Int J Parallel Prog
DOI 10.1007/s10766-013-0287-0

Data Reduction Analysis for Climate Data Sets

Songbin Liu · Xiaomeng Huang · Haohuan Fu · Guangwen Yang · Zhenya Song

Received: 16 April 2013 / Accepted: 4 October 2013

Abstract Global climate modeling not only requires computation capabilities, but also brings tough challenges for data storage systems. The input and output data sets generally require hundreds or even thousands of terabytes storage. Therefore, storage reduction methods, such as content deduplication and various data compression methods, are extremely important for reducing the storage size requirement in climate modeling. However, little work has been done on investigating the effectiveness of these data reduction methods for climate data sets. In this paper, the potential benefit of data reduction for climate data is studied by investigating a total of 46.5 TB climate data sets, including 3 observation data sets (14.1 TB) and 3 climate model output data sets (32.4 TB). Five different data compression algorithms and two types of content deduplication mechanisms are applied to these data sets to study the possible data reduction effectiveness. Further more, the compressibility of different climate component data is also examined. Our work demonstrates the potential of applying

S. Liu · X. Huang (✉) · H. Fu · G. Yang
Ministry of Education Key Laboratory for Earth System Modeling, and Center for Earth System Science, Tsinghua University, Beijing 100084, China
e-mail: hxm@mail.tsinghua.edu.cn

S. Liu
e-mail: liusb08@mails.tsinghua.edu.cn

H. Fu
e-mail: haohuan@mail.tsinghua.edu.cn

G. Yang
e-mail: ygw@mail.tsinghua.edu.cn

Z. Song
The First Institute of Oceanography, Qingdao 266061, China
e-mail: songroy@fio.org.cn

Published online: 18 October 2013

data reduction methods in climate modeling platforms, and provides guidance for selecting the suitable methods for different kinds of climate data sets. We find that the compression method $LCFP$ can provide the best compression ratio; however, its throughputs, especially the inflate throughputs are much lower than all the others. To strike a better balance between compression ratio and throughputs, we propose a new compression method for the model output data. The new compression method can achieve comparable compression ratio, while attain about 20 times higher inflate throughput than that of $LCFP$.

Keywords Data reduction · Data compression · Data deduplication · High throughput · Climate modeling · Climate data

1 Introduction

In climatology, there are huge volumes of data from observation and climate model simulations. These data are critical for understanding the climate mechanisms and predicting the future climate change. Observation data are fundamental for climatology to monitor and predict the climate, and to verify the climate models. Hundreds of observation stations generate data every couple of minutes or hours for decades. Highly sophisticated measurement technologies such as Satellite and Radar stations, have also been elaborated over the last few decades, producing a huge amount of data [9,24,25]. In the meantime, there has been an explosion in data from numerical climate model simulations, which have increased significantly in complexity and size. With the increasing knowledge of climate processes and the fast growing high-performance supercomputer facilities, more and more complex climate models are developed. Data from these models are expected to become the largest and the fastest-growing segment of the global archive, and it is predicted that there will be more model output data than observation data in the near future [26]. For example there were 16 models and 35 TB data for CMIP3 in 2007, while there will be 21 Models and 3.1 PB data by the year of 2013 for CMIP5 [40]. Thus climate data archiving and transferring will be a great challenge to storage space and I/O bandwidth. Even though faster (and more expensive) storage and network can be adopted to mitigate this problem, reducing the size of data sets itself can be a more effective solution.

In general, data compression and data deduplication are two efficient ways of data reduction. Data compression is a longstanding technique to eliminate unneeded data when data are stored or sent [30,37,38]. It removes the redundancy internal to an object and generally reduces textual data by factors of two to six [16]. However, climate data sets are usually composed of floating point numbers, which are believed inherent random in nature and hard to be compressed losslessly [3,17]. Because of the nature of floating point numbers, even a small change to the numbers may cause significant changes to the byte stream.

Data deduplication suppresses duplication from a more coarse grain (such as data chunks or files) than data compression, and is very efficient in reducing data size for workloads that have many similar files [23,32,42]; storage space requirements can be reduced by a factor of 10–20 or more when backup data is deduplicated [2,27]. For

primary work loads, data dedupication can eliminate 20–30 % redundancy [18,22,32]. However, there are few studies about data deduplication on climate data sets. The most related work is to study the data deduplication in a HPC storage system, and only a small part of its data is output from climate models [22].

In this paper, we examined the data reduction effectiveness of various data compression methods and deduplicaton methods on several climate data sets. The potential benefit of data reduction for climate data is studied by investigating a total of 46.5 TB climate data sets, including 3 observation data sets (14.1 TB) and 3 climate model output data sets (32.4 TB). Different data compression algorithms and content deduplication policies are applied to these data sets to study the possible data reduction effectiveness. In addition, the compressibility of data for different climate components are also examined. Our major contributions are as follows:

- (1) We perform an extensive exploration of five different compression schemas and two content deduplication methods on six different climate data sets.
- (2) We find that *LZO* provides significantly higher throughput (one or two orders of magnitude) than the other compression methods at the cost of compression ratio, while *LCFP* can achieve the best compression ratio with poor throughput.*ZLIB* provides a good trade-off between compression ratio and throughput.
- (3) We provide a new compression methods *nzip*, which gains better trade-off between compression ratio and throughput. *nzip* can achieve competitive compression ratio over *LCFP*, while attaining even higher throughput than *ZLIB*.

The rest of this paper is organized as follows: The next section gives an overview of various kinds of data reduction methods that will be used in our evaluation. Section 3 describes properties of the data sets, and some of our general presentation and analysis techniques. Section 4 presents evaluation results and our corresponding analysis. Section 5 provides the main idea of our new compression methods for model output data sets, and its effectiveness on the model output data sets. Section 6 reviews previous work. Section 7 concludes the paper.

2 Overview of Data Reduction Methods

As stated in the previous section, data compression and data deduplication are two different ways of reducing redundancy in data sets. Usually, they are on different granularities of data with different mechanisms, and can be used jointly to further eliminate redundancy [16,18]. This section will give an overview of the data compression algorithms and deduplication methods used in our evaluation.

2.1 Data Compression

Data compression algorithms can be lossy or lossless. With lossy compression noncritical data is not always restored exactly as it was in the original file, and it is often used to process images or voice signals. With lossless compression data can always be restored to its original format. High data precision for climate computing is important

to maintain numerical stability, so lossless compression algorithms are chosen for climate data reduction.

The compression algorithms in our evaluation fall into two categories: generic byte oriented compressors, such as ZLIB, BZIP2 and LZO, and special-purpose floating-point oriented compressors, such as SZIP and LCFP. Generic compressors operate at byte granularity, while the floating-point compressors has 'knowledge' of the floating-point format and operate at 4 bytes (single-precision floating-point numbers in our data sets) granularity.

ZLIB: ZLIB [44] is an variant implementation of Lempel-Ziv (LZ77) compression algorithm [43]: it records the offset and length of the recurrence of bytes sequence in the history sliding window; then these information is further encoded with Huffman coding. ZLIB is widely adopted in many applications. The compression algorithm used in ZLIB is essentially the same as that in Gzip and Zip. The netCDF-4 library allows users to create compressed data with the ZLIB library [34].
LZO:LZO focuses on speed rather than compression ratios, and it offers fast compression and ultra fast decompression; it uses 64 KB memory for compression while requiring no extra memory for decompression [20]. There are many algorithms of LZO implemented, and the one used in this paper is LZO1X, which is often the best choice among the variants [19].
BZIP2:BZIP2 [4] compresses files using the Burrows-Wheeler block sorting text compression algorithm [39]. BZIP2 is a popular compressor because its compression ratio is generally considerably better than that achieved by more conventional LZ77, and BZIP2 approaches the performance of the PPM family of statistical compressors.
LCFP: LCFP [14] compresses floating-point values with predictive coding in a lossless manner. The predicted and the actual floating-point values are broken up into sign, exponent and mantissa; and their difference is compressed separately with context-based arithmetic coding. In our study, unit-delayed predictor is adopted for prediction: last accessed value as the predicted value.
SZIP: SZIP [41] is an extended implementation of the Rice algorithm [29] to compress science data [1]. The HDF5 library provides interfaces to store and retrieve compressed data with SZIP [10]. The netCDF4 library can also read data compressed with SZIP [34]. In this paper, szip2.1 distributed by the HDF Group is used, and the BITS_PER_PIXEL is set to 32 for the single-precision floating-point values.

2.2 Data Deduplication

Data deduplication is very efficient in eliminating redundancy data across the whole data set in workloads that have many identical data chunks. Usually there are two steps in data deduplication: first, the content of a file is split into non-overlapping data chunks; second, the SHA-1 hash of each chunk is used to determine whether the chunk is already stored among all previously stored data. In the first step, chunking the file into chunks is the most important for the quality of redundancy detection

and therefore the overall data reduction. Two different chunking methods are usually adopted in deduplication systems:

Fixed-Size Chunking (FSC): From the beginning of each file, a file is divided into non-overlapping fixed size data chunks (except the last chunk) [12,28]. The advantage of *FSC* is that it is simple and less CPU-intensive; however, the drawback of *FSC* is that it is sensitive to content shifting among nearly identical files.

Content Defined Chunking (CDC): The generated chunks of a file are defined by the content of the file. Content defined chunking can tolerate the content shifting by finding special anchors in the file content, and using these anchors as boundary to split a file into chunks. Rabin fingerprint is employed to help find the anchors in most *CDC* based deduplication systems [16,23,42]. To split a file into content defined chunks, the Rabin fingerprints of a sliding window (size of the window is usually 48 bytes) over the file content are calculated, and some of these Rabin fingerprints are used as anchors. Each of these special Rabin fingerprints (f) should fulfill the equation:

$$f \bmod c == m \tag{1}$$

If a fingerprint (f) fulfills Eq. 1, an anchor is found, and these anchors are used to indicate chunk boundaries. The chunks generated by this method have a variable size with an expected size c. To avoid too small and too large chunks, minimal and maximal chunk sizes are enforced. In this paper we use the Two Thresholds Two Divisions (TTTD) chunking method to further control the variations of chunk sizes [8]: the TTTD algorithm uses the minimum and maximum thresholds to eliminate large-sized and small-sized chunks, and a second divisor c' to assist the algorithm to determine a backup anchor.

CDC works more efficiently than *FSC* in most cases where there are a lot of similar files [21]; however, when processing rarely changed files, there is little difference in redundancy elimination between *CDC* and *FSC*.

3 Evaluation Methodology

3.1 Data Sets

To have a comprehensive understanding of the climate data, we examined two types of data sets: observation data sets and model output data sets. The observation data sets come from satellite, radar and land observations. The model output data sets are from earth system models and are submitted to CMIP5 [33].

3.1.1 Observation Data Sets

There are three types of observation data sets in our research. One is satellite data from *European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)*. The main purpose of *EUMETSAT* is to deliver weather and climate-related satellite data, images and products–24 h a day, 365 days a year. A sub data set is chosen from *EUMETSAT* during the period of 26 months from 2010.11 to

Table 1 Observation data sets

Data set	File count	Size (TB)	Mean FS.	Median FS.
EUSat	2,250	2.9	1,313	763.6 KB
Radar	3,682	6.8	1,930	27.9 MB
NewOpr	2,577,389	4.4	1.80	550.0 KB

Table 2 Model output data sets

Data set	File count	Size (TB)	Mean FS.	Median FS.
BNU	3,359	8.5	2.7 GB	342 MB
LASG	119,775	13.3	116 MB	32 MB
FIO	1,444,215	10.6	7.7 MB	960.6 KB

2012.12, and is noted as $EuSat$. The second observation data set comes from a cluster of radar observation stations in China, which is noted as $Radar$ in this paper. The last observation data set (noted as $Newopr$) is from the land observation stations in China, and its content is about aerosol. There is about 14.1 TB observation data in total as shown in Table 1. The size of each data set in Table 1 are the accumulations of the logical size of each file in the data sets. These three data sets are hosted in the data center of *China Meteorological Administration (CMA)*.

3.1.2 Climate Model Output Data Sets

There has been an explosion in data from numerical climate model simulations, which have increased greatly in complexity and size. Data from these models are expected to become the largest and the fastest-growing segment of the global archive [26]. Among the model output data sets, the data sets from the fifth phase of the Coupled Model Intercomparison Project (CMIP5) models are the most important ones [33]. CMIP5 provides valuable data sets for examining climate predictability, assessing model mechanisms, and improving the ability of models to predict climate on larger time scales. Currently, there are 19 institutes providing 41 models along with their output data sets. All the CMIP5 data are climate data at global scale, and is in netCDF formate following the CMOR standard. All the data sets can be downloaded through *earth system grid federation* (ESGF) [7].

In this paper, three data sets from three different Chinese institutes are examined: BNU-ESM model by the researchers in Beijing Normal University (BNU), FIO-ESM model by the researchers in The First Institute of Oceanography (FIO), and LASG-CESS model by Chinese Academy of Sciences (LASG) and Tsinghua University. There are about 32.4 TB output data in total from these three models which have been submitted to the CMIP5. Table 2 gives an overview of the properties of these three data sets. Almost all of the output data are in netCDF classic format, and the high dimension variables in the netCDF files are IEEE 32-bit single precision floating-point arrays. To evaluate the performance of *SZIP* and *LCFP*, we extract these floating-point arrays and then apply the *SZIP* and *LCFP* onto these arrays.

3.2 Data Processing

To analyze the potential benefits of these different data reduction methods, we developed a toolkit that can apply these data reduction methods to the data sets in the data centers. The toolkit contains three kinds of tools as well as some automation scripts for deployment:

(1) *data compression tools*: We use the python modules *zlib* and *bz2* (Python 2.7.3) to implement the parallel versions of ZLIB and BZIP2. A parallel version of LZO compressor/decompressor is implemented based on the miniLZO [20]; each thread compresses data chunks at the size of 32 MB. The compression level of these compressors are all in their default settings: 6 for *ZLIB*, 9 for *BZIP2*, and 7 for *LZO*. Our *LCFP* compressor is based on Isenburg's source [11], and compresses data chunks of 2 GB. The SZIP compressor is based on szip-2.1, and BITS_PER_PIXEL is set to 32, PIXELS_PER_BLOCK is 32, SCANLINE is 128. As mentioned above, SZIP and LCFP work on the extracted floating-point arrays, while the other compressing methods compress the whole netCDF files in the three model output data sets.

Some tricks are adopted to achieve better performance when studying the compressibility of the data sets. Since the compressed files are not needed when evaluating the compression ratio of the data sets, only the length of the output stream of the compressors is accumulated, and the content of the output stream is discarded without touching the disk, thus improving performance and reducing extra storage requirement. When to evaluate the throughputs of these compression algorithms, the compressed file of the compressor is put into the memory file system *tmpfs*, the decompressor read the compressed file from the memory file system and delete the compressed file afterwards.

(2) *data deduplication tools*: parallel versions of TTTD based chunking and FSC based chunking are implemented in c programming language; the output formats of these two chunking methods are the same: containing file name, file size, the 20 bytes SHA-1 hashes of its chunks, the length of each chunk. Due to the huge amount of the chunk hashs, a distributed parallel program is implemented to statistics the deduplicated chunks. A set of other Python scripts are also developed to make further analysis of the chunks.

(3) *combination of deduplication and compression*: A framework of compressing the duplicated chunks to estimate the joint data reduction effect of data deduplication and data compression methods.

Most of the tools are deployed onto the data centers where the data sets are resident. During our study, there is no update to these data sets. The data reduction ratio R in this paper is define as:

$$R = \frac{ReducedSize}{OriginalSize} \tag{2}$$

The smaller the reduction ratio is, the better the data reduction works. A reduction ratio of 75 % indicates that 25 % of the data could be removed by compression or deduplication, and only 75 % of the original data size would be actually stored.

4 Results and Analysis

In this section, we first compare the data reduction ratio and throughput of the five compression algorithms on the climate data sets, and then compare the data reduction efficiency of different chunking strategies of data deduplication methods. The data reduction effect of the combination of data deduplication and data compression algorithms is also studied in this section. In addition, since climate model data can usually be grouped into five different components, the reduction efficiency of each component is also examined.

4.1 Data Reduction by Compression

To find the most efficient compression algorithm for the climate data sets, we evaluate the performance of the five compression algorithms over the climate data sets. The efficiency of a compression algorithm can be measured from two aspects, data size reduction (compression ratio), and the throughput of deflate and inflate. In this section, the compression ratios of the data sets by these compression algorithms are examined first; then the throughputs of these algorithms are also studied.

4.1.1 Comparison of Compression Ratios

Figure 1 presents the compression ratio by the five compression methods over the climate data sets. Note that since *SZIP* and *LCFP* should work on byte stream of floating-point numbers; we first extracted the floating-point variables from the netCDF files of the three model data output sets, and then fed the *SZIP* and *LCFP* compressors with the floating-point byte streams. The extracted data weighs more than 90 % of the total storage.

Figure 1 shows that *LCFP* provides the best data reduction for the three model output data sets. The reduction ratio of *LCFP* is always 10 % better than the best

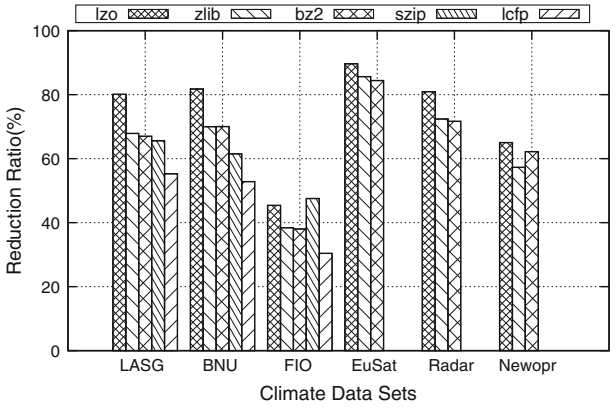

Fig. 1 Data reduction for the climate data sets by different data compression methods: *LZO*, *BZIP2*, *ZLIB*, *SZIP* and *LCFP*. It should be noted that *SZIP* and *LCFP* only work on the three model output data sets

compression ratio achieved by all the other four methods. For example, for the $LASG$ data set, the reduction ratio by *LCFP* is 55.25 %, while the reduction ratio by *ZLIB* is 67.87 %. However, another numeric oriented compressor *SZIP* is not always better than the generic compressors. For the BNU data set, *SZIP* achieve significantly better reduction ratio than *ZLIB*, while on the FIO data set, the reduction ratio of *SZIP* is much poorer than that of *ZLIB*. The good compression ratio by *LCFP* indicates that *LCFP* should be adopted to compress the floating-point numbers when better data reduction is expected.

As shown in Fig. 1, the compressibility of each data sets is different from the others. Among the three model output data sets, the FIO data set can be significantly compressed by all the three compression algorithms: achieving more than 60 % reduction in size. In contrast, the other two model output data sets, $LASG$ and BNU, can only be reduced around 40 % in size. The reason for the good compressibility of the FIO data set is that there are many zero chunks in the data set; *cdc4k* (see Sect. 4.2) detects that 16.7 % of the FIO data set are zero chunks. As to the observation data sets, the $EuSat$ data set shows the least compressibility. This is because the content of $EuSat$ is comprised of satellite images, which are already in some compressed format (JPEG for example). The $Newopr$ data set shows the best compressibility among the observation data sets, about 40 % of its storage requirement can be reduced by compression.

From Fig. 1, it can also be seen that the *BZIP2* algorithm is not much better than the *ZLIB* algorithm with regard to the reduction ratio, which is different from the results in other benchmarks [5]. In our experiment, in most cases, the compression efficiency by *ZLIB* is only around 1 % less than that by the *BZIP2*. For the $Newopr$ data set, *ZLIB* even outperforms *BZIP2*. As shown in Sect. 4.1.2, both the deflate and inflate throughputs of *ZLIB* are much higher than those of *BZIP2*. So we can get the conclusion that *ZLIB* is a better choice than *BZIP2* for generic climate data compression.

It is worth noting that the current *netCDF-4* library provides interface to compress data with *ZLIB*, and the *HDF-5* library allows user to compress data with *SZIP*. According to the findings in this section, we can see that *SZIP* does not have obvious advantage over *ZLIB* in terms of compression ratio. To achieve even better compression ratio, these libraries should provide interface to allow users compress data with the *LCFP* compressor.

Compression ratio is not the only factor to evaluate a data compressor, its deflate and inflate throughputs are also important in climate applications. In the followings section, we will evaluate the throughput performance of these five compressors over the climate data sets.

4.1.2 Comparison of Throughput

Climate computing also demands high deflate and inflate throughput if compression is applied to the climate data. Figure 2 shows the single-thread throughput of deflating and inflating the data sets with the five data compression algorithms. It should be noted that the throughputs do not include the overhead of disk I/O: the compression and decompression time is measured while the data is in the main memory. It should also be pointed out that the throughputs for the three model output data sets, *LASG*,

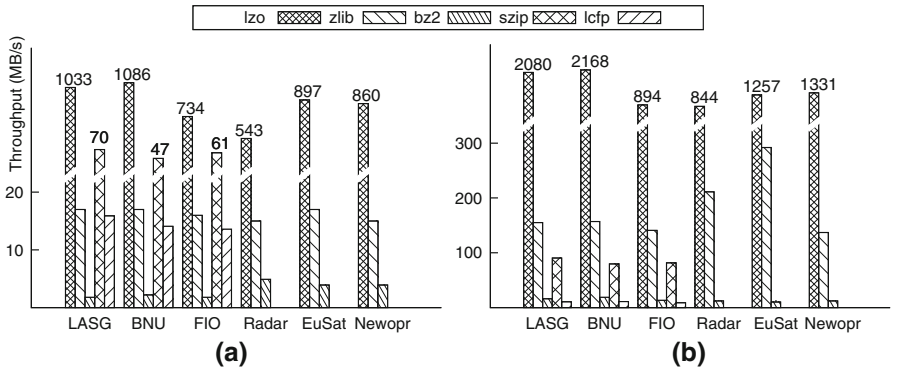

Fig. 2 Single thread deflate and inflate throughput (in memory) of the data compression algorithms on each data set. As the throughputs of *lzo* and *szip* are much higher than those of *zlib*, *bz2* and *lcfp*, the histograms for them are shown in two segments, and the numbers on top of the bars are the throughputs of them **a** deflate throughput (MB/s) **b** inflate throughput (MB/s)

BNU and *FIO*, were evaluated on three different servers, and the throughputs for the observation data sets were evaluated on one server in *CMA*.

As can be seen in Fig. 2, *LZO* is the best of the five data compression algorithms in terms of throughput. For deflation, *LZO* is about 36–66 times faster than *ZLIB*, and can be as much as 575 times faster than *BZIP2* over the *LASG* data set. As to inflation, the throughput of *LZO* is about one order of magnitude faster than that of *ZLIB*, and around two orders faster than that of *BZIP2* and *LCFP*. So if an application needs extremely high throughput and can tolerate higher data redundancy (around 10 % according to previous subsection), *LZO* shall be the first choice of the five compression algorithms.

Though *LCFP* can achieve significantly better compression ratio than the other compressors, its throughput is relatively low. As shown in Fig. 2, the deflate throughput of *LCFP* is only better than that of *BZIP2*, and its inflate throughput is the lowest of the five compressors. This is because *LCFP* has to switch compression context frequently to achieve better compression ratio, and there are as much as 512 contexts in the *LCFP* compressor. Each floating-point number is split into two or three parts, and each part belongs to one of the 512 compression contexts.

One phenomenon should be noted about observation data sets: the *LZO* throughputs for *Radar* is the lowest among the three observation data sets, though the compression ratios for *Radar* are not the worst. This may be explained that for the easiest compressed data set (*Newopr* in this case), the *LZO* can easily find enough matching strings when compressing; and for the hardest compressed data set (*Eusat* in this case), *LZO* can make the decision of non-match earlier. Thus the throughputs for the easiest and the hardest compressed data sets will be much higher. This inference is confirmed in Table 4: The hardest compressed component *atmos* and the easiest compressed component *seaIce* have the highest throughputs among the five components.

According to Figs. 1 and 2, we can find that *ZLIB* achieve a good trade-off between compression ratio and throughput. The compression ratio of *ZLIB* is only less than that of *LCFP*, while its throughput is much higher than that of *LCFP*. The inflate throughput of *ZLIB* is better than the other compressors except *LZO*. The generic

compressor *BZIP2* does not have obvious advantage over *ZLIB* in terms of compression ratio, and has much worse throughput than *ZLIB*. So *BZIP2* should be avoided when compressing climate data; and *ZLIB* should be the default compressor because of its good balance of compression ratio and throughput.

4.2 Comparison of Different Chunking Strategies of Data Deduplication

In this section, the effectiveness of data reduction by different data deduplication methods is examined. As mentioned in Sect. 2.2, fixed-size chunking (FSC) and content-defined chunking (CDC) are two popular chunking methods; and 4, 8 and 16 KB are common chunking size. In this section we compare the effectiveness of FSC and CDC; in addition, different chunking sizes are also examined. Figure 3 shows the data reduction ratio of the six data sets by FSC and CDC with different chunking sizes. Note that the data deduplication of each data set is done in their own domains.

As shown in Fig. 3, the data reduction ratios by deduplication methods vary over different data sets. Among the three model output data sets, data deduplication methods can detect around 30 % redundancy for the FIO data set; while the other two model output data sets, $LASG$ and BNU, can only be reduced by around 7 % through deduplication methods. Again, the zero chunks in FIO data set contribute a lot to the redundancy. As to the observation data sets, both CDC and FSC methods can hardly find any redundancy in $EuSat$ and $Radar$ data sets. However, the $Newopr$ data set can benefit a lot from data deduplication methods. For example, *cdc4k* can eliminate about 25 % redundancy data for $Newopr$ data set.

Figure 3 also shows that content-defined chunking (cdc4k) based data deduplication is barely better than that of fix-sized chunking (fsc4k), 3 % better at most for $Newopr$ data set. In general, content-defined chunking, which can overcome the vulnerability to data shifting, outperforms fixed-size chunking significantly; for example, in a backup

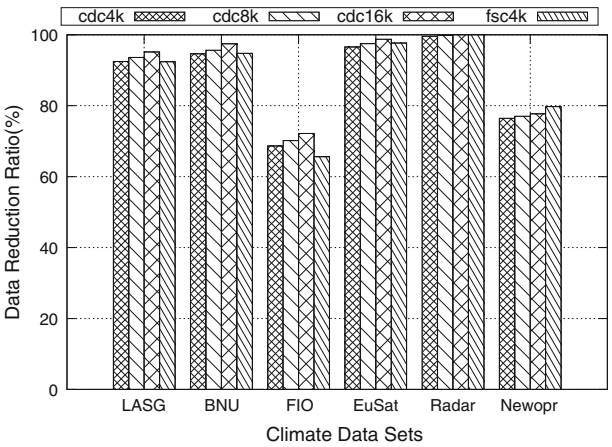

Fig. 3 Data reduction for the data sets by different data chunking methods and sizes. Value is reduced size over the original (shown here as percentage); smaller is better

scenario, content-defined chunking can find 10 % more redundancy than fix-sized chunking [21]. However, in our experiments, since there is no backup data, there is no data shifting problem; thus CDC does not outperform FSC. As to the FIO data set, FSC is even obviously better than CDC; we examined the chunks generated by FSC and CDC, and found that FSC can detect more duplicate zero chunks than CDC for data set FIO. This observation suggests that the less CPU intensive FSC method shall be used for climate data set deduplication without losing much storage savings.

Another phenomenon that shall be noted in Fig. 3 is that the deduplication quality drops slowly with the increasing of the chunk sizes, which is confirmed in other literatures on backup scenarios [21,32]. For example, while the chunk size increase from 1 KB (not shown in Fig. 3) to 16 KB for the $Newopr$ data set, the redundancy detected by data deduplication only drops from 24.67 to 22.30 %. This means larger chunk size can be adopted to reduce the metadata needed while maintaining a relatively good data deduplication quality. The reason that the drop in deduplication ratio is less than linear with increasing chunk size is the spatial locality of the duplicated data. To investigate the spatial locality of duplicated chunks, we measure the distance of a duplicated chunk and the next duplicated chunk following it in the same file for all the files in the data sets. The distance of two duplicated chunks is the number of data chunks between them in one file. For example, if duplicated chunk B is next to another duplicated chunk A, then the distance of chunk A and B is zero. We find that around 90 % of the distances of the duplicated chunks in data sets chunked by *cdc4k* is zero, which means that the duplicated chunks are clustered.

4.3 Data Deduplication and Compression

Data compression methods outperform the deduplication methods in terms of data reduction over all the six climate data sets, which can be seen by comparing Figs. 1 and 3. For example, *cdc4k* can remove merely around 7 % of the two module output data sets $LASG$ and BNU, while compression methods can eliminate around 30 % storage requirement for these two data sets. *cdc4k* can hardly find any data redundancy on the satellite observation data set $EuSat$, especially for the radar based observation data set $Radar$; on the other hand, compression methods can eliminate around 20 % redundancy in these two observation data sets.

Although data deduplication contributes less to the data reduction than that of compression methods, the two types of methods are orthogonal, and can be combined together [32] to reduce redundancy further. To study the joint effect of the data deduplication and data compression, we first deduplicate the data sets by *cdc4k*, then each unique data chunk is compressed with *ZLIB*. Figure 4 shows the combined data reduction effectiveness of *cdc4k* and *ZLIB*. As the $EuSat$ and $Radar$ can hardly be deduplicated, Fig. 4 does not include the results over them. As shown in Fig. 4, the reduction effectiveness of the two methods can be added up. However, the ideal effectiveness of the combination of the two is slightly better than the experimental results. This is because *ZLIB* usually needs a chunk size of 32 KB to achieve its best compression result, while the average chunk size of *cdc4k* is only 4 KB.

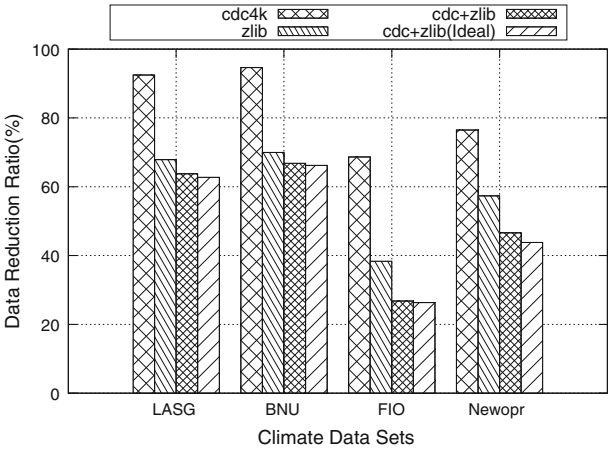

Fig. 4 Data reduction for the combination of cdc4k and zlib. Value is reduced size over the original (shown here as percentage); smaller is better

4.4 Data Reduction of Climate Components

The research on global climate change usually involves the studies of atmosphere *(atmos)*, ocean *(ocean)*, land *(land)*, sea ice *(seaIce)* and land ice *(landIce)*, and a coupler that combines these components together *(cpl)*. However, due to the different observation measures and different physical models, the data of these components vary from each other, and in most cases, these different components are accessed separately. Thus it is necessary to study the data reduction potential of these components individually. Table 3 shows the sizes of each component for the three model output data sets. It can be seen that the atmosphere usually has the largest volume of data, and the ocean comes the second; while the data volumes of land ice and sea ice are much smaller. One main reason is that sea ice and land ice only cover a small part of the earth.

The data reduction results of the components for the three model output data sets are shown in Fig. 5. It should be noted that deduplication(*cdc4k*) was done in each component domain for each data set, respectively. The first observation from Fig. 5 is that the data reduction ratios of the components are different from each other. The *land, landIce* and *seaIce* components shows good compression ratios; while the *atmos* component can not be compressed easily by most of the data reduction methods. For example, the *land, landIce*, and *seaIce* components have reduction ratios around 30 % by compression; the *atmos* component on the other hand, has a compression ratio

Table 3 Component size (GB) of model output data sets

Data set	atmos	ocean	land	seaIce	landIce	cpl	Other
BNU	5,902	1,944	200	506	11	0	147.3
LASG	9,272	3,548	294	454	28	0	0
FIO	2,702	3,374	1,256	75.8	0	3,405	8.4

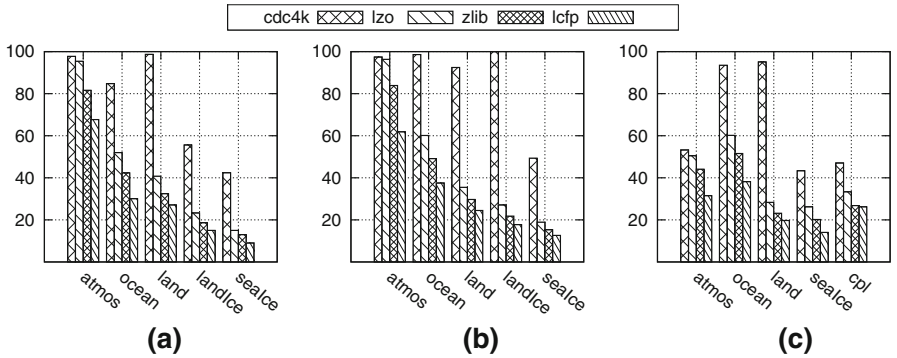

Fig. 5 Reduction ratio of each climate component. **a** LASG, **b** BNU, **c** FIO

Table 4 Throughput of climate components of *BNU*

	Deflate (MB/s)			Inflate (MB/s)		
	lzo	zlib	lcfp	lzo	zlib	lcfp
atmos	1,163	15.7	13.2	2,262	156.1	10.2
ocean	902	17.4	15.3	2,075	153.0	11.9
land	760	28.9	17.1	1,470	224.6	13.4
landIce	949	21.8	19.4	1,344	263.2	14.5
seaIce	1,425	15.3	22.3	1,976	157.4	15.7

around 60 %. This is because land, sea ice and land ice cover a relative smaller part of the earth, and all component models are assigned the same global grid; thus most values of the *land, landIce* and *seaIce* components data are set to a constant, resulting higher compressibility.

The second observation from Fig. 5 is that compression based methods(*zlib,lzo, lcfp*) outperform the CDC method(*cdc4k*) in term of data reduction effectiveness for all the components; especially for the *land* component, *cdc4k* can hardly find any redundancy, while the compression based methods can reduce 80–85 % storage requirement. This is because even for the more compressible components, there are shorter repeated byte sequences, which are out of the capable of CDC methods.

Another interesting observation from Fig. 5 is that the gaps among the data reduction ratios by the three compression methods *LZO, ZLIB* and *LCFP* are narrowing with the increasing of the data reduction possibility; this trend can be seen clearly in Fig. 5a, b. For example, the gap between the reduction ratios by *ZLIB* and *LCFP* is around 12 % (83.88 vs. 61.85) over the *atmos* component of *LASG*, and drops to less than 3 % (15.28 vs. 12.60) over the *seaIce* component. This observation indicates that for more compressible data set, high throughput compression methods (such as *LZO* and *ZLIB*) can be adopted, while attaining a comparable compression ratio with *LCFP*.

Not only is the compressibility of the components different from one another, but also the throughput of compression and decompression over these components varies a lot. Table 4 shows the throughput of deflating and inflating the components of the *BNU* data set with the three compression algorithms: *LZO*, *ZLIB* and *LCFP*. As can be seen in Table 4, the throughput over each component is different from each

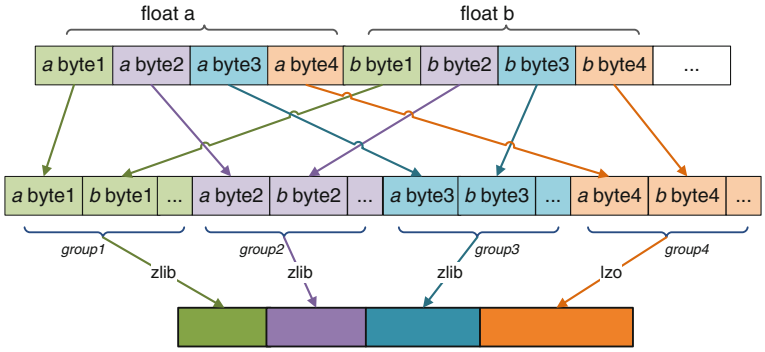

Fig. 6 Re-arrangement the bytes of the IEEE 32-bit single precision floating-point numbers for compression. There are four different compression streams, and each stream can be compressed with different byte-stream oriented compression method. In this paper, all the four streams are compressed with *zlib*

other by all the three algorithms. For example, the deflate throughput by *LZO* varies from 760 to 1,425 MB/s. An interesting phenomenon can be seen from Table 4 is that the throughputs of *ZLIB* and *LCFP* are increasing with the compressibility of the components (the exception is the *seaIce* component); however, *LZO* does not have this trend. From Table 4, we can notice again that the throughput, especially the inflate throughput of *LCFP* is very poor: the inflate throughput of *LCFP* is one order slower than that of *ZLIB*, and two orders slower than that of *LZO*.

5 A New Compression Method

According to the findings in previous sections, we can see that *LCFP* can achieve outstanding compression ratio over the other data reduction methods. However, its weakness is also obvious: the inflate throughput of *LCFP* is the lowest among the compression methods. Inspired by *LCFP* that sign, exponent and mantissa bits are treated separately, we provide a new compression method, named ***nzip***, to compress floating-point numbers with better trade-off between compression ratio and throughput.

The main idea of our new compression method *nzip* is to rearrange the four bytes of a series of IEEE 32-bit single precision floating-point numbers; the rearranged byte-stream can make better use of the advantages of the byte-stream oriented compressors, such as *ZLIB* and *LZO*. As shown in Fig. 6, *nzip* first reads into a chunk (32 MB in our evaluation) of 32-bit floating-point numbers; then it splits each 32-bit floating-point number into four bytes and appends each byte to its corresponding byte-group; in the final step, it feeds the rearranged data chunk to a byte-stream oriented compressor (*ZLIB* in our evaluation). The intuition behind *nzip* stems from the following two observations. First, the floating-point numbers of the same variable in the same netCDF file tend to be on the same order of magnitude, so the sign bit and exponent bits shall be similar; for example, in Fig. 6 *group1*, which is made of the sign bit and higher 7 exponent bits, has a compression ratio of 5 % for even the *atmos* component. Second, in many cases the adjacent floating-point numbers are also close to each other in value;

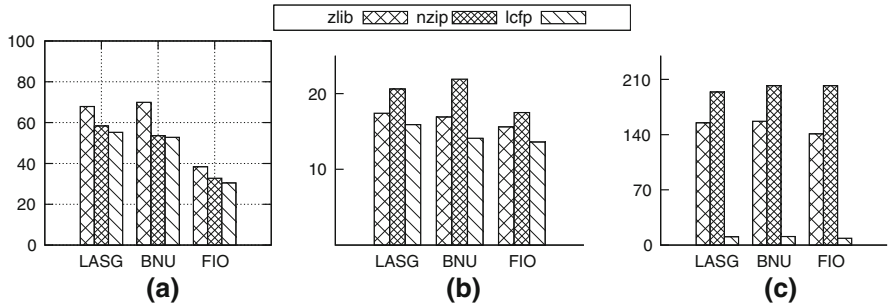

Fig. 7 Comparison of reduction efficiency among the nzip, *ZLIB* and *LCFP*. **a** compression ratio, **b** deflate throughput **c** inflate throughput

this is due to the fact that atmosphere and ocean circulations are basically orientated in an east to west direction because of the rotation of the earth. Thus, on a given latitude, the adjacent floating-point numbers are also close to each other in value. As a result, the corresponding bytes of adjacent floats have higher chance to be the same. According to these two observations, the rearranged byte-stream will be more 'meaningful' to the compressor, so a better compression ratio can be expected.

To evaluate the effectiveness of our new compression methods, we use *ZLIB* as compression engine to implement a prototype compressor of *nzip*, and apply it on the three model output data sets. Figure 7 presents the compression ratio(Fig. 7a), deflate throughput (Fig. 7b) and inflate throughput (Fig. 7c), by *ZLIB*, *LCFP* and our new compression methods *nzip* over data set $LASG$, BNU and FIO. Figure 7a shows that *nzip* can achieve much better compression ratio than that of *ZLIB*, approaching to the compression ratio by *LCFP*. As to the inflate throughput, *nzip* provides around 20 times higher throughput than *LCFP*, as is shown in Fig. 7c. Another interesting phenomenon in Fig. 7 is that the deflate and inflate throughput of *nzip* are even better than that of *ZLIB*, despite of the additional byte rearrangement by *nzip*. There are two reasons for this phenomenon. First, the bytes rearrangement involves only simple memory copy operation, which can incur only negligible overhead when compared to the more time consuming compression process. Second, *ZLIB* tends to have higher throughput when data are more compressible, as shown in Table 4; after byte rearrangement, the byte stream is much more compressible as shown in Fig. 7a. And since *nzip* uses *ZLIB* as its compression engine, thus *nzip* can achieve better throughput than the pure *ZLIB*. From the evaluation results shown in Fig. 7, we believe that *nzip* have achieved a good trade-off between compression and throughput: it can achieve comparable compression ratio with *LCFP*, and in the meanwhile it can attain about 20 times higher throughput than *LCFP*.

6 Related Work

Prior to our work, there are several studies on data reduction and its usage in scientific computing environment. The most related work conducted by Meister et al. [22] examined the deduplication potential of the online storage in 4 high performance

computing (HPC) data centers; they reported that 20–30 % of the data is redundant and could be removed by deduplication techniques; they find more redundancy than our work possibly because their data are from the home directories and research project directories where users always generate many similar results. Schmalzl investigated the usefulness of several image compression algorithms for the storage of data from computational fluid dynamics [31]. Wessel proposed to compress the gridded data sets in Earth science to reduce both transmission time and disk storage [38]. Wang et al. suggested a way to achieve interactive rendering by compressing large-scale time-varying data [36]. Compression is also used to improve I/O throughput for I/O forwarding layer in current peta-scale HPC environment [30,37].

There are also many studies that explore the data reduction in other context. Jin and Miller examined the effectiveness of deduplication on virtual machine disk images [15]; they found that deduplication of VM disk images can save 80 % or more space, and chunk-wise compression can reduce the size of the stored chunks by 40 %. Lu et al. evaluated the data reduction opportunity by deduplication and compression, separately and combined, in primary storage systems [6,18]. Meister and Brinkmann compared the influence of different chunking approaches on deduplication in a backup scenario [21]. Wallace et al. presented a study on the data redundancy of backup workloads in production systems [35].

Compared with the studies mentioned above, our work focuses on the data reduction potential on climate data sets, including climate model output data and climate observation data. And we study data reduction effectiveness by deduplication methods and compression methods, separately as well as the joint of the two types of methods.

7 Conclusions and Future Work

In this paper, we investigated the data reduction opportunity of climate data sets by different data deduplication methods and data compression algorithms, and the combination of data deduplication and compression. We found that climate data has less redundancy and compressibility than data sets in other workloads. As to the data reduction effectiveness, compression based methods outperforms data deduplication based methods. However, the two types of methods can be combined together to gain further data reduction. Among the five compression algorithms, *LZO* is featured with orders higher throughput than the other compression methods at the cost of compression ratio. *LCFP* on the other hand, can achieve the best compression ratio, with poor throughput. *ZLIB* stands between *LZO* and *LCFP*, providing a trade-off between compression ratio and throughput. *BZIP2* should be avoided for climate data compression, because it does not have obvious advantage on compression ratio over *ZLIB*, while has much worse throughput than *ZLIB*. As to data deduplication methods, different chunking sizes have little impact on the data deduplication results, which can be partly explained by chunk locality that the duplicated chunks clustered together. And the fix-sized chunking (FSC) can find almost as much redundancy as content-defined chunking (CDC) does.

According to the observations of our study, we propose a new compression method that achieves an even better balance between compression ratio and throughput for the

model output floating-point data. Our compression method can provide comparable compression ratio with *LCFP*, while achieves about 20 times higher throughput than *LCFP*. This compression method shall be applicable to other data sets of floating-point numbers, as long as these numbers are similar in values. In the future, we should further study the value characteristics of the model output data and apply proper prediction methods [3,13] to optimize our compression method, and evaluate its performance on more data sets.

In our future work, we should also do more research on the characteristics of observation-based climate data sets. In this paper only preliminary quantitative analysis of this kind of data sets are present; because of their data types, record structures and data sources are much different from one to another, we can not give a common conclusion of the characteristics of the observation-based climate data sets currently. In the future, we will continue study the characteristics of different kinds of observation-based climate data sets, and try to provide a more effective compression methods for them.

Acknowledgments We would like to thanks Ma Qiang from China Meteorological Administration, professor Lanning Wang from Beijing Normal University, and the researchers from the First Institute of Oceanography and Chinese Academy of Sciences for providing access to their data sets. This research was sponsored by the National High Technology Development Program of China (2010AA012401, 2011AA01A203).

References

1. 120.0-G-2, C. Lossless data compression. In: Report Concerning Space Data System Standards (2006), Green Book, Issue 2
2. Biggar, H.: Experiencing data de-duplication: improving efficiency and reducing capacity requirements. The Enterprise Strategy Group (2007)
3. Burtscher, M., Ratanaworabhan, P.: Fpc: a high-speed compressor for double-precision floating-point data. IEEE Trans. Comput. **58**(1), 18–31 (2009)
4. bzip2. http://www.bzip.org
5. compression-rating. http://compressionratings.com
6. Constantinescu, C., Glider, J., Chambliss, D.: Mixing deduplication and compression on active data sets. In: Data Compression Conference (DCC), 2011, IEEE, pp. 393–402 (2011)
7. Earth System Grid Federation. http://pcmdi9.llnl.gov/esgf-web-fe/
8. Eshghi, K., Tang, H.: A framework for analyzing and improving content-based chunking algorithms. Hewlett-Packard Labs Technical Report TR 30 (2005)
9. EUMETSAT. http://www.eumetsat.int
10. HDF group—HDF5. http://www.hdfgroup.org/HDF5/
11. Homepage of Martin Isenburg. http://www.cs.unc.edu/~isenburg/
12. Hong, B., Plantenberg, D., Long, D., Sivan-Zimet, M.: Duplicate data elimination in a san file system. In: Proceedings of the 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies), pp. 301–314 (2004)
13. Ibarria, L., Lindstrom, P., Rossignac, J., Szymczak, A.: Out-of-core compression and decompression of large n-dimensional scalar fields. In: Computer Graphics Forum (2003), vol. 22, Wiley Online Library, pp. 343–348
14. Isenburg, M., Lindstrom, P., Snoeyink, J.: Lossless compression of predicted floating-point geometry. Comput.-Aided Des. **37**(8), 869–877 (2005)
15. Jin, K., Miller, E.: The effectiveness of deduplication on virtual machine disk images. In: Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, ACM, p. 7 (2009)
16. Kulkarni, P., Douglis, F., LaVoie, J., Tracey, J.M.: Redundancy elimination within large collections of files. In: Proceedings of the USENIX Annual Technical Conference, pp. 59–72 (2004)

17. Lakshminarasimhan, S., Shah, N., Ethier, S., Klasky, S., Latham, R., Ross, R., Samatova, N.: Compressing the incompressible with isabela: in-situ reduction of spatio-temporal data. Euro-Par 2011 Parallel Processing, pp. 366–379 (2011)
18. Lu, M., Chambliss, D., Glider, J., Constantinescu, C.: Insights for data reduction in primary storage: a practical analysis. In: Proceedings of the 5th Annual International Systems and Storage Conference, ACM, p. 17 (2012)
19. LZO Documentation. http://www.oberhumer.com/opensource/lzo/lzodoc.php
20. LZO real-time data compression library. http://www.oberhumer.com/opensource/lzo/
21. Meister, D., Brinkmann, A.: Multi-level comparison of data deduplication in a backup scenario. In: Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, ACM, p. 8 (2009)
22. Meister, D., Kaiser, J., Brinkmann, A., Cortes, T., Kuhn, M., Kunkel, J.: A study on data deduplication in hpc storage systems. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press, p. 7 (2012)
23. Muthitacharoen, A., Chen, B., Mazieres, D.: A low-bandwidth network file system. In: ACM SIGOPS Operating Systems Review, vol. 35. ACM, pp. 174–187 (2001)
24. NOAA Radar Data. http://www.ncdc.noaa.gov/radar-data
25. NOAA Satellite Data. http://www.ncdc.noaa.gov/satellite-data
26. Overpeck, J., Meehl, G., Bony, S., Easterling, D.: Climate data challenges in the 21st century. Science **331**(6018), 700–702 (2011)
27. Park, N., Lilja, D.J.: Characterizing datasets for data deduplication in backup applications. In: Workload Characterization (IISWC), 2010 IEEE International Symposium on (2010), IEEE, pp. 1–10
28. Quinlan, S., Dorward, S.: Venti: a new approach to archival storage. In: Proceedings of the FAST 2002 Conference on File and Storage Technologies, vol. 4 (2002)
29. Rice, R.F.: Practical universal noiseless coding. In: 23rd Annual Technical Symposium. International Society for Optics and Photonics, pp. 247–267 (1979)
30. Schendel, E.R., Pendse, S.V., Jenkins, J., Boyuka II, D.A., Gong, Z., Lakshminarasimhan, S., Liu, Q., Kolla, H., Chen, J., Klasky, S., et al.: Isobar hybrid compression-i/o interleaving for large-scale parallel i/o optimization. In: Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing, ACM, pp. 61–72 (2012)
31. Schmalzl, J.: Using standard image compression algorithms to store data from computational fluid dynamics. Comput. Geosci. **29**(8), 1021–1031 (2003)
32. Srinivasan, K., Bisson, T., Goodson, G., Voruganti, K.: idedup: latency-aware, inline data deduplication for primary storage. In: Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST12), San Jose, CA (2012)
33. Taylor, K., Stouffer, R., Meehl, G.: An overview of cmip5 and the experiment design. Bull. Am. Meteorol. Soc. **93**(4), 485 (2012)
34. The netCDF-4 format. http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/NetCDF_002d4-Format.html
35. Wallace, G., Douglis, F., Qian, H., Shilane, P., Smaldone, S., Chamness, M., Hsu, W.: Characteristics of backup workloads in production systems. In: Proceedings of the 10th USENIX Conference on File and Storage Technologies (Berkeley, CA, USA, 2012), FAST'12, USENIX Association, pp. 4–4
36. Wang, C., Yu, H., Ma, K.-L.: Application-driven compression for visualizing large-scale time-varying data. IEEE Comput. Gr. Appl. **30**(1), 59–69 (2010)
37. Welton, B., Kimpe, D., Cope, J., Patrick, C.M., Iskra, K., Ross, R.: Improving i/o forwarding throughput with data compression. In: Cluster Computing (CLUSTER), 2011 IEEE International Conference on (2011), IEEE, pp. 438–445
38. Wessel, P.: Compression of large data grids for internet transmission. Comput. Geosci. **29**(5), 665–671 (2003)
39. Wheeler, D., Burrows, M.: A block-sorting lossless data compression algorithm. Digital Systems Research Center Report 124 (1994)
40. Williams, D.N.: Climate Science Responds to 'Big Data' Challenges: Accessing Analyzing Model Output and Observations. http://downloads.usgcrp.gov/downloads/igim/05_Williams.pdf
41. Yeh, P.-S., Xia-Serafino, W., Miles, L., Kobler, B., Menasce, D.: Implementation of ccsds lossless data compression in hdf. In: Earth Science Technology Conference (2002)
42. Zhu, B., Li, K., Patterson, H.: Avoiding the disk bottleneck in the data domain deduplication file system. In: Proceedings of the 6th USENIX Conference on File and Storage Technologies, vol. 18 (2008)

43. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Trans. Inf. Theory **23**(3), 337–343 (1977)
44. zlib. http://www.zlib.net