Comparing Ext4 and ZFS for Onboard Data Processing: A Systematic Mapping and Experimental Evaluation

Stephanie Liza Johansson
Div. of Intelligent Future Technologies
Mälardalen University
Västerås, Sweden
ljn18009@student.mdu.se

Hassan Omer Said
Div. of Intelligent Future Technologies
Mälardalen University
Västerås, Sweden
hsd18001@student.mdu.se

Håkan Forsberg
Div. of Intelligent Future Technologies
Mälardalen University
Västerås, Sweden
hakan.forsberg@mdu.se

Nandinbaatar Tsog
Div. of Intelligent Future Technologies
Mälardalen University
Västerås, Sweden
nandinbaatar.tsog@mdu.se

Oskar Flordal
Chief Technology Officer
Unibap AB
Uppsala, Sweden
oskar.flordal@unibap.com

***Abstract*—Selecting the correct file system is critical for space applications where risks are present. This study systematically maps and tests Ext4 versus ZFS for onboard data processing on the *iX10-100* and *iX5-100* payload processors. The test sets are presented along with results on several performance metrics. The conclusion is that both ZFS and Ext4 are useful, but based on certain considerations of onboard data processing, Ext4 is better than the other.**

***Index Terms*—File System Performance, Ext4, ZFS, Systematic Mapping Study, Benchmarking, Onboard Data Processing**

I. Introduction

Onboard processing of data in satellites requires data to be processed and stored correctly to achieve mission objectives. This is true for space-cloud payload processors, where the objective is to handle the onboard processing of a satellite's payload data through a cloud network. To support a space mission, it is critical that the underlying mechanisms, such as the type of operating system and file system used in a satellite, are carefully chosen. In fact, according to Karim et al. [1] the topic of a file system is always neglected in storage unless significant demands are present. Selecting the correct file system is necessary since it is responsible for handling data smoothly and without errors in a storage device such as Solid State Drive (SSD) and Hard Disk Drive (HDD). However, selecting the correct file system for space application is not an easy task due to the harsh space environment.

Since 2006, many research articles have been published in which several disadvantages and advantages of commonly used file systems are discussed. Some of the published research articles that compare the performance between two or more file systems include Karim et al. [1], Heger [2], and Promchana et al. [3]. To the best of our knowledge, there is currently no state-of-the-art that systematically maps the differences between the Extended 4 (Ext4) file system and the Zettabyte File System (ZFS) in various feature categories other than performance and reliability. Due to the lack of such a study, commercial space firms can find it harder to choose the best option for their clients when deciding between Ext4 and ZFS. This offered an incentive for undertaking a thorough comparison study and an experimental evaluation of Ext4 and ZFS. Thus, the objective of this research contains two aspects:

1) to present a systematic mapping of the state-of-the-art knowledge of Ext4 versus ZFS for onboard data processing by studying relevant literature from 2006 and onwards
2) to test Ext4 and ZFS in a next-generation space-cloud payload processor called *iX10-100* [4] and using *iX5-100* [5] as the space heritage reference by conducting the same tests on it as well. Both payload processors are from the Swedish space company Unibap AB [6].

The former aspect offers a logical approach to a theoretical study, whereas the latter aspect offers an experimental evaluation. Furthermore, the aforementioned tests contribute to research that does not entail such a test environment.

This paper is organised in the following manner: Section II will give an overview of the background including related work. Section III will describe the process applied to conduct the systematic mapping. In section IV, the gathered results of the systematic comparison study are illustrated and described. Section V describes the setup for conducting the experiments. This is followed by an illustration of the gathered experiment results in section VI. Finally, a conclusion is drawn based on the presented results in section VII.[1]

II. Background

As mentioned, an efficient and secure way to access and manipulate the data stored on computer storage devices is through the use of a file system. File systems organise and

[1]**This work is supported by the Swedish Knowledge Foundation within the project Dependable Platforms for Autonomous systems and Control (DPAC) and Vinnova within the project SafeDeep.**

manage data on computer storage devices to offer the desired data reliability, scalability, performance, storage capacity, and more. This is important for a space mission operation; storage in space is impacted negatively due to radiation, limited storage capacity, high power consumption, limited bandwidth, and poor robustness. Therefore, it is crucial to consider these factors when designing or selecting a file system for space applications. Nowadays, Ext4 and ZFS are some of the popular file systems that are commonly used by users (e.g., data storage companies and space community). ZFS is designed to provide redundancy and data integrity, i.e., ZFS supports not losing data when a drive fails. It also ensures data integrity in the event of system failures [7]. On the other hand, the Ext4 is designed to handle large storage capacities and offer high performance unlike its predecessors (e.g., Ext2 and Ext3) [8].

Section I mentioned some of the existing research papers which are relevant to this study. Karim et al. [1] compares the I/O performance, flexibility, and ease of use of Linux file systems Ext4, Extents File System (XFS), and B-tree File System (BtrFS). These file systems are running on storage stack systems such as Logical Volume Manager (LVM) and ZFS with Rados Block Devices (RBDs), replacing physical drives. The experiments are conducted to evaluate the performance of selected file systems. In addition, the paper highlights the limitations of LVM and ZFS on physical drives and provides a detailed analysis. The analysis is related to the hardware, software and network configurations of the testing environment.

The objective of the research conducted by Djordjevic and Timcenko [8] is to identify and compare the new features that are added to Ext4 compared to its predecessors. Features related to storage, performance, and reliability are discussed where simulation results showed that Ext4 is indeed superior to its predecessors.

Heger [2] compares BtrFS to ZFS in his paper, although a comparison to an Ext4 setup is also presented. In addition to the comparison, Heger offers sufficient details on ZFS characteristics that relate to reliability, performance, and scalability. This information is a useful addition to the research done by Djordjevic and Timcenko [8] since it makes it simpler to draw comparisons between Ext4 and ZFS.

In their study, Widianto et al. [7] focus on a distinct file system, namely ZFS. Similar to this research, their work revolves around the design of a storage system, albeit one built upon the ZFS framework. This particular approach holds relevance for the current study on two fronts. Firstly, the chosen system configuration becomes noteworthy, particularly due to its insights into software RAID within ZFS, known as RAIDZ. Secondly, Widianto et al. delve into an analysis of the reliability of their ZFS-based storage solution by integrating compression algorithms. This assessment serves as an indicator of the reliability support inherent to ZFS. Both of these vantage points carry significance as they align with the objectives of the comparative analysis conducted in this study.

Based on studies of related work, it is observed that Ext4 and ZFS are currently not compared against each other in a systematic way in more than two or three file system feature categories. In this regard and as discussed, a file system organises and manages data in such a way that it offers data reliability, scalability, performance, and storage capacity to a desired degree. The four feature categories mentioned are not only crucial in the context of file systems but are also essential to the systematic mapping conducted in this research. By using the four feature categories to perform a systematic comparison study, this research can contribute to a better understanding of the state of research in the field of file systems related to Ext4 and ZFS. Another contribution is apparent in the tests that will be performed on Ext4 and ZFS based on the chosen workload as well as hardware setup. The tests involve benchmarking operations where several performance metrics (e.g., throughput, latency, CPU usage, and IOPS) are deemed sufficient indicators of Ext4 and ZFS performance behaviour within the context of this research. In addition, a third test on disc performance is made to aid in understanding the file system performance results better.

III. Systematic Comparison Study Process

The process of the systematic comparison study is similar to a systematic literature review. As a matter of fact, Kitchenham [9] clarifies that a systematic literature review entails researchers gathering data by examining and evaluating research that is relevant to the respective research problem. In this context, the research problem is related to mapping, extracting, and evaluating Ext4 versus ZFS using an organisational scheme, classification scheme, and graphical visualisation. Note that the experimental evaluation is another process with its own setup. However, the results from the comparison study and experimental evaluation are equally crucial to: 1) drawing a conclusion on Ext4 versus ZFS and 2) contributing to research. Nonetheless, the process of the systematic comparison study is summarised in four activities (see Fig. 1). Fig. 1 shows the process begins by selecting appropriate related work. To select appropriate related work, a search strategy needs to be applied. To achieve the latter, Kitchenham [9] recommends some of the following parameters:

1) several databases
2) search string
3) inclusion-, and exclusion criteria
4) amount of search hits.

In this research, IEEE, Semantic Scholar, and Scopus are the three databases that are used. The search string will be used in each database to generate an initial number of hits. In addition, to obtain papers that are relevant to the theme of the comparison study, i.e., Ext4 versus ZFS, the search string is selected based on keywords related to the theme. Therefore, the search string is constructed using the keywords ”File systems”, ”ext4”, and ”zfs”. Also, Boolean operators **OR** and **AND** are included in the search string to refine the

search [9]. Hence, the following search string is used:

("File systems" **AND** (zfs **AND** ext4)) **OR** ("File systems" **AND** ("ext4" **OR** "zfs")).

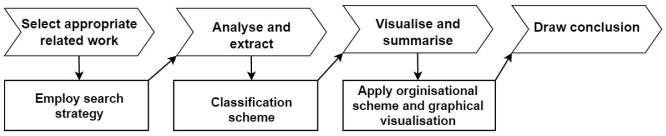

Fig. 1: An illustration of the systematic comparison study process.

To scale down the initial number of hits received by the search string, inclusion-, and exclusion criteria are applied on two levels. The first level entails applying criteria to the title and abstract of each research paper. After the first level, the second level entails exposing the remaining papers to a new set of criteria, this time applied to the entirety of each paper's content. The inclusion-, and exclusion criteria are the following:

1) **Inclusion Criteria**:
   a) Papers are between the years 2006 and 2023 (IC_1).
   b) Papers are written in English (IC_2).
   c) Papers must address one or more feature categories in relation to Ext4 and/or ZFS (IC_3).
   d) Papers must be peer-reviewed (IC_4).
2) **Exclusion Criteria:**
   a) Papers mention the same aspects with the same benchmark environment implementation (EC_1).
   b) Papers that do not illustrate I/O implementation of Ext4 and/or ZFS (EC_2).
   c) Papers requiring payment (outside normal institutional sign-in) and/or request of access from authors (EC_3).
   d) Papers that are not published in journals and conferences (EC_4).

The remaining papers from the second level are gathered and input to the second activity in the systematic process which is "Analyse and extract", as depicted in Fig. 1. Here, a "classification scheme" (see Fig. 2) is applied when analysing each of the gathered papers. This makes it easier to group each paper into one or more categories for file system features. As a result, it is now evident what can be extracted from each document that is crucial to this research; providing guidance for each researcher when they reread the papers helps solidify the findings. Moreover, Fig. 2 shows an improved version of the "classification scheme" since extracted information makes each feature category more detailed. In turn, the researchers can easier map the differences and/or similarities between ZFS and Ext4 within each feature category. From here, a visualisation and summary of what has been extracted are made in the third activity. To achieve this, an organisational scheme called "point-by-point" and histograms are used. A "point-by-point" organisational scheme refers to alternating points about Ext4 with comparable points about ZFS, or vice versa. Finally, a conclusion based on the outcomes of the third activity is reached.

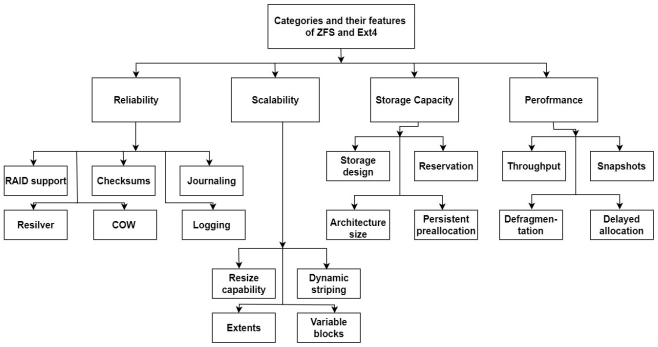

Fig. 2: A detailed classification scheme of the file system categories and their features in relation to Ext4 and ZFS.

IV. Systematic Comparison Study Results

Inserting the search string (mentioned in section III) into the three chosen databases yields an initial total of 781 search hits (see TABLE I). After the implementation of inclusion criterion IC_1, the initial count of research findings is reduced to 766. All these 766 articles present in both IEEE and Scopus conform to inclusion criterion IC_4, as well as exclusion criterion EC_4. Among the 396 papers on Semantic Scholar, six papers do not appear in English, in accordance with inclusion criterion IC_2. This adjustment revises the total count from 766 papers down to 760.

At this stage, it is not possible to ascertain whether any of the 760 papers necessitate payment. Consequently, exclusion criterion EC_3 will be employed subsequently to confirm the full accessibility of the final papers under the MDU license.

Before applying inclusion criterion IC_3 and exclusion criterion EC_2 to the title and abstract of each paper, any duplicates must be eliminated. To facilitate the application of exclusion criterion EC_1, the results from each database are compiled into an Excel sheet for manual review. A total of 28 duplicate papers are identified in Scopus, two duplicates in IEEE, and 20 duplicates in Semantic Scholar. After eliminating these duplicates, the remaining paper count becomes 710. Further refinement involves filtering out papers that lack relevance to the comparative study theme based on their titles and abstracts, leading to a subset of 70 papers. A second round of the same criteria (IC_3 and EC_2) is then applied to the full text of these 70 papers. For efficiency, the workload is divided between researchers, with one handling 35 papers and the other tackling the rest. Ultimately, four papers are identified ([1], [2], [7], [8]), and it is also established that none of them require additional payment to be accessed. The four papers are discussed previously in section II as relevant related work.

It is established that none of the four papers achieve full coverage of all feature categories. However, in combination, they are most relevant to achieving full coverage which is one

of this paper's objectives. Thus, the theoretical findings from evaluating the four papers with respect to each feature category are:

1) ZFS triumphs over Ext4 in reliability aspects. This is due to the higher degree of reliability that it offers within checksumming, RAID support, resilvering, and Copy-On-Write (COW).
2) ZFS and Ext4 are uniform when it comes to offering scalability. This is the result of both file systems offering resize capabilities and extents with identical benefits.
3) ZFS offers greater storage capacity than Ext4 with respect to its storage design and persistent preallocation. However, regarding architecture size, the benefits for storage capacity outweigh each other in both filesystems.
4) Ext4 has an overall better performance in read-, and write IO that are sequential. For mixed workloads, the results tend to lead to Ext4 outputting better performance in terms of throughput. ZFS has a better throughput in regard to random workloads.

Note that these results are still theoretical results and thus, to conclude anything about ZFS versus Ext4 in relation to onboard data processing, experimental testing is required. In specific, clause 4) is examined further in this study.

TABLE I: Illustrates the number of hits of papers by search.

Database	Amounts of hits	After IC_1	Date of search
IEEE	83	82	2023-03-09
Scopus	291	288	2023-03-09
Semantic Scholar	407	396	2023-03-09
Total	781	766	

V. Experimental Setup

For the experimental evaluation, the test environment consisted of the following: *iX10-100* capable of having up to 8 Streaming Multiprocessors (SMs) [4] and equipped with an M.2 1TB NVMe SSD drive, *iX5-100* which uses "Qseven e20xx/e21xx compute modules" [5] and equipped with an M.2 128GB SATA SSD drive. Also, a network router was needed to configure Ext4 and ZFS. To evaluate the performance of the file systems, Test 1 and Test 2 are conducted on both hardware modules (see tables III and IV). As for a disc performance evaluation, Test 3 is conducted (see Table V). All three tests required the use of a standardised benchmarking tool known as *Fio* [10]. The chosen operating system version was Ubuntu 20.04 and this was installed on each hardware module using a bootable USB drive.

After each installation, mounting partitions with Ext4 and ZFS on each hardware respectively were ensured for Test 1 and Test 2. On the *iX10-100*, a single drive setup was used where Ext4 was created on an SSD partition of 100GB, i.e., /dev/nvme0n1p5. The mount point for Ext4 was /mnt/disk5. For ZFS, a pool called "new-pool", containing a single partition of 100GB, i.e., /dev/nvme0n1p4 was created. The mount point for ZFS was /mnt/disk6. Setting up the *iX5-100* entailed a similar approach. However, for Ext4 a partition of 40GB, i.e., /dev/sda6 was created and the mount point for Ext4 was /mnt/disk1. For ZFS, the pool was instead named "mypool" and it contained the partition /dev/sda7 of size 40GB. Furthermore, its mount point was /mnt/temp. Test 3 was conducted last since it needed to be done on raw disks that were not mounted, i.e., /dev/sda and /dev/nvme0n1. See Table II for the exact software versions with respect to Linux kernel, *Fio*, file systems, and operating system.

TABLE II: Software versions.

Software	**Version**
OS	Ubuntu 20.04 LTS Focal
Linux kernel	5.15.32
Benchmarking	FIO 3.16 -1
File system	
ZFS	0.8.3-1 ubuntu 12.15
Ext4	1.45.5-2ubuntu1

Table III shows the first *Fio* benchmark that was conducted. In Test 1, the throughput of Ext4 and ZFS is measured using random writes and reads with block sizes ranging from 1 kB to 100000 kB. The IOdepth in *iX5-100* was set to 16 and for the *iX10-100*, IOdepth was set to 128. The increase in IOdepth value is due to the parallelism of the NVMe drive. The runtime in each case was 60s.

TABLE III: Illustrates the initial Fio benchmark.

Test 1	
Hardware	iX5-100, iX10-100
Fio setup	◇ Random Read and Write ◇ Block Size: 1kB - 100000kB ◇ File Size: 1GB ◇ Jobs: 1 ◇ Focus: Throughput IO

Table IV shows the second *Fio* benchmark that was conducted. In comparison to Test 1, Test 2 contains file size ranges from 2GB to 10GB with a fixed block size of 128kB. For instance, the first script would run by creating a test file of 2GB on top of each respective file system. Finally, the last script would run by creating a test file of 10GB on top of each file system. The IOdepth was set to 256 and runtime in each case was 60s. The number of jobs was set to 16 and the addition of sequential testing serves as a further difference from Test 1. The focus of this benchmark is related to IOPS versus the number of jobs, latency numbers as well as the total CPU utilisation versus file size.

TABLE IV: Illustrates the second Fio benchmark.

Test 2	
Hardware	iX5-100, iX10-100
Fio setup	◇ Random, Sequential Read and Write ◇ Block Size: 128kB ◇ File Size: 2 - 10GB ◇ Jobs: 16 ◇ Focus: IOPS, Latency, CPU usage

Table V illustrates the last conducted experiment for the sake of capturing disc performance. Since the *iX5-100* does not

come with support for an M.2 NVMe SSD drive, Test 3 was deemed a necessary benchmark to add. This benchmark consists of a mixed IO workload that uses the default values of *Fio* for amounts of reads and writes. Both sequential-, and random testing are made, and the number of jobs is set to 5. The focus of this benchmark is to gather results on throughput performance, latency numbers, and CPU utilisation of how many IO operations are made per CPU percentage. The runtime is first 60s and later increased to 20 minutes.

TABLE V: Illustrates the last conducted *Fio* benchmark.

Test 3	
Hardware	iX5-100, iX10-100
Fio setup	◇ Mixed Random, Sequential Read and Write ◇ Block Size: 1kB - 100000kB ◇ Jobs: 5 ◇ Focus: Latency, Throughput IO, CPU usage

VI. Experimental Results

The experimental results of conducting Test 1 indicate that ZFS achieves an overall higher total throughput for the job than Ext4 (see Fig. 3). This is clear when running tests on the *iX10-100*. In Fig. 3, however, the results differ between hardware. When running on the *iX5-100*, which is the space heritage component, it is clear that Ext4 is actually outputting a closer or better behaviour to that of ZFS; Ext4 achieves better throughput in random writes. The overall throughput for Ext4 in random reads is 202MB/s and for ZFS it is 267MB/s. Whereas in random writes, the overall throughput for Ext4 is 132MB/s and ZFS has a throughput value of 50MB/s. Nonetheless, the expectation of ZFS performing better or close to Ext4 in a single drive setup for random testing is accurate against the indications of past research [2].

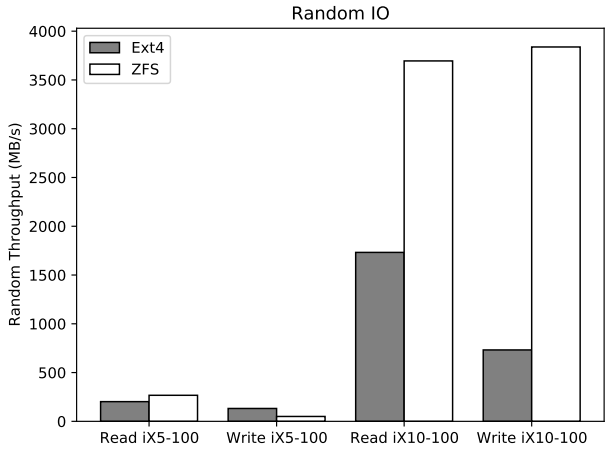

Fig. 3: Random throughput results of Test 1.

Figures 4-7 show the result of conducting Test 2 with respect to IOPS. Almost all figures illustrate linear behaviours of each random and sequential IOPS depending on whether it is a read or write operation. In Fig. 4 and Fig. 5, ZFS has higher slopes than Ext4 with respect to the number of IOPS per thread size. These results are conducted on the *iX10-100*. For 16 threads, the generated amount of random read operations per second is 20799 whereas the amount of sequential read operations per second at 16 threads is 25777, seen in Fig. 5. For random write operations, the IOPS value at 16 threads is 10893 and 15989 for sequential write operations. In Fig. 4, it should be noted that "IOPS_Read_Ext4" and "IOPS_Write_Ext4" are not overlapping. In comparison to ZFS, the Ext4 file system does not reach the high amounts of IOPS per thread size in both random and sequential cases of testing on the *iX10-100*. Fig. 4 indicates that Ext4 has an IOPS value of 4115 at thread size 16 for random read operations. For the same thread size and sequential read operations, this value increases to 6931 IOPS, as seen in Fig. 5. At the same thread size, the IOPS for random write operations with respect to Ext4 is 4316 (see Fig. 4) and it increases to 4682 during sequential write operations (see Fig. 5). Conducting the same test on the *iX5-100* (seen in figures 6 and 7) yields interesting results; "IOPS_Read_ZFS" has the highest slope in both cases of sequential tests when comparing Fig. 5 and Fig. 7.

On the *iX5-100*, the IOPS values reached per thread size are

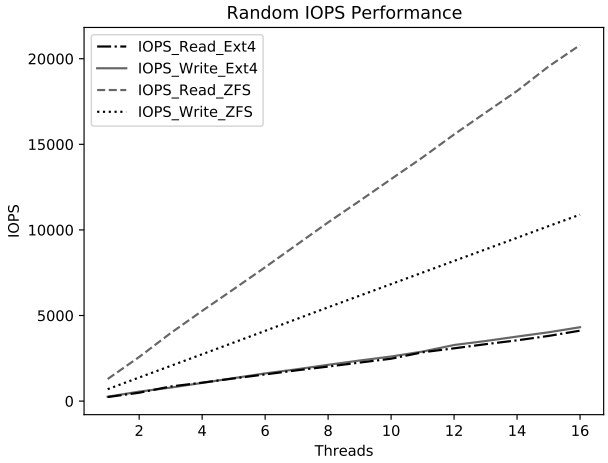

Fig. 4: Random IOPS versus thread size results of Test 2 on *iX10-100*.

not as high as in the case of the *iX10-100*. An underlying factor for this could be the benefits of having an NVMe SSD that can process IO requests faster than the SATA SSD. Specifically, the achieved IOPS values in both sequential-, and random cases at 16 threads for Ext4 (seen in figures 6 and 7) are still low similar to the ones from testing on the *iX10-100*. Even if "IOPS_Read_Ext4" has the highest slope in Fig. 6, its behaviour is still within the ranges of the Ext4 lines seen in Fig. 4. Interestingly, when >12 threads are running, the generated number of random IOPS by Ext4 increases rapidly until it drops at 14 threads, and finally continues linearly from 15-16 threads. This behaviour might be due to an unexpected increase/decrease in read latency. In all the discussed figures with respect to IOPS, one can see that there is a higher slope of read operations than there are write operations. This makes

sense since writing a file entails the creation of metadata whereas reading a file is less expensive.

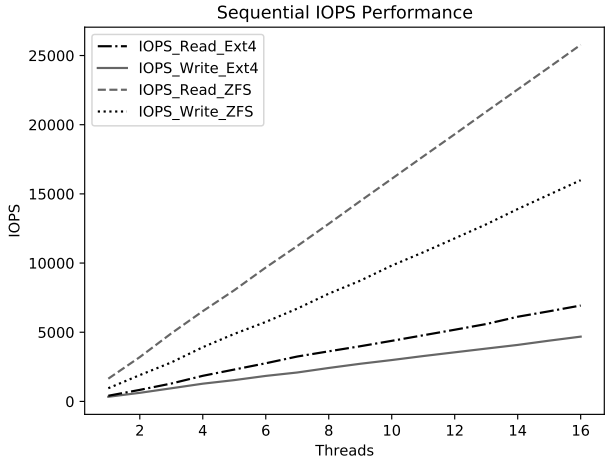

Fig. 5: Sequential IOPS versus thread size results of Test 2 on *iX10-100*.

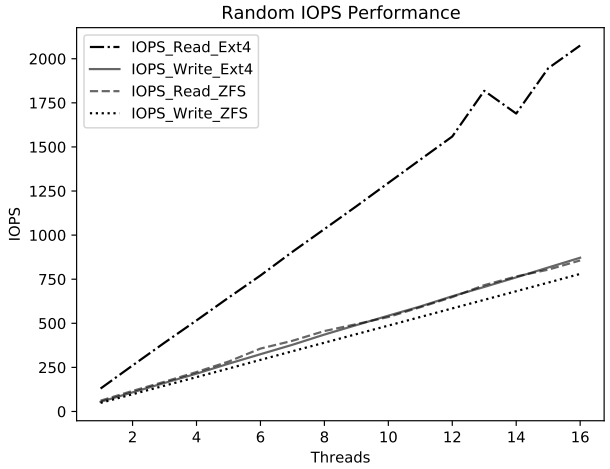

Fig. 6: Random IOPS versus thread size results of Test 2 on *iX5-100*.

The CPU utilisation on the *iX10-100* with respect to each file size created on top of the file systems, respectively, exhibit similar outputs in random-, and sequential cases (see figures 8 and 9). For instance, in Fig. 8 and Fig. 9, reading files up to 10GB using ZFS requires a steady value of roughly 40% of the processor's resources. This is quite ideal and it does not increase towards 100% CPU utilisation which is positively viewed as these tests portray smaller onboard data processing operations. Furthermore, no other user applications were running while conducting the tests. Hence, it is expected that CPU utilisation should not be greatly affected. In Fig. 8 and Fig. 9, Ext4 does not require much of the CPU to carry out sequential-, and random read and write operations. There is a very notable behaviour of the "Write_ZFS" during sequential testing. Fig. 9 shows that the CPU is utilised at 30% and as the file size increases from 6Gb to 8GB, there is a significant drop. To write an 8GB file, ZFS requires roughly 10% of

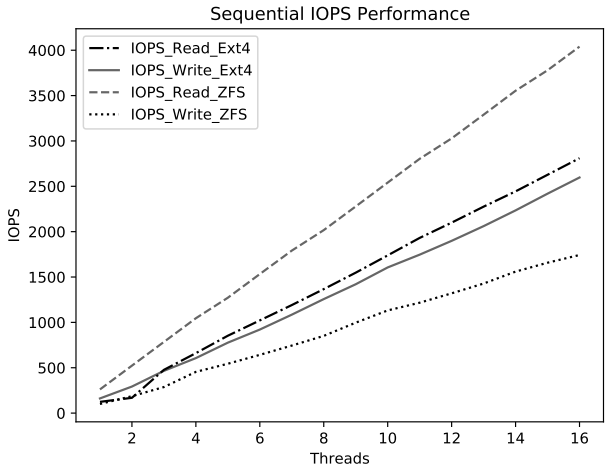

Fig. 7: Sequential IOPS versus thread size results of Test 2 on *iX5-100*. The number of generated IOPS follows a somewhat linear slope for both file systems per increasing thread size.

CPU resources and this is almost constant even when writing a 10GB file. Conducting these tests on the *iX5-100* shows some similar characteristics, however, much lower CPU utilisation is required for all operations (see figures 10 and 11). Both Fig. 10 and Fig. 11 indicate that ZFS achieves higher CPU utilisation values, just like in figures 8 and 9. The lower values of CPU utilisation, in this case, could be connected to the fact that the iX5-100 has a slow SATA SSD in comparison to the much faster NVMe SSD on the *iX10-100*. It is worth mentioning that lower CPU usage means the significance of a continuous time processing task like a low power mode task could then pull the CPU down to low power consumption during these processes to save power resources on space missions. In reality, though, there will be many other applications going while onboard data processing is happening.

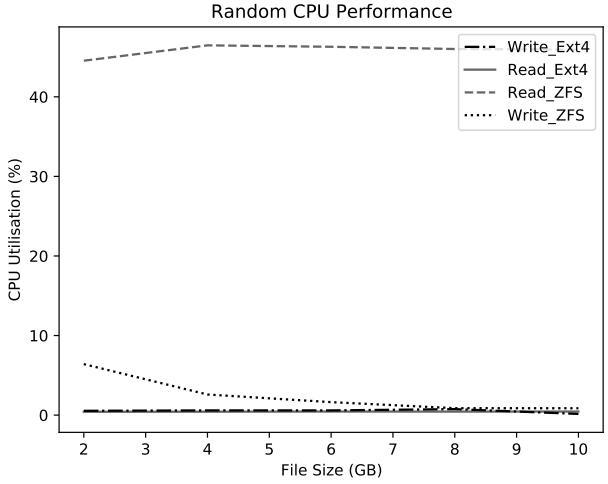

Fig. 8: Random CPU usage results of Test 2 on *iX10-100*.

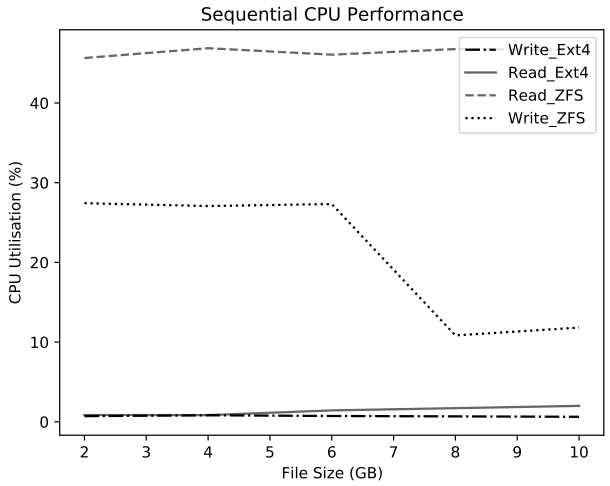

Fig. 9: Sequential CPU usage results of Test 2 on *iX10-100*.

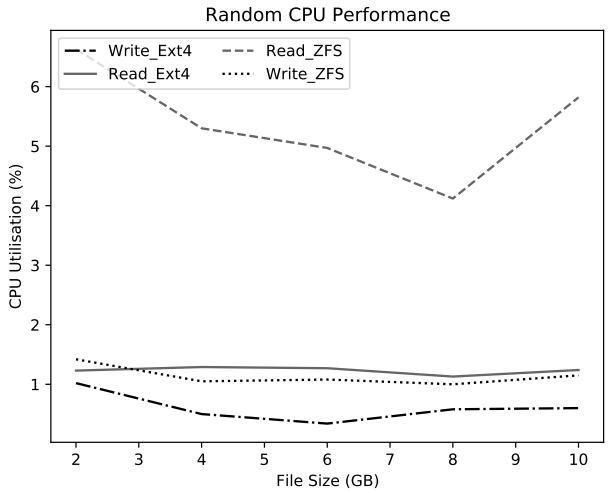

Fig. 10: Random CPU usage results of Test 2 on *iX5-100*.

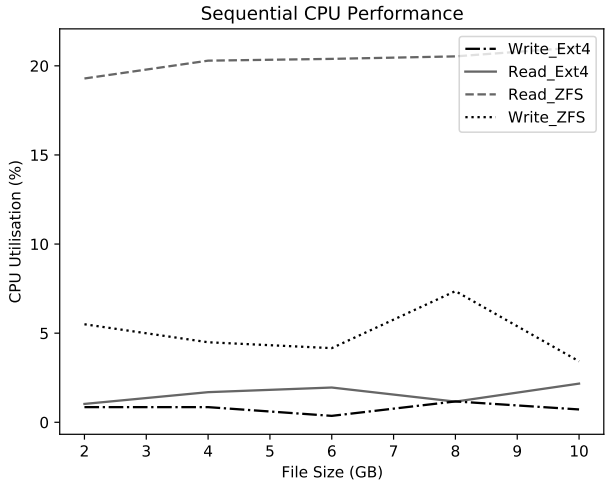

Fig. 11: Sequential CPU usage results of Test 2 on *iX5-100*.

Finally, the disc performance test results are seen in Fig. 12 and Fig. 13. Fig. 12 shows the throughput results when running towards the raw SATA SSD of *iX5-100* versus the raw NVMe SSD of *iX10-100*. Here, the NVMe SSD shows a dominant performance in both random-, and sequential mixed workloads. This comes as no surprise given the characteristics of NVMe SSDs versus SATA SSDs [11]. As for the efficiency, Fig. 13 must not be misleading when viewing it. The number of mixed workload operations per CPU percentage is greater when running the benchmark towards the SATA SSD, however, this is with a runtime of 60s. When increasing the runtime to 20 minutes, the SATA SSD could not keep up and the task was aborted. Hence, it lacks bars for a runtime of 20 minutes. The authors believe this to be due to a high queue depth and increasing the runtime may have stressed the SATA SSD enough to its performance limits. With an increased runtime, the true efficiency of the NVMe SSD could be analysed. Its efficiency shot up quite rapidly as was expected from the beginning. The number of operations per CPU % for a mixed workload of reads is 22262. For a mixed workload of writes, the value is 23096 Ops/%CPU.

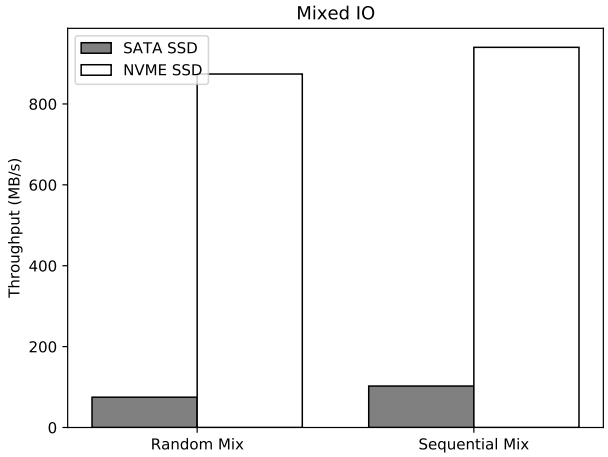

Fig. 12: Throughput results of random-, and sequential mixed IO of Test 3.

A. Latency Results Test 2

Total latency results (from when *Fio* created the IO to its completion) for Test 2 on *iX10-100* with respect to Ext4 and file size of 10GB showed the following:

1) The sequential IO that took the longest to complete had a maximum latency time of 420ms.
2) The random IO that took the longest to complete had a latency of $\approx$ 1221ms.

With respect to ZFS, the results were the following:

1) The sequential IO that took the longest to complete had a maximum latency time of $\approx$ 35ms.
2) The random IO that took the longest to complete had a maximum latency time of $\approx$ 38ms.

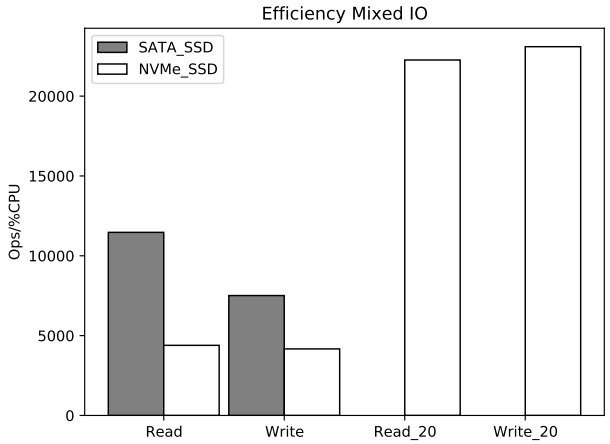

Fig. 13: Efficiency performance of disc types with respect to mixed workload of Test 3, where the NVMe SSD is more efficient than the SATA SSD.

Total latency results (from when *Fio* created the IO to its completion) for Test 2 on *iX5-100* with respect to Ext4 and same file size as above showed:

1) The sequential IO that took the longest to complete had a maximum latency time of ≈ 3655ms.
2) The random IO that took the longest to complete had a maximum latency time of 6182ms.

With respect to ZFS, the results were the following:

1) The sequential IO that took the longest to complete had a maximum latency time of ≈ 287ms.
2) The random IO that took the longest to complete had a maximum latency time of 583ms.

B. Latency Results Test 3

Total latency results (from when *Fio* created the IO to its completion) for Test 3 on *iX10-100* showed the following: the sequential mixed IO that took the longest to complete had a maximum latency time of ≈ 1365ms. As for the random mixed IO that took the longest to complete, its maximum latency time is ≈ 1240ms. The results from testing on *iX5-100* showed the following: the sequential mixed IO that took the longest to complete had a maximum latency time of 6600ms. Whereas the random mixed IO that took the longest to complete had a maximum latency time of ≈ 4845ms.

VII. Conclusion

From the theoretical aspect of this paper, the authors can draw a conclusion on the performance of the two file systems. This conclusion is that in terms of throughput, both ZFS and Ext4 are recommended as great candidates for onboard data processing. However, it is not enough to draw a conclusion on the performance of the two file systems based solely on the conducted systematic comparison study and one metric. Thus, from the experimental results, several solid conclusions are made; solid in terms of the degree of technical evidence that has been presented.

Firstly, on the latency and IOPS, in Fig.4 and Fig.5 together with the discussions from section VI-A, Ext4 generates a lower amount of IOPS and the latency for this is higher than what is experienced with ZFS. This is true for random and sequential IO. The number of operations per second is much higher using ZFS and the overall experienced latency is fairly low compared to Ext4. The writers consider that this benefit comes from ZFS being a full storage stack solution and not just a traditional file system such as Ext4. Furthermore, the latency results for sequential IO outperform the latency results of random IO regardless of the file system. Another conclusion drawn is related to throughput. For small onboard data processing operations, ZFS outputs greater throughput during runtime for random IO. Hence, if one has a lot of random data then ZFS is a better candidate. On average, ZFS uses more CPU resources than Ext4 for random-, and sequential IO per file size whereas Ext4 maintains a low CPU usage throughout runs. Opting for a resource-constrained perspective, Ext4 is the recommended choice for onboard data processing of similar workloads in combination with other running tasks. In addition, the *iX10-100* with its updated processor architecture, in comparison to the *iX5-100*, provides a better foundation for testing the file systems against each other in terms of the presented results related to throughput, CPU usage, efficiency, and IOPS. Lastly, based on the results of running the disc performance test, i.e., Test 3, the authors conclude that NVMe SSDs are more efficient and output better throughput in terms of mixed workloads.

References

[1] M. B. Ab Karim, J.-Y. Luke, M.-T. Wong, P.-Y. Sian, and O. Hong, "Ext4, xfs, btrfs and zfs linux file systems on rados block devices (rbd): I/o performance, flexibility and ease of use comparisons," in *2016 IEEE Conference on Open Systems (ICOS)*, Langkawi, pp. 18–23.

[2] D. A. Heger, "Workload dependent performance evaluation of the btrfs and zfs filesystems," in *35. International Computer Measurement Group Conference, Dallas, TX*, 2009.

[3] V. Phromchana, N. Nupairoj, and K. Piromsopa, "Performance evaluation of zfs and lvm (with ext4) for scalable storage system," in *2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Bangkok, BKK, pp. 250–253.

[4] "Unibap", "*iX10-100 SpaceCloud solution*," https://unibap.com/en/our-offer/space/spacecloud-solutions/ix10100/, (*Last accessed: Aug 25, 2023*).

[5] ——, "*iX5-100 SpaceCloud solution*," https://unibap.com/en/our-offer/space/spacecloud-solutions/ix5100/, (*Last accessed: Aug 25, 2023*).

[6] Unibap AB, "Rymd Industri Innovation," https://www.unibap.com/, (*Last accessed: Aug 25, 2023*).

[7] W. Eko D, P. Agung B, and G. Ahmed, "On the implementation of zfs (zettabyte file system) storage system," in *Proc. of 2016 3:rd Int. Conf. on Information Tech, Computer, and Electrical Engineering (ICITACEE), Semarang*, pp. 408–413.

[8] B. Djordjevic and V. Timcenko, "Ext4 file system in linux environment: features and performance," in *International Journal of Computers*, 2012, pp. 37–45.

[9] B. Kitchenham, "Procedures for Performing Systematic Reviews," Department of Computer Science Keele University, Keele, UK, Tech. Rep. ISSN:1353-7776, 2004.

[10] J. Axboe, "Flexible I/O Tester," 2022, (*Last accessed: Aug 25, 2023*). [Online]. Available: https://github.com/axboe/fio

[11] Y. Son, H. Kang, H. Y. Yeom, and H. Han, "An empirical evaluation and analysis of the performance of nvm express solid state drive," in *Cluster Computing*, 2016, p. 1541–1553.