Towards Green Scientific Data Compression Through High-Level I/O Interfaces

Yevhen Alforov*, Anastasiia Novikova†, Michael Kuhn†, Julian Kunkel‡, Thomas Ludwig*

*Deutsches Klimarechenzentrum GmbH, Hamburg, Germany, {alforov, ludwig}@dkrz.de
†Universität Hamburg, Hamburg, Germany, {michael.kuhn, novikova}@informatik.uni-hamburg.de
‡University of Reading, Reading, England, juliankunkel@googlemail.com

***Abstract*—Every HPC system today has to cope with a deluge of data generated by scientific applications, simulations or large-scale experiments. The upscaling of supercomputer systems and infrastructures, generally results in a dramatic increase of their energy consumption. In this paper, we argue that techniques like data compression can lead to significant gains in terms of power efficiency by reducing both network and storage requirements. To that end, we propose a novel methodology for achieving on-the-fly intelligent determination of energy efficient data reduction for a given data set by leveraging state-of-the-art compression algorithms and meta data at application-level I/O. We motivate our work by analyzing the energy and storage saving needs of real-life scientific HPC applications, and review the various compression techniques that can be applied. We find that the resulting data reduction can decrease the data volume transferred and stored by as much as 80 % in some cases, consequently leading to significant savings in storage and networking costs.**

***Index Terms*—HPC, I/O interfaces, data reduction, compression, energy consumption**

I. Introduction and Motivation

Energy saving became an imminent problem for many researchers and computer scientists. Their aim is to reduce energy consumption in supercomputers as far as possible without decreasing the runtime performance. Even though a number of approaches and mechanisms to reduce energy consumption in supercomputers have been suggested at the different levels of computing systems (CPU and GPU, storage disks, I/O, network, etc.), more and more HPC applications still produce enormous volumes of data sets that need more storage space to keep information saved. This also results in additional costs for energy because new hardware used in computing systems (for example, SSDs and HDDs) requires additional power. Therefore, storage, energy and overall costs increase for supercomputing facilities. For instance, for each PB of HDD storage space, the German Climate Computing Center (DKRZ) roughly has to pay investment costs of 100,000 € and annual electricity costs of 3,680 €; for its 54 PiB storage system, this amounts to almost 200,000 € per year for electricity alone.[1] These costs do not even include maintenance (approximately 15 % of storage costs) and tapes for long term archives (70 000 tape slots) [1].

Reduction of data volumes is a straight-forward *solution* to minimize energy consumption in storage systems. It can be achieved by leveraging different data reduction techniques like compression, transforms or deduplication. For storage systems, data reduction directly results in less storage hardware that has to be procured and operated. The main benefits of reduction techniques are storage capacity optimization, network bandwidth reduction, minimization of operational costs and, of course, energy saving.

In this connection, HPC users have a great interest in data reduction. Especially in those methods and algorithms that are the most appropriate for their data sets. Of course, the chosen data reduction must be able to provide the highest reduction ratio without decreasing the whole runtime performance or using additional resources including energy consumption. In our paper, we are focusing on scientific applications (typical users of HPC) where choosing reduction strategies with high performance and energy efficiency for the generated deluge of scientific data is a challenging task today. For these users, the choice of a compression algorithm is a technical decision that is difficult to make, since a well suited algorithm for one data set might be suboptimal for another data set or on another machine. Therefore, we aim to automatize the decision making process on behalf of the users.

In this work, we aim to define the methodology for intelligent selection of algorithms from a variety of state-of-the-art reduction techniques with an emphasis on their energy consumption. This methodology will be applied in the further development of our framework for scientific data reduction. Regarding this purpose, we make the following *contributions* in this paper:

- We first highlight in Section II main drawbacks and benefits of reduction techniques deployment on determined levels of data path through the common HPC I/O stack. After that we introduce our framework for scientific data compression though high-level I/O interfaces.
- In Section III, we introduce experimental setup for investigation of application-level data compression techniques.
- We present in Section IV the preliminary results obtained from a series of experiments and outline the main principles of our methodology for reduction method selection.
- After a concise review of related work in Section VI, we conclude in Section VII with a summary of our findings, and describe our future work on this topic.

[1]One PB of storage needs 3 kW of power and 1 kWh of energy costs 0.14 €.

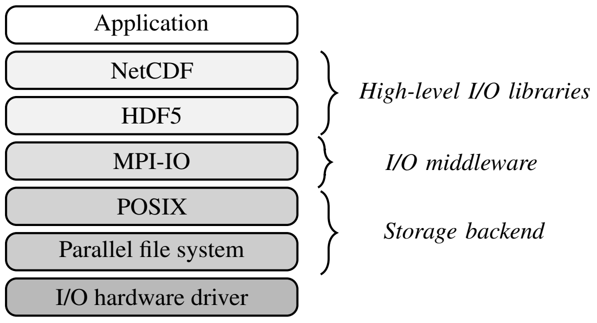

Figure 1. HPC I/O stack commonly used in life sciences

Table I
DRAWBACKS AND BENEFITS PROVIDED BY DEPLOYMENT OF REDUCTION TECHNIQUES IN HIGH AND LOW LEVELS OF I/O

	SYSTEM LEVEL	APPLICATION LEVEL
DRAWBACK	**Uncertainty** due to the lack of access to application-specific semantic information (e.g., data structures, important variables, etc.) only lossless reduction can be considered	**Clarity** insight into the code and requirements of applications is needed for tuning the performance of data reduction techniques
BENEFIT	**Transparency** no need to modify applications, even if they are very diverse or do not use a common I/O software stack	**Flexibility** semantic information is easily accessible, hence more reduction techniques can be leveraged (even for specific portions of data)

II. DATA REDUCTION THROUGH HIGH-LEVEL I/O

All data has to cross the full *I/O stack* during manipulation or retrieval. It consists of a file system, middleware and I/O libraries as depicted in Figure 1.

Two of the most popular and common high-level I/O interfaces in the scientific community to access data in both serial and parallel manner are *HDF5* (Hierarchical Data Format 5) [2] and *NetCDF* (Network Common Data Form) [3]. They allow HPC applications written in various programming languages (e.g., C, C++, Fortran, Python, etc.) to manipulate data and store it in a self-describing portable way by using multidimensional arrays [4]. Using self-describing data formats gives the opportunity to store a description of the data file layout as an additional meta information in the header part.

HDF5 and *NetCDF* perform I/O in a layered manner. The *NetCDF* programming interface delegates data storing to *HDF5*, and *HDF5* uses the I/O implementation of *MPI* (Message Passing Interface) [5]. *MPI* employs the I/O operations of the underlying parallel file system (backends for specific file systems or the more generic *POSIX* backend [6]). In the end, I/O is performed by the I/O driver. If the application performs data writing, it uses the high-level I/O library, and the data is going through the stack down until it is placed in the driver layer. A data read works in the opposite direction.

It must be also clearly noticed that *NetCDF* and *HDF5* interfaces provide parallel I/O. In this case it is necessary to use *HDF5*'s *MPI-IO* backend and have an underlying parallel file system which allows multiple processes to access a file. Otherwise, the I/O operations will be serialized.

On the basis of these considerations, it is possible to determine in general two main levels of the data path where data reduction mechanisms can be deployed. They are *system (low)* and *application (high)* levels. Depending on where in the *I/O stack* data reduction is employed, different benefits and drawbacks become apparent. As can be seen from Table I, data reduction usage on higher levels of *HPC I/O stack* is advantageous. Unlike low layers, it is possible to access and exploit additional meta information stored in the header part of files like data types. Different HPC applications (e.g. for climate change and weather forecasting, bioinformatics, etc.) are using a common I/O stack (Figure 1), making it easier to employ application-level data reduction for them. Thus, data reduction leverage at the application level is possible to be fine-tuned by taking application requirements and meta data into account. Techniques which can be deployed in a way that is transparent for users of these applications are deduplication [7], compression and transforms [8], [9].

With regards to the problem mentioned before and bearing in mind the data path, we are developing Scientific Compression Library (SCIL)[2] - a framework for data reduction though high-level I/O interfaces (see Figure 2). Its main goal is providing the most appropriate data reduction strategy for a given scientific data set on the basis of semantic information and performance of algorithms. It currently supports different lossless and lossy techniques.

SCIL is a meta-compressor that aims to exploit knowledge on the application level [10], [11]; it decouples the selection of various error quantities and the expected performance behavior from the selection of the algorithm. For example, a newer and better algorithm could be selected by the library without change in the application code once it becomes available. The library should ultimately pick a suitable chain of algorithms yielding the user's requirements. Initially, this is done based on the capabilities of the algorithms but the ongoing work is a preliminary stage for the design of an improved algorithm selector that could benefit from energy-aware selection of the algorithms.

An application can either use the NetCDF4, HDF5 or the SCIL C interface, directly. As the majority of climate models is stored in NetCDF file format, but NetCDF doesn't have compression feature itself, and uses for this purpose HDF5 compression filters, a new one was developed. Using this filter and some others we are launching our tests.

Apart from this, SCIL is rich of useful features. It provides tools to:

- Create random patterns or add noise.
- Compress CSV or NetCDF3 files.
- Compress/Decompress and plot the results.

In this paper, the main focus of our research will be on energy consumption of data compression techniques widely used to safe storage space.

[2]The current version of library under LGPL license: https://github.com/JulianKunkel/scil

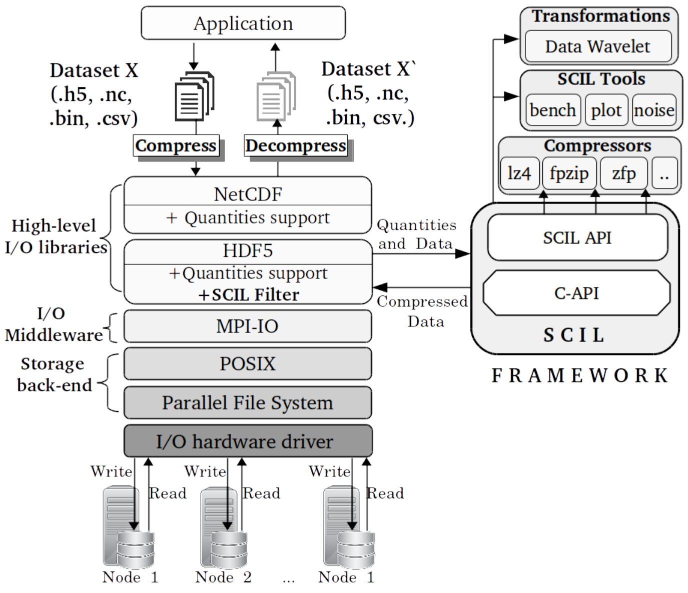

Figure 2. General architecture of Scientific Compression Library SCIL

III. Experimental Setup

Apart from the points described above, a question still to be examined in detail are there any dependencies between performance of compressor (including its energy efficiency) and the structure of the data. If yes, then it will be extremely useful to know how such dependencies can be employed in selection of power-aware reduction techniques for a given data set when its meta data is available at the hand.

In the next sections, we will try to find an answer through evaluation of HDF5 filters at the high-level I/O and their distinct performance and energy consumption characteristics. Algorithms like LZ4 [12] and Zstd [13] are fast and provide high throughputs. However, their compression ratios can be lower compared to slower algorithms that consume more energy (such as LZMA [14]).

Environment setup. In order to investigate the performance of data compression at the application level, we used a cluster which operates with the parallel distributed file system Lustre. The maximum throughput was limited to roughly 110 MB/s due to using only one node (outfitted with two 2.80 GHz quad-core Intel Xeon X5560 processors and 12 GB of RAM). The experiments were conducted by repacking the source files located on the same node and storing them in a local file system. For the energy consumption measurements, the *ArduPower* [15] wattmeter was used. It is designed to simultaneously measure the DC power consumption of different components (e.g., motherboard, CPU, GPU, disks) inside computing systems even at very large scale. *ArduPower* provides 16 channels to monitor the power consumption with a sampling rate varying from 480 to 5,880 Hz.

Metrics. The main metrics in which we were interested are the compression ratio (CR)[3] to quantify the data reduction, runtime of each algorithm to see how slow or fast is it, average CPU utilization and consumed energy.

[3]We define compression ratio as $CR = \frac{\text{size original}}{\text{size compressed}}$.

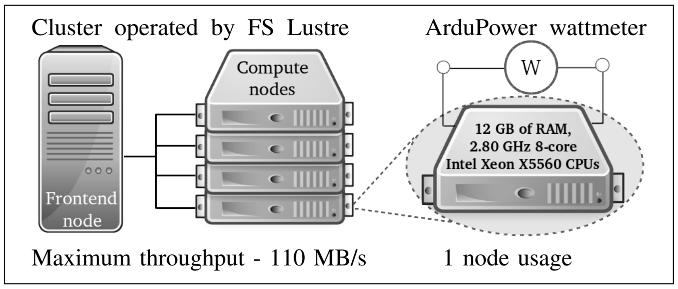
Cluster operated by FS Lustre
ArduPower wattmeter
W
Compute nodes
12 GB of RAM, 2.80 GHz 8-core Intel Xeon X5560 CPUs
Frontend node
Maximum throughput - 110 MB/s
1 node usage

Figure 3. Experimental setup

Dataset and workload. For data reduction techniques evaluation at the high-level, two data sets with roughly the same size have been chosen and one smaller data was taken for additional experiments:

- 17 GB data set of 3-dimensional ecosystem model for the North Sea *ECOHAM* [16], [17], [18] (from Climate Science)
- 14 GB data set of tomography experiments from *PETRA III*'s PCO 4000 detector [19] (from High Energy Physics)
- 4 GB data set of *ECHAM* atmospheric model [20] (from Climate science)

Evaluated techniques. To perform the reduction of data sets, different HDF5 compression filters have been leveraged. In experimental evaluation we compared the following algorithms:

- `off`: No filtering is applied. This represents the baseline.
- `blosc`: The Blosc meta-compressor using LZ4 compressor. Additionally, Blosc's shuffle pre-conditioner was used.
- `mafisc`: The MAFISC compression algorithm that uses several pre-conditioners and LZMA compressor [21].
- `lz4`: The LZ4 compression algorithm using its default acceleration factor.
- `zstd`: The Zstd compression algorithm using its default aggression parameter. The `zstd-11` and `zstd-22` variants represent Zstd with aggression parameters of 11 and 22, respectively.
- `scil`: HDF5 plugin applying LZ4 compression algorithm with some pre-conditioners to each variable in a data set.

IV. Preliminary results and discussion

The obtained results of compression ratio and consumed energy for the *ECOHAM* and *PETRA III* data sets are plotted in Figure 4. Average CPU utilization for both data sets are plotted in Figure 5 and Figure 6 accordingly. Figure 7 shows that overall the runtimes vary wildly, even though both data sets have roughly the same size (17 GB for *ECOHAM* and 14 GB for *PETRA III*). This gives rise to the view that it is related to the different data set structures: While the *ECOHAM* data set contains more than 300 4-dimensional variables of double precision floats, the *PETRA III* data set contains around ten

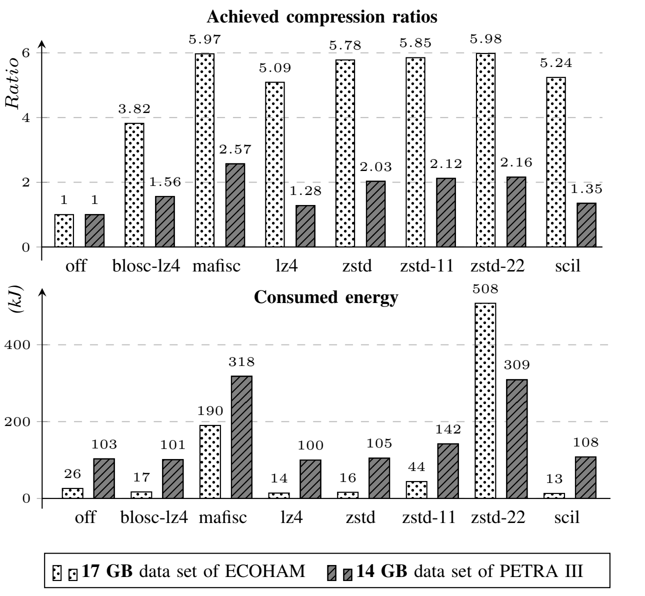

Figure 4. Average compression ratios and energy consumption depending on the HDF5 filter used for compression

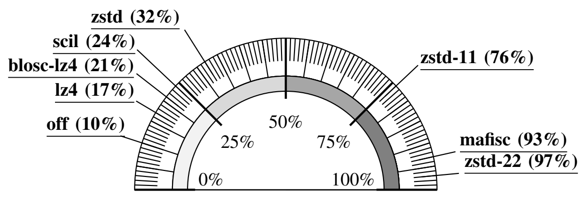

Figure 5. Average CPU utilization with **ECOHAM** data set

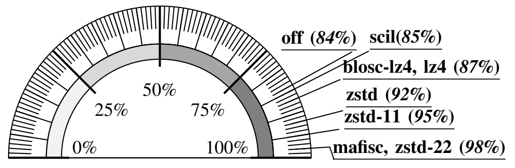

Figure 6. Average CPU utilization with **PETRA** data set

3-dimensional variables of 16-bit integers.[4] Consequently, the pre-conditioners were tuned for the data sets where appropriate. Blosc was set up to use a 8-byte data size for *ECOHAM* and a 2-byte data size for *PETRA III*. In addition, *ECOHAM* contains many (repeating) fill values, which explains higher compression ratios in comparison to the *PETRA III* data set.

Blosc's pre-conditioners shuffle the data in such a way that compressors should achieve higher compression ratios. In case of *PETRA III*, this approach works well because `blosc-lz4` achieves a compression ratio of 1.56 while `lz4` only achieves a compression ratio of 1.28. Due to a very similar runtime and CPU utilization (and thus, energy consumption), the additional pre-conditioning provides benefits. However, in case of *ECOHAM*, Blosc's pre-conditioner significantly reduces the achievable compression ratio from 5.09 for `lz4` to 3.82 for `blosc-lz4`. In addition to that, it also increases the runtime and CPU utilization and, therefore, consumes 50 % more energy.

For both data sets, Zstd achieves significantly better compression ratios than LZ4 even with its lowest compression level. The increases in runtime and energy consumption are negligible for the *PETRA III* data set. Zstd has also been tested with levels 11 and 22 besides its lowest compression level. As can be seen from Figure 4 and Figure 7, level 11 improves both the achieved compression ratios and runtimes moderately. Level 22 gives even higher compression ratios but with significantly increased runtime and, thus, energy consumption.

MAFISC achieves the highest compression ratios in both cases due to its advanced pre-conditioners. However, this achievement is expensive because runtime and CPU utilization are increased significantly. Regarding energy consumption, it needs 8x and 3.5x the energy compared to running without compression for *ECOHAM* and *PETRA III*, respectively.

Concluding, SCIL was run with only one algorithm - LZ4 for all variables. Comparing to LZ4, its compression ratio is slightly better. This happened because more pre-conditioners were applied to algorithm in SCIL filter. However, as for the average energy consumption, also for the average CPU utilization, the values are almost the same as in the case of LZ4. Thus, we can state that SCIL itself makes almost no influence on the performance of chosen algorithm.

Additionally we have tested *ECHAM* data which is smaller (4 GB only) than *ECOHAM* and *PETRA III* data (see Table II). ECHAM data set contains 135 double and float variables. Average CPU utilization is 99 % for all the algorithms, excluding SCIL, where CPU utilization equals 71 %. The blosc-lz4 saves more energy for this dataset than others.

Data set	Filter	Comp.	Runtime	CPU	Energy
ECOHAM	off	1.00	06:58	10 %	24 J
	blosc-lz4	3.82	04:48	21 %	16,5 J
	mafisc	5.97	49:25	93 %	189,4 J
	lz4	5.09	04:05	17 %	14,2 J
	zstd	5.78	04:35	32 %	16,2 J
	zstd-11	5.85	11:47	76 %	44,3 J
	zstd-22	5.98	2:11:50	97 %	508 J
	scil	5.24	03:47	24 %	15 J
PETRA III	off	1.00	27:28	84 %	103 J
	blosc-lz4	1.56	26:58	87 %	101 J
	mafisc	2.57	1:23:04	97 %	318 J
	lz4	1.28	26:57	87 %	100 J
	zstd	2.03	28:12	92 %	105.4 J
	zstd-11	2.12	38:08	95 %	142 J
	zstd-22	2.16	1:20:49	98 %	309 J
	scil	1.35	29:03	85 %	108 J
ECHAM	off	1.00	2:59	99 %	11 kJ
	blosc-lz4	1.95	3:05	99 %	11,7 kJ
	mafisc	2.36	20:12	99 %	77 kJ
	lz4	1.5	3:03	99 %	11,7 kJ
	zstd	1.8	3:17	99 %	12,6 kJ
	zstd-11	1.82	4:50	99 %	18,2 kJ
	zstd-22	1.91	34:30	99 %	132 kJ
	scil	1.5	4:58	71 %	18 kJ

Table II
ECOHAM, PETRA III AND ECHAM DATA SETS COMPRESSED USING DIFFERENT HDF5 FILTERS

[4]This runtime difference only occurs when using `h5repack` but not when using the simpler `h5copy`.

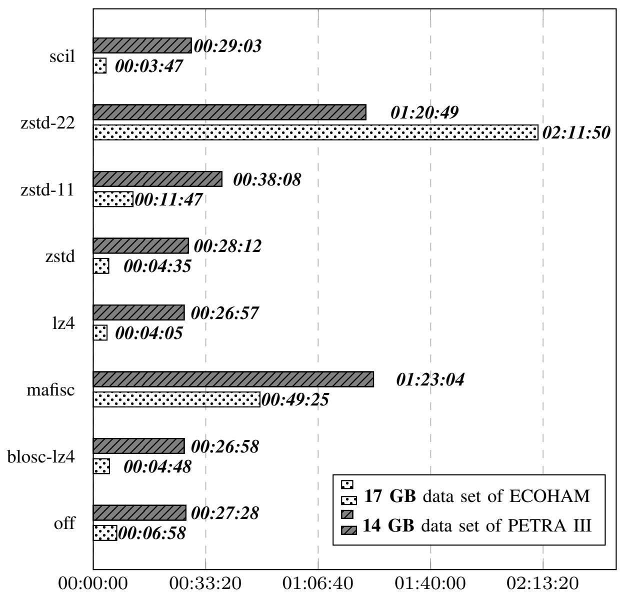

Figure 7. Runtime *T(HH:MM:SS)* of evaluated compressors

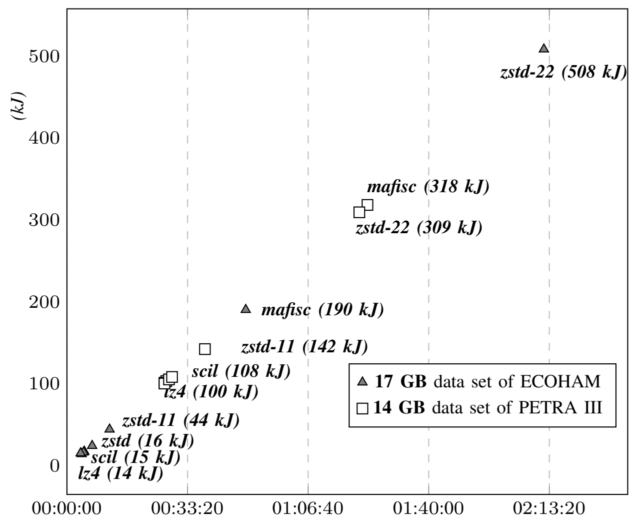

Figure 8. Consumed Energy vs. Runtime of compressors

V. Outcome and methodology

After collecting all the metrics for each of applied compressors, now it is possible to look at the dependencies between metrics. It is possible to compare results for all combinations (CR vs. Time, CR vs. Energy, CR vs. CPU, Time vs. CPU, Time vs. Energy, CPU vs. Energy), however most of the graphs show chaotic results, except of one, which is depicted on Figure 8 where energy `E` is directly proportional to time `T`. This can be easily explained by the formula: `Energy=Power*Time`. Therefore, the less time an algorithm needs for compression, the less energy it consumes. This can be taken as a base line for a power-aware compression selection. To this end, we can now observe that the most efficient by energy, time and CPU usage for both data sets is `lz4` compressor with acceptable compression ratio (CR). Hence, this algorithm can be employed for data compression by default.

As can be seen from the evaluation results, there is a trade-off between compression ratio and energy consumption. With defined accuracy for these metrics (when small increases may be negligible) more appropriate algorithms can be mapped for each data set by all the parameters: `ZSTD-11` for *ECOHAM* with CR=5.85 and `ZSTD` for *PETRA III* with CR=2.03. In this way, if the input data has approximately the same data structure as ECOHAM for instance, SCIL can leverage `ZSTD-11`. Thus, based on the results of performed compression for given datasets we can describe basic steps at the beginning of reduction strategy selection:

- `LZ4` is set up by default. This option takes into account all the metrics.
- If only CR is important for a user then `MAFISC` can be applied to the data set.
- If CR is important for a user with taking into account the energy consumption, then the data structure must be checked and compared to the obtained results (here user can establish the accuracy and semantic meta information should be available):
  – Regarding the characteristics of input data set one of the following compressors can be used (`BLOSC-LZ4`, `ZSTD`, `ZSTD-11`).

All these observations are of great significance for the methodology of compression algorithm selection. `MAFISC` can be employed when only the ratio matters, and `LZ4` or `ZSTD` when runtime, energy or CPU load are also important. Thus, if the given data set at the input has a similar structure as *ECOHAM* or *PETRA III*, we already have a defined reduction strategies. In order to make the strategy selection more precise, evaluation of various compressors on different data must be performed with collection of various metrics. For taking an intelligent decision on what compression can be applied to the given data it is possible to leverage a trained decision tree from machine learning which can provide to the user these different options in algorithm selection based on the meta data and performance of algorithms. Design and implementation of such a decision support unit is an ongoing work and is out of scope in this paper.

VI. Related work

The impact of compression on I/O throughput is studied in [22]. The results show that the achievable throughput is highly dependent on the chosen algorithm and data because slow algorithms or incompressible data can decrease throughput significantly.One way to compensate for this drawback is to implement these algorithms in hardware [23]. Authors of [24] have implemented gzip on FPGAs using OpenCL. Their implementation offers a throughput of 3 GB/s in comparison to 300 MB/s for a highly-optimized CPU implementation. Moreover, performance-per-watt ratio from the FPGA implementation is twelve times better than the one from the CPU implementation. However, not all accelerator-based implementations are faster than CPU-based implementations. In [25], the authors have implemented bzip2 on an NVIDIA GTX 460 and found that their implementation is more than two times slower than the original. One of the reasons for this is the fact that all data has to be transferred via the PCIe bus to the GPU and back. Thus, compressing data already on the compute nodes can be much more beneficial than porting compression methods to the accelerators.

It is furthermore important to foresee which reduction method will produce the best results. For example, [26] presents a decision algorithm for MapReduce users to decide whether to use compression or not. The key factor here is a data compressibility which determines the cases when compression is worthwhile. With the introduced algorithm, the MapReduce framework becomes a more powerful tool for data centers. After studying the impact of compression on performance and energy efficiency for MapReduce data-intensive workloads, the authors of this work reported that compression provides up to 60 % energy savings for some jobs. Therefore, prediction is crucial.

In [27], the authors present a compression algorithm for arrays of 64-bit floating-point values. It predicts the next value in the array based on previous values and uses XOR to encode the difference between the predicted and actual value. For this, data analysis is very useful and also helps to determine appropriate reduction and to avoid negative information loss. In [28], the authors examine properties of every individual variable of a data set produced by Community Earth System Model (CESM) [29] and suggest to apply compression on a per variable basis. Further, they target to implement an automated tool for appropriate lossy compressor identification which, however, focuses only on CESM's workflow [30]. Suchlike approach has been considered in [31] when semantic data from high-level I/O can be taken into account, which is unfortunately problematic for the underlying file system.

However, to the best of our knowledge, almost none of the previous works focus on data reduction at the high levels of the HPC I/O stack with usage of additional meta information for tuning the chosen technique or for identifying an appropriate reduction strategy. Moreover, energy consumption is not considered as a rule. It is precisely this gap we aim to address in our work.

VII. Conclusion and Future Work

Our preliminary results show that the amount of data which can be saved after using reduction techniques like compression heavily depends on the structure of data. Pre-conditioners such as byte- and bit-shuffling might work well for one data set, but they might worsen data savings for others. Moreover, one can observe that there is a delicate trade-off between compression ratio and energy consumption. Data reduction algorithms like MAFISC can provide high compression ratios but at the same time increase energy costs and reduce throughput. In addition, different approaches are appropriate depending on the use case. Files for archival can be compressed with slower algorithms while parallel I/O should be handled as fast as possible.

In the future, we plan to extend our evaluation of compression algorithms on various data sets in order to build more strategies and to discover what exactly influences on energy consumption during reduction process. Besides this, we aim also to experiment with additional application-specific data reduction techniques. Using semantic information available at the application level about data enables other techniques such as lossy compression or other transforms to reduce their precision. Moreover, deploying data reduction at the high levels of the HPC I/O stack allows their fine-tuning to take application requirements into account (different compression algorithms could be used for different types of variables). For instance, lossy compression could be used for less important variables. We also plan to experiment with techniques that are complex and/or expensive to employ at the system level such as deduplication. Using it only on a per-variable basis could help significantly reduce the costs in terms of memory that is required for the deduplication tables.

Obtained preliminary results in our paper now can be taken into account during ongoing work when implementing the decision unit for intelligent algorithms selection in a SCIL framework for scientific data reduction. It will identify appropriate data reduction strategies for HPC users based on relevant semantic information and performance metrics. For our purposes, collected metrics of `lz4`, `mafisc` and `zstd` algorithms will be employed at the start.

Acknowledgment

This work is part of the "BigStorage: Storage-based Convergence between HPC and Cloud to handle Big Data" project, funded by the European Union under the Marie Skłodowska-Curie Actions (H2020-MSCA-ITN-2014-642963). This work was supported in part by the German Research Foundation (DFG) through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) (GZ: LU 1353/11-1). We would also like to thank André Rothkirch from DESY for providing us with access to parts of their data.

References

[1] N. Hübbe and J. M. Kunkel, "Reducing the HPC-datastorage footprint with MAFISC - Multidimensional Adaptive Filtering Improved Scientific data Compression," *Computer Science - R&D*, vol. 28, no. 2-3, pp. 231–239, 2013. [Online]. Available: https://doi.org/10.1007/s00450-012-0222-4

[2] HDF Group, "HDF5 home page," https://support.hdfgroup.org/HDF5/, 2017.

[3] UCAR, "NetCDF," http://www.unidata.ucar.edu/software/netcdf/, 2017.

[4] C. Bartz, K. Chasapis, M. Kuhn, P. Nerge, and T. Ludwig, "A Best Practice Analysis of HDF5 and NetCDF-4 Using Lustre," in *High Performance Computing*, ser. Lecture Notes in Computer Science, J. M. Kunkel and T. Ludwig, Eds., no. 9137. Switzerland: Springer International Publishing, 06 2015, pp. 274–281.

[5] H. Taki and G. Utard, "MPI-IO on a Parallel File System for Cluster of Workstations," in *1st International Workshop on Cluster Computing (IWCC '99), 2-3 December 1999, Melbourne, Australia*, 1999, pp. 150–157. [Online]. Available: http://dx.doi.org/10.1109/IWCC.1999.810820

[6] The Open Group, "POSIX.1 FAQ," http://www.opengroup.org/austin/papers/posix_faq.html, 2011.

[7] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A Comprehensive Study of the Past, Present, and Future of Data Deduplication," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1681–1710, 2016. [Online]. Available: http://dx.doi.org/10.1109/JPROC.2016.2571298

[8] J. Schlachter, V. Camus, and C. C. Enz, "Design of energy-efficient discrete cosine transform using pruned arithmetic circuits," in *IEEE International Symposium on Circuits and Systems, ISCAS 2016, Montréal, QC, Canada, May 22-25, 2016*, 2016, pp. 341–344. [Online]. Available: https://doi.org/10.1109/ISCAS.2016.7527240

[9] H. J. Nussbaumer, *Fast Fourier transform and convolution algorithms*. Springer Science & Business Media, 2012, vol. 2.

[10] J. Kunkel, A. Novikova, E. Betke, and A. Schaare, "Toward decoupling the selection of compression algorithms from quality constraints," in *High Performance Computing*, J. M. Kunkel, R. Yokota, M. Taufer, and J. Shalf, Eds. Cham: Springer International Publishing, 2017, pp. 3–14.

[11] J. Kunkel, A. Novikova, and E. Betke, "Towards Decoupling the Selection of Compression Algorithms from Quality Constraints – an Investigation of Lossy Compression Efficiency," *Supercomputing Frontiers and Innovations*, pp. 17–33, 12 2017. [Online]. Available: http://superfri.org/superfri/article/view/149

[12] Yann Collet, "lz4," http://lz4.github.io/lz4/, 01 2017.

[13] Facebook, "Zstandard," http://facebook.github.io/zstd/, 2018.

[14] Z. B. Tariq, N. Arshad, and M. Nabeel, "Enhanced LZMA and BZIP2 for improved energy data compression," in *SMARTGREENS 2015 - Proceedings of the 4th International Conference on Smart Cities and Green ICT Systems, Lisbon, Portugal, 20-22 May, 2015.*, 2015, pp. 256–263. [Online]. Available: https://doi.org/10.5220/0005454202560263

[15] M. F. Dolz, M. R. Heidari, M. Kuhn, T. Ludwig, and G. Fabregat, "ArduPower: A low-cost wattmeter to improve energy efficiency of HPC applications," in *IGSC*. IEEE, 2015, pp. 1–8.

[16] U. H. Institute of Oceanography, "ECOHAM," https://wiki.zmaw.de/ifm/ECOHAM, 2015.

[17] F. Große, N. Greenwood, M. Kreus, H. Lenhart, D. Machoczek, J. Pätsch, L. A. Salt, and H. Thomas, "Looking beyond stratification: a model-based analysis of the biological drivers of oxygen depletion in the North Sea," *Biogeosciences Discussions*, pp. 2511–2535, 2015. [Online]. Available: http://www.biogeosciences-discuss.net/12/12543/2015/bgd-12-12543-2015.pdf

[18] I. Lorkowski, J. Pätsch, A. Moll, and W. Kühn, "Interannual variability of carbon fluxes in the North Sea from 1970 to 2006–Competing effects of abiotic and biotic drivers on the gas-exchange of CO 2," *Estuarine, Coastal and Shelf Science*, vol. 100, pp. 38–57, 2012.

[19] DESY, "PETRA III," http://petra3.desy.de/index_eng.html, 2015.

[20] E. Roeckner, G. Bäuml, L. Bonaventura, R. Brokopf, M. Esch, M. Giorgetta, S. Hagemann, I. Kirchner, L. Kornblueh, E. Manzini *et al.*, "The atmospheric general circulation model echam 5. part i: Model description," 2003.

[21] N. Hübbe, "MAFISC," https://wr.informatik.uni-hamburg.de/research/projects/icomex/mafisc, 2016.

[22] B. Welton, D. Kimpe, J. Cope, C. M. Patrick, K. Iskra, and R. B. Ross, "Improving I/O Forwarding Throughput with Data Compression," in *2011 IEEE International Conference on Cluster Computing (CLUSTER), Austin, TX, USA, September 26-30, 2011*, 2011, pp. 438–445. [Online]. Available: http://dx.doi.org/10.1109/CLUSTER.2011.80

[23] L. Benini, D. Bruni, A. Macii, and E. Macii, "Hardware-Assisted Data Compression for Energy Minimization in Systems with Embedded Processors," in *2002 Design, Automation and Test in Europe Conference and Exposition (DATE 2002), 4-8 March 2002, Paris, France*, 2002, pp. 449–453. [Online]. Available: http://dx.doi.org/10.1109/DATE.2002.998312

[24] M. S. Abdelfattah, A. Hagiescu, and D. Singh, "Gzip on a chip: high performance lossless data compression on FPGAs using OpenCL," in *Proceedings of the International Workshop on OpenCL, IWOCL 2013 & 2014, May 13-14, 2013, Georgia Tech, Atlanta, GA, USA / Bristol, UK, May 12-13, 2014*, 2014, pp. 4:1–4:9. [Online]. Available: http://doi.acm.org/10.1145/2664666.2664670

[25] R. A. Patel, Y. Zhang, J. Mak, A. Davidson, and J. D. Owens, "Parallel lossless data compression on the GPU," in *2012 Innovative Parallel Computing (InPar)*, May 2012, pp. 1–9.

[26] Y. Chen, A. Ganapathi, and R. H. Katz, "To compress or not to compress - compute vs. IO tradeoffs for mapreduce energy efficiency," in *Proceedings of the 1st ACM SIGCOMM Workshop on Green Networking 2010, New Delhi, India, August 30, 2010*, 2010, pp. 23–28. [Online]. Available: http://doi.acm.org/10.1145/1851290.1851296

[27] P. Ratanaworabhan, J. Ke, and M. Burtscher, "Fast Lossless Compression of Scientific Floating-Point Data," in *2006 Data Compression Conference (DCC 2006), 28-30 March 2006, Snowbird, UT, USA*, 2006, pp. 133–142. [Online]. Available: http://dx.doi.org/10.1109/DCC.2006.35

[28] A. H. Baker, D. M. Hammerling, S. A. Mickelson, H. Xu, M. B. Stolpe, P. Naveau, B. Sanderson, I. Ebert-Uphoff, S. Samarasinghe, F. De Simone *et al.*, "Evaluating lossy data compression on climate simulation data within a large ensemble," *Geoscientific Model Development*, vol. 9, no. 12, p. 4381, 2016.

[29] J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, J.-F. Lamarque, W. G. Large, D. Lawrence, K. Lindsay *et al.*, "The community earth system model: a framework for collaborative research," *Bulletin of the American Meteorological Society*, vol. 94, no. 9, pp. 1339–1360, 2013.

[30] A. H. Baker, H. Xu, D. M. Hammerling, S. Li, and J. P. Clyne, "Toward a multi-method approach: Lossy data compression for climate simulation data," in *High Performance Computing - ISC High Performance 2017 International Workshops, DRBSD, ExaComm, HCPM, HPC-IODC, IWOPH, IXPUG, P^3MA, VHPC, Visualization at Scale, WOPSSS, Frankfurt, Germany, June 18-22, 2017, Revised Selected Papers*, 2017, pp. 30–42. [Online]. Available: https://doi.org/10.1007/978-3-319-67630-2_3

[31] M. Kuhn, "A Semantics-Aware I/O Interface for High Performance Computing," in *Supercomputing*, ser. Lecture Notes in Computer Science, J. M. Kunkel, T. Ludwig, and H. W. Meuer, Eds., no. 7905. Berlin, Heidelberg: Springer, 06 2013, pp. 408–421.