Evaluating Power-Performace Benefits of Data Compression in HPC Storage Servers

Konstantinos Chasapis, Manuel F. Dolz, Michael Kuhn, Thomas Ludwig
Department of Informatics
University of Hamburg
Hamburg, Germany
E-mails: {konstantinos.chasapis,manuel.dolz,michael.kuhn,thomas.ludwig}@informatik.uni-hamburg.de

***Abstract*—Both energy and storage are becoming key issues in high-performance (HPC) systems, especially when thinking about upcoming Exascale systems. The amount of energy consumption and storage capacity needed to solve future problems is growing in a marked curve that the HPC community must face in cost-/energy-efficient ways. In this paper we provide a power-performance evaluation of HPC storage servers that take over tasks other than simply storing the data to disk. We use the Lustre parallel distributed file system with its ZFS back-end, which natively supports compression, to show that data compression can help to alleviate capacity and energy problems. In the first step of our analysis we study different compression algorithms with regards to their CPU and power overhead with a real dataset. Then, we use a modified version of the IOR benchmark to verify our claims for the HPC environment. The results demonstrate that the energy consumption can be reduced by up to 30 % in the write phase of the application and 7 % for write-intensive applications. At the same time, the required storage capacity can be reduced by approximately 50 %. These savings can help in designing more power-efficient and leaner storage systems.**

Keywords-power consumption; high performance computing; parallel distributed file system; storage servers; compression;

I. INTRODUCTION

As we progress towards Exascale systems, the economic cost of energy consumption and the pressure exerted by power dissipation on cooling equipment are rapidly becoming major hurdles to the deployment of new HPC facilities. As of today, the most energy-efficient HPC supercomputers deliver close to 4.5 GFLOPS/W (10^9 floating-point operations per second, per watt) [1]. Simple arithmetic shows that building an ExaFLOPS system based on this technology would require about 220 MW, yielding this approach economically unfeasible. Even if we can maintain the considerable improvements experienced by the most energy efficient systems, the goal of building a 20 MW Exascale system will still be largely exceeded. If we want to continue enjoying the significant advances enabled by scientific computing and supercomputers during these past decades, a holistic investigation is needed to improve energy efficiency of HPC hardware and software.

High performance I/O is also a major stumbling block to reach the ExaFLOPS barrier. Considering that large-scale computations and simulations conducted to solve complex problems in several scientific domains increasingly produce large amounts of data, the ExaFLOPS performance goal is still far away. While CPU speed and HDD capacity have increased by roughly a factor of 1,000 every 10 years [2], HDD speed only develops at a slow pace: only a 300-fold throughput increase over the last 25 years. Minimizing the amount of data that is being stored in the storage subsystem can also help to improve application performance. Thus, data compression can be used for this purpose. Furthermore, due to the increasing electricity footprints, energy used for storage represents an important portion of the Total Cost of Ownership (TCO) [3]. For instance, the German Climate Computing Center's (DKRZ) HDD-based storage system is using 10,000 HDDs to provide a 7 PB file system for earth system science. Assuming a power consumption of 5–10 W for typical HDDs, this results in energy costs of 50,000–100,000 € per year for the HDDs alone. To make the problem worse, the growth of HDD capacity has recently also started to slow down, requiring additional investment to keep up with the increasing processing power.

In this paper, we analyze the usefulness of HPC storage servers that compress file data in order to reduce the required amount of storage hardware and the overall energy consumption, and thus the total cost of storage hardware usually found in supercomputers. As current-generation CPUs provide ample performance for data processing such as compression or encryption, we provide a case for turning on compression by default to reduce the number of required storage devices. In particular, the paper includes the following contributions:

1) We review the architecture of parallel distributed file systems for the special case of Lustre and present a power-performance profiling and tracing environment for these platforms.
2) We evaluate the performance of different compression algorithms using different synthetic and real scientific sets of input data.
3) We employ the parallel I/O Performance Benchmark (IOR) [4] and real scientific data to assess different compression algorithms of the Lustre parallel distributed file system with its Zettabyte File System (ZFS) [5] back-end.
4) We evaluate the impact of slowly-spinning, energy-aware HDDs used in storage servers as a way to reduce energy consumption in large storage servers.

The paper is structured as follows: In Section II, we present the parallel distributed file system Lustre and our power-performance measurement framework used to analyze the benefits of our proposed approach. Section III contains an extensive evaluation of our proposal to compress all data produced by HPC systems. We present related work in Section IV. Finally, we draw conclusions and discuss further work in Section V.

 ISBN: 978-1-61208-332-2

II. Background

In this section, we explain the power-performance framework that we use to conduct our evaluation. First, we describe the distributed parallel file system and the power-performance measurement framework we used. Parallel distributed file systems are used in HPC to aggregate storage devices existing in several storage servers. File systems provide an abstraction layer between the applications and the actual storage hardware such that application developers do not have to worry about the organizational layout or technology of the underlying storage hardware. To meet the high demands of current high performance computing applications, they distribute data across multiple storage servers. Moreover, they are designed to allow parallel access to multiple clients and cooperatively work with the same data concurrently. On the other hand, power measurements are needed to design new energy-aware approaches and meet consumption constraints. In this case, wattmeters attached to the server nodes are leveraged in a profiling and tracing framework that allows developers to correlate performance and power data.

A. Parallel distributed file systems

To perform our evaluation we use Lustre [6]. Lustre is an open-source parallel distributed file system and is widely used on current supercomputers. Lustre powers around half of the TOP100 supercomputers and almost one third of all TOP500 supercomputers [7]. In contrast to other proprietary solutions such as the General Parallel File System (GPFS) [8], it is possible to adapt, extend and improve Lustre since it is licensed under the GPL (version 2).

Lustre distinguishes between file system clients and servers. While all clients are identical, the servers can have different roles. First, the *object storage servers* (OSSs) manage the file system's data and are connected to possibly multiple *object storage targets* (OSTs) that store the actual file data. Second, the *meta-data servers* (MDSs) handle the file system's meta-data, such as directories, file names and permissions. Each *MDS* is connected to possibly multiple *meta-data targets* (MDTs) that store the actual meta-data. The general architecture of Lustre is illustrated in Figure 1 on this page. Both *MDTs* and *OSTs* use an underlying back-end file system to store their data. This file system is responsible for block allocation and hiding the internal disk layout from Lustre. Past versions of Lustre supported only a modified version of `ext4` [9] called `ldiskfs`. Current versions of Lustre also support ZFS [10].

ZFS is a local file system that offers a rich set of advanced features. Among others, it provides advanced storage device management, data integrity, as well as transparent compression and deduplication. It supports several compression algorithms: zero-length encoding `zle`, `lzjb` (a modified version of `LZRW1` [11]), `lz4` [12] and `gzip` (a variation of LZ77 [13]). Performing the compression on the servers has several advantages: the compression is completely transparent to other applications (including Lustre), no modifications of the client libraries or operating systems are necessary, and computations of the clients are not influenced by the CPU overhead of the compression.

B. Power-Performance measurement framework

To analyze power and performance metrics of the Lustre storage servers, we employ a version of the integrated framework presented in [14] that works in combination with

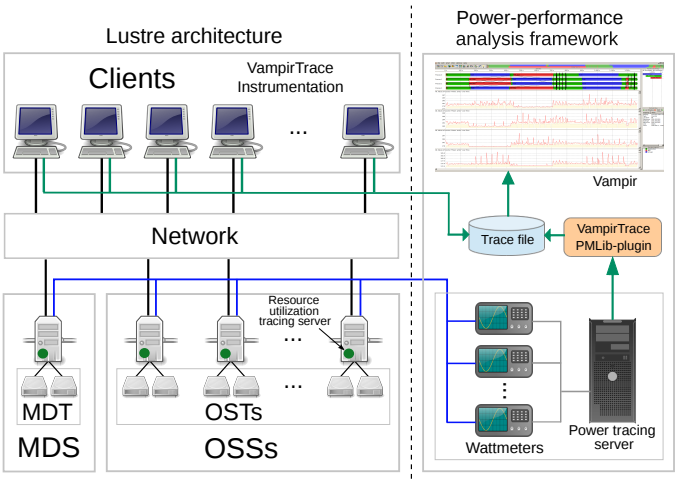

Figure 1. Lustre architecture in combination with the power-performance analysis framework.

VampirTrace and Vampir, which are profiling/tracing and visualization tools, respectively. The left part of the Figure 1 on the current page offers a graphical representation of the Lustre architecture; the right depicts the tracing and profiling framework. To use our approach, the application running on the clients is compiled using the VampirTrace compiler wrappers, which automatically instrument the code. Next, the application is run on the client nodes, thus consuming a certain amount of energy, both on the client and server nodes, due to the I/O instructions. The server nodes are connected to power measurement devices that account for the consumed energy and send the power data to the tracing server. The attached VampirTrace `pmlib` plugin employs the client API that sends start/stop primitives in order to gather captured data by the wattmeters onto the tracing server, where an instance of the `pmlib` server is running. Once the application's run has finished, the VampirTrace `pmlib` plugin receives the power data from the tracing server. The instrumentation post-process generates the performance trace files and the `pmlib` plugin inserts the power data into them.

In addition to the power measurements, we also account for the resource utilization values of the Lustre servers: CPU load, memory usage and storage device utilization. We run special `pmlib` server instances on the server nodes that retrieve these values from the `proc` file system (leveraging the `psutil` Python library). Thus, `pmlib` plugin instances running with the instrumented application connect with the `pmlib` servers and also receive resource utilization data of the Lustre servers. Finally, using the Vampir visualization tool, the power-performance traces can be easily analyzed through a series of plots and statistics.

III. Evaluation

A. Methodology

We leverage Lustre 2.5 with its ZFS back-end file system for our power-performance analysis of data compression in HPC storage servers. As the first step, we evaluate the compression algorithms that ZFS supports. For this purpose, we choose a real data-set that contains input and output data of the Max Planck Institute Ocean Model (MPI-OM), which is the ocean/sea-ice component of the MPI Earth System Model [15]. The data-set has a size of 540 GB, consisting of around 73 % binary data and 27 % NetCDF data [16]. In this evaluation, we also collect system statistics such as the average CPU load

 ISBN: 978-1-61208-332-2

and the execution time of the tests. To extract the compression ratio of the data-set we use the `zfs get compressratio` command that reports the actual compression ratio of the file system. Next, we evaluate the compression algorithms that are suitable for our needs with repeated and random patterns in order to understand the behavior in extreme cases. Using above information, we pick the best suitable algorithms in terms of compression ratio, CPU load and execution time to run our Lustre experiments. For now, we do not consider decompression because the evaluated compression algorithms are expected to have higher overheads when compressing than when decompressing.

In order to evaluate data compression in HPC storage servers we use the IOR, which is a commonly-used, configurable benchmark for parallel distributed file systems. We employ IOR to simulate realistic write activity of the Lustre storage servers and collect the execution time and throughput. To fulfill our needs we have modified IOR in the following ways: *(i)* we have inserted a sleep operation to simulate the computation phase, *(ii)* we have added a read phase in the initialization of the program in order to fill the write buffer with part of our scientific data-set, and *(iii)* we have instrumented it using the VampirTrace compiler wrappers to obtain power-performance-related traces.

To simulate the application's computation phase, we introduce a sleep instruction into our modified IOR. This simulates a common scenario in real-world applications, which alternate between computation and I/O phases. Consequently, the I/O systems see bursts of activity during the I/O phases and are otherwise relatively idle. For our cases, we choose to sleep approximately four times more than the time spent during write operations, which is reasonable for write-intensive applications. The data used to initialize the write buffer corresponds to the first 10 GB of an appropriate file from our scientific data-set with compression ratios of 1.7 and 2.53 for the `lz4` and `gzip` algorithms, respectively.

B. Environment Setup

The experimental setup includes a cluster with ten client and ten server nodes. Each client node is equipped with two Intel Xeon Westmere X5650 processors (6 cores each) running at 2.66 GHz, 12 GB of RAM and one 250 GB Seagate Barracuda HDD. Each storage server node has one Intel Xeon Sandy Bridge E31275 processor (4 cores) running at 3.40 GHz, 16 GB of RAM, one 160 GB Intel SSD and three 2 TB Western Digital Caviar Green HDDs. An energy-saving feature of these HDDs is the IntelliPark technology (also referred to as `idle3`) [17], which positions the read/write heads unloaded in a parking position and turns off certain drive electronics after a pre-defined inactivity time (8 seconds by default).

The client and servers nodes are interconnected using Gigabit Ethernet. Furthermore, a Lustre file system is provided by the ten server nodes. All ten nodes are configured as OSSs and use a ZFS pool containing one of their HDDs. Additionally, one node also fulfills the role of the MDS, using a ZFS pool containing the SSD; we use the SSD to exclude the influence of meta-data operations. We configure Lustre to stripe file data among all OSSs to get the best possible performance. The total amount of data that we write is 600 GB, that is, almost four times more than the total amount of RAM that the the OSSs are equipped with. When simulating client-side computation, we write 150 GB per I/O phase and then issue a sleep command to the clients for 600 seconds, which is approximately four times the length of the I/O phase.

TABLE I. COMPARISON OF DIFFERENT COMPRESSION ALGORITHMS.

Comp. Algorithm	Comp. Ratio	Avg. CPU Util. (%)	Runtime Ratio
`off`	1.00	23.7	1.00 (2:11 h)
`zle`	1.13	23.8	1.04
`lzjb`	1.57	24.8	1.09
`lz4`	1.52	22.8	1.09
`gzip-1`	2.04	56.6	1.06
`gzip-2`	2.05	62.3	1.05
`gzip-3`	2.02	71.9	1.11
`gzip-4`	2.08	73.0	1.07
`gzip-5`	2.06	80.2	1.21
`gzip-6`	2.04	84.7	1.88
`gzip-7`	2.05	85.1	2.36
`gzip-8`	2.06	86.8	4.79
`gzip-9`	2.08	83.1	13.66

We have observed that the best results were delivered with private files per process and consequently use this configuration. These results are to be expected because there is no locking overhead implied from the file system's POSIX semantics. We configure IOR with a block size of 256 KB for the following reasons: *(i)* it is aligned to the file system stripe size, *(ii)* it is large enough to minimize performance penalties and *(iii)* it fits the guidelines found in [18], which describes energy-efficient best practices for file system operations. Moreover, to avoid any caching effects on the client side, we allocate 85 % of their main memory. This helps achieving realistic measurements because real-world applications usually allocate as much memory as possible for computations.

Finally, to account for the power consumption we employ four external, calibrated ZES ZIMMER LMG450 devices connected between the power supply unit of the I/O servers and the electrical outlets. For the experiments we set the sampling frequency to 20 Hz.

C. Experimental Results

To evaluate the different compression algorithms supported by ZFS we copied the data-set into an uncompressed ZFS pool and set up another separate ZFS pool with the compression algorithm that we want to test. After copying the data-set into the compressed ZFS pool, we measured the compression ratio as well as the CPU utilization and runtime. Table I on this page shows a comparison of different compression algorithms supported by ZFS. As can be seen, disabling the compression yields the best runtime and lowest CPU utilization. The zero-length encoding (`zle`) adds negligible runtime and CPU overhead and already achieves a compression ratio of 1.13, which reduces our data-set to 470 GB. Both `lzjb` and `lz4` increase runtime and CPU utilization slightly but offer a significant boost for the compression ratio; even though the average CPU utilization is actually reduced with `lz4`, more CPU time is consumed due to the increased runtime. `lzjb` and `lz4` compress our data-set to 343 GB and 354 GB, respectively. The `gzip` compression algorithm additionally allows the compression level (1–9) to be specified. Even the lowest `gzip` compression level (`gzip-1`) further increases the compression ratio to 2.04 and compresses our data-set to 267 GB. While the runtime overhead is negligible, the CPU utilization more than doubles. The higher `gzip` compression levels do not significantly improve the compression ratio but increase the runtime and CPU utilization. Consequently, we choose `lz4` and `gzip-1` as the compression algorithms for our further analysis.

 ISBN: 978-1-61208-332-2

TABLE II. COMPARISON OF DIFFERENT DATA PATTERNS WITH SELECTED COMPRESSION ALGORITHMS.

Pattern	Comp. Algorithm	Comp. Ratio	Avg. CPU Util. (%)	Runtime Ratio
repeated	off	1.00	23.7	1.00 (11:13 min)
	lz4	126.96	15.8	1.28
	gzip-1	126.96	23.3	1.24
random	off	1.00	23.5	1.00 (11:21 min)
	lz4	1.00	24.1	0.97
	gzip-1	1.00	66.1	1.03

To have a better understanding of the algorithms that we use in our evaluation of the parallel distributed file system, we examine them with different data patterns. First, we use easily-compressible (repeated data obtained using the `yes` utility) and non-compressible (random data from the `frandom` RNG) data patterns. In this case, we have used a pre-generated 50 GB file and proceeded as in the previous evaluation. Table II on the current page shows a comparison of different data patterns using the previously selected compression algorithms `lz4` and `gzip-1`. As can be seen, compressing the repeated data incurs computational overhead. Both algorithms achieve the same compression ratio and almost the same runtime ratio. However, the `lz4` algorithm uses significantly less CPU time than the `gzip-1` algorithm (around 68 %). Both algorithms have a lower CPU utilization than the default configuration without compression; we believe that this is due to the fact that the dramatically reduced amount of data almost completely eliminates the overhead incurred by ZFS's checksumming and copy-on-write. When using the random data, again, both algorithms need approximately the same runtime and achieve the same compression ratio. In this case, the difference in CPU utilization is even more pronounced with the `lz4` algorithm using it only around 36 % as much as the `gzip-1` algorithm.

While the previous tests only used a single machine to get a basic understanding of the different compression algorithms and their associated overheads, the following parallel tests use the underlying Lustre file system. Table III on the following page shows the results for our write-only IOR benchmark, which does not include any computation. As can be seen, the measurement without compression (`off`) was used to set the baseline. The CPUs were moderately utilized with 116 %, while the HDDs were heavily utilized with around 87 %. The benchmark achieved a maximum throughput of 723 MB/s, which led to a runtime of 852 s. All ten I/O servers combined consumed 530 W during the benchmark run, which resulted in an energy consumption of 125 Wh. When using the `lz4` and `gzip-1` compression algorithms, the written data could be reduced significantly; `lz4` achieved a compression ratio of 1.92, while `gzip-1` even managed a ratio of 2.49. While `lz4` did not increase CPU utilization significantly (153 %), `gzip-1` produced a much higher computational overhead (402 %). In both cases, this reduced the HDD utilization and, at the same time, tremendously increased the throughput to 1,114 MB/s, which is the maximum our cluster can theoretically achieve. As a result, both benchmarks using compression finished much faster (553 s). However, in `gzip-1`'s case, the high power consumption of 848 W increased the energy consumption to 130 Wh. In `lz4`'s case, the moderate CPU and HDD utilization led to a slight increase in the power consumption (592 W). Combined with the shortened runtime, this resulted in significant savings regarding the energy consumption (91 Wh). Overall, `lz4` manages to reduce the total energy consumption to roughly 70 %.

TABLE IV. PARALLEL I/O BENCHMARK WITH SELECTED COMPRESSION ALGORITHMS (WRITE AND COMPUTATION).

Comp. Algorithm	Time (s)	Avg. Power (W)	Energy (Wh)	`idle3` Timer
off	3,212.15	456.03	406.90	Disabled
lz4	2,951.01	461.72	378.48	
gzip-1	2,950.54	503.63	412.77	
off	3,181.35	436.18	385.46	Enabled (8 s)
lz4	2,951.11	437.33	358.50	
gzip-1	2,950.60	484.55	397.14	

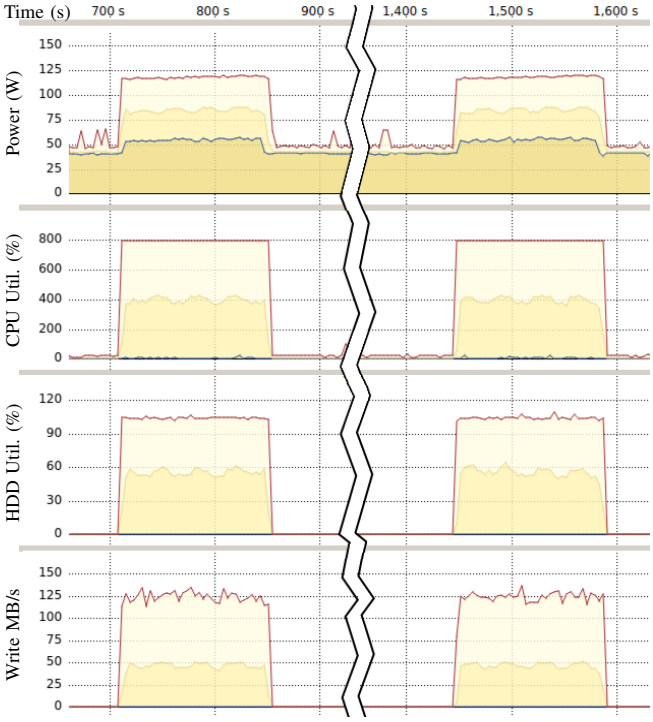

Figure 2. Vampir trace with the modified version of IOR benchmark using the `ondemand` governor and the `gzip-1` algorithm.

Next, we present our final experiments that are closer to real applications. Figure 2 on the current page shows a Vampir trace as recorded on one of the I/O storage servers while running our modified version of IOR benchmark. In this scenario the server alternates between idle and I/O state that correspond to the I/O activity and computation phase on the clients side.

The top of the figure shows the trace's timeline. As can be seen, the trace excerpt starts at around 670 s into the benchmark run and ends at approximately 1,620 s. The spikes on the left and right of the figure show I/O activity. To be able to focus on these important parts, we do not show most of the idle period in the middle. Our `pmlib` server collected data for the power consumption, CPU utilization, as well as HDD utilization and throughput. As can be seen, while the I/O server is being heavily utilized during the I/O phases, it is mostly idle during the computation phases. Since Vampir is not able to show all recorded values due to the lower screen resolution, it draws three lines representing the minimum, the average and the maximum for each of the values.

Table IV shows the results for our modified IOR benchmark, which simulates client-side computation; the rest of the configuration was not changed in any way. Due to the resulting idle phases on the I/O servers, we omit the HDD utilization and throughput. Additionally, we evaluated the IntelliPark (`idle3`)

 ISBN: 978-1-61208-332-2

TABLE III. PARALLEL I/O BENCHMARK WITH SELECTED COMPRESSION ALGORITHMS (WRITE-ONLY).

Comp. Algorithm	Comp. Ratio	Avg. CPU Util. (%)	Avg. Dev. Util. (%)	Throughput (MB/s)	Time (s)	Avg. Power (W)	Energy (Wh)
`off`	1.00	115.5	87.0	723.40	851.95	529.88	125.39
`lz4`	1.92	152.6	74.8	1114.03	553.29	592.07	90.99
`gzip-1`	2.49	402.3	55.8	1113.71	553.50	847.63	130.32

power saving feature supported by our Western Digital HDDs. As in the previous case, we repeated the benchmark using different compression algorithms; a run without compression (`off`) was used as the baseline.

The first set of results was conducted with the IntelliPark feature disabled. As can be seen, with 3,212 s, the benchmark now takes significantly longer to complete. While the average power consumption decreases to 456 W, the increase in run time leads to a total energy consumption of 407 Wh. For the `lz4` and `gzip-1` compression algorithms, the results look similar to the write-only test: Both algorithms help to significantly reduce the overall runtime by increasing the I/O throughput while, at the same time, increasing the average power consumption. In `gzip-1`'s case, the power consumption of 504 W leads to a higher energy consumption of 413 Wh. `lz4`, however, only slightly increases the power consumption to 462 W, which results in an energy consumption of 378 Wh. While these energy savings are not as significant as the ones on the write-only test, `lz4` still manages to reduce the total energy consumption to 93 %.

For the second set of tests, we enabled the IntelliPark mechanism and set the timeout to the default of 8 s. While IntelliPark helps minimizing power consumption of unutilized hard disk drives by intelligently parking the head of the disk, it also has its drawbacks: The disks may break sooner due to the high frequency of parking and unparking the disk head. Additionally, the warm up period required to resume operating from the park state adds extra latency to disk accesses. However, in HPC the hard drives are either fully utilized or idle for longer periods, which means that the IntelliPark technology fits perfectly for this use case. We measured the power consumption of a single server by enabling and disabling the IntelliPark technology on a single hard drive and observed a difference of approximately 1 W. The results of our experiment with IOR confirm that we can decrease the power and energy consumption by about 5 % across the board. Apart from that, we do not observe any other meaningful change.

IV. RELATED WORK

The energy efficiency of parallel I/O has been examined in several papers. Rong Ge et al. [18] provide an evaluation of parallel I/O energy efficiency using PVFS and NFS. They evaluated different I/O access patterns and block sizes, concluding that accessing larger data sets is more energy efficient. Moreover, they examined the potential benefits of leveraging DVFS in the compute nodes for I/O-intensive applications where lowering the clock frequency to half reduced energy consumption by around 25 % without compromising application performance. Our benchmarks are configured according to this paper. The authors in [19] propose Sera I/O, a portable and transparent middleware to improve the energy efficiency of I/O systems. Sera I/O creates a table from profiling information of micro-benchmarks. At run time, Sera I/O analyzes the I/O pattern of the application using the pre-created table and applies DVFS techniques accordingly. Takafumi et al. [20] propose an energy-aware I/O optimizer for check-pointing and restart on a NAND flash memory system. Based on profiles, the optimizer applies DVFS and controls the I/O processes, resulting in energy savings.

Mais Nijim et al. [21] integrate flash memory in the storage architecture to improve the energy efficiency of the disk subsystem. Using flash memory as a cache to keep the frequently used data sets, they are able to serve most of the I/O from the cache and put the majority of the disk drives into standby mode to save energy. The authors of DARAW [22] added write buffer disks to the system architecture to minimize energy consumption. Using a set of disks to temporarily store I/O accesses allows them to spin down storage disks for longer periods and reduce the power consumption. The impact of the disk speed on energy consumption is also presented in [23], [24]. In contrast to these papers, in our evaluation we investigate the impact of techniques that can be implemented in the disk controller to reduce energy consumption and do not require any modification in any other part of the system. However, above techniques can be combined with our proposal to improve even more the energy savings.

The authors of [25] explore the impact of compression in the storage servers by examining several file formats. However, our work is complementary to this and we are specially targeting HPC storage servers. Previous studies [26] have shown that scientific data can achieve high compression ratios depending on the used algorithm. Moreover, in comparison to our approach, which is server side compression, the authors in [27] propose compression on the client side, minimizing the amount of transmitted data between compute and storage nodes by sacrificing CPU cycles on the compute nodes.

Our previously conducted deduplication study for HPC data already showed great potential for data savings, allowing 20–30 % of redundant data to be eliminated on average [28]. However, deduplication can be very expensive in terms of memory overhead. The advantage of compression in comparison of deduplication is that is does not require any kind of lookup tables and is thus much cheaper to deploy because no additional hardware is required.

V. CONCLUSIONS AND FUTURE WORK

Our evaluation shows that data compression in HPC storage servers can be used to save energy and improve I/O performance. On the one hand, less HDDs are required to store the same amount of data due to the compression. On the other hand, it is also possible to achieve a higher throughput by storing more data in the same amount of time; this is especially relevant in I/O-intensive cases. These two advantages lead to lower procurement and operational costs.

However, it is important to carefully choose compression algorithms due to their inherent CPU overhead. Using expensive algorithms will increase power consumption and can theoretically even decrease performance. We have identified `lz4` as a suitable compression algorithm for scientific data and will use it for further analysis in the future. Lustre's ZFS backend provides a convenient possibility to leverage these compression algorithms without modifying or influencing client

 ISBN: 978-1-61208-332-2

applications. In more detail, using a real world data-set we demonstrated that we can achieve compression ratios of more than 1.5 without any significant increase of CPU utilization. Additionally, we observed a reduction in energy consumption of 30 % during the write phases and 7 % in write-intensive applications.

In the future, we plan to extend our evaluations by including additional factors such as energy-efficient SSDs, different CPU and network configurations as well as the overhead caused by decompression. Additionally, we want to take a closer look at other technologies, such as encryption, which are suitable to be handled by the storage servers. With current parallel distributed file systems, it is not possible to shut down individual storage nodes during operation; we will also investigate this possibility to enable additional energy savings.

ACKNOWLEDGMENTS

We would like to thank the German Helmholtz Association's LSDMA project, the EU's EXA2GREEN project (FP7-318793) and the EU's SCALUS (FP7-PEOPLE-ITN-2008-238808) project for their support. We also wish to thank Petra Nerge from our group for providing access to the data-set used for the experiments.

REFERENCES

[1] The Green500 Editors, "Green500," http://www.green500.org/, 12 2013, last accessed: 2013-12.

[2] R. Freitas, J. Slember, W. Sawdon, and L. Chiu, "GPFS scans 10 billion files in 43 minutes," IBM Advanced Storage Laborator. IBM Almaden Research Center. San Jose, CA, vol. 95120, 2011.

[3] M. L. Curry, H. L. Ward, G. Grider, J. Gemmill, J. Harris, and D. Martinez, "Power Use of Disk Subsystems in Supercomputers," in Proceedings of the Sixth Workshop on Parallel Data Storage, ser. PDSW '11. New York, NY, USA: ACM, 2011, pp. 49–54. [Online]. Available: http://doi.acm.org/10.1145/2159352.2159364

[4] Lawrence Livermore National Lab, "The IOR benchmark," http://sourceforge.net/projects/ior-sio, 2008, last accessed: 2013-12.

[5] J. Bonwick, M. Ahrens, V. Henson, M. Maybee, and M. Shellenbaum, "The Zettabyte File System," 2003.

[6] P. Schwan, "Lustre: Building a file system for 1000-node clusters," in Proceedings of the 2003 Linux Symposium, vol. 2003, 2003.

[7] D. Fellinger, "The State of the Lustre® File System and The Lustre Development Ecosystem," http://www.opensfs.org/wp-content/uploads/2013/04/LUG_2013_vFinal.pdf, 04 2013, last accessed: 2013-10.

[8] F. Schmuck and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," in Proceedings of the 1st USENIX Conference on File and Storage Technologies, ser. FAST '02. Berkeley, CA, USA: USENIX Association, 2002, pp. 231–244. [Online]. Available: http://dl.acm.org/citation.cfm?id=1083323.1083349

[9] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, "The new ext4 filesystem: current status and future plans," in Proceedings of the Linux Symposium, vol. 2. Citeseer, 2007, pp. 21–33.

[10] Lawrence Livermore National Lab, "Native ZFS on Linux," http://zfsonlinux.org, last accessed: 2013-12.

[11] R. Williams, "An extremely fast Ziv-Lempel data compression algorithm," in Data Compression Conference, 1991. DCC '91., Apr 1991, pp. 362–371.

[12] Yann Collet, "LZ4 Explained," http://fastcompression.blogspot.com/2011/05/lz4-explained.html, 2008, last accessed: 2013-12.

[13] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," Information Theory, IEEE Transactions on, vol. 23, no. 3, May 1977, pp. 337–343.

[14] S. Barrachina et al., "An Integrated Framework for Power-Performance Analysis of Parallel Scientific Workloads," 3rd Int. Conf. on Smart Grids, Green Communications and IT Energy-aware Technologies, 2013, pp. 114–119.

[15] J. H. Jungclaus et al., "Characteristics of the ocean simulations in the Max Planck Institute Ocean Model (MPIOM) the ocean component of the MPI-Earth system model," Journal of Advances in Modeling Earth Systems, vol. 5, no. 2, 2013, pp. 422–446. [Online]. Available: http://dx.doi.org/10.1002/jame.20023

[16] R. Rew and G. Davis, "Data Management: NetCDF: an Interface for Scientific Data Access," IEEE Computer Graphics and Applications, no. 10-4, 1990, pp. 76–82. [Online]. Available: http://dx.doi.org/10.1109/38.56302

[17] Western Digital, "Western Digital Green," http://www.wdc.com/en/products/products.aspx?id=780, 12 2013, last accessed: 2013-11.

[18] R. Ge, "Evaluating Parallel I/O Energy Efficiency," in Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing. IEEE Computer Society, 2010, pp. 213–220.

[19] R. Ge, X. Feng, and X.-H. Sun, "SERA-IO: Integrating Energy Consciousness into Parallel I/O Middleware," in Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, May 2012, pp. 204–211.

[20] T. Saito, K. Sato, H. Sato, and S. Matsuoka, "Energy-aware I/O Optimization for Checkpoint and Restart on a NAND Flash Memory System," in Proceedings of the 3rd Workshop on Fault-tolerance for HPC at Extreme Scale, ser. FTXS '13. New York, NY, USA: ACM, 2013, pp. 41–48. [Online]. Available: http://doi.acm.org/10.1145/2465813.2465822

[21] M. Nijim, A. Manzanares, X. Ruan, and X. Qin, "HYBUD: An Energy-Efficient Architecture for Hybrid Parallel Disk Systems," in Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on, Aug 2009, pp. 1–6.

[22] X. Ruan, A. Manzanares, S. Yin, Z. Zong, and X. Qin, "Performance Evaluation of Energy-Efficient Parallel I/O Systems with Write Buffer Disks," in Parallel Processing, 2009. ICPP '09. International Conference on, Sept 2009, pp. 164–171.

[23] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "DRPM: dynamic speed control for power management in server class disks," in Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on, June 2003, pp. 169–179.

[24] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes, "Hibernator: Helping Disk Arrays Sleep Through the Winter," vol. 39, no. 5. New York, NY, USA: ACM, Oct. 2005, pp. 177–190. [Online]. Available: http://doi.acm.org/10.1145/1095809.1095828

[25] R. Kothiyal, V. Tarasov, P. Sehgal, and E. Zadok, "Energy and Performance Evaluation of Lossless File Data Compression on Server Systems," in Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, ser. SYSTOR '09. New York, NY, USA: ACM, 2009, pp. 4:1–4:12. [Online]. Available: http://doi.acm.org/10.1145/1534530.1534536

[26] N. Hübbe and J. Kunkel, "Reducing the HPC-Datastorage Footprint with MAFISC - Multidimensional Adaptive Filtering Improved Scientific data Compression," in Computer Science - Research and Development, Executive Committee. Hamburg, Berlin, Heidelberg: Springer, 2012.

[27] B. Welton, D. Kimpe, J. Cope, C. M. Patrick, K. Iskra, and R. Ross, "Improving I/O Forwarding Throughput with Data Compression," in Proceedings of the 2011 IEEE International Conference on Cluster Computing, ser. CLUSTER '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 438–445. [Online]. Available: http://dx.doi.org/10.1109/CLUSTER.2011.80

[28] D. Meister, J. Kaiser, A. Brinkmann, T. Cortes, M. Kuhn, and J. Kunkel, "A Study on Data Deduplication in HPC Storage Systems," in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 7:1–7:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389006

 ISBN: 978-1-61208-332-2