Toward Decoupling the Selection of Compression Algorithms from Quality Constraints

Extended Abstract

Julian Kunkel
Deutsches Klimarechenzentrum
Bundestrasse 45a
Hamburg, Germany 20146

Anastasiia Novikova
Universität Hamburg
Bundestrasse 45a
Hamburg, Germany 20146

Eugen Betke
Deutsches Klimarechenzentrum
Bundestrasse 45a
Hamburg, Germany 20146

ABSTRACT

With the Scientific Compression Library (SCIL), we are developing a meta-compressor that allows users to set various quantities that define the acceptable error and the expected performance behavior. The library then chooses the appropriate chain of algorithms to yield the users requirements. This approach is a crucial step towards a scientifically safe use of much-needed lossy data compression, because it disentangles the tasks of determining scientific ground characteristics of tolerable noise, from the task of determining an optimal compression strategy given target noise levels and constraints. Without changing applications, it allows these codes to utilize future algorithms once they are integrated into the library.

KEYWORDS

Compression, climate science, data reduction

ACM Reference format:
Julian Kunkel, Anastasiia Novikova, and Eugen Betke. 2017. Toward Decoupling the Selection of Compression Algorithms from Quality Constraints. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 2 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Compression offers a chance to increase the provided storage space or to provide virtually the same storage space but with less costs. Analysis has shown that with proper preconditioning and algorithm, a compression factor of roughly 2.5:1 can be achieved with lossless compression, i.e., without loss of information / precision [4]. However, the throughput of compressing data with the best available option is rather low (2 MiB/s per core). By using the statistical method in [7] to estimate the actual compression factor that can be achieved on our system, we saw that LZ4fast yield a compression ratio[1] of 0.68 but with a throughput of more than 2 GiB/s on a single core. Therefore, on our system it even outperforms algorithms for optimizing memory utilization such as BLOSC.

In the AIMES project we develop libraries and methods to utilize lossy compression. The SCIL library[2] provides a rich set of user quantities to define from, e.g., HDF5. Once set, the library shall ensure that the defined data quality meets all criteria. Its plugin architecture utilizes existing algorithms and aims to select the best algorithm depending on the user qualities and the data properties.

Contributions of this poster are: 1) Introduction of quantities for acceptable tolerance and performance; 2) Analysis of (lossless) compression for climate data (and two new algorithms).

2 RELATED WORK

Lossless algorithms: The LZ77[8] algorithm is dictionary-based and uses a "sliding window". The concept behind this algorithm is simple: It scans uncompressed data for two largest windows containing the same data and replaces the second occurrence with a pointer to the first window. DEFLATE[6] is a variation of LZ77 and uses Huffman coding[3]. GZIP[2] is a popular lossless algorithm based on DEFLATE.

Lossy algorithms for floating point data: FPZIP[9] was primarily designed for lossless compression of floating point data. It also supports lossy compression and allows the user to specify the bit precision. The error-bounded compression of ZFP[9] for up to 3 dimensional data is accurate within machine epsilon in lossless mode. The dimensionality is insufficient for the climate scientific data. SZ[1] is a newer and effective HPC data compression method. Its compression ratio is at least 2x better than the second-best solution of ZFP. In [4], compression results for the analysis of typical climate data were presented. Within that work, the lossless compression scheme MAFISC with pre-conditioners was introduced; its compression ratio was compared to that of standard compression tools reducing data 10% more than the second best algorithm. In [5], two lossy compression algorithms (GRIB2, APAX) were evaluated regarding to loss of data precision, compression ratio, and processing time on synthetic and climate dataset. These two algorithms have equivalent compression ratios and depending on the dataset APAX signal quality exceeds GRIB2 and vice versa.

3 DESIGN

The main goal of the compression library SCIL is to provide a framework to compress structured and unstructured data using the best available (lossy) compression algorithms. SCIL offers a user interface for defining the tolerable loss of accuracy and expected performance as various quantities. It supports various data types. An application can either use the NetCDF4, HDF5 or the SCIL C interface, directly. SCIL acts as a meta-compressor providing various back-ends such as the existing algorithms: LZ4, ZFP, FPZIP, and SZ. Based on the defined quantities, their values and the characteristics

[1] We define compression ratio as $r = \frac{\text{size compressed}}{\text{size original}}$; inverse is the compr. factor.

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

Conference'17, July 2017, Washington, DC, USA
2017. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

of the data to compress, the appropriate compression algorithm is chosen [3].

Supported Quantities. The tolerable error on lossy compression and the expected performance behavior can be defined. Quantities define the properties of the residual error ($r = v - \hat{v}$):

- **absolute tolerance**: compressed value $\hat{v} = v \pm$ abstol
- **relative tolerance**: $\hat{v} = v \cdot (1 \pm reltol)$, $reltol < 1$
- **relative error finest tolerance**: used together with rel tolerance; absolute tolerable error for small v's. If relfinest $> |v \cdot (1 \pm reltol)|$, then $\hat{v} = v \pm$ relfinest
- **significant digits**: number of significant decimal digits
- **significant bits**: number of significant digits in bits

Additional, the performance behavior can be defined for both compression and decompression (on the same system). The value can be defined according to: 1) absolute throughput in MiB or GiB; or 2) relative to network or storage speed. Thus, SCIL must estimate the compression rates for the data. The system's performance must be trained for each system using machine learning.

Algorithms. The development of the two algorithms sigbits and abstol has been guided by the definition of the user quantities. Both algorithms aim to pack the number of required bits as tightly as possible into the data buffer. We also consider these algorithms useful baselines when comparing any other algorithm. 1) Abstol: This algorithm guarantees the defined absolute tolerance. 2) Sigbits: This algorithm preserves the user-defined number of precision bits from the floating point data.

4 EVALUATION

In the evaluation, we utilize SCIL to compress the data with various algorithms. In all cases, we manually select the algorithm. The test system is an Intel(R) Core(TM) i3-2120 with 4 cores @ 3.30GHz.

A pool of (single precision floating point) data is created from several synthetic patterns generated by SCIL's pattern library such as constant, random, linear steps, polynomial, sinusoidal or by the OpenSimplex [10] algorithm. Additionally, we utilize the Hurricane Isabel data produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR, and the U.S. National Science Foundation (NSF) [4].

4.1 Experiments

In each test, only one thread of the system is used for the compression/decompression. Each configuration is run 3 times measuring compression and decompression time and compression ratio.

1) Compression Ratio Depending On Tolerance: Firstly, we investigate the compression factor depending on the tolerance level for all scientific data files varying the precision for the algorithms ZFP, SZ, Sigbits and Abstol. For the precision bits, when preserving three mantissa bits, roughly 9:1 could be achieved with sigbits+LZ4. Note that in roughly half the cases, ZFP could not hold the required precision, as it defines the number of bits for the output and not in terms of guaranteed precision[5]. 2) Fixed Absolute Tolerance: To analyze throughput and compression ratio across variables, we selected an absolute tolerance of 1% of the maximum value. Synthetic random patterns serve as baseline to understand the benefit of the lossy compression; we provide the means for 5 different random patterns. 3) Fixed Precision Bits: Similarly to our previous experiment, we now aim to preserve 9 precision bits for the mantissa. It can be seen that Sigbits+LZ4 outperforms ZFP mostly, although ZFP does typically not hold the defined tolerance.

[3] The implementation for the automatic algorithm selection is ongoing effort and not the focus of this paper. SCIL will utilize a model for performance and compression ratio for the different algorithms, data properties and user settings.

[4] http://www.vets.ucar.edu/vg/isabeldata/readme.html

[5] Even when we added the number of bits necessary for encoding the mantissa to ZFP.

5 SUMMARY

This poster introduces the concepts for the scientific compression library (SCIL) and compares novel algorithms implemented with the state-of-the-art compressors. It shows that these algorithms can compete with ZFP/SZ when setting the absolute tolerance or precision bits. In cases with steady data, SZ compresses better than abstol. Since SCIL aims to choose the best algorithm, it ultimately should be able to take benefit of both algorithms. Ongoing work is the development of a single algorithm honoring all quantities and the automatic chooser for the best algorithm.

ACKNOWLEDGEMENTS

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).

REFERENCES

[1] Sheng Di and Franck Cappello. 2015. Fast Error-bounded Lossy HPC Data Compression with SZ. (2015).
[2] Jean-Loup Gailly and Mark Adler. [n. d.]. GZIP algorithm. http://www.gzip.org/algorithm.txt. ([n. d.]). [Online; accessed 04-10-2016].
[3] David A. Huffman. [n. d.]. Huffman coding. AMethodfortheConstructionofMinimum-RedundancyCodes. ([n. d.]). [Online; accessed 04-10-2016].
[4] Nathanel Hübbe and Julian Kunkel. 2013. Reducing the HPC-Datastorage Footprint with MAFISC – Multidimensional Adaptive Filtering Improved Scientific data Compression. *Computer Science - Research and Development* (05 2013), 231–239. http://link.springer.com/article/10.1007/s00450-012-0222-4
[5] Nathanael Hübbe, Al Wegener, Julian Kunkel, Yi Ling, and Thomas Ludwig. 2013. Evaluating Lossy Compression on Climate Data. In *Supercomputing (Lecture Notes in Computer Science)*, Julian Martin Kunkel, Thomas Ludwig, and Hans Werner Meuer (Eds.). Springer, Berlin, Heidelberg, 343–356. https://doi.org/10.1007/978-3-642-38750-0_26
[6] Phil Katz. [n. d.]. DEFLATE algorithm. https://en.wikipedia.org/wiki/DEFLATE. ([n. d.]).
[7] Julian Kunkel. 2016. Analyzing Data Properties using Statistical Sampling Techniques – Illustrated on Scientific File Formats and Compression Features. In *High Performance Computing: ISC High Performance 2016 International Workshops, ExaComm, E-MuCoCoS, HPC-IODC, IXPUG, IWOPH, P3MA, VHPC, WOPSSS (Lecture Notes in Computer Science)*, Michaela Taufer, Bernd Mohr, and Julian Kunkel (Eds.). Springer, 130–141. https://doi.org/chapter/10.1007/978-3-319-46079-6_10
[8] Abraham Lempel and Jacob Ziv. [n. d.]. LZ77. https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossless/lz77/example.htm. ([n. d.]). [Online; accessed 04-10-2016].
[9] Peter Lindstrom and Martin Isenburg. 2006. Fast and Efficient Compression of Floating-Point Data. *IEEE Transactions on Visualization and Computer Graphics, 12(5):1245-1250* (2006).
[10] Kurt Spencer. [n. d.]. OpenSimplex Noise in Java. https://gist.github.com/KdotJPG/b1270127455a94ac5d19. ([n. d.]). [Online; accessed 05-02-2017].