high power processing but also generate a massive volume of data. Data acquisition, storage, analysis, and visualisation 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 analysis of these large volumes of data is another important step in this scenario. It consumes much power processing from resources to generate understandable information through the visualisation process for post-human analysis.

With this big data usability, a data management problem arises. I/O-related characteristics, indeed, are important for application performance. Researchers are proposing solutions to improve the I/O architecture from different perspectives. Many contributions focus on data distribution over the years. Some of them consider heterogeneous storage systems (Zhou et al., 2016a, 2016b; Xie et al., 2015) others present hash tables algorithms for data mapping (Liu et al., 2014; Wozniak et al., 2010) or even focus on le stripe layout in heterogeneous environments (He et al., 2015a, 2015b). Other works consider hardware combinations to enhance data access. Some of these solutions utilise hash memory, as SSD (Solid-State Disk) together with HDD (Hard Disk Drive) to support the I/O management problems, improving then the performance of these applications. Finally, other dimensions adopt the software improvement in the upper layer to enhance the I/O performance, as illustrated in Gorton and Klein (2014).

The overall overview of the actual efforts could be divided into a macro view of three basic elements (i.e., software, hardware, and storage systems). Our proposal can be understood, as Figure 1 shows, as a characterisation model related to the I/O improvements, studying each component separately and their interconnections. In a previous literature review, we found a gap that indicates a necessary set of experiments to better understand the relation between the three components. To evaluate our proposal, we present a set of experiments performed inside the Grid'5000, a large distributed computational environment, which targeted to indicate aspects related to I/O performance especially considering the view of scientific and industrial applications.

The remainder of this paper is structured as follows: Section 2 presents some elements related to high-performance storage and applications. Related works are illustrated in Section 3. Section 4 highlights the proposed research work of the paper. The used environment and the factors used in the experimentation process are presented in Section 5. Experimental results in the Grid'5000 are presented in Section 6. Finally, Section 7 presents conclusions and future works after this investigation effort.

## 2 High-performance storage and applications

High-performance environments and applications are usually faced with I/O bottleneck issues. A factor that increases this issue is the existing gap between power processing and storage latency. As predicted by Moore's Law in the 1960s, the number of transistors on an integrated circuit would double every eighteen months. However, computing systems are composed not only by processing but also by memory and storage hierarchy that support this processing.

The storage technologies evolve at a slower velocity than processing and this contributes to the increasing of the main I/O performance problem. High-performance applications usually move exabytes of computed or generated data between environmental nodes, specifically computed nodes (CN) and storage nodes (SN), and their performance depends on the I/O process. Typically, a clustered architecture is used to execute these applications. Because of that, I/O performance bottlenecks are a hot area of study, and researchers propose solutions to increase the I/O performance architecture from many perspectives.

### *2.1 Data-intensive scalable computing (DISC)*

Data-Intensive Scalable Computing (DISC) systems usually deal with a massive and expressive amount of data. The necessity to employ a system that processes and organises these data arises nowadays. To acquire, update, share and archive these datasets in an organised way is a challenging task. DISC systems came to fill these requirements that emphasise data management. An inspiration factor to DISC systems creation is the increased growth of the internet infrastructure companies. These leader companies create new methods and technologies to solve and deal with their particular problems. Scalability, fault-tolerance, availability, and cost-performance are some goals that are target to these systems and applications. These applications come from diverse scientific domains and are usually concerned with the role of data management and computation.

Bryant (2011) pointed some key principles related to DISC systems. Some of them are related due to the hardware nature. The first principle relates to the intrinsic data. Instead, associate the data with the users, the system has to collect and maintain these data.

Another characteristic of these systems is that they should implement reliability functionalities such as replication and error correction. The second is directly related to the high-level programming models. Targeting data processing consistently and independently, those systems usually employ high-level programming models considering parallelisation. Interactive access is another considered vital principle of DISC systems. It states that the requirement for computing and storage should be independent and allow variety of set up. Using these provided resources, users can execute these programs interactively, and the system should return an input query quickly while performing computations in the background without losing systems performance. The last principle relates to reliability and availability. Implementing mechanisms that ensure these principles increases the quality of the system. As mentioned before, DISC systems came to fill data requirements and making these last principles more important.

Usually, data-intensive computing facilities are specifically projected to provide higher data performance without losing cost performance. Although DISC infrastructure varies depending on the objective, they presented common hardware aspects already found in Warehouse-Scale Computers (WSC) environments. These environments are usually composed of servers grouped into racks. These agglomerated racks create a cluster. Each server is composed of several processor sockets, each with its microprocessors and cache hierarchy attached to the local RAM chip. The servers are connected to the racks through a 1-gigabit-per-second (Gbps) Ethernet switch and the racks are connected to the cluster-level switch through a 1 or 10 Gbps Ethernet switch. This architecture connects all servers. All of them contain its memory hierarchy, processing core, and storage media locally forming a clustered environment.

### *2.2 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 organised computing resource designed to address problems that require exhaustive processing. These workloads usually refer to simulations and modelling problems commonly found in the scientific and industrial fields that are infeasible to be processed on a unique hardware capability. Their design mainly focuses on providing high processing power for large scale distributed and parallel applications, even though low communication and data access latency is considered increasingly important requirements (Lucas et al., 2014; Chang et al., 2017a).

Usually, this environment-class targets aggregating computing power in a way that provides much higher performance than a single computational machine. One observed characteristic in these environments is the separation of computing and storage resources, which results in massive data movements through layers of the I/O path during the execution of a data-intensive application. In such complex computing environments, many factors can affect the I/O performance perceived by an application: workload characteristics, system architecture and topology model, configurations in the many layers of the parallel I/O software stack, properties of hardware components, interference and system noises, to name a few.

Research works that address I/O performance variability proposed inter-application coordination approaches focusing on reducing the impact of multiple applications simultaneously executing on the HPC system (Dorier et al., 2012; Kuo et al., 2014; Dorier et al., 2014; Yildiz et al., 2016; Wan et al., 2017). However, on extreme-scale computational science applications, denoting applications with dedicated access to all resources of the HPC environment for execution, I/O performance variability can be mostly attributed to intra-application interference. One major, yet intrinsic, source of intra-application interference in this context relates to the load balance on PFS's data servers.

## 3 Related work

Conducting performance studies on storage devices and systems is challenging for researchers as they need to deal with various low-level concepts and new technologies. Applications that produce and process many data in a short time interval almost always need to store and retrieve data and usually they encounter latency problems to perform such operations. Much of these problems are related to the device and technology that are being used. The related works shown below exemplify some issues that are being addressed by researchers to improve I/O performance.

Saif et al. (2018) presented an I/O tracer, called IO scope, 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, altering-based profiling over fine-grained criteria inside 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).

Calzarossa et al. (2016) also presented a survey considering a characterisation model but differently from the previous one, they consider the importance of the workload characterisation exploiting its importance in popular applications domains. Their focus is directed to workload from the web and also with workloads associated with online social networks, video services, mobile apps, and cloud computing infrastructure. They also present and analyse a modelling technique applied for this characterisation. Their proposed characterisation model does not consider the three basic elements (e.g. software, hardware, and storage systems) as in our presented model. They present studies in a cloud computing infrastructure, but their concerns are also related to the characteristics of cloud workloads.

Finally, Traeger et al. (2008) described and presented a survey considering a nine-year examination of a range of file system and storage benchmarks. They survey 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 they can influence the performance in an evaluation process of a storage system.

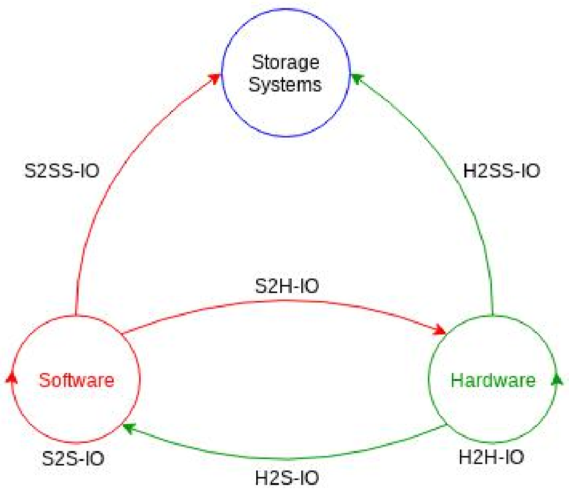
## 4 Characterisation model

A characterisation model for classifying research works on I/O performance is presented in this section. Several researchers are proposing solutions to overcome the existential gap between power processing and storage latency. They combine different hardware devices and software applications in the most varied way possible. Sometimes these new approaches require the development of new software drivers to interface the proposed architecture. Figure 1 presents the characterisation model that was given in Pioli et al. (2019). It is composed of three basics elements: software, hardware, and storage systems.

Software investigations could be understood as an improvement where the object that is being proposed as a solution is an algorithm, method, framework, or any programmable solution. Hardware investigations could be characterised as improvements when the object that is proposed as an improvement is a physical component or something palpable. We found that these two groups can relate and improve each other or a storage system targeting the improvement of I/O performance. With this approach, we argue that if researchers classify their papers before publishing, it could improve and save time by the other researchers when trying to improve objects in this field.

Previous works exemplify some issues that are being addressed by researchers to improve I/O performance on storage devices and systems. These works were selected because it is being realised a systematic review concerning I/O improvements on storage devices and systems that were proposed in the last ten years. There is no additional intention in the presentation of the works exposed besides showing that they have purposes in improving a similar class of objects. These solutions were divided into numbered groups for better viewing.

**Figure 1** Proposed characterisation I/O improvements



### 4.1 *Group 1 – software solution to improve I/O performance on hardware (S2H-IO)*

All the contributions presented in group 1 are some kind of 'software' solution and could be characterised as a 'solution' that is being proposed to improve I/O performance on a storage device. This group of improvements was characterised and shown in Figure 1 as the arrow that leaves the red circle (Software) and arrives in the green circle (Hardware). For better viewing, the acronym S2H-IO which means 'Software solution to improve I/O performance on Hardware' was created and added above the arrow.

Below we expose some researches that are part of group 1. Chang et al. (2017b) proposed an approach to operate wear levelling on virtual erase counts instead of real erase counts using SSD devices. Kim et al. (2016) proposed an I/O architecture that optimises the I/O path to take full advantage of NVMe SSDs. The authors' approach works by eliminating the overhead of user-level threads, bypassing unnecessary I/O routines, and enhancing the interrupt delivery delay. Ramasamy and Karantharaj (2015) proposed an algorithm called random first flash enlargement to improve the performance of write operation on the flash-memory-based SSDs. Shen and Park (2013) proposed an I/O scheduler where the design of the solution is motivated by unique characteristics on Flash-Based SSDs. Yang et al. (2019) proposed (WARCIP) which means 'write amplification reduction by clustering I/O pages' to minimise the negative impact of garbage collection (GC) on SSD devices. They used a clustering algorithm to minimise the rewrite interval variance of pages in a flash block.

### 4.2 *Group 2 – hardware solution to improve I/O performance on hardware (H2H-IO)*

Papers presented in group 2 relate to improvements targeting I/O performance on the hardware device. The authors do that using hardware technology as an object to perform this improvement. In all of these cases, the solution involves different hardware technology as a solution. This group of improvements was characterised and shown in Figure 1 as the arrow that is circling the hardware circle. The acronym H2H-IO which means 'Hardware solution to improve I/O performance on Hardware' was created and added below the green circle (Hardware).

Below we expose some researches that are part of group 2. Lee et al. (2014) proposed a Stacked DRAM with a microbump interface. They built a High-Bandwidth Memory (HBM) introducing four DRAM memories into a chip-on-wafer. 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 128 Gb NAND flash. Lee et al. (2017) proposed the design of FESSD that uses an on-chip access control memory (ACM) introducing any nature of on-chip non-volatile memory (NVM) into the micro-controller of an SSD.

### 4.3 *Group 3 – software solution to improve I/O performance on software (S2S-IO)*

All contributions presented in group 3 are some kind of 'software' approach which could be characterised into a 'solution' that is being proposed to improve I/O performance on another 'software' object differently from group 1. It is important to notice that although the improvements are from software to software, they take into account the storage technologies that they are using. This group of improvements receives the acronym S2S-IO which means 'Software solution to improve I/O performance on Software'. It was created and added below the red circle (Software) in Figure 1.

Below we expose some researches that are part of group 3. Yang et al. (2017) proposed a content look-aside buffer (CLB) for simultaneously providing redundancy-free virtual disk I/O and caching. They implemented a CLB on the KVM hypervisor and demonstrate that CLB delivers considerably improved I/O performance with realistic workloads. Huo et al. (2015) proposed a caching management algorithm, sometimes called a framework named ACSH, which is based on SSD devices and DRAM and is focused on the improvement of metadata I/O on the le systems. Ou et al. (2015) proposed a le index scheme for a flash file system called NIS. In this scheme, they are concerned about the performance of le systems when using NAND ash as a storage device. Wu et al. (2018) proposed a priority-based data placement method for databases using SSDs. They consider a mechanism and a migration rule for performing migrations between HDDs and SSDs.

## 4.4 *Group 4 – hardware solution to improve I/O performance on software (H2S-IO)*

Papers presented in group 4 relate to improvements targeting I/O performance into the software. Different from group 3, in this group, they consider some 'hardware' objects as a 'solution' to perform its improvement. Although this approach is less common than the others, it was classified and characterised as presented below. This group of improvements receives the acronym H2S-IO which means 'Hardware solution to improve I/O performance on Software'. It was created and added below the arrow that leaves the green circle (Hardware) in Figure 1.

Below we expose some researches that are part of group 4. Min et al. (2015) proposed a method using NVMe SSDs to enhance I/O resource management of Linux Cgroup on NUMA systems. Ouyang et al. (2010) proposed a method which is an aggregation staging I/O to enhance checkpoint writing performance using staging I/O and SSD on the data server archiving better write bandwidth. Kannan et al. (2011) proposed a mechanism using an Active NVRAM-based approach for I/O staging. In the considered method, each physical node has an additional active NVRAM component to stage I/O. Bhattacharjee et al. (2011) proposed utilising SSD to enhance recovery and restart through random access capability in a database engine. Nakashima et al. (2018) proposed a method for improving the I/O performance of a big data application using SSD as cache. The results presented by the authors demonstrate that the method can improve I/O performance.

Group 5 and 6 are concerned about the I/O improvements targeting storage systems. It is important to notice that in these characterisations "Storage Systems" does not mean some software that is present in a storage device but rather a group of technologies and software working together and asynchronously in an environment. Because the Storage Systems are composed of both hardware and software, the improvements proposed by researchers can be either software or hardware improvements or both of them.

## 4.5 *Group 5 – software solution to improve I/O performance on storage systems (S2SS-IO)*

Papers presented in group 5 relate to improvements targeting I/O performance into a 'storage system'. As well as groups 1 and 3 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, a Storage Systems. This fact leads us to relate group 5 with the arrow that leaves the red circle (Software) and arrives in the blue circle (Storage Systems) presented in Figure 1. The acronym S2SS-IO which means 'Software solution to improve I/O performance on Storage Systems' was created and added above the related arrow.

Below we expose some researches that are part of group 5. Zhou et al. (2018) proposed an algorithm that can make better use of heterogeneous devices for storage systems and is based on consistent hashing. Du et al. (2015) proposed a balanced partial stripe (BPS) write scheme to improve the write performance of RAID-6 systems. Oh et al. (2012) proposed a schema that organises the cache space into reading and writes and manages these spaces according to the workload characteristics for improving the performance of hybrid storage solutions.

## 4.6 *Group 6 – hardware solution to improve I/O performance on storage systems (H2SS-IO)*

In group 6 the papers also concern about I/O improvements targeting storage systems, but unlike group 5, it considers some 'hardware' objects as a 'solution' to perform the improvement into a storage system. This fact leads us to relate group 6 with the arrow that leaves the green circle (Hardware) and arrives in the blue circle (Storage Systems). The acronym H2SS-IO which means 'Hardware solution to improve I/O performance Storage Systems' was created and added above the related arrow presented in Figure 1.

Below we expose some researches that are part of group 6. 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. (2012) introduce a Hybrid SSD approach that combines DRAM, phase changed memory (PCM), and flash memory into a hierarchical storage system. Dae-Sik and Seung-Kook (2009) designed and analysed a high-end class DRAM-based SSD storage using DDR-1 memory and PCI-e interface.

## 5 Empirical scenario and factors definition

The experimental environment and the used factors to carry out the experiments are presented in this section. We also relate some of these factors with the elements presented in Figure 1.

### 5.1 Empirical scenario

Grid'5000 which is a large-scale testbed was considered in our experimentation process. It focuses on parallel and distributed computing including cloud computing, HPC, and big data. Each cluster provides a huge amount of technologies including different CPU processors, GPUs, storage devices such as SSD, HDD, NVMe, and Ethernet, Infiniband, and Omni-Path network interconnectors. The Grid'5000 testbed is a secure and powerful environment composed of 8 different sites located in France providing a huge amount of devices and technologies working in parallel to solve huge problems of science.

In our experimental process, we used 24 nodes of the dahu cluster located in Grenoble. The nodes are composed of a Dell PowerEdge C6420 model interconnected with a Gigabit Ethernet network. Each node has 2 CPUs Intel Xeon Gold 6130 2.10GHz with 16 cores/CPU. The storage of each of them has 240GB SSD SATA, 480 GB SSD SATA, and 4.0 TB HDD SATA, and the memory RAM 192 GiB. Centos7 was used as an Operational System, kernel 3.10.0-957.21.2.el7.x86 64 and ext4 le system. In this experiment, 16 nodes were used as Compute Nodes (CN) and 8 as Storage Nodes (SN).

The used file system was the OrangeFS (version 2.9.7). The software used to perform the experimentation was the IOR-EXTENDED (IORE) benchmark (Inacio and Dantas, 2018). IORE benchmark is a flexible and distributed I/O performance evaluation tool that was designed for hyperscale storage systems. It supports a whole experimental variety of workloads. The requests were generated running IORE on a CN with MPICH 3.0.4 version.

### 5.2 Factors definition

This subsection presents the factors that we used in the analysis. Below we explain and correlate some of these factors to the model presented in Figure 1.

#### 5.2.1 Storage device

Providing incentives for researchers to continue proposing hardware solutions that improve IO performance, this factor relates to the hardware storage devices used in the experiment with the green circle Hardware in Figure 1. The right usability of the storage can improve the I/O performance of applications that needs to execute I/O operations frequently. To verify that the technology device usually can influence the performance, this research takes care of three common approaches to store data. The first one is to store all kinds of data, and metadata, on an HDD device. The second is to use SSD to store metadata while the data are stored on the HDD devices. Finally, we used an SSD device to store data and metadata.

#### 5.2.2 Linux I/O schedulers

Three Linux I/O schedulers were considered in this experiment. Complete Fairness Queuing (CFQ) (Axboe, 2004), deadline, and noop are these schedulers that were used as factors to the experiment complementing the scenario to store data shown above. 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 to the schedulers, with are software line codes, with the red circle (software) presented in Figure 1. Many contributions are performed by researchers to improve schedulers because I/O schedulers play an essential role in those environments.

#### 5.2.3 Task numbers

The number of tasks that we used on the experimentation is an important factor that we considered. In this experiment, we consider 32 and 64 as the number tasks because we believe that different workloads and size numbers can be easily found in DISC applications and it probably influences the performance results.

#### 5.2.4 Access pattern

The last experimental parameter is related to the access pattern. The workload access patterns used in this experimentation were 'random' and 'sequential' for each scenario. We performed the experimentation with these two options because it could give us different information and present broader results. The data access pattern is among the most important characteristics of disk drive workloads because it is related to the disk service process (Riska and Riedel, 2006). This factor could be related to the arrow that leaves the red circle (software) and arrives on the green circle (hardware) presented in Figure 1 because some software improvement could be proposed to improve the data management workload access on the storage devices.

In conclusion, we performed the experimentation with a total of 36 different scenarios where 3 came from the different approaches to store data and metadata, 3 came from the different I/O schedulers used on the requests, 2 came from the number of tasks used and 2 came from the data access pattern used by the benchmark. In order 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 Evaluation results

Through this section, we present the 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.

The selected schedulers defined in section 5 were set to the respective device that was being used to store the data and metadata in each running execution. Indeed there are many ways of analysing the presented results due to the huge number of factors defined in section 5, but we are concerned with analysing through two different perspectives. The results were classified into two different groups with different analysis purposes.

The first group is concerned to present results in a view of the storage approach, which is in some way a hardware approach, to store data. Presenting results from this perspective we believe that researchers could keep increasing and proposing hardware devices to increase the I/O layer.

The second group presents results considering the schedulers presented on Linux operational system. Presenting results from this perspective, the results could lead researchers to keep increasing the schedulers presented on Linux that could be related as a software improvement solution in Figure 1.

We provide a brief discussion of how the data and information were disposed of in the graphics. In each figure presented, 24 outcomes for the read and write operations are shown. 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.

## 6.1 *Scheduler overview*

Figures 2, 3, and 4 present the average latency time for the read and write operations switching the three presented schedulers in section 5 (CFQ, deadline, and noop).

Figure 2 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 file, a 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 the CFQ scheduler, it is also possible to notice in Figure 2 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 storing data on HDD and metadata on SSD. Analysing 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 the only SSD compared with the hybrid approach when the access pattern equal to sequential and the number of tasks equal to 64.

**Figure 2** Latency analysis using CFQ scheduler

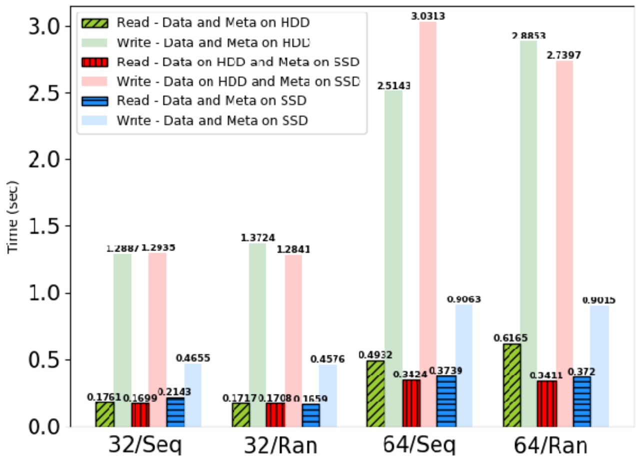


Table 1 presents the average latency value for all the scenarios presented in Figure 2 when using the deadline scheduler. It is possible to see that storing data on HDD and metadata on SSD was the configuration storage that gave us the lowest read latency time. To the write operation, the best configuration storage was storing both data on the SSD device with almost three times less than the lowest configuration which uses HDD.

**Table 1** Average of latency presented in Figure 2

| *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 3 presents the results using the deadline Linux scheduler. The presented results were similar to the results presented by the CFQ scheduler. However, comparing Figure 2 with Figure 3 we verify that when using a 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 deadline scheduler, almost in all write results, storing data and metadata in an HDD was the worst approach providing the higher latency time.

Table 2 presents the average latency value for all the scenarios presented in Figure 3 when using the deadline scheduler. Just like Table 1, Table 2 also presented the approach of storing data on HDD and metadata on 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.

**Figure 3** Latency analysis using deadline scheduler

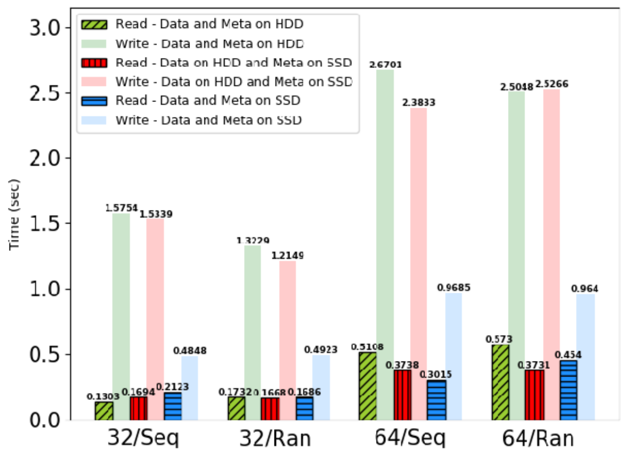


**Table 2** Average of latency presented in Figure 3

| *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 4 presents the latency results when using the Linux noop scheduler. It is possible to verify in this figure that the write operation latency when using only HDD as a device to store both data and metadata was the worst storage approach. An important fact that we noticed is that Figure 4 follows the same latency pattern presented by Figure 2 for the read operation in all cases. For instance, in Figure 2 when the storage configuration is access pattern equal to sequential and the number of tasks is 32, the storage approach which presented the worst latency for reading operation was when storing both data and metadata on SSD.

**Figure 4** Latency analysis using noop scheduler

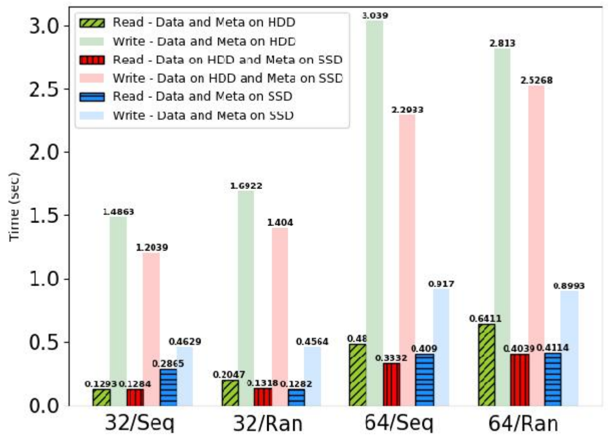


Table 3 presents the average latency value for all the scenarios presented in Figure 4 when using the noop scheduler. As Table 1 and Table 2, Table 3 also presented the approach of storing data on HDD and metadata on SSD as the best results with a smaller latency time to read operation and the approach which stores both data and metadata on SSD as the best approach to decrease the latency time for the write operation.

**Table 3** Average of latency presented in Figure 4

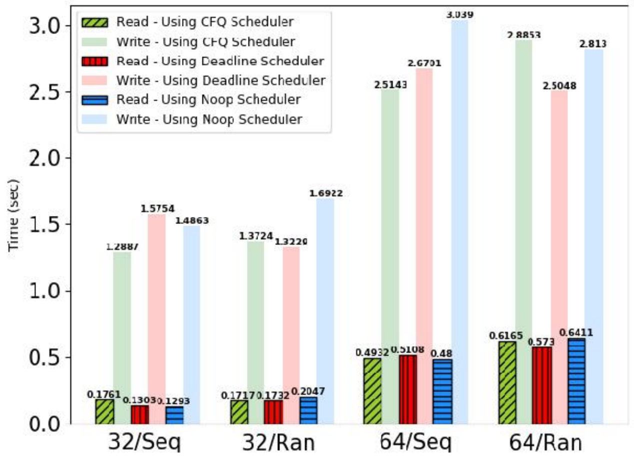
| *Storage approach* | *Read* | *Write* |
|---|---|---|
| Data and Meta on HDD | 0.3638 | 2.2576 |
| Data on HDD and Meta on SSD | 0.2493 | 1.8570 |
| Data and Meta on SSD | 0.3088 | 0.6839 |

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.

## 6.2 *Storage overview*

In Figures 5, 6, and 7, we present the average latency time for reading and writing operations switching the three storage approaches presented in section 5 (Storing Both Data and Metadata on HDD, Storing Data on HDD and Metadata on SSD and Storing Both Data and Metadata on SSD.)

**Figure 5** Data and metadata on HDD



The latency results when storing both data and metadata on the HDD device switching then the I/O schedulers are presented in Figure 5. Analysing it we verify that when the 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 similar to sequential, the scheduler which presented the lowest latency for the reading operation was the noop scheduler, and for the write operation was the CFQ scheduler. Analysing 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 write operation was the deadline scheduler.

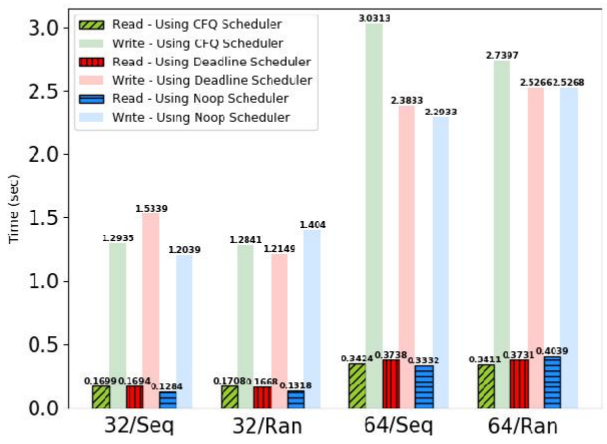
In Table 4 we present the average latency time for all the scenarios presented in Figure 5. We present this average because DISC and HPC applications can treat and use heterogeneous kinds of data with different access patterns and the number of tasks on the same application. Considering this, it's possible to see in Table 4 that when we are storing both data and metadata on the HDD device the scheduler that presents the big latency time to execute read operation is the CFQ, and to perform the write operation is the Noop scheduler.

**Table 4** Average of latency presented in Figure 5

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

The results when storing data on HDD and metadata on SSD device switching then the I/O schedulers are presented in Figure 6. Just as it happened in Figure 5, in Figure 6 all latency values to perform the write operation are greater than the value to perform the read operation. However, we verify that the latency rate when storing by this way was lower in almost all read operation cases compared to the values presented in Figure 5. The unique configuration where the read latency was bigger than the presented in Figure 5 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 operations has decreased compared with the first storage approach. The cases where it has not happened were the twice for the CFQ scheduler with the sequential access pattern and the deadline scheduler with the 64 number of tasks equal and random access pattern. All the other write values have the latency time decreased when changing the way to store data and metadata.

**Figure 6** Data on HDD and metadata on SSD



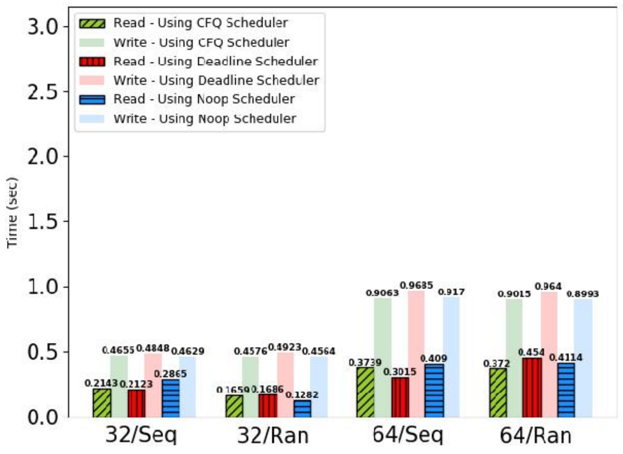
We noticed that in Table 5 the latency average to execute the read operation for all schedulers are smaller than when storing data on HDD presented in Table 4. It means that for all schedulers, all corresponding latency times for the read operation were decreased when the use of an SSD was applied. However, for the write operation, unlike the values presented in Table 4, Table 5 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 compared with the previous storage approach. It is also possible to notice that when we are storing data on HDD and metadata on SSD the scheduler that presents the lowest average latency time to perform read and write operation is the noop scheduler.

**Table 5** Average of latency presented in Figure 6

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

The results when storing both data and metadata on SSD that is a device that does not have mechanical components are presented in Figure 7.

**Figure 7** Data and metadata on SSD



Comparing the latency time for the read operation with the values presented in Figure 6, we noticed 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 in Figure 5 we verify that when using only an SSD device to store data and metadata, all schedulers presented a higher latency time. These same pattern results we verify if compared to Figure 6. By these interesting results, we 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 6, presented better results from six if compared to Figure 5.

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 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 was more expressive if compared to the previous results.

It is possible to notice in Table 6, that when performing the read operation, the latency ratio was higher for all schedulers if compared to Table 5. It enforces our analysis that this storage approach did not present significant improvement as the previous results when performing read operations. However, when compared to Table 4, it presented a smaller and better overall latency time. Analysing the write value presented in Table 6 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 6** Average of latency presented in Figure 7

| *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 is 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.

## 7 Conclusion

The huge amount of created data together with the growing gap between power processing and storage latency imposes a challenge in terms of data transfer and storage. Researchers are proposing interesting and important solutions to reduce and approximate the power processing to the memory and storage development. HPC and DISC applications are kinds of such systems that are faced with data challenges due to the need to deal with many parameters when managing data. This study described our characterisation model for classifying research works on I/O performance improvements for storage systems and devices. We consider that they are divided in a macro view of the software, hardware, and storage systems approaches. Our proposal can be understood as a characterisation related to the I/O improvements, where we are studying each component separately (i.e. software, hardware, and storage systems) and their interconnections. We present a set of experiments performed inside the Grid’5000 and we demonstrated that the latency when performing I/O operations can undergo many variations if we take into account the presented factors evaluated in the experiments. Among many presented results achieved by the 36 different scenarios relating to the latency when performing I/O operations we classify and show the results separated through two different perspectives called scheduler overview and storage overview. In the first evaluated perspective, we discuss how the performance of each Linux scheduler behaved when changing the perspective of storing data and metadata on different devices. In the second evaluation, we discuss how each storage approach behaved when switching the Linux schedulers. In order to improve this research, we intend to perform experiments in a large cloud environment considering different storage technologies and software approaches to also validate it using the cloud concepts.

## References

Axboe, J. (2004) ‘Linux block IO - present and future’, *Ottawa Linux Symp*, pp.51–61.

Bhattacharjee, B., Ross, K.A., Lang, C., Mihaila, G.A. and Banikazemi, M. (2011) ‘Enhancing recovery using an SSD buffer pool extension’, *Proceedings of the 7th International Workshop on Data Management on New Hardware*, pp.10–16.

Bryant, R.E. (2011) ‘Data-intensive scalable computing for scientific applications’, *Computing in Science & Engineering*, Vol. 13, No. 6, pp.25–33.

Bu, K., Wang, M., Nie, H., Huang, W. and Li, B. (2012) ‘The optimization of the hierarchical storage system based on the hybrid SSD technology’, *Second International Conference on Intelligent System Design and Engineering Application*, pp.1323–1326.

Calzarossa, M.C., Massari, L. and Tessera, D. (2016) ‘Workload characterization: a survey revisited’, *ACM Computing Surveys (CSUR)*, Vol. 48, No. 3, pp.1–43.

Chang, C., Greenwald, M., Riley, K. and others. (2017a) *Fusion Energy Sciences Exascale Requirements Review*, An Office of Science review sponsored jointly by Advanced Scientific Computing Research and Fusion Energy Sciences in USDOE Office of Science (SC).

Chang, L., Huang, S. and Chou, K. (2017b) ‘Relieving self-healing SSDs of heal storms’, *10th ACM International Systems and Storage Conference*, p.5.

Dae-Sik, K. and Seung-Kook, C. (2009) ‘A design of DDR-1 solid-state drive using PCI-e interface’, *15th Asia-Pacific Conference on Communications*, pp.889–891.

Dorier, M., Antoniu, G., Cappello, F., Snir, M. and Orf, L. (2012) ‘Damaris: How to efficiently leverage multicore parallelism to achieve scalable, jitter-free I/O’, *IEEE International Conference on Cluster Computing*, pp.155–163.

Dorier, M., Antoniu, G., Ross, R., Kimpe, D. and Ibrahim, S. (2014) ‘CALCioM: mitigating I/O interference in HPC systems through cross-application coordination’, *IEEE 28th Int. Parallel and Distributed Processing Symposium*, pp.155–164.

Du, C., Wu, C., Li, J., Guo, M. and He, X. (2015) ‘Bps: a balanced partial stripe write scheme to improve the write performance of raid-6’, *IEEE International Conference on Cluster Computing*, pp.204–213.

Gorton, I. and Klein, J. (2014) ‘Distribution, data, deployment: software architecture convergence in big data systems’, *IEEE Software*, Vol. 32, No. 3, pp.78–85.

He, S., Sun, X.H. and Haider, A. (2015) 'HAS: heterogeneity-aware selective data layout scheme for parallel systems on hybrid servers', *IEEE International Parallel and Distributed Processing Symposium*, pp.613–622.

He, S., Sun, X.H., Wang, Y., Kougkas, A. and Haider, A. (2015) 'A heterogeneity-aware region-level data layout for hybrid parallel systems', *44th International Conference on Parallel Processing*, pp.340–349.

Huo, Z. et al. (2015) 'A metadata cooperative caching architecture based on SSD and DRAM for le systems', *International Conference on Algorithms and Architectures for Parallel Processing*, pp.31–51.

Inacio, E.C. and Dantas, M.A.R. (2018) 'IORE: a flexible and distributed I/O performance evaluation tool for hyperscale storage systems', *Symposium on Computers and Communications (ISCC)*, pp.1026–1031.

Kannan, S., Gavrilovska, A., Schwan, K., Milojicic, D. and Talwar, V. (2011) 'Using active NVRAM for I/O staging', *Proceedings of the 2nd international workshop on Petascal data analytics: challenges and opportunities*, pp.15–22.

Kim, H.J., Lim, J.D., Lee, J.W., Na, D.H., Shin, J.H., Kim, C.H. et al. (2015) '7.6 1GB/s 2Tb NAND ash multi-chip package with frequency-boosting interface chip', *IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, pp.1–3.

Kim, J., Ahn, S., La, K. and Chang, W. (2016) 'Improving I/O performance of NVMe SSD on virtual machines', *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp.1852–1857.

Kuo, C., Shah, A., Nomura, A., Matsuoka, S. and Wolf, F. (2014) 'How le access patterns influence interference among cluster applications', *International Conference on Cluster Computing (CLUSTER)*, pp.185–193.

Lee, D.U., Kim, K.W., Kim, K.W., Lee, K.S., Byeon, S.J., Kim, J.H. et al. (2014) '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*, Vol. 50, pp.191–203.

Lee, J., Ganesh, K., Lee, H.J. and Kim, Y. (2017) 'FESSD: a fast encrypted ssd employing on-chip access-control memory', *IEEE Computer Architecture Letters*, Vol. 16, No. 2, pp.115–118.

Liu, J., Byna, S., Dong, B., Wu, K. and Chen, Y. (2014) 'Model-driven data layout selection for improving read performance', *IEEE International Parallel & Distributed Processing Symposium Workshops*, pp.1708–1716.

Lucas, R., Ang, J., Bergman, K., Borkar, S., Carlson, W., Carrington, L. et al. (2014) 'Top ten exascale research challenges', *DOE ASCAC subcommittee report*, pp.1–86.

Min, J., Ahn, S., La, K., Chang, W. and Kim, J. (2015) 'Cgroup++: enhancing I/O resource management of Linux Cgroup on NUMA systems with NVMe SSDs', *Proceedings of the Posters and Demos Session of the 16th International Middleware Conference*, p.7.

Nakashima, K., Kon, J. and Yamaguchi, S. (2018) 'I/O performance improvement of secure big data analyses with application support on SSD cache', *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, p.90.

Oh, Y., Choi, J., Lee, D. and Noh, S.H. (2012) 'Caching less for better performance: balancing cache size and update cost of ash memory cache in hybrid storage systems', *FAST*, Vol. 12.

Ou, Y., Wu, X., Xiao, N., Liu, F. and Chen, W. (2015) 'NIS: a new index scheme for flash file system', *29th Symposium on Mass Storage Systems and Technologies (MSST)*, pp.44–51.

Ouyang, X., Marcarelli, S. and Panda, D.K. (2010) 'Enhancing checkpoint performance with staging IO and SSD', *International Workshop on Storage Network Architecture and Parallel I/Os*, pp.13–20.

Pioli, L., Stroele, A.M. and Dantas, M.A.R. (2019) 'Research characterization on I/O improvements of storage environments', *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp.287–298.

Ramasamy, A.S. and Karantharaj, P. (2015) 'A buffer cache management algorithm for ash-memory-based SSD to improve write performance', *Canadian Journal of Electrical and Computer Engineering*, Vol. 38, pp.219–231.

Riska, A. and Riedel, E. (2006) 'Disk drive level workload characterization', *USENIX Annual Technical Conference*, Vol. 2006, pp.97–102.

Saif, A., Nussbaum, L. and Song, Y. (2018) 'IOscope: a flexible I/O tracer for workloads, I/O pattern characterization', *International Conference on High Performance Computing*, pp.103–116.

Shen, K. and Park, S. (2013) 'Flashfq: a fair queueing i/o scheduler for ash-based ssds', paper presented as part of the *2013 USENIX Annual Technical Conference USENIX ATC 13*, pp.67–78.

Traeger, A., Zadok, E., Joukov, N. and Wright, C.P., (2008) 'A nine year study of le system and storage benchmarking', *ACM Transactions on Storage (TOS)*, Vol. 4, No. 2, pp.1–56.

Wan, L., Wolf, M., Wang, F., Choi, J.Y., Ostrouchov, G. and Klasky, S. (2017) 'Comprehensive measurement and analysis of the user-perceived I/O performance in a production leadership-class storage system', *International Conference on Distributed Computing Systems (ICDCS)*, pp.1022–1031.

Wozniak, J., Jacobs, B., Latham, R., Lang, S., Son, S.W. and Ross, R. (2010) 'C-mpi: a dht implementation for grid and hpc environments', Preprint ANL/MCS-P1746-0410, Vol. 4.

Wu, C., Huang, C. and Chang, C. (2018) 'A priority-based data placement method for databases using solid-state drives', *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*, pp.175–182.

Xie, W., Zhou, J., Reyes, M., Noble, J. and Chen, Y. (2015) 'Two-mode data distribution scheme for heterogeneous storage in data centers', *IEEE International Conference on Big Data (Big Data)*, pp.327–332.

Yang, C., Liu, X. and Cheng, X. (2017) 'Content look-aside buffer for redundancy-free virtual disk I/O and caching', *ACM SIGPLAN Notices*, Vol. 52, No. 7, pp.214–227.

Yang, J., Pei, S. and Yang, Q. (2019) 'WARCIP: write amplification reduction by clustering I/O pages', *12th ACM International Conference on Systems and Storage*, pp.155–166.

Yildiz, O., Dorier, M., Ibrahim, S., Ross, R. and Antoniu, G. (2016) 'On the root causes of cross-application I/O interference in HPC storage systems', *International Parallel and Distributed Processing Symposium (IPDPS)*, pp.750–759.

Zhou, J., Chen, Y. and Wang, W. (2018) 'Attributed consistent hashing for heterogeneous storage systems', *PACT*, pp.1–12.

Zhou, J., Xie, W., Gu, Q. and Chen, Y. (2016) 'Hierarchical consistent hashing for heterogeneous object-based storage', *IEEE Trustcom/BigDataSE/ISPA*, pp.1597–1604.

Zhou, J., Xie, W., Noble, J., Echo, K. and Chen, Y. (2016) 'SUORA: a scalable and uniform data distribution algorithm for heterogeneous storage systems', *IEEE International Conference on Networking, Architecture and Storage (NAS)*, pp.1–10.