Analyzing Data Properties using Statistical Sampling Techniques – Illustrated on Scientific File Formats and Compression Features

Julian M. Kunkel

Deutsches Klimarechenzentrum
Bundesstraße 45a
20146 Hamburg
kunkel@dkrz.de

Abstract. Understanding the characteristics of data stored in data centers helps computer scientists in identifying the most suitable storage infrastructure to deal with these workloads. For example, knowing the relevance of file formats allows optimizing the relevant formats but also helps in a procurement to define benchmarks that cover these formats.
Existing studies that investigate performance improvements and techniques for data reduction such as deduplication and compression operate on a small set of data. Some of those studies claim the selected data is representative and scale their result to the scale of the data center. One hurdle of running novel schemes on the complete data is the vast amount of data stored and, thus, the resources required to analyze the complete data set. Even if this would be feasible, the costs for running many of those experiments must be justified.
This paper investigates stochastic sampling methods to compute and analyze quantities of interest on file numbers but also on the occupied storage space. It will be demonstrated that on our production system, scanning 1% of files and data volume is sufficient to deduct conclusions. This speeds up the analysis process and reduces costs of such studies significantly. The **contributions** of this paper are: 1) the systematic investigation of the inherent analysis error when operating only on a subset of data, 2) the demonstration of methods that help future studies to mitigate this error, 3) the illustration of the approach on a study for scientific file types and compression for a data center.

Keywords: Scientific Data, Compression, Analyzing Data Properties

1 Introduction

Understanding the characteristics of data stored in the data center helps computer scientists optimizing the storage. The quantities of interest could cover proportions, i.e. the percentage of files with a certain property, or means of certain metrics such as achievable read/write performance, compression speed and ratio. For example, knowing the relevance of file formats may shift the effort

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-46079-6_10

towards the most represented formats. When 80% of the capacity is utilized by NetCDF4 files, performance analysis and optimization should target this file format first. Understanding the achievable compression ratio of available compression schemes helps in choosing not only the best one for user-specific compression but also for file system compression as provided by ZFS.

Assessing the benefit of any modification in production systems requires to either deploy those system-wide, or to evaluate its potential in advance on a small data set. For data centers with storage capacity of tens of Petabytes and file numbers in the order of hundreds of millions, it is not feasible to change the production system frequently. Thus, prototyping and evaluation on small scale is necessary. However, as a scientist involved in such prototyping, how can we estimate the benefit from small scale to large scale? Usually, this is done by picking a representative or relevant set of data for the evaluation and assuming those values can be scaled up to the full system.

In the literature, studies can be found that investigate compression ratio, deduplication factor or improve performance of scientific middleware. Due to the long running time to apply any improvement on large amounts of data, many studies assume the benefit measured on a small data sample can be transferred to the scale on the data center. However, usually these studies do not pay attention if the data set is actually representative, with other words, they do not take into account the fraction of the workload that can actually benefit from the advancement. In statistics, the big field of sampling theory addresses this issue. Due to the law of large numbers, there are methods to draw instances appropriately and deduce properties from the sample set to the population with high confidence. However, this process is non-trivial and a research discipline in statistics by itself [1], [2].

This paper investigates statistical sampling to estimate file properties on the scale of data centers using small data sets and statistical simulation. The computation time used for this project was 517 core days. With 24 cores per node, a complete system scan of DKRZ's system would have needed about 475 node days which would have cost at least about 4000 €[1] – while not revealing additional insight. Instead with 1% of scanned files or capacity, similar results are achievable.

The paper is structured as follows: an excerpt to related studies analyzing scientific data is given in Section 2. The method to create test data for this research is described in Section 3. To show the variability of data and importance of proper sampling, the data is explored in Section 4. It also describes some interesting properties of DKRZ's scientific data. The strategies to pick appropriate samples for studies analyzing data by file count and by occupied space is given in Section 5. Finally, the paper is concluded.

[1] The value is an estimate based on the TCO of the system for 5 years. It is conservative and does not include secondary costs such as jitter introduced to other models by the caused I/O.

2 State of the art

Existing research that analyzes properties of scientific data can be classified into performance analysis, compression ratio and data deduplication. Effort that investigates and optimizes performance usually picks a certain workload to demonstrate that the new approach is superior than existing strategies. A few studies analytically analyze typical patterns and optimize for a large range of access patterns. An example is the study in [3], which analyzes the access pattern for several workloads and discusses general implications. Research for optimization techniques, as far as known to the author, do not check how many people actually benefit from these optimizations and the implications on system level.

In the field of compression, many studies have been conducted on pre-selected workloads, for example, see [4–7]. Some of those studies are used to estimate the benefit of the compression on the data center level, for example, Hübbe et al. investigate the cost-benefit for long-term archival. Similarly in [6], the compression ratio is investigated. However, in this case the selected data is a particular volume from a small system.

Modern file systems such as BTRFS and ZFS offer compression on system-side [8]. It is also considered to embed compression into storage devices such as SSDs [9] and evaluate it for OLTP workloads. In [10], Jin et.al investigate the benefit for compressing and de-duplicating data for virtual machines. They created a diverse pool of 52 virtual images and analyze the impact.

The de-duplication study for data centers in [11], analyzes a larger fraction of scientific data on different sites, but due to the long run-time did not manage to analyze the full data set. When looking at all these studies, one may ask the question how would those techniques behave on a full system?

3 Sampling of Test Data

To assess and demonstrate the impact of statistical sampling, firstly, a subset of data of DKRZ's supercomputer Mistral is scanned and relevant data properties about data compression and scientific file types are extracted. The goal of the following strategy was not to gain a completely representative sample, since this is to be developed within this paper. Since the global Lustre file system hosts about 320 million files and 12 Petabytes of space is occupied, only a subset is scanned: 380k (0.12%) accounting for an (aggregated size) of 53.1 TiB of data (0.44%). Note that the mean file size scanned is about 145 MiB but on our single file system it is 38.8 MiB. The discrepancy is due to the fact that project directories contain usually larger files compared to home directories that were not scanned. To prevent data loss and ensure data privacy, the scanning process is performed using a regular user account and, thus, it cannot access all files. There are still 58 million files and 160 out of 270 project directories accessible.

The scanning process used as baseline in the paper works as follows:

1. Run a parallel scan for accessible files of each project directory independently using `find`; store them in individual file lists.

2. Select 10.000 files from each project directory randomly (or all, if less files exist in a project directory) and merge them into a single file list.
3. Create a random permutation of the file list and partition the result into one process list for each thread that shall be used.
4. Distribute the threads across different nodes, each thread processes its fixed file list sequentially and writes an individual output file.
5. After 300k files have been scanned, the threads are prematurely terminated and the resulting data from all threads is ingested into a SQLlite database.

The strategy increases the likelihood, that a representative sample of files is chosen from accessible projects[2]. It is to be expected that projects differ in their file characteristics as they may use different scientific simulations and analysis workflows. From perspective of statisticians, this process creates a simple random sample [1] but incorporates a systematic stratified sampling approach to balance between projects. The limitations of this sampling strategy to investigate properties based on occupied file size, will be shown later.

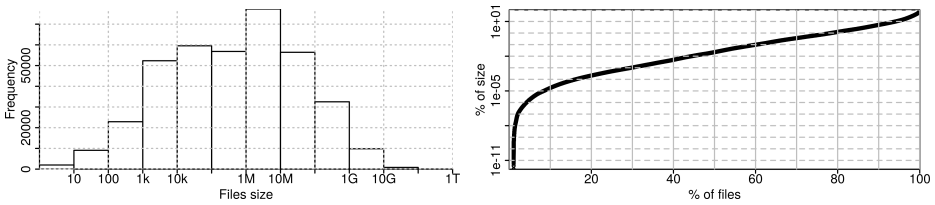
Processing of the threads: a few threads are started concurrently on the system, to prevent overwhelming the file system and exploit compute time on interactive nodes that is not utilized by users. Since the file list is created only once, but the threads are executed over the course of several weeks, the processing ignores non-existing files. A thread iterates through the file list, for each file it first copies it to a temporary location – this prevents concurrent file modifications, then it runs: 1) the CDO [12] command to identify the scientific file type, 2) the `file` command to identify file types checking the file header, and 3) each compressor under investigation (LZMA, GZIP, BZIP2, ZIP) in compression and de-compression mode. The `time` command is used to capture runtime information of each step. To assess compression speed, user time is extracted – this covers the actual computing time and ignores system and I/O times as well as competing jobs.

4 Exploring Analyzed Data

In this section, we investigate several quantities of interest on the complete data set, they are computed either on file count, i.e. each file is weighted identically, or by weighting each file with its occupied size. This section will show that both types of analyses lead to different results.

In Figure 1, the distribution of file sizes is shown. Fig 1a) shows a histogram with logarithmic file sizes. In Fig. 1b) the relation between file size and file count is illustrated; to construct the figure, files have been sorted by size in ascending order and then the cumulative sum is computed. While the histogram suggests similarities between size distribution and a normal distribution, this is due to the logarithmic x-axis. In the cumulative view, it can be seen that aggregated

[2] Obviously, if those 160 projects are not representative, deducing properties for the full data is not valid. Still the introduced analysis and approaches are correct. The number of 10k files was choosen as it would ensure to scan at most 0.5% of the files.



(a) Histogram **(b)** Cumulative file sizes (y-axis in log scale)

Fig. 1. Distribution of file sizes

20% of files consume one millionth of storage space and 90% still consume less than 10% space (these files are below 100 MiB in size). If a study takes small files as representatives, those fail to represent the storage capacity. Similar large file fail to represent the typical (small) file that must be handled by the storage.

4.1 Scientific File Formats

The usage of file formats is shown in Figure 2. The figure shows the relative relevance in terms of occupied space and number of files of each file format. About 60% of the number of files and 47% of aggregated file size is non-scientific and cannot be resolved with the CDO tool. The dominant scientific formats are NetCDF3, GRIB1 and NetCDF2. The `file` command cannot identify and distinguish scientific formats as reliable as CDO but can shed light over the distribution of the 60%. Looking at its output, the 60% of capacity seems to be dominated by TAR (7%) and GZIP compressed files (5%)[3] – it classifies 43% of capacity as "data" and 40% as NetCDF (no version information provided). Looking at the proportions in terms of file count, roughly 30% are classified as data, 30% as text (e.g., code), 24% as NetCDF files, 4% as HDF5 and 3.5% as images. Other file types are negligible.

[3] From the GZIP files, the extension tar.gz is observed on 9% of files, representing 53% of GZIP data overall size. Thus most GZIP files are also TAR files.

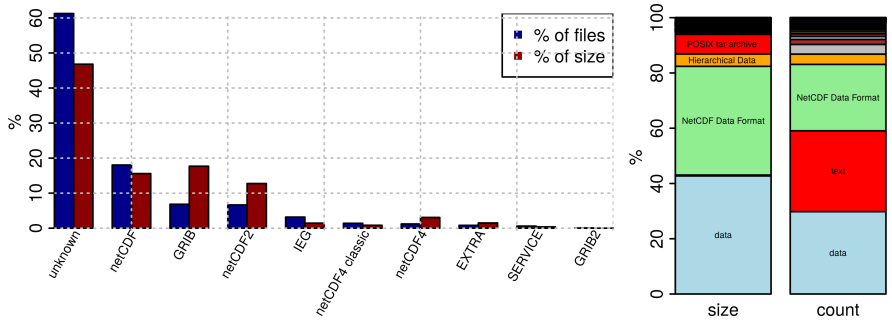
4.2 Compression Ratio

To evaluate compression ratio, files with a size below 4 KiB (about 15% of all files) are not taken into consideration, as compressing them is not expected to be beneficial because of the additional header and file system block size. As metric we will use the inverse of the compression ratio that is the fraction the compressed file occupies in comparison to the original, e.g., after compressing a file might be down to 10% of its original size. For simplicity, we label this metric as **compressed %**; it is computed as $c_{\%}(f) = \frac{\text{compressed size}(f)}{\text{file size}(f)}$. The mean compression can be determined as the arithmetic mean ratio all files c_{files} (Eq. 1), i.e. averaging it by file number; or, by file size, i.e. considering the total space saved c_{size} (Eq. 2).

$$c_{files} = \frac{\sum_{f=1}^{\text{files}} \frac{\text{comp.size}(f)}{\text{file size}(f)}}{\text{file count}} \tag{1}$$

$$c_{size} = \frac{\sum_f \text{compr.size}(f)}{\sum_f \text{file size}(f)} \tag{2}$$

In Figure 3, both, the mean relative compression by count and by size is shown for each compression scheme and file format. The column "all" shows the reduction when compressing the full data set. In average, the compressors except LZMA achieve similar results – but a compression scheme can be slightly better than another on individual file sets. LZMA, performs much better on all data except GRIB2. While overall the computation by file and by size is similar, there are differences in details. For example, in average LZMA packs each file down to 40% of its size, but in terms of overall storage space only 50% is saved. Thus, smaller files typically compress better with LZMA than the larger files. This is mainly caused by the fraction of non-scientific file formats (the unknowns) that compress much worse in terms of capacity. For example, GZIP compressed files occupy a significant portion of space.



(a) CDO types **(b)** File types

Fig. 2. Relative usage of file formats determined using CDO and `file`.

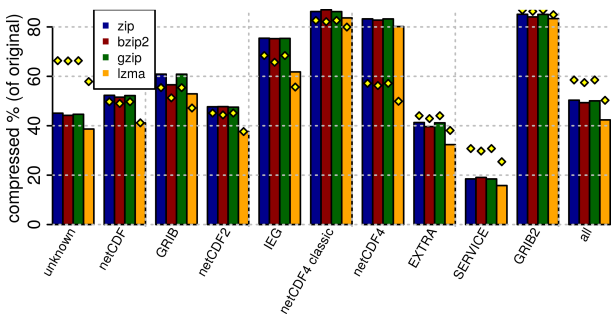
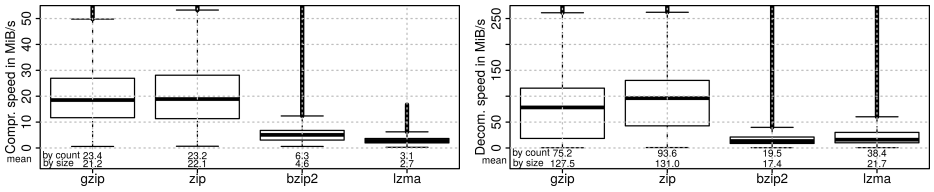


Fig. 3. Arithmetic mean compression of the full data set for each scientific file format computed on file number. The column "all" shows the mean values for the whole data set. Yellow diamonds show compress % computed by file size.

4.3 Performance of Compression Algorithms

Fast compression and decompression speed and, thus, required computation time, is crucial for embedding compression schemes into existing workflows without delaying them. The performance spectrum across the files is shown in Figure 4. The boxplots show the median as black line; the box indicates the limits of quartile 1 and quartile 3, whiskers can go up to 1.5 interquartile range. Many outliers with faster performance are not visualized. While on average, BZIP2 does not perform much better than the other schemes on the data set, it is much slower. LZMA achieved the best compression ratio but is much slower than gzip. Again, the compression ratio by file count differs from the mean speed computed by size. For LZMA, the mean is 38.4 and 21.7 MiB/s, computed by count and by size, respectively. Thus the difference in decompression is roughly 50% as bigger files need more time to compress and decompress.



(a) Compression **(b)** Decompression

Fig. 4. Boxplots showing compression/decompression speed per file, the arithmetic means are shown as text under the plot.

4.4 Variance Between Different Projects

It is expected that scientific projects exhibit different data characteristics, for example, they may utilize file formats differently. This makes it difficult to pick a random sample from just a few projects. To investigate this issue, all projects for which more than 1,000 files have been scanned were analyzed individually. A few characteristics across the projects are shown in the boxplot in Figure 5: The file count indicates how many files have been scanned for each project, how

much storage is occupied by the files, the compression % of LZMA, proportion of NetCDF and GRIB files in terms of the projects overall file count and occupied size. It can be seen that while between 2,000 and 3,000 files have been scanned due to random file selection, other metrics vary significantly. For example, some projects have nearly 100% of NetCDF files while others use other formats. Thus, it is very important to sample properly across projects.

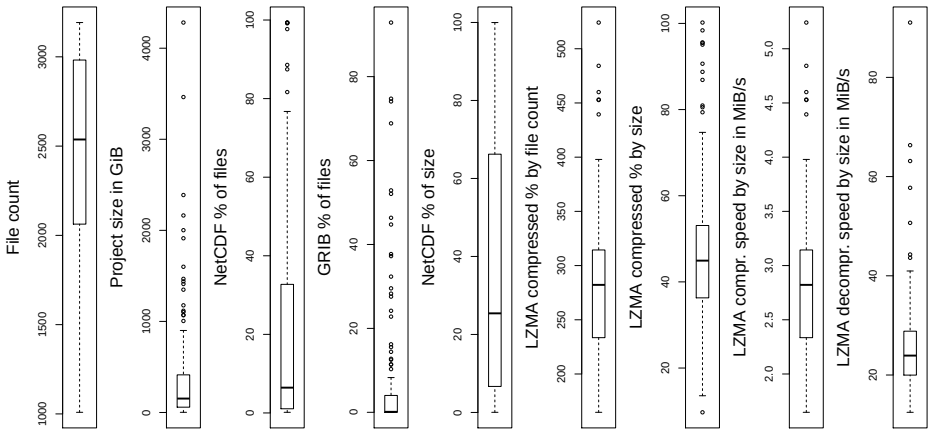


Fig. 5. Several quantities of interest, this barchart covers 125 individual projects. Each point represents the arithmetic mean value computed on one project.

5 Stochastic Sampling of Data

The way the quantities of interest are computed are either by file count, i.e. we predict properties of the population based on individual files, or by weighting the individual files with their size. From the perspective of statistics, we analyze variables for quantities that are continuous values or proportions, i.e. the fraction of samples for which a certain property holds. To select the number of observations that allows inference about the population, statistics knows methods for determining sample size. These methods require that we can make certain assumptions about the distribution of values such as normally distributed data. To determine the number of samples needed, the tolerable error between prediction and real value must be chosen, it depends on the used analysis method and distribution of values. Usually the error is defined by the size of an interval with the predicted value as center in which is extremely likely (e.g., 95%), that the true value resides. The interval is called confidence interval and the probability is the confidence level.

For estimating proportions there are easy approaches – that work regardless of probability distribution: With Cochran's sample size formula, to achieve an error bound of ±5% and ±1%, roughly 400 and 10000 samples are needed,

respectively [1]. Note that the sample number does not increase even for large population sizes.

Estimating a continuous variable, e.g., the arithmetic mean compression ratio or performance, is more complex as sampling ratio is based on the expected distribution of values. But we do not know it a-priori, and not necessarily the property we are looking for is normally distributed. All these methods have in common that they determine the mean value, i.e., the proportion in all files and not weight them by occupied storage size. As the size is a-priori is unbounded, it is not unlikely that file size distribution follows heavy-tailed distributions, increasing the analysis of sample size determinination [13, 14]. The detailed analysis is out of scope of this paper, but we will show that the means are converging for typical use cases. The methods to obtain a representative sample are as follows.

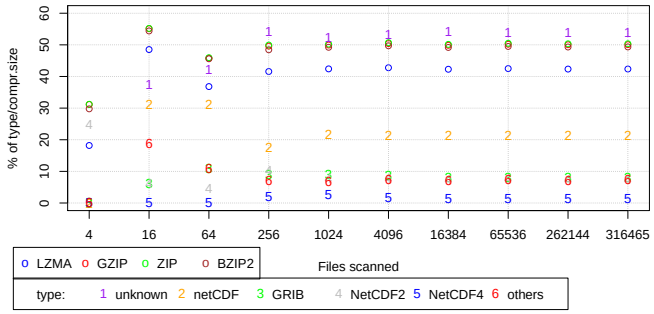
Sampling method to compute by file count. When computing the proportion or the mean of a variable for files, a strategy is to enumerate all files on the storage system and then create a simple random sample, i.e., choose a number of files for which the property is computed. For proportion variables, Cochran's sample size formula is applicable and the number of files can be easily deducted.

Sampling method to compute by file size. Estimating values and weighting them based on file size requires to enumerate all files and determine their size, then pick a random sample from the file list based on the probability defined by filesize/totalsize. Draws from the list must be done with replacement, i.e., we never remove any picked file. Once all chosen files are determined, the quantities of interest are computed once for each unique file. Then, each time we have chosen a file, we add our quantity of interest without weighting the file size, e.g., the arithmetic mean can be computed just across all samples. Thus large files are more likely to be picked but each time their property is accounted identically as for small files.

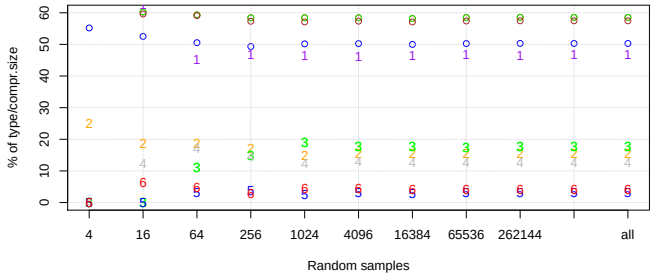
Evaluation for mean compressed file size and proportions of file types. To demonstrate this approach, a simulation has been done by drawing a variable number of samples from the data. The result is shown in Figure 6. It can be seen that this quickly converges to the correct mean value of the full data. With 4096 samples, that are slightly more than 1% of files, the value is close to the correct mean.

Robustness. To illustrate the stability of the approach, the simulation is repeated 100 times and a boxplot is rendered with the deviations. Naturally, the repeats of a robust method should have little variance and converge towards the correct mean value. The result for the proportion of GRIB files are given as an example but the results for all variables behave similar. In Figure 7, it can be clearly seen that the error becomes smaller but Cochran's approximation is yield.

The sampling strategy to compute quantities on file size is shown in Figure 8b). Similarly, to the correct method for sampling by file count it converges quickly. However, if we would simply use a file scanner to compute the metrics on size but it would choose files randomly without considering file sizes, we



(a) Correct sampling method to compute mean by count



(b) Correct sampling method to compute mean by size

Fig. 6. Evaluating various metrics (proportions and compressed %) for an increasing number of samples. Only one simulation is done for each count.

would achieve highly unstable results (Figure 8a). Indeed the error margin with even one fifth of all files (64k) is comparable to the correct sampling strategy with only 1024 samples. Thus, it is vital to apply the right sampling method, and, therefore, the initial approach used to gather the test data as described in Section 3 is suboptimal.

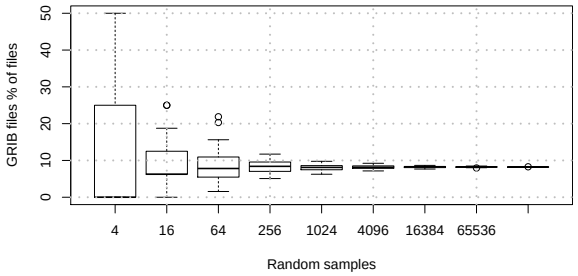
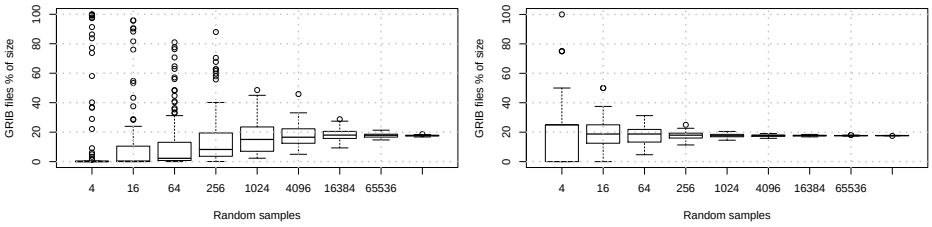


Fig. 7. Simulation of sampling by file count to compute compr.% by file count.



(a) By file count (this is suboptimal!) **(b)** Sampling (With replacement) with a probability proportional to size

Fig. 8. Simulation of sampling to compute proportions of types by size.

6 Summary & Conclusions

In this paper, sampling techniques from statistics are applied to estimate data properties. These techniques are demonstrated to be useful approximate the proportions of scientific file types, the compressed % and speed. The paper also highlighted some interesting data properties of DKRZ's data. It has been demonstrated that a random file scanner is not efficient to estimate quantities that are computed on file size. Instead, sampling with replacement and a probability equal to the proportion of file size leads to stable results. Tools using such techniques can estimate properties of data robust without the need to analyze the huge data volumes of data centers. We will be working on such tools to evaluate the benefit of optimization strategies.

Acknowledgements

I thank Charlotte Jentzsch for the fruitful discussions.

References

1. Kotrlik, J., Higgins, C.: Organizational Research: Determining Appropriate Sample Size in Survey Research Appropriate Sample Size in Survey Research. Information technology, learning, and performance journal **19**(1) (2001) 43
2. Newcombe, R.G.: Two-sided confidence intervals for the single proportion: comparison of seven methods. Statistics in medicine **17**(8) (1998) 857–872
3. Lofstead, J., Polte, M., Gibson, G., Klasky, S., Schwan, K., Oldfield, R., Wolf, M., Liu, Q.: Six Degrees of Scientific Data: Reading Patterns for Extreme Scale Science IO. In: Proceedings of the 20th international symposium on High performance distributed computing, ACM (2011) 49–60
4. Lakshminarasimhan, S., Shah, N., Ethier, S., Ku, S.H., Chang, C.S., Klasky, S., Latham, R., Ross, R., Samatova, N.F.: ISABELA for Effective in Situ Compression of Scientific Data. Concurrency and Computation: Practice and Experience **25**(4) (2013) 524–540
5. Kunkel, J., Kuhn, M., Ludwig, T.: Exascale Storage Systems – An Analytical Study of Expenses. Supercomputing Frontiers and Innovations (06 2014) 116–134
6. Kuhn, M., Chasapis, K., Dolz, M., Ludwig, T.: Compression By Default – Reducing Total Cost of Ownership of Storage Systems (06 2014)
7. Hübbe, N., Kunkel, J.: Reducing the HPC-Datastorage Footprint with MAFISC – Multidimensional Adaptive Filtering Improved Scientific data Compression. Computer Science - Research and Development (05 2013) 231–239
8. Legesse, S.D.: Performance Evaluation of File Systems Compression Features. Master's thesis, University of Oslo (2014)
9. Zuck, A., Toledo, S., Sotnikov, D., Harnik, D.: Compression and SSDs: Where and How? In: 2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads (INFLOW 14), Broomfield, CO, USENIX Association (October 2014)
10. Jin, K., Miller, E.L.: The Effectiveness of Deduplication on Virtual Machine Disk Images. In: Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, ACM (2009) 7
11. Meister, D., Kaiser, J., Brinkmann, A., Kuhn, M., Kunkel, J., Cortes, T.: A Study on Data Deduplication in HPC Storage Systems. In: Proceedings of the ACM/IEEE Conference on High Performance Computing (SC), IEEE Computer Society (11 2012)
12. Schulzweida, U., Kornblueh, L., Quast, R.: CDO User's guide: Climate Data Operators Version 1.6. 1 (2006)
13. Resnick, S.I.: Heavy-tail phenomena: probabilistic and statistical modeling. Springer Science & Business Media (2007)
14. Tursunalieva, A., Silvapulle, P.: Estimation of Confidence Intervals for the Mean of Heavy Tailed Loss Distributions: A Comparative Study Using a Simulation Method. (2009)