Evaluation of ZFS as an efficient WLCG storage backend

M Ebert and A Washbrook
School of Physics and Astronomy, University of Edinburgh, Edinburgh EH9 3FD, Scotland, UK
E-mail: marcus.ebert@ed.ac.uk, awashbro@ph.ed.ac.uk

Abstract. A ZFS based software raid system was tested for performance against a hardware raid system providing storage based on the traditional Linux file systems XFS and EXT4. These tests were done for a healthy raid array as well as for a degraded raid array and during the rebuild of a raid array. It was found that ZFS performs better in almost all test scenarios. In addition, distinct features of ZFS were tested for WLCG data storage use, like compression and higher raid levels with triple redundancy information. The long term reliability was observed after converting all production storage servers at the Edinburgh WLCG Tier-2 site to ZFS, resulting in about 1.2PB of ZFS based storage at this site.

1. Introduction
Within WLCG large storage servers are used, traditionally based on hardware raid systems and a Linux file system like EXT4 or XFS. While the capacity of single disks used in such servers was at the order of 2TB some years ago, now disks with capacities of 8TB and even 10TB are available and used in newer installations. Using hardware raid systems in such high capacity servers lead to different problems. The time that is needed for rebuilding a raid array after a disk failure will increase to days or even weeks. Also, the probability that other disks fail during a rebuild operation increases which makes a traditional raid6 system unreliable. In addition to that, a hardware raid system can not provide any data safety since it does not know anything about the data written to disk. A traditional hardware raid system only makes sure that data is read from disk and delivered to the client, without any possibility to check if the data read from disk is also the data that was written to disk. As a potential solution to these problems with hardware raid systems, ZFS on Linux [1] was tested against a similar set up hardware raid system based on EXT4 and XFS.

1.1. ZFS
ZFS was developed by SUN Microsystems for their Solaris OS and made Open Source. On Solaris, ZFS is known for its reliability and high performance since a long time. ZFS is a 128 bit pooled storage system that combines a software raid, volume manager, and file system, and it uses Copy On Write which transforms random writes into sequential writes. Different than hardware raid implementations, ZFS knows about the disk hardware as well as the file system itself. This makes it possible to have a very efficient raid rebuild process since only the used space needs to be rebuilt and not always a whole disk. ZFS can also make sure that data read
from disk is exactly the data that was written to disk by using checksums for every written block. Therefore, ZFS can protect against silent data corruption and in addition it also restores corrupted data blocks from redundancy automatically. To protect against multiple disk failures at the same time, which can happen especially during rebuilds of large capacity disks, ZFS also implements raidz3 which uses an additional parity compared to raid6. In addition to the safety features, ZFS also supports on-disk compression and has a very easy to use administration interface.

A stable Linux implementation as a kernel module became only available recently. To see how the current Linux implementation works in the long run and as a storage backend used for WLCG data storage and access, the latest version of ZFS on Linux was also tested in a long time study over several months.

2. Test setup
For all tests the following hardware was used:

- Machine: Dell R610
- CPU: 2x E5620 2.4GHz, 8 cores and with Hyperthreading enabled
- RAM: 12 GB
- Disk controller: 1 Dell PERC H800 with 1GB onboard cache
- Disks: 36x 2TB (7.2k, NL-SAS 6Gbps, in 3x MD1200)

The disks were configured in a raid60 configuration with 11 disks per raid6 when a hardware raid was used, and in a stripped raidz2 with 11 disks per raidz2 for the ZFS configuration. The hardware raid controller was used in its default configuration (with write-back and adaptive read-ahead cache). For ZFS, the options `xattr=sa` and `relatime=on` were also enabled.

For the data transfer tests, only a workload similar to what is seen in production when used as a WLCG storage server was of interest. Therefore, only sequential read and write access was tested. These read and write tests were done for parallel and non-parallel access, as well as for very large and for very small files. For all read and write tests an own script was used to be able to control the parameters for the file access [2].

The read tests were done on a mix of files from different experiments (Atlas, LHCb, CMS, LSST, ILC, and HyperK) that were on site. A total of 5.1TB was used. The read tests were done sequentially, one by one, and by reading 10 files in parallel. For the ZFS based read tests, compression was used in its default setting.

The write tests were divided in writes of large files and writes of small files. 54GB was chosen as the file size for the large files to eliminate any caching effects and 131kB as the size for the small files. 100 large files and 200,000 small files were written in the different tests. Similar like for the read tests, write tests were done two times, first sequentially by a single process and then with more processes in parallel. For the parallel write tests, 10 of the large files were written in parallel and 20 of the small files. All the above mentioned tests were done for a healthy raid array, a degraded raid array, and while a raid array is rebuilding a failed disk.

As a final test, it was also decided to do a massively parallel access test. These tests were the same as the parallel tests mentioned before but with 400 parallel reads, 100 parallel writes for the large files, and with 400 parallel writes for the small files.

3. Comparison of ZFS against EXT4 and XFS based hardware raid systems

3.1. Performance of a healthy raid array
In this test series, the raid arrays were all fully functional with all disks present and healthy.

As shown in table 1, ZFS showed a much better performance than EXT4 and XFS when the raid array is healthy. The largest performance difference between ZFS and XFS/EXT4 was seen
Table 1. Time needed for a healthy raid system based on ZFS, XFS, and EXT4 to complete the different read and write tests; percentages are relative to the ZFS values.

|                          | ZFS 3xraidz2 | XFS raid60 | EXT4 raid60 |
|--------------------------|--------------|------------|-------------|
| **read, 5.1TB, 18,335 files** |              |            |             |
| sequential read          | 9,981s       | 6,246s     | 8,815s      |
|                          | -37%         | -12%       |             |
| 10 reads parallel        | 7,164s       | 9,936s     | 10,036s     |
|                          | +39%         | +40%       |             |
| **100x(write of 54GB, sync, rm)** |            |            |             |
| seq. write               | 7,849s       | 8,296s     | 15,902s     |
|                          | +6%          | +103%      |             |
| 5 writes in parallel     | 8,133s       | 17,596s    | 39,460s     |
|                          | +116%        | +385%      |             |
| **200,000x(write of 131kb, sync, rm)** |          |            |             |
| seq. write               | 2,453s       | 10,232s    | 10,816s     |
|                          | +317%        | +341%      |             |
| 20 writes in parallel    | 348s         | 1,651s     | 1,274s      |
|                          | +374%        | +266%      |             |

for parallel file access. This is especially important because file access in a production use case is mostly parallel too when used for WLCG data storage.

Interestingly, XFS and also EXT4 showed a better read performance when only one file at a time was read. This can be explained by the cache used within the raid controller. When a hardware raid is used then the controller can efficiently cache the next blocks of a file when a sequential read is performed. However, this performance advantage is expected to go away when more files are read in parallel. This can be seen in the results for the parallel reads for which the performance of the hardware raid dropped behind the ZFS performance.

3.2. Performance of a degraded raid array

During normal operation, a disk can not always be replaced within a very short time period after its failure. Therefore, in this test series, one disk of the raid array was removed and not replaced to test the performance during such degraded time periods.

From the results presented in table 2, it can be seen that the performance for ZFS was nearly unchanged when compared to a healthy raid array. One of the reasons for that could be that ZFS uses variable strip sizes which depend on the file size. That means that small files or the last block of a large file could end up to be not distributed over all disks because it is a too small amount of data. In the case of a degraded raid array, this means that not for all files parts of a logical data block are missing and therefore do not need to be restored from redundancy.

Hardware raid arrays, at least with the controller used here, use a fixed strip size. Therefore, each logical block needs to be restored using the redundancy. This has a large performance impact for the reading of files.
Table 2. Time needed for a degraded raid system (1 failed disk) based on ZFS, XFS, and EXT4 to complete the different read and write tests; percentages are relative to the corresponding values of a healthy raid system.

|                          | ZFS 3xraidz2 | XFS raid60 | EXT4 raid60 |
|--------------------------|--------------|------------|-------------|
| **read, 5.1TB, 18,335 files** |              |            |             |
| sequential read          | 10,059s      | 19,535s    | 20,091s     |
|                          | +1%          | +213%      | +128%       |
| 10 reads parallel        | 7,568s       | 19,094s    | 17,949s     |
|                          | +6%          | +92%       | +79%        |
| **100x(write of 54GB, sync, rm)** |          |            |             |
| seq. write               | 7,828s       | 8,411s     | 18,584s     |
|                          | 0%           | +1%        | +17%        |
| 5 writes in parallel     | 7,955s       | 17,327s    | 39,170s     |
|                          | -2%          | -2%        | -1%         |
| **200,000x(write of 131kb, sync, rm)** |        |            |             |
| seq. write               | 2,441s       | 10,293s    | 10,706s     |
|                          | +0%          | +0%        | -1%         |
| 20 writes in parallel    | 348s         | 1,524s     | 1,653s      |
|                          | +0%          | -8%        | +30%        |

3.3. **Performance of a degraded raid array during rebuild**

With the current disk sizes and their expected increase in the future, the time that is needed to rebuild a degraded raid array is at the order of days or even weeks. Therefore, this will also have impact on production performance. In this test series, the disk that was removed previously was replaced with a new disk and the rebuild process started. The results are shown in table 3.

For the reading of files, there was a very large decrease of performance seen when a hardware raid was used with XFS or EXT4, while the performance for a ZFS based raid array was only slightly decreased. For XFS and EXT4, the write performance was also decreased especially when only one file at a time was written to disk. However, this write test for large files was not possible to perform on ZFS because the rebuild had finished before the write test finished. Therefore the ZFS write results would be a mix of write access during rebuild and write access for a healthy raid array. Using these results would not be a fair comparison to the hardware raid based write tests and therefore are not presented here.

A hardware raid system does not know about the data on the disks and needs to read blocks equal to the maximum amount of data that can be stored in the raid array and needs to write out blocks equal to the capacity of a single disk. In the test case with 11x2TB disks per raid6 array, each raid6 provides 9x2TB of usable disk space. In the rebuild case, this means that 9x2TB need to be read and restored from redundancy, while 2TB need to be written to the new disk. This rebuild process took about 21h.

ZFS however knows about the data stored in the raid system and therefore needs to restore only data blocks that are actively used. In this test case, the 5.1TB for the read test and the large files from the write test while it was ongoing were in the array, stripped over three raidz2. That means that during the rebuild process only the data blocks that were used on the failed disk needed to be rebuild which is a small amount compared to the overall capacity. The ZFS
Table 3. Time needed for a degraded raid system during rebuild based on ZFS, XFS, and EXT4 to complete the different read and write tests; percentages are relative to the corresponding values of a healthy raid system.

|                              | ZFS 3xraidz2 | XFS raid60 | EXT4 raid60 |
|------------------------------|--------------|------------|-------------|
| **read, 5.1TB, 18,335 files** |              |            |             |
| sequential read              | 13,327s      | 21,655s    | 21,735s     |
| 10 reads parallel            | 10,036s      | 20,055s    | 19,500s     |

|                              |              |            |             |
| 100x(write of 54GB, sync, rm) |              |            |             |
| seq. write                   | -            | 15,126s    | 23,127s     |
| 5 writes in parallel         | -            | +82%       | +45%        |
|                                | -            | 19,807     | 42,813s     |
|                                | -            | +13%       | +8%         |

| **200,000x(write of 131kb, sync, rm)** | | | |
| seq. write                    | 4,210        | 9,584s     | 11,974s     |
|                                | +72%         | -6%        | +11%        |
| 20 writes in parallel         | 820s         | 1,549s     | 1,377s      |
|                                | +136%        | -6%        | +8%         |

rebuild process finished within 2:15h.

While for small files a decrease in write performance was seen for ZFS, it still performed much better than XFS or EXT4 on a hardware raid.

### 3.4. Performance for a healthy raid array and highly parallel file access

During this test series, the performance for highly parallel file access was tested. This scenario will most likely not happen during regular production use. However, there can be times when hundreds of jobs access files on a single server at a site. Because it is not expected that such data access happens over a longer period of time in production, the tests were limited to test parallel file access for a healthy raid array.

The same tests as before were used, however the number of processes accessing files in parallel was increased. For the read test, 400 read processes were used in parallel instead of the default value of 10. For the write test using small files, the number of parallel processes was also increased to 400, while all 100 files were written in parallel for the write test using large files.

As shown in table 4, the read performance on ZFS increased compared to the default parallel file access shown in table 1, while the read performance on a hardware raid dropped by a large amount for XFS and EXT4. The write performance of large files decreased for all three file systems, however ZFS performed still much better than XFS and EXT4 in this scenario. The performance for small writes increased for all three file systems when using more parallel processes, with ZFS showing the best performance writing out all 200,000 files about three times faster than XFS.
Table 4. Time needed for a healthy raid system based on ZFS, XFS, and EXT4 to complete the different read and write tests with a large number of parallel processes accessing files; percentages are relative to the values for default parallel access reported in table 1.

| Test Description | ZFS 3xraidz2 | XFS raid60 | EXT4 raid60 |
|------------------|--------------|------------|-------------|
| **read, 5.1TB, 18,335 files** | 5,253s | 21,512s | 23,400 |
| 400 reads in parallel | -27% | +117% | +133% |
| **100x write of 54GB, sync, rm** | 10,800s | 27,751s | 46,468s |
| 100 writes in parallel | +33% | +58% | +18% |
| **200,000x write of 131kb, sync, rm** | 176s | 511s | 469s |
| 400 writes in parallel | -49% | -69% | -63% |

4. Other ZFS features useful for WLCG data storage

4.1. Compression

One of the distinct features of ZFS is providing block level based compression. The different algorithms that can be used are lz4(default), lzjb, and gzip. All three were tested for their compression result as well as for the impact of the overhead coming from decompression while reading files.

The default algorithm, lz4, showed very good results with much lower overhead than all other algorithms and with no negative impact on the overall performance. It was found that using lz4 an average compression of about 5% could be achieved for Atlas data files. The best compression results for data files were achieved for RDO and ESD files. Data files from astronomy groups were also used to test compression and a compression ratio of 1.18 was found using the lz4 algorithm. This could become more important for sites in the future when new projects, like LSST, start to take data. The data volume for such projects is expected to be at the order of tens of petabytes and compression could provide substantial savings of disk space.

All compression results are summarized in table 5 and table 6. The results shown in table 5 are based on all available on-site data for LHCb, CMS, ILC, and LSST. Due to the low statistics, the results that are based on data files from the LHCb, CMS, and ILC experiments should be verified on other sites that have a larger amount of such files available.

4.2. Raid levels

With single hard disk capacities of 8TB or more, the rebuild times for raid6 based raid arrays can be at the order of weeks depending on the overall array size. This makes it much more likely that additional disks will fail during the rebuild especially since the remaining disks will be put under higher load than during normal production use. For a reliable data storage production system, raid6 will stop to be usable very soon for the same reasons raid5 should not be in use for large storage systems today.

ZFS provides additional levels of protection here. Instead of using a raidz2, which provides double redundancy similar to a hardware raid6, one has also the possibility to use raidz3 which provides an additional redundancy. A raidz3 over 35 disks was tested against the 3xraidz2 used in the previous tests. While it is in general expected that a single array with a large number of
Table 5. Compression results shown for different algorithms usable with ZFS applied to files from different experiments; lz4 is the default algorithm with ZFS.

| compression algorithm | ATLAS 4.3TB | LHCb 0.5TB | CMS 0.2TB | ILC 0.5TB | LSST 1TB | HyperK 0.7TB |
|-----------------------|-------------|------------|-----------|-----------|---------|-------------|
| lz4                   | 1.04        | 1.03       | 1.01      | 1.00      | 1.18    | 1.02        |
| lzjb                  | 1.02        | 1.01       | 1.00      | 1.00      | 1.16    | 1.01        |
| gzip                  | 1.05        | 1.03       | 1.01      | 1.00      | 1.24    | 1.03        |
| gzip1                 | 1.05        | 1.03       | 1.01      | 1.00      | 1.21    | 1.03        |
| gzip5                 | 1.05        | 1.03       | 1.01      | 1.00      | 1.23    | 1.03        |
| gzip9                 | 1.05        | 1.03       | 1.01      | 1.00      | 1.24    | 1.03        |

Table 6. Compression results shown for different file types of the Atlas experiment using lz4.

| file type | number of files | du – apparent-size report | du report | ratio |
|-----------|----------------|--------------------------|-----------|-------|
| tgz       | 1.1M           | 2.7TB                    | 2.5TB     | 1.08  |
| RDO       | 1.1k           | 1.1TB                    | 0.87TB    | 1.26  |
| ESD       | 44k            | 5.6TB                    | 4.9TB     | 1.14  |
| EVNT      | 52k            | 11.7TB                   | 11.6TB    | 1.01  |
| HITS      | 120k           | 34.6TB                   | 34.6TB    | 1.00  |
| AOD       | 346k           | 401.5TB                  | 388.7TB   | 1.03  |
| other     | -              | 215.1TB                  | 205.8TB   | 1.05  |

disks performs worst than a stripped array with a small number of disks in each array, it was found that the performance when using a large raidz3 compared to the stripped raidz2 arrays did not decrease too much.

While there was no significant difference in the write performance seen, the read performance decreased by about 30% when using a large raidz3. This was found consistently for a healthy raid array, a degraded raid array, as well as for a degraded raid array during rebuild. Due to the large number of disks in the raidz3 (35 disks) compared to the stripped raidz2 with 11 disks per raidz2, the rebuild time also increased when raidz3 is used.

For production use, a limiting factor for data access is usually the network bandwidth. It was found that while the performance of a 35 disk array using raidz3 was lower than for a stripped raidz2, a 10Gbps network could still be saturated. That makes the raidz3 configuration a well-suited storage solution for large and reliable storage systems. An additional benefit of the raidz3 usage is the increased storage space. On the test system with 3xraid6 and 11 disks per raid6, the space of 6 disks is used for redundancy information. Using a raidz3, only half of that space is needed for redundancy while providing a better reliability.
5. Long term reliability and usage
At the Edinburgh Tier-2 site [3] which is part of ScotGrid and GridPP [4], the first server was switched to ZFS at the end of 2015. After initial tests, the whole storage on site was converted to use ZFS over the next months. Since May 2016, 11 storage servers are providing between 56TB and 225TB of local storage space using ZFS. These machines are similar like the test machine, using 36 disks with a single disk capacity of 2TB, 4TB, or 8TB. A single raidz3 is used over all disks on a server. The whole storage space is managed by DPM [5] and file access is done mostly by using xrootd [6]. The overall usable ZFS based storage space on site is about 1.2PB, not taking compression into account.

Since the first server was switched to ZFS, the ZFS version and kernels were updated more than once. No single problem related to ZFS was observed so far while the administration of the storage became much easier to handle. Also, the setup of a new raid array became much easier with ZFS [7].

6. Conclusion
It was found that a software raid system based on ZFS can not only be reliable and performing well, but also that it can perform much better than a hardware raid system combined with a traditional Linux file system. The better performance of ZFS was observed for healthy raid arrays as well as for degraded raid arrays and during the rebuild of a degraded raid array. ZFS performed especially better than XFS and EXT4 based hardware raid systems when different files were accessed in parallel. The better performance was observed for reading from storage as well as for writing to storage.

It was also found that ZFS is a future-proof system by providing additional raid levels, like raidz3, which account for the higher probability of disk failures when having a large amount of disks in a raid array with large single disk capacities. In addition, it was found that block level compression provided by ZFS has no impact on the performance when used in default mode and can achieve a substantial saving of storage space taking the whole data amount within WLCG into account.

ZFS was found to be a high performance storage backend and very reliable also in the long term. Due to having no need for hardware raid controllers, it can also provide cost savings for future storage purchases.

References
[1] Native ZFS on Linux, Online: http://zfsonlinux.org/
[2] Ebert M 2016 "File system tests", Online:
http://gridpp-storage.blogspot.co.uk/2017/01/file-system-tests.html
[3] Edinburgh Tier-2 GridPP site, Online:
http://www.ph.ed.ac.uk/particle-physics-experiment/our-activities/gridpp
[4] Britton D et al. 2009 "GridPP: the UK grid for particle physics", Phil. Trans. R. Soc. A 367 2447-2457
[5] Disk Pool Manager (DPM), Online: http://lcgdm.web.cern.ch/dpm
[6] XRootD, Online: http://xrootd.org/
[7] Ebert M 2016, "ZFSonLinux, Installation and experience at Edinburgh", Online:
https://indico.cern.ch/event/539135/contributions/2214537/