01 Nov 2019

Use Cases of Lossy Compression for Floating-Point Data in Scientific Data Sets

Franck Cappello

Sheng Di

Sihuan Li

Xin Liang
Missouri University of Science and Technology, xliang@mst.edu

et. al. For a complete list of authors, see https://scholarsmine.mst.edu/comsci_facwork/1105

Recommended Citation

F. Cappello et al., "Use Cases of Lossy Compression for Floating-Point Data in Scientific Data Sets," *International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201-1220, SAGE Publications, Nov 2019.
The definitive version is available at https://doi.org/10.1177/1094342019853336

Special Issue: CCDSC 2018

International Journal of HIGH PERFORMANCE COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
2019, Vol. 33(6) 1201–1220
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1094342019853336
journals.sagepub.com/home/hpc
SAGE

Use cases of lossy compression for floating-point data in scientific data sets

Franck Cappello[1,2], Sheng Di[1], Sihuan Li[3], Xin Liang[3], Ali Murat Gok[4], Dingwen Tao[5], Chun Hong Yoon[6], Xin-Chuan Wu[7], Yuri Alexeev[1] and Frederic T Chong[7]

Abstract
Architectural and technological trends of systems used for scientific computing call for a significant reduction of scientific data sets that are composed mainly of floating-point data. This article surveys and presents experimental results of currently identified use cases of generic lossy compression to address the different limitations of scientific computing systems. The article shows from a collection of experiments run on parallel systems of a leadership facility that lossy data compression not only can reduce the footprint of scientific data sets on storage but also can reduce I/O and checkpoint/restart times, accelerate computation, and even allow significantly larger problems to be run than without lossy compression. These results suggest that lossy compression will become an important technology in many aspects of high performance scientific computing. Because the constraints for each use case are different and often conflicting, this collection of results also indicates the need for more specialization of the compression pipelines.

Keywords
Lossy compression, floating-point data, scientific data set, applications, use cases

1. Introduction

Lossy compression for scientific data and floating-point numbers is becoming more popular as the limitations in terms of storage/memory size (MS) and transfer bandwidth of scientific equipment, such as instruments and supercomputers used for numerical simulation and analysis, are becoming more severe. In the past, the main use case for lossy compression of scientific data (comprising floating-point numbers) was visualization. Currently, one of the main drivers for new use cases of lossy compression is the slow increase in the storage bandwidth (SB) in supercomputers across recent generations compared with the increase in the MS and computing performance. Table 1 shows the main characteristics of three classes of systems: early petascale, petascale, and pre-exascale systems.

From Table 1, we can observe that the ratio of MS on nodes to total SB is increasing except in the case of CORI with burst buffers. The MS/SB ratios actually reflect the time it will take in an ideal situation to store the full memory content on the file system. In practice, the time is even higher as a result of different congestion factors in the I/O system (Gainaru et al., 2015). Since executions are larger (more cores, more memory) in newer systems, the consequence of this trend is that it takes more time to save execution results and states (checkpoints) than before. To avoid severe performance degradation, applications need to reduce the size of the data (results or states) saved on file systems. We can also observe an important increase in the ratio of petaflops to SB, which represents the number of floating-point operations needed for each byte transferred

[1] Argonne National Laboratory, Lemont, IL, USA
[2] Department of Computer Science, University of Illinois Urbana-Champaign, Urbana, IL, USA
[3] Department of Computer Science and Engineering, University of California, Riverside, CA, USA
[4] Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA
[5] Department of Computer Science, University of Alabama, Tuscaloosa, AL, USA
[6] Linac Coherent Light Source, SLAC National Accelerator Laboratory, Menlo Park, CA, USA
[7] Department of Computer Science, University of Chicago, Chicago, IL, USA

Corresponding author:
Franck Cappello, Argonne National Laboratory, Lemont, IL 60439, USA.
Email: fci@lri.fr

Table 1. Three classes of supercomputers showing their performance, MS, and SB.

Supercomputers	Year	Class	PF	MS	SB	Ratio MS/SB	Ratio PF/SB
Cray Jaguar	2008	1 Pflops	1.75 Pflops	360 TB	240 GB/s	1.5k	7.3k
CRAY Blue Waters	2012	10 Pflops	13.3 Pflops	1.5 PB	1.1 TB/s	1.3k	13k
CRAY CORI	2017	10 Pflops	30 Pflops	1.4 PB	1.7 TB/s[a]	0.8k	17k
IBM Summit	2018	100 Pflops	200 Pflops	>10 PB[b]	2.5 TB/s	>4k	80k

PF: peak flops; MS: memory size; SB: storage bandwidth.
[a]When using burst buffer.
[b]Counting only DDR4.

to the file system in order to keep the processor busy. This ratio is one order of magnitude higher in 2018 than it was in 2008. The table is consistent with the figure presented in Foster et al. (2017). The impact of this trend is that applications should be optimized to reduce significantly their I/O traffic during execution.

In addition to the slow progress of the SB in supercomputers, other trends motivate the use of lossy compression for scientific data. First, the average selling price of dynamic random access memory (DRAM) doubled between 2012 and 2017 (and tripled since 2007). Since the DRAM represents a significant portion of the cost of a supercomputer, for a given system cost (e.g. $200 million), the size of the system DRAM between 2007 and 2017 could not grow as fast as the DRAM density grew, and the ratio of the DRAM bytes per flop decreased during that period: from 0.2 (Jaguar) in 2008 to 0.05 in 2017–2018 (CORI and Summit). Consequently, the applications need to be adapted to use relatively less memory than before. Similarly, the total performance per socket has increased sharply (+50% to +60%/year) in the past decade, thanks to the multi-many-core design, but the bandwidth between the processor socket and the memory has not increased at the same pace (+23%/year) (McCalpin, 2016). This situation results in a widening gap between the processor and the memory system performance that the cache hierarchy can compensate for only partially and not for all applications. Thus, applications should be improved to require less memory bandwidth.

Data reduction has been an important technique for recent large-scale instruments such as the Large Hadron Collider. However, the development of new scientific instruments is facing even more severe constraints with respect to data reduction because the storage and communication systems are improving at a much lower speed than the measurement resolution and accuracy of these instruments are. This is particularly true for upgrade projects such as the Advanced Photon Source APS-U (Fornek, 2017) and the Linac Coherent Light Source II (Marcus et al., 2015) and also for other flagship projects such as the Square Kilometre Array (Domingos Barbosa, 2016) that will produce gigantic amounts of data. In the case of instruments, the data need to be reduced online while it is produced, thus representing another level of difficulty.

This evolution of supercomputer and instrument characteristics has motivated new research and development of lossy compression software for scientific data. In the following section, we describe the progress made in the past 5 years and how this progress has enabled new use cases of lossy compression.

The article is organized as follow. The next section reviews the progress in lossy compression technologies for scientific data sets and compares state-of-the-art lossy compressors with wavelet-based compressors and decimation. The following section presents seven use cases of lossy compression for scientific data: visualization, reduction of data stream intensity, reduction of the storage footprint, reduction of I/O time, accelerating checkpoint/restart, reduction of memory footprint and accelerating execution. From the observations made through the different use cases, the next section suggests to design and implement specialized compression pipelines to serve the different use cases effectively. The final section concludes this article.

2. Progress in lossy compression technologies

The past 5 years have seen exceptional progress in the consideration of lossy compression for scientific data sets. This is due mainly to the significant evolution of the design of lossy compressors for scientific data. In the past, lossy compressors for scientific data focused almost exclusively on data reduction for visualization. The lossy compressors used techniques directly inherited from lossy compression of images such as variations of wavelet transforms, coefficient prioritization, and vector quantization (Goldschneider, 1997; Li et al., 2018b). Lossy compressors for image processing are designed and optimized considering human perception. While such compressors may be adequate for scientific visualization, they do not provide enough compression error control to address the supercomputer limitations discussed in the Section 1. For example, most lossy compressors for visualization do not provide a global upper bound on the compression error (the maximum compression error, or $L\ \infty$ norm of the compression error). Our experience and discussions with application developers and users indicate that a strict user-set pointwise error control is needed for the data analysis with lossy data sets (after simulation or in situ) and for the execution restarting from lossy states and calculation from lossy data in memory. This is precisely what the new generation of lossy compressors for scientific data (SZ (Di and Cappello, 2016;

Liang et al., 2018; Tao et al., 2017c), ZFP (Lindstrom, 2014), and MGARD (Ainsworth et al., 2018)[1]) has introduced in the past 5 years, opening multiple new research directions. The use cases presented later in this article are the currently identified usages of lossy compression for scientific data.

2.1. General architecture of lossy compressors strictly respecting error bounds

Only a few lossy compressors are currently known and have been tested to respect strictly user-set error bounds. In the following paragraphs, we describe the SZ and ZFP compressors, which have been extensively tested and, in some cases, have been pushed to their limits in order to understand the nature of their compression error.

SZ and ZFP follow the classic structure of lossy compressors: they are multistage compressors featuring decorrelation, quantization, and encoding stages. In ZFP, the quantization and encoding stages are combined and form an embedded coding stage. SZ and ZFP use different decorrelation, quantization, and encoding techniques to compress scientific data sets and respect user-set error bounds. Both compressors compress data sets block by block. ZFP has been implemented initially for 3-D blocks; SZ can compress 1-D, 2-D, and 3-D blocks. The 3-D block size is different for SZ and ZFP; the size has been established after an optimization process related to their specific compression approach. Both ZFP and SZ allow random access decompression: each block can be decompressed individually. SZ and ZFP support absolute user-set error bounds. SZ also supports directly relative user-set error bounds, whereas users need to use ZFP in its fixed-rate mode to control relative error bounds.

SZ relies on prediction for the decorrelation stage. Currently, three types of predictors are selectable in this stage: Lorenzo, regression, and pattern. All prediction schemes produce predicted values that are compared with the actual values. For each value, the difference between the predicted and the actual value is quantized by using a linear scale (linear quantization). Each bin of the scale has a size equal to twice the user-set error bound. SZ strictly respects the user-set error bound (absolute or relative) because of (i) the prediction scheme that uses previously predicted values instead of actual ones and (ii) the quantization scheme that measures the prediction error as a multiple of the user-set error bound. ZFP combines a transform-based decorrelation scheme with an embedded coding scheme. The decorrelation stage operates on fixed 3-D blocks. Before applying the transforms, all the values within each block are aligned to the same exponent.[2] This allows the transform to work on integer values. The transform by itself uses custom coefficients established from extensive experiments and comparisons on many different data sets. The embedded coding scheme performs the quantization and coding simultaneously. Removing low-energy coefficients is a solution adopted by many transform-based lossy compressors, JPEG for example. But with this solution, the compressor cannot respect user-set error bounds. ZFP's solution is to truncate the precision of the coefficients based on the error bound. This leads to using a higher number of bits for high-energy coefficients and a lower number of bits for low-energy coefficients. ZFP guarantees the respect of the user-set error bound, introducing in each coefficient an error that is much lower (typically 1 order of magnitude) than the user-set error bound.

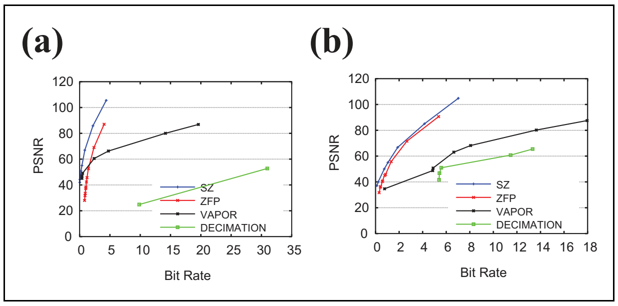

Figure 1. Comparison of rate–distortion based on different lossy compression strategies on (a) CESM-ATM and (b) NYX data sets. CESM: Community Earth System Model. The results about SZ, ZFP and VAPOR are extracted from (Liang et al., 2018).

2.2. Comparison with wavelet-based compressor and decimation

Until few years ago, lossy compressors for scientific data were only mostly used for visualization. It is only because of the recent progress in lossy compressor performance and the pressing needs to reduce data sets that other use cases have appeared. To illustrate the performance progress made by recent lossy compressors for scientific data (floating-point values), we compare SZ and ZFP with VAPOR (Clyne et al., 2007) (a wavelet-based compressor without user-set error control, published in 2007) and decimation in space (a technique often used in visualization that does not provide user-set error control). Specifically, the decimation-in-space scheme performs downsampling of the data set for compression and reconstructs the missing data by tricubic interpolation. We use two types of data sets representative of large classes of applications. The first data set is CESM-ATM. It is a 2.5-D data set (pseudo-3D) generated from a climate simulation code that uses a variation of computational fluid dynamics. The second data set is from the NYX cosmology application. NYX performs particle simulation using a combination of adaptive mesh hydrodynamics and an N-body cosmological simulation code. The data set produced by NYC simulation is 3-D.

Figure 1 presents the rate–distortion graphs for these four lossy compression techniques. Rate–distortion graphs show peak signal-to-noise ratios (PSNRs) in decibels (DB) for different bit rates (how many bits in the compressed format represent, on average, each value in the uncompressed data). The PSNR reflects an average compression error. It is computed from the root mean square error. Note that since PSNR is in DB, the y-axis in the graph is in log scale: linear differences are much larger than the ones shown in the graphs.

From Figure 1, we can observe that recent error-bounded generic compressors exhibit a lower distortion (higher PSNR) than do previous compression techniques. Since the rate–distortion graphs indicate the average compression error, they do not tell us the maximum compression error. We know that SZ and ZFP respect user-set error bounds. Since VAPOR and decimation do not provide user-set error control, there is a risk that they exhibit a much higher max compression error than do SZ and ZFP. To confirm this issue, we compare the max error of the four compression techniques for a single field (velocity_x) of the NYX data set and a single compression ratio (CR), 32. For similar CRs (20 for VAPOR, 27 for decimation), the maximum relative errors (maximum error over value range) of VAPOR and decimation are 0.015 and 0.25, respectively, while those of SZ and ZFP are 7e−4 and 3.5e−3. Experiments on other fields of NYX and other data sets lead to the same conclusion: not only does the new compression software achieve better rate–distortions, but the software also has lower max errors for similar CRs.

Comparison of recent lossy compressors with time-based decimation reaches the same conclusion (Li et al., 2018a).

These accuracy progresses in lossy compression technologies were indispensable for the adoption by scientific users of lossy compression beyond visualization.

3. Lossy compression use cases

In this section, we describe the seven identified use cases of lossy compression for floating-point data in scientific data sets. For the classic visualization use case, we show the progress provided by recent lossy compressors. For other user cases, we show how lossy compression can respond to the limitations in supercomputers and instruments mentioned in the Section 1.

For each use case, we describe the motivations for using lossy compression, the specific constraints in terms of compression rate, ratios, error control, and experimental results. Our objective is not to compare the performance of recent compressors (specifically SZ and ZFP). Rather, it is to show their applicability and suitability to the different use cases.

Most of the experiments presented in the following section were performed on the Argonne National Laboratory Bebop cluster that features 8192 cores (i.e. 256 nodes; each node has two Intel Xeon E5-2695 v4 processors and 128 GB of memory, and each processor has 16 cores). The Bebop storage system adopts General Parallel File System (GPFS), which is located on a raid array and served by multiple I/O nodes. The I/O and storage systems are typical high-end supercomputer facilities.

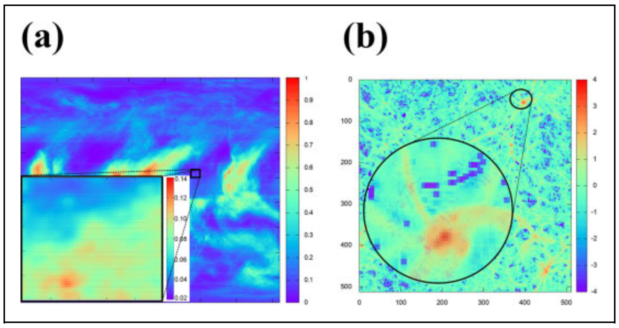

Figure 2. Visualization of original raw data. (a) CESM-ATM (CLDHGH) and (b) NYX (dark matter dens). CESM: Community Earth System Model.

3.1. Visualization

Visualization is an important topic in the scientific simulation community. It usually serves as the first and foremost step in the post-analysis of the simulation results by researchers. On the other hand, since human cognitive systems can tolerate visual distortion to some extent, the result for visualization does not need to be exact. However, there are no general constraints on the acceptable distortion level since they are really application dependent. For instance, the minimal allowable structural similarity index (SSIM) for medical image compression ranges from 0.95 to 0.99 to achieve "diagnostically lossless" compression (Baker et al., 2017), while the *just-noticeable distortion* (JND) profile indicates that a PSNR around 30 is unnoticeable for general image/video processing (Wei and Ngan, 2009). Also, since scientific visualization is often applied with the post-processing where I/O is the bottleneck (Li et al., 2018b), the time constraints of visualization are similar to those on reducing I/O time, which is introduced later.

In this case, lossy compression can be applied to reduce the data size and transfer time while maintaining acceptable visual quality, similar to what JPEG (Wallace, 1992) does for 2-D images. Error-bounded lossy compression, in addition, provides users with means to control the error (which affects the final distortion); hence, it is able to offer user-guided compression given a desired reduction size or visual quality. Moreover, many researchers (Ballester-Ripoll et al., 2018; Liang et al., 2018; Lindstrom, 2014; Tao et al., 2017c) assess the compression quality in terms of visualization-related metrics such as PSNR. More detailed discussions of the lossy compression techniques regarding the visual quality can be found in Li et al.'s (2018b) survey.

In what follows, we present the visualization quality of four lossy compressors (VAPOR (Clyne et al., 2007), TTHRESH (Ballester-Ripoll et al., 2018), ZFP (Lindstrom, 2014), and SZ (Di and Cappello, 2016; Liang et al., 2018; Tao et al., 2017c)) under different reduction sizes to illustrate the effectiveness of lossy compression in terms of visualization. This comparison also illustrates the progress of the recent generation of lossy compressors

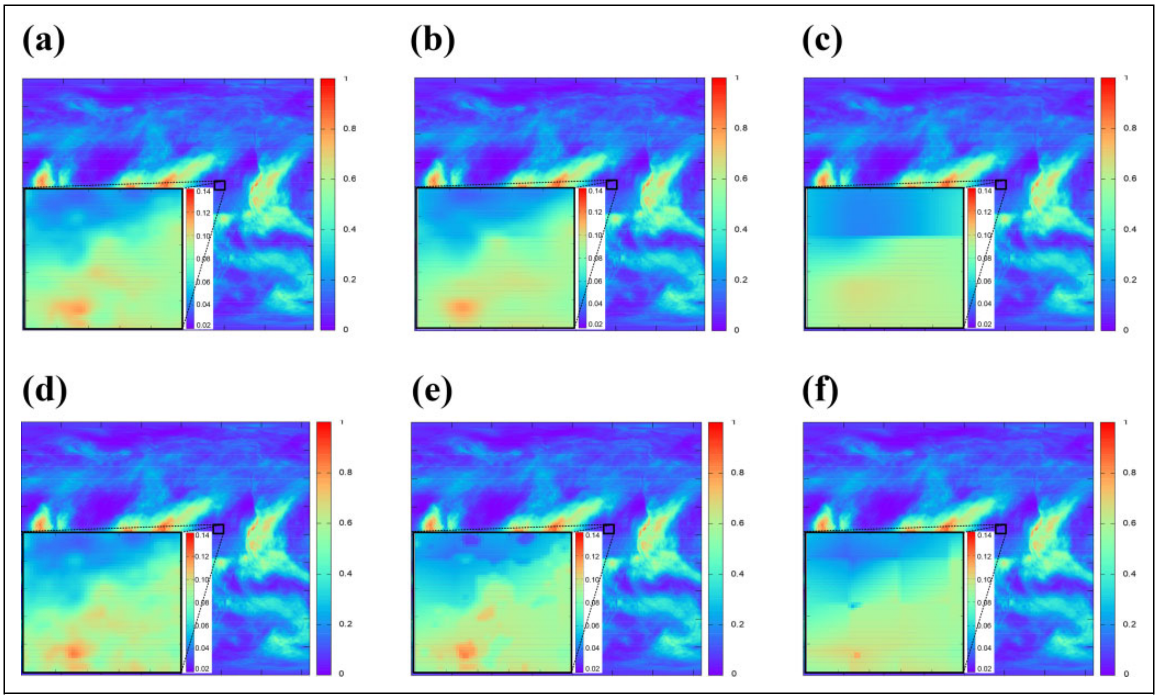
Figure 3. Visualization of the decompressed data on CLDHGH field in CESM-ATM. (a) VAPOR, CR = 9, PSNR = 52, SSIM = 0.9987. (b) VAPOR, CR = 37, PSNR = 46, SSIM = 0.995. (c) VAPOR, CR = 138, PSNR = 38, SSIM = 0.9845. (d) SZ, CR = 9, PSNR = 75.3, SSIM = 1. (e) SZ, CR = 38, PSNR = 49, SSIM = 0.997. (f) SZ, CR = 137, PSNR = 43.2, SSIM = 0.9911. CESM: Community Earth System Model.

(SZ, ZFP, TTHRESH) compared with an older compressor (VAPOR).

We first show the CLDHGH field in the CESM-ATM data set as an example of 2-D cases (CESM-ATM is a 2.5-D data set; here we use only one vertical slice). The visualization of the original 1800 × 3600 data is displayed in Figure 2(a) with enlarged details in a local 50 × 50 region. Since TTHRESH does not support 2-D data sets and ZFP is better optimized for 3-D data sets, only VAPOR and SZ are evaluated given similar reduction size (CRs of 9, 37, and 138). The visualization results are shown in Figure 3, with two popular visualization metrics, PSNR and SSIM, listed below each subfigure (SSIM is the structural similarity index that compute a value from three components: luminance, contrast, and structure). According to Figure 2, both compressors lead to almost no difference for visualization under the value range of the whole data set even when the CR goes up to 138. However, a significant difference can be seen when we zoom in on the local region. Figure 3(a) to (c) and Figure 3(d) to (f) indicate that the details in the local region disappear for both compressors as the CR increases. When the CR is around 9 (Figure 3(a) and (d)), the visualization looks similar to the raw data, even for the data in the local region. For the last compression CR (Figure 3(c) and (f)), the block artifacts manifest in the visualization are due to the blockwise design in the compression algorithm (128 × 128 block for VAPOR and 16 × 16 block for SZ). Nevertheless, these cases are still useful if only the visualization of the whole data set is needed. Also, users who desire more accurate results can switch the desired ratio (VAPOR) or the error bound (SZ) to lower CRs for better quality, thus showing the flexibility of lossy compression.

We then show the dark_matter_density field in the NYX data set as an example for 3-D data. We visualize the 256th slice of the original 512 × 512 × 512 data for demonstration purposes. Similar to the 2-D example, the visualization of the original data is displayed in Figure 2(b), and those visualizations of the decompressed data from different lossy compressors are shown in Figure 4 with PSNR and SSIM. We note that PSNR is computed over the entire data set whereas SSIM is calculated only for the current slice in this case. We also fixed the CRs of different lossy compressors to a similar level, namely, 8, 32, and 128. They correspond to the three rows in the figure, respectively. From a top-down view, we can observe a similar pattern in that the visualization quality drops as the CRs increase. Of interest is how the visualization quality changes as the CR differs across compressors, due to the difference in the compression algorithms. For example, the visualization results of VAPOR (Figure 4(a), (e), and (i)) become more and more blurry because of more discarded wavelet coefficients, but it is able to keep a rough shape because the coefficients with the highest priority are kept. On the other hand, ZFP has a block mosaic effect when the CR is high (Figure 4(k)) because of the exponent alignment in each 4 × 4 × 4 block.

In summary, general-purpose visualization of whole data sets does not require high accuracy for acceptable

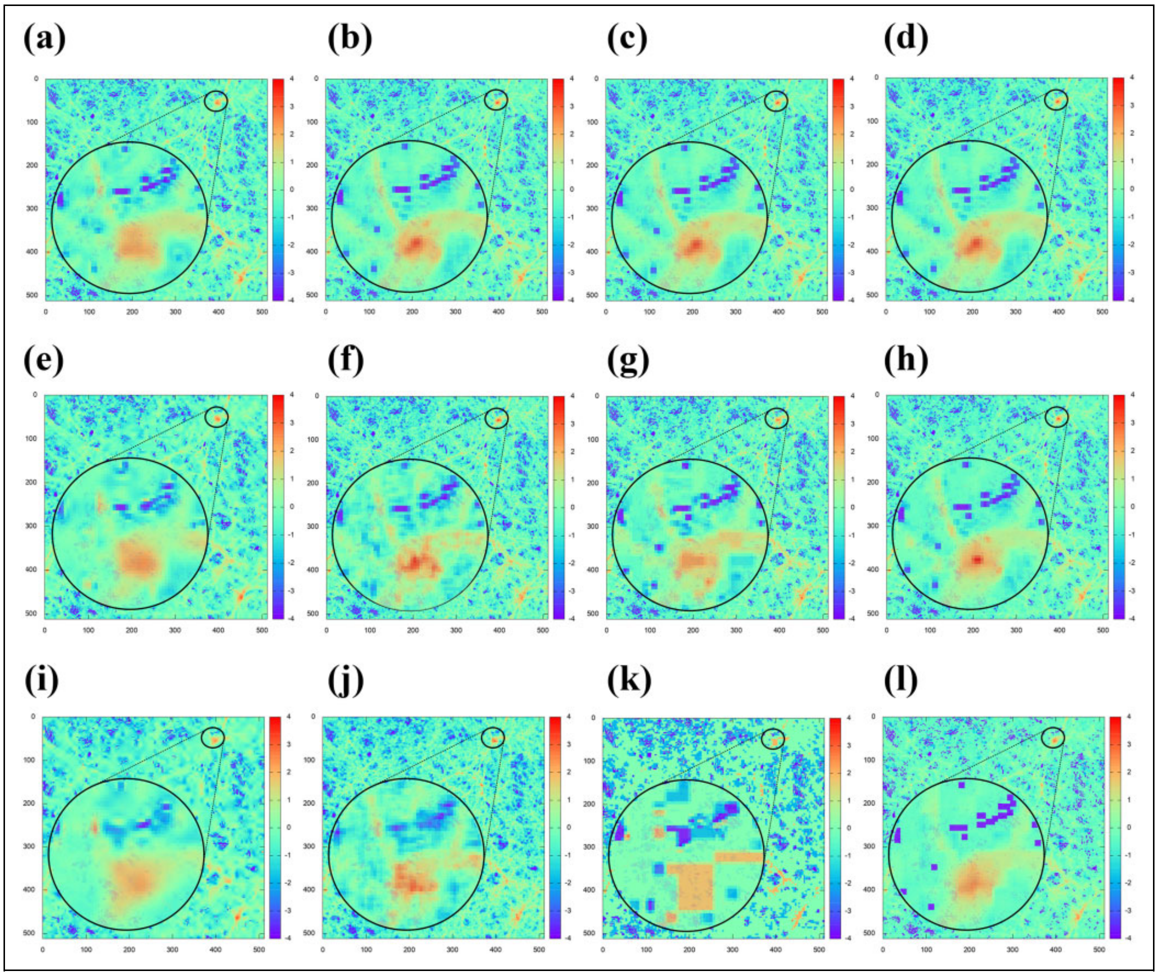

Figure 4. Visualization of the decompressed data (slice 256) of the dark_matter_density field in NYX. (a) VAPOR, CR = 9.65, PSNR = 26.8, SSIM = 0.469. (b) TTHRESH, CR = 8.2, PSNR = 39.7, SSIM = 0.9051. (c) ZFP, CR = 7.6, PSNR = 44.3, SSIM = 0.9736. (d) SZ, CR = 7.9, PSNR = 49.2, SSIM = 0.9896. (e) VAPOR, CR = 38.6, PSNR = 22.6, SSIM = 0.2552. (f) TTHRESH, CR = 30, PSNR = 25.3, SSIM = 0.4841. (g) ZFP, CR = 34.3, PSNR = 24.8, SSIM = 0.4483. (h) SZ, CR = 31.4, PSNR = 32.8, SSIM = 0.6784. (i) VAPOR, CR = 154, PSNR = 20.6, SSIM = 0.1384. (j) TTHRESH, CR = 130, PSNR = 21.5, SSIM = 0.2767. (k) ZFP, CR = 110, PSNR = 19.5, SSIM = 0.1655. (l) SZ, CR = 127, PSNR = 20.6, SSIM = 0.3937.

results meeting the JND, which in turn leads to high CRs. However, for some special cases, such as medical images or scenarios where zoom-in is needed, higher accuracy is preferred. Error-bounded lossy compressors could fit in both cases because of its error-bounded nature, requiring less storage or transfer time while maintaining acceptable accuracy.

3.2. Reducing data stream intensity

Reducing data stream intensity is a typical need of large-scale instruments such as accelerators and telescopes, and several lossy compression techniques have been proposed or tested specifically for these instruments (Nicolaucig et al., 2003; Offringa, 2016; Pataumer, 2011; Peters and Kitaeff, 2014; Röhrich and Vestø, 2006; Vohl et al., 2015, 2017). In this section, we focus on light source facilities such as the Advanced Photon Source (APS) (Fornek, 2017) (a synchrotron-radiation light source research facility at Argonne) and the Linear Coherent Light Source (LCLS) (Marcus et al., 2015) (a free-electron laser facility located at SLAC). Using the high-brilliance X-ray beams generated from the light source facilities, scientists can conduct basic and applied research in many fields including biological science, materials science, chemistry, physics, geophysics, environmental science, and planetary science.

The light source facilities generate extremely large volumes of data. Usually, the raw data cannot be stored because of the limited I/O bandwidth and storage space. Therefore, in order to transfer and store the data, significant data reduction is needed. Whereas for other scientific instruments generating extreme volumes of data, such as the Large Hadron Collider, the data reduction is performed by ad hoc techniques and infrastructures specific to the nature of the instrument and experiments, researchers involved in the design of the LCLS-II data reduction infrastructure are investigating the potential use of available, generic lossy compressors.

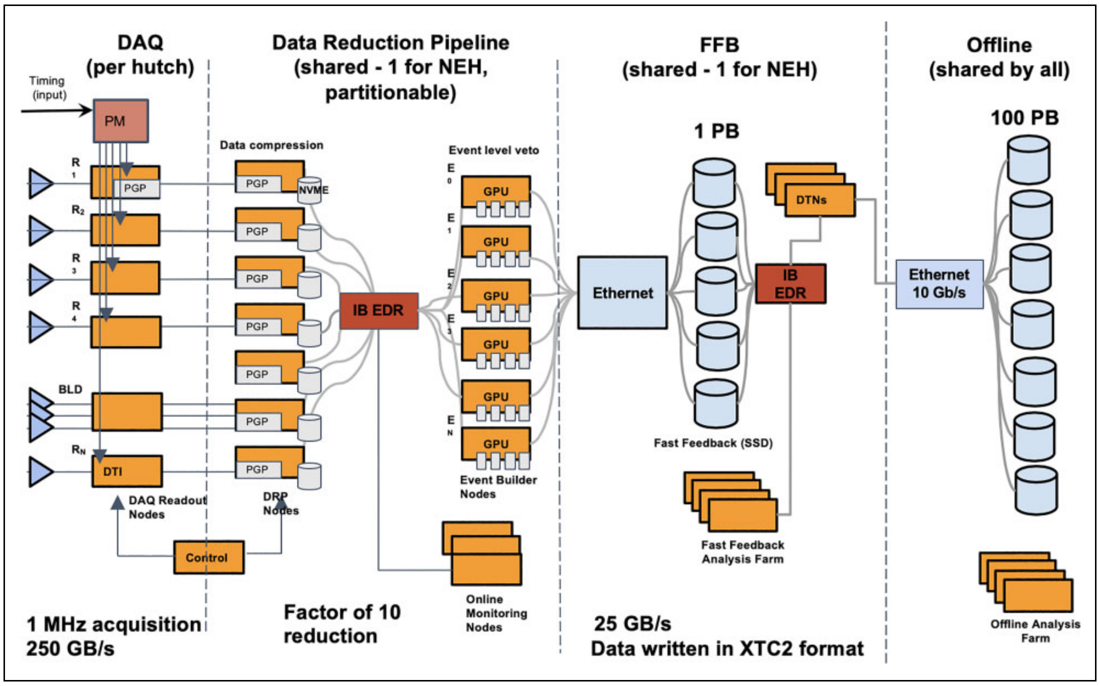

Figure 5. Data system for the LCLS-II (extracted from Thayer et al. (2017)). LCLS: Linear Coherent Light Source.

Figure 5 presents the data system for the LCLS-II. The raw data acquisition rate is 250 GB/s. Storing the raw data without reduction would require thousands of discs to sustain the data production rate. The designers of the LCLS-II data system have fixed a data reduction objective of 10 to reduce the required SB to 25 GB/s. The requirement in terms of compression error is 30 analogue-to-digital units (ADUs), which corresponds to 30 in integer (16-bit) representation.

For such applications, the lossy compressors should have both medium CRs (10) and very high compression speed ($>$ 250 GB/s), such that the overall data compression rate is higher than the data production rate. The requirement relative to the absolute error can be translated to a medium-accuracy requirement of 10^{-3} relative to the value range. This set of constraints is challenging for available lossy compressor software even if the software is used in a cluster of nodes equipped with GPUs.

In what follows, we present some early experimental results based on LCLS-II crystallography analysis data using two state-of-the-art lossy compressors, SZ and ZFP. They exhibit better compression results on this data set than do other lossy compressors such as FPZIP (Lindstrom and Isenburg, 2006) and ISABELLA (Lakshminarasimhan et al., 2011), which also do not support absolute error controls required by the application users. The original data set (called calibrated data) was stored in the form of integers (in 4 dimensions, $10 \times 32 \times 185 \times 388$), which we convert to floating-point values in our experiment for fairness, because floating-point data compression is the optimized mode for the two compressors. The visualization of the crystallography analysis data is demonstrated in Figure 6.

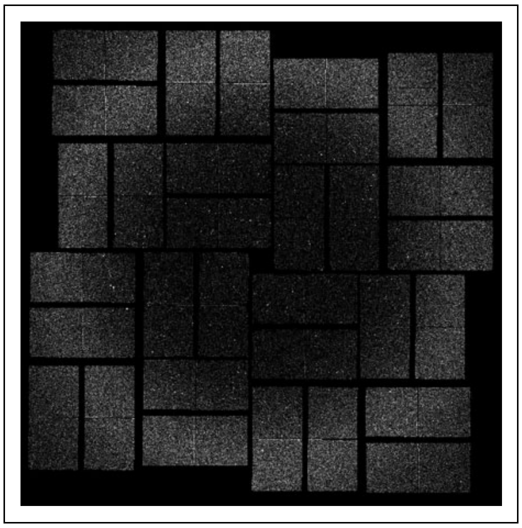

Figure 6. Visualization of LCLS-II crystallography analysis data (more details can be found in Yoon et al. (2017) and Mark et al. (2016)). LCLS: Linear Coherent Light Source.

We present the compression results in Table 2. The compression was conducted on one node of the Bebop cluster (LCRC Bebop cluster, 2018) at Argonne National Laboratory, indicating the single-core compression performance. We compress the data by treating it as a 2-D array and a 1-D array, respectively, in that we observe that 1-D-style compression will lead to higher compression quality probably because of the nonsmoothness feature of the data in space. From the table, we can observe that when setting the absolute error bound to 30 ADU for SZ 2.0, the CR is slightly better when treating the data set as a 1-D array than as a 2-D array. By comparing the error bound setting and

Table 2. Compression results on crystallography data.

Compressors	Error bound	Max error	CR	PSNR	Compression rate
SZ 2.0 (2-D)	30 ADU	30 ADU	7.73	71.6	121 MB/s
SZ 2.0 (1-D)	30 ADU	29.5 ADU	8.04	71.6	107 MB/s
ZFP 0.5 (2-D)	30 ADU	30.25 ADU	4.46	81.5	184 MB/s
ZFP 0.5 (1-D)	30 ADU	47 ADU	3.86	75.6	121 MB/s

CR: compression ratio; PSNR: peak signal-to-noise ratio; ADU: analogue-to-digital unit.

real maximum compression errors, we can see that the ZFP compressor does not respect the error bound. We also observe that ZFP is 50% faster than SZ, while SZ has about $2\times$ higher CRs. Considering that the original data are stored in 16-bit integers while the floating-point values used in our experiments are in 32-bit format, the real CRs is half of the numbers shown in the table.

In summary, Table 2 shows promising results concerning the CR (close to 10 for SZ 2.0 in 1-D) of generic lossy compression for the LCLS-II data. A significant gap still remains, however, concerning the compression rate: At the current compression rate, about 2000 cores will be needed to sustain the data acquisition rate. A collaboration between LCLS-II and the SZ team is exploring solutions to improve the CRs and rates.

3.3. Reducing the storage footprint

Some scientific simulations (e.g., geoscience and fluid dynamics) may involve large problem sizes with long simulation timescales, in order to exploit as many scientific findings as possible. These simulations generate data for post-execution analysis that includes complex structure calculations and feature finding. At extreme scale, they would generate overwhelming volumes of data if all the data produced during their execution (over multiple iterations) was to be saved. However, the limited storage of supercomputers hinders scientists from running such simulations or from saving all the produced data. Adding more storage units to a supercomputer is not a practical solution because of the expense and the superfluous storage for other high performance computing (HPC) applications that do not need as much storage.

Thus, to reduce the storage of scientific simulations, researchers have developed lossless compression algorithms (Engelson et al., 2000). However, lossless compression has limited compression, a factor of at most 2 in most cases (Son et al., 2014) because of the random mantissa bits. Such a low CR is not enough for the high volume of simulation data. Therefore, researchers have been exploring lossy compressors.

Since the post-analysis of scientific simulation results generally allows a certain level of data distortion, error-controlled lossy compression is a good choice to significantly shrink the simulation data, and thus it will greatly mitigate the demand for extremely large capacity of storage systems on a supercomputer. With lossy compression, one can run a simulation with a much larger problem size and/or more timesteps/snapshots on the existing HPC storage system without any hardware modifications. Also, lossy compression can effectively save the cost paid for the use of storage. In the following, to demonstrate the benefits of reducing the storage footprint, we present two typical real-world simulations: climate and cosmology.

In climate simulation, the Community Earth System Model (CESM) is the most widely used simulation code (Hurrell et al., 2013). In the CESM set of simulations, the CESM-Large Ensemble (CESM-LE) project involves large-scale and 180-year climate simulations at high resolution. Such simulations produce huge amounts of data. Because of storage constraints, CESM-LE users have to enlarge the time period for output between every two adjacent simulation steps. Moreover, the simulation results have to be deleted monthly. Specifically, in terms of data volume and the storage limits, the first 30 ensemble member CESM-LE simulations produce more than 300 TB of data, of which only 200 TB could be kept in the disk (Kay et al., 2016). With the help of lossy compression, CESM-LE could run at larger scale in longer simulations time, which clearly is beneficial to more scientific investigations.

This case uses lossy compression in an online fashion, which means that lossy compression should be incorporated into the CESM-LE simulation. We then show that offline lossy compression is also meaningful for CESM-LE simulation data. Since the raw simulation data would take a huge amount of storage and since users have to pay for the storage they use, using lossy compression can shrink the data size and save the cost of storage. Furthermore, lossy compression is suitable for users who need to make space for other application executions without deleting the existing simulation results.

In cosmology simulation, the Hardware/Hybrid Accelerated Cosmology Code (HACC) (Habib et al., 2016) has been designed to enable portability across diverse computing platforms and scalability at millions of cores. HACC starts simulating the later half of the history of the universe from 50 million years after the Big Bang. It has been able to run at the trillion-particle scale. However, such simulations produce about 500 snapshots with more than 40 TB of data per snapshot. Since the total storage capacity of Mira is 24 PB and the full data of a trillion-particle simulation would occupy 20 PB, users need to reduce the data significantly. They currently use decimation in time, keeping only 1 snapshot in 5 or even 1 snapshot in 10. To reconstruct the missing snapshots, users rely on interpolation. However, recent results show that decimation in time generates much more errors and much higher error rates than does compression for a given data reduction ratio (Li et al., 2018a). For exascale simulations, the number of involved particles is projected to be 125 trillion, which will produce 5 PB of data per snapshot (Habib, 2017). The full data of a 500-iteration execution would require more than 2 EB, while projected

Table 3. CRs on CESM and HACC data (higher PSNR means lower average error).

Compressors	PSNR = 43	PSNR = 55	PSNR = 67	PSNR = 86
CESM-ATM				
SZ	350	70	35	14
ZFP	29	22	14	8
Compressors	PSNR = 65	PSNR = 85	PSNR = 105	PSNR = 125
HACC-x				
SZ_vlct	118	36	13	6
SZ	29	10	5	3

CESM: Community Earth System Model; HACC: Hardware/Hybrid Accelerated Cosmology Code; PSNR: peak signal-to-noise ratio. Extracted from our conference papers Liang et al., 2018; Li et al., 2018a.

exascale systems would provide on the order of .5 EB of storage. Thus, lossy compression can facilitate the success of exascale cosmological simulation.

We now discuss the constraints on the lossy compressors when they are used for reducing storage in offline mode. (The online mode has different constraints, which will be addressed in the next section.) The compressors should have CRs of tens, but they can be slow since the primary goal is to reduce storage. However, the lossy compression should have high accuracy for post-analysis depending on the simulation and the analysis. For example, in the evaluation of lossy compression effects on CESM simulation data (Baker et al., 2017), the absolute error bounds can be small, on the scale of 10^{-6}. Table 3 shows the CRs of several state-of-the-art lossy compressors (Liang et al., 2018; Li et al., 2018a) for CESM and HACC simulation data. The results are obtained by changing the absolute error bound relative to value range. Lower error bound results imply higher PSNR values. For HACC data, we include the results only in the position field, x. We can see from the table that lossy compressors are efficient in terms of CR for CESM and HACC data—more than 100 in some cases. For HACC data, although SZ_vlct has only one-third of the compression rate compared with that of SZ, here for the goal of reducing storage we prefer SZ_vlct to SZ because of its much better CR.

3.4. Reducing I/O time

The I/O bottleneck is becoming a serious issue because of the ever-increasing volume of data produced by today's HPC scientific simulations at runtime and the limited I/O bandwidth of the parallel file system (PFS). Our experiments (Liang et al., 2018; Li et al., 2018a), for instance, indicate that the I/O bandwidth of the Bebop cluster (LCRC Bebop cluster, 2018) at Argonne National Laboratory is only 1–2 GB/s when simultaneously writing/reading a large amount of data by different ranks to/from the PFS. In this case, the total data writing time will go up to hours if the total size of the data to store is 10+ TB. Such a volume could be easily reached since one HPC simulation may involve a large number of ranks/cores, each producing a portion of simulation data. Suppose there are 40k ranks in a simulation and each rank may produce 2 GB of data to store; then the total data size would be about 80 TB. Although other more powerful supercomputers have higher I/O bandwidths in their PFS than the Bebop cluster does, they still have an upper-bound I/O bandwidth on the data writing/reading because of the separate I/O devices (such as I/O racks, drawers, or nodes) deployed, and such an upper bound would still become a bottleneck when the volume of data produced by simulations is large enough. For instance, according to cosmological simulation users and developers at Argonne, one HACC (Habib et al., 2016) simulation may simulate trillions of particles through 500 snapshots, which means dozens of petabytes of data to be produced during the simulation.

In order to improve the I/O performance, many existing I/O libraries designed for scientific data sets allow compression of the data before writing it to the PFS. The HDF5 library (HDF5 Library, 2018), for instance, allows users to specify a lossless or lossy compression filter (HDF5 Filter Plugin, 2018) when storing the data in the HDF5 format. The compression filter will call a specific lossless or lossy compressor (such as Gzip (Deutsch, 1996), Zstd (Zstandard compressor, 2018), SZ (Di and Cappello, 2016; Liang et al., 2018; Tao et al., 2017c), or ZFP (Lindstrom, 2014)) to perform the compression before the data dumping and perform the decompression automatically during the data loading, actions that are totally transparent to users.

Compared with lossless compressors, error-controlled lossy compression techniques can significantly reduce the total size of the data to be stored during the simulation with respected data distortion, thus effectively reducing the runtime data dumping time as well as the data loading time for post-analysis. The total data-dumping time can be approximated as the total compression time plus the total time of writing the compressed data. Obviously, the important constraint of reducing I/O time by compression techniques for HPC simulations is that the total data dumping/loading time should be less than the time of writing/reading the original data. This constraint involves many factors such as compression/decompression time, CR, and I/O bandwidth.

Note that the compression operation could be performed by each rank in parallel without any communication; the total compression time is equal to the maximum compression time on one rank. As such, the compression time would be negligible when the execution scale is very large (such as 10k+ ranks). The total data-dumping time is actually dominated by the data writing time for a large-scale simulation with vast volumes of data to write/read, while it is dominated by the compression time for a small-scale simulation in general. This situation has been observed in many studies based on real-world simulations (Liang et al., 2018; Li et al., 2018a; Tao et al., 2017a, 2017b, 2017c).

In the following, we further illustrate the use cases of adopting lossy compressors to reduce I/O time, using the experimental results with real-world simulation data (called NYX) from the cosmology research project.

The experiments are conducted on Bebop. For reading/writing data in parallel, we adopt file-per-process mode with POSIX I/O (Welch, 2005) on each process.[3]

We evaluate the overall performance of dumping/loading data in the NYX simulation using various state-of-the-art lossy compressors with the same data distortion level. Specifically, the PSNR is set to 60 for each field except for dark matter density (PSNR = 30) and baryon density (PSNR = 40), since such a setting already reaches a high visual quality. We conduct a weak-scaling evaluation in which each rank processes 3 GB data and the total data size increases linearly with the number of cores. We assess the performance by running different scales (2048 cores–8192 cores). The original uncompressed data size is about 24 TB when using 8192 cores, which may cost over 6 h to store on the PFS. We present the breakdown of the data-dumping performance (sum of compression time and data-writing time) and data-loading performance (sum of data-reading time and decompression time) in Figure 6 for the lossy compressors (SZ 1.4, ZFP, and SZ 2.0) with the best CRs on this data set, in order to clearly observe the performance difference. Other compressors such as VAPOR (VAPOR, 2018), FPZIP (Lindstrom and Isenburg, 2006), and ISABELA (Lakshminarasimhan et al., 2011) would cause much longer time because of much lower CRs.

We observe that the overall data-dumping times with the three compressors can be reduced significantly compared with the original data-writing time (about 6 h). The evaluation results (Figure 7) also show that different compressors may lead to different performance. Under SZ 2.0, for example, it takes only 24% and 54% of the time cost by SZ 1.4 and ZFP 0.5 when adopting 8192 cores, which correspond to 4.12X and 1.86X performance gain, respectively. The key reason is that SZ 2.0 leads to 1.5–8X higher CRs than do the other two compressors with the same PSNR in the range of [30,60] dB, as shown with SZ 2.0 results in Figure 1(b). SZ 2.0 can also obtain 1.95X higher data-loading performance (49% lower time cost) than the second best solution (ZFP 0.5) does, when running the experiment with 8192 cores. It is slightly higher compared with the data-dumping performance (1.86X) because of the higher decompression rate than the compression rate. We note that ZFP 0.5 is generally faster than SZ 2.0 when compressing the data set, so its overall data dumping/loading performance is higher than that of SZ 2.0 when the data size is relatively low (such as running small-scale simulation with fewer than 64 ranks/cores each producing 2 GB of data). The reason is that the overall I/O performance is dominated by the compression/decompression time in the small-scale simulation with relatively small amounts of data to write/read, as discussed previously.

3.5. Accelerating checkpoint/restart

With ever-increasing scales of scientific research problems, scientists need to run their applications on hundreds of thousands and even millions of cores in parallel. At these scales, resilience is a critical issue, because experiencing interruptions or failure events without fault tolerance would mean the loss of the execution products and a drastic waste of resources and energy. The existing fault tolerance strategy used in HPC environments is already the source of considerable performance overhead. Extreme-scale application executions on future and larger systems will need dramatic improvements in the fault tolerance strategy in order to keep its performance overhead acceptable.

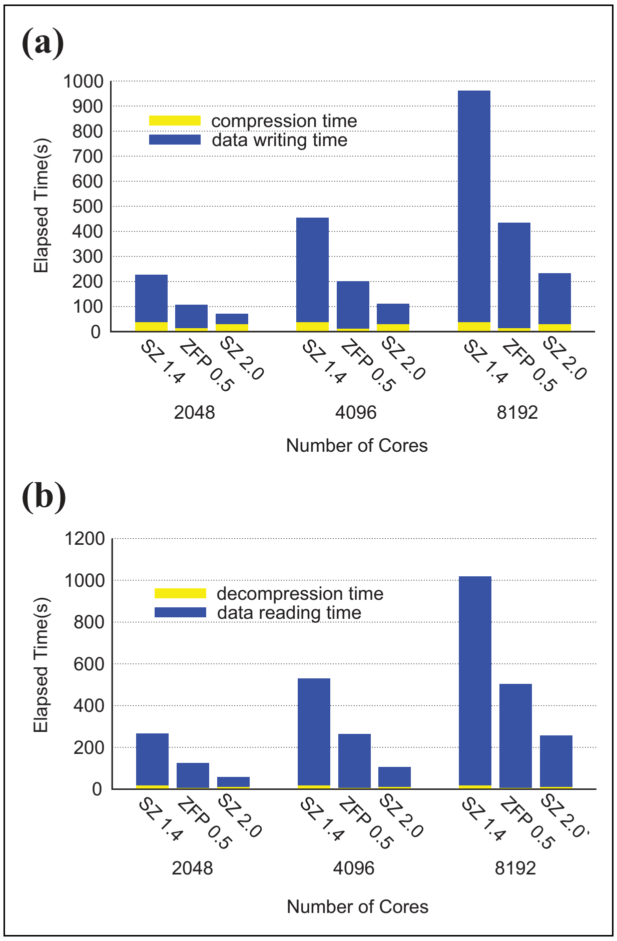

Figure 7. Performance evaluation using NYX (extracted from our conference paper (Liang et al., 2018)). (a) Data dumping performance and (b) data loading performance.

The current approach used by most applications is checkpoint/restart. The role of checkpointing is to capture and save the execution state in a reliable storage system. Upon failure, the execution is restarted from the checkpoint accessed on the storage system. File systems are designed to be extremely reliable, and they represent a place of choice to save checkpoints. However, while the memory of extreme-scale systems continues to grow (by a factor of 5 or more for the next generation of systems compared with the current one), the file system bandwidth is increasing relatively slowly (as shown in Table 1), meaning that saving application state (that will be much larger since applications tend to use all available memory) on file systems will take much longer than in current systems.

Optimization strategies of checkpoint/restart model have been studied for decades. A multilevel checkpoint/restart model (Bautista-Gomez et al., 2011; Moody et al., 2010), for instance, was proposed to provide tolerance for different types of failures. Recently, a few studies have demonstrated the feasibility of using compression techniques to improve the checkpoint/restart performance. Islam et al. (2013) adopted data-aware aggregation and lossless data compression to improve the checkpoint/restart performance. Sasaki et al. (2015) proposed a lossy compression technique based on wavelet transformation for checkpointing and explored its impact in a production climate application. Calhoun et al. (2018) verified the feasibility of using lossy compression in checkpointing two specific PDE simulations experimentally. Their results show that the compression errors in the checkpointing files can be masked by the numerical errors in the discretization, leading to improved performance without degraded overall accuracy in the simulation.

Generally, in the context of checkpoint/restart with lossy compression, researchers have to address three questions: (1) How does the lossyness level of checkpoints correlate with application completion time, or even completion at all? (2) Can lossy checkpoint/restart provide significant performance gain compared with classic checkpoint/restart, not only for the time gained on I/O and storage operations and the time lost on compression/decompression, but also for the extra application iterations that might be needed to reach convergence from lossy recovery? (3) How can one optimize performance in the presence of lossy checkpointing and failures?

Previous work (Calhoun et al., 2018 Sasaki et al., 2015) studied a specific scientific application such as climate simulation or cosmology simulation. In comparison, we will illustrate how to use lossy compression for checkpoints in the context of fundamental iterative methods being used by the scientific community more widely. These methods are the Jacobi stationary iterative method, conjugate gradient (CG) method, and generalized minimal residual (GMRES) method. The following context will illustrate that lossy compression of checkpoints can significantly improve the overall performance for these popular iterative methods in numerical linear algebra.

For demonstration purposes, we use the sparse matrix arising from discretizing a 3-D Poisson's equation. We refer readers to (Tao et al., 2018) for the matrix details. We use the PETSc (v3.8) (Balay et al., 2018) library for GMRES and its default preconditioner (block Jacobi with ILU/IC). We set the relative convergence tolerance to 10^{-6}. For GMRES, we use PETSc's recommended setting 30 as its restarted step (i.e. GMRES(30)). For the lossy compressor, we adopt the SZ lossy compression library (v1.4.12) (Tao et al., 2017c). We use a relative error bound of 10^{-4} for all the experiments. We choose the Gzip (Deutsch, 1996) lossless compressor to represent the lossless compression for comparison with the lossy compression strategy. We call the checkpointing without any compression techniques *traditional checkpointing*, the checkpointing with lossy compression *lossy checkpointing*, and the checkpointing with lossless compression *lossless checkpointing*.

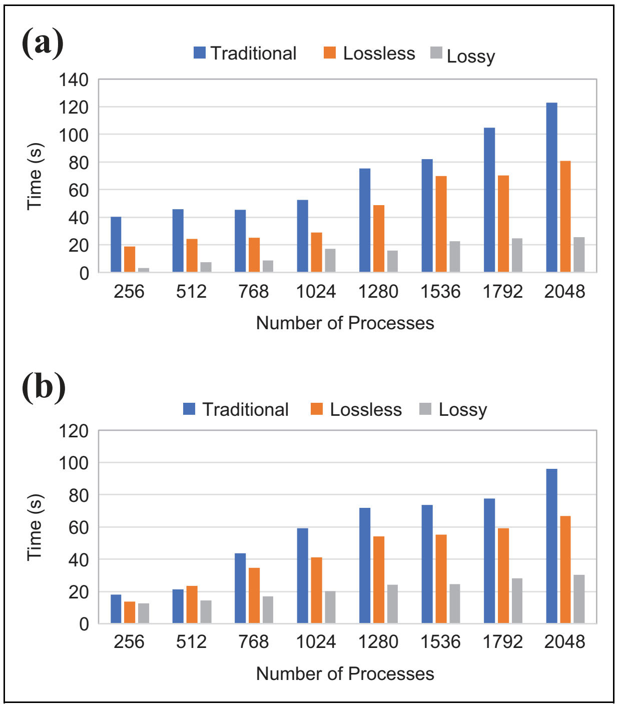

Figure 8. Average time of one checkpoint and recovery for GMRES with different checkpointing techniques (extracted from our prior conference paper (Tao et al., 2018)). (a) Checkpoint and (b) recovery. GMRES: generalized minimal residual.

We evaluate the lossy checkpointing technique for the iterative methods using 2048 processes/cores from the Bebop cluster (LCRC Bebop cluster, 2018) at Argonne National Laboratory. The checkpoints are stored on a parallel file system. We characterize the checkpointing and recovery overheads by running the iterative methods five times, with a total of about 80 checkpoints and 15 recoveries for each execution scale.

Figure 8 shows the average checkpointing time and recovery time for GMRES based on different settings: traditional, lossless, and lossy checkpointing. We observe that the checkpoint/recovery time can be reduced significantly by storing the checkpoint data compressed by the SZ lossy compressor, as compared with the other two solutions. In absolute terms, the lossy checkpointing time is only one-fifth of the traditional checkpointing time and about one-third to one-half of the lossless checkpointing time, because of inevitable significant I/O bottleneck when processing the large volumes of data on the parallel file system. Jacobi and CG simulations exhibit similar results for the checkpoint/restart cost.

Figure 9 shows the average overall extra time of running the three iterative methods (Jacobi, GMRES, and CG) with different checkpointing solutions with their corresponding checkpoint intervals using 2048 cores on Bebop in the

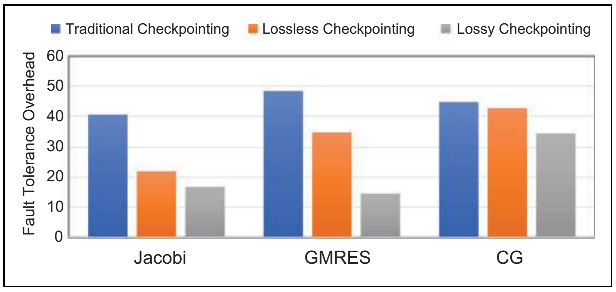

Figure 9. Fault-tolerant overhead of Jacobi, GMRES, and CG method with different checkpointing techniques. GMRES: generalized minimal residual; CG: conjugate gradient. Extracted from our prior conference paper (Tao et al., 2018).

presence of our injected failures. We observe that lossy checkpointing improves the overall performance of the iterative methods significantly. Specifically, for Jacobi, lossy checkpointing reduces the fault tolerance overhead by 59% compared with the traditional checkpointing and 24% compared with the lossless checkpointing; for GMRES, lossy checkpointing outperforms the traditional checkpointing and the lossless checkpointing by 70% and 58%, respectively; for CG, lossy checkpointing reduces the fault tolerance overhead by 23% and 20% compared with the traditional and lossless checkpointing, respectively. Note that for simplicity we used Young's formula (Young, 1974) to calculate the corresponding checkpoint intervals.

Lossy compression of checkpoints can be also exploited in context of iterative methods for nonlinear systems of equations. Lossy checkpointing, for instance, is currently being investigated for adjoint computations (Boehm et al., 2016; Kukreja et al., 2018).

The constraints on the lossy compressors when they are adopted for improving the performance of checkpoint/restart include the following aspects. First, the application still needs to reach the convergence from lossy recovery. This convergence can be because either compression errors in checkpoints were masked by numerical errors or compression errors caused extra application iterations even though they were not masked by numerical errors. Second, the performance degradation due to the extra application iterations must be mitigated by the time saved on I/O and storage operations. Third, in order to optimize the overall execution performance in the presence of failures, one needs to calculate the checkpoint intervals based on the revised performance model proposed in (Tao et al., 2018). According to the model, the compression and decompression speeds need to be able to be estimated based on the (compressed) checkpoint size and user-set compression error bound.

3.6. Reducing the memory footprint

Quantum circuit simulation is a good example of a use case for reducing memory footprint. In quantum computing research and development, using classical HPC systems to simulate quantum computers is important for understanding the operations and behaviors of quantum computing systems. Such simulations allow developers to evaluate the complexity of new quantum algorithms and validate the design of quantum devices. To simulate a quantum system of n quantum bits (qubits), one needs 2^n amplitudes to describe the quantum system. All the amplitudes are written as a vector, called state vector. Each amplitude is a complex number, represented by two double-precision floating-point data points, one for the real part and the other for the imaginary part. Thus, given n qubits, the state vector requires 2^{n+4} bytes. Since the number of quantum state amplitudes grows exponentially with the number of qubits in the system we want to simulate, the size of the quantum circuit simulation is limited by the memory capacity of the classical computing system. For example, in order to store the full quantum state of a 45-qubit system, the memory requirement is 0.5 petabytes. Quantum systems with more than 49 qubits would require too much memory to simulate. Since all the amplitudes are generally involved in every quantum gate computation, storing the state vector on hard disk would introduce too much I/O overhead to simulate. In order to reduce the footprint in memory, the latest approach is to apply lossy compression (SZ compressor) for floating-point data (quantum state amplitudes) to the quantum circuit simulation (Wu et al. 2018a, 2018b).

By using data compression to reduce the memory requirement of storing the full quantum state, the simulators are able to simulate larger quantum systems within the same memory capacity. The state vector CR is the key factor that determines how large the quantum system can be. In general, lossy compressors achieve higher CRs than lossless compressors do, while introducing errors to a certain extent. This approach thus involves trading data accuracy for memory footprint reduction.

There are two constraints on the lossy compressor when we apply this technique to quantum circuit simulation. First, the compression errors must be low enough to get meaningful results. For the simulations generating only one output solution, we may tolerate a larger compression error. However, if the simulation is expected to deliver a probability distribution of a certain problem, we need to have a smaller compression error. Second, when we want to increase the simulation size by n qubits, the CR must be greater than 2^n.

3.6.1. Simulation steps. The quantum circuit simulator with lossy compression divides the whole state vector into several strides $(s_1, s_2, ..., s_n)$, and all the strides are stored in the compressed format in memory, so that the memory footprint for storing the state vector is reduced. The pseudo-code of the simulation process is shown in Algorithm 1. The simulation of a quantum program is processed gate by gate. When a gate is applied to the quantum state, only the stride, s_j, under processing is decompressed; the unitary computation is performed on the decompressed stride, and then the stride is compressed. This is a complete operation cycle for a stride. After a stride is finished, the simulator processes the next stride.

Algorithm 1. Quantum state vector stride update.

```
1: for gate $U_i$ in the program do
2:     for stride $s_j$ in the state vector do
3:         decompress($s_j$)
4:         gateComputation($U_i, s_j$)
5:         compress($s_j$)
6:     end for
7: end for
```

3.6.2. Simulation results. Table 4 shows the simulation results with the lossy compression approach. The first benchmark is a case of a quantum approximate optimization algorithm (QAOA), which is one of the most promising quantum applications that can be executed in the intermediate-scale quantum computers. The second benchmark is a quantum Fourier transform (QFT), which is one of the most popular quantum core functions of quantum applications (e.g. Shor's factorization algorithm). The third, Grover's Search algorithm, is one of the most famous quantum algorithms. The simulation quality is assessed by the state fidelity, CR, and performance overhead. State fidelity is a measure of the similarity of two quantum states (Nielsen and Chuang, 2002). If the fidelity value is 1, then the two quantum states are identical. The CR determines the memory footprint reduction. In general, the SZ lossy compressor successfully reduces the memory requirement of quantum circuit simulation.

3.7. Accelerating execution

Accelerating the computation has been one of the most important driving forces for advancements in both computer software and hardware since the development of electronic computers. Apart from the conspicuous consequence of completing the given job earlier, faster computer software can also have other beneficial properties such as releasing computing resources earlier or reducing the total system energy consumption. On the other hand, the efficiency of data storage is becoming increasingly important for modern computers (Goda and Kitsuregawa, 2012), especially with the emergence of the supercomputers. Consequently, many scientific works are proposing high-speed compression algorithms, which attempt to solve both of the aforementioned problems: speed and data storage efficiency. Even though such a compression stage adds more computations, it may still be possible to improve overall execution speed by relieving the memory bandwidth and size bottlenecks.

For some applications, it may be possible to exploit the repetition and self-similarities in the data to accelerate execution. One such application is GAMESS (Schmidt et al., 1993), where two-electron repulsion integrals are computed for quantum chemistry simulations. The GAMESS application calculates a large number of data blocks that contain two-electron repulsion integral (ERI) data in single-precision floating-point numbers. These blocks are generated and then consumed repeatedly in quantum chemistry simulations during the runtime. Most of these blocks have high computation costs; however, many of these calculated blocks are identical to each other but still calculated over and over again nonetheless. Furthermore, within each are a large number of self-similarities. As shown in Gok et al., (2018), one can exploit the repetition of blocks and self-similarities within such blocks to improve the execution time while maintaining the accuracy and CR requirements.

Table 4. Quantum circuit simulation results.

Benchmark	No. of qubits	No. of gates	CR	Fidelity	Performance overhead
QAOA	17	260	8×	99.9%	6×
QFT	26	1710	16×	99.9%	15×
Grover's Search	20	199	4.59×10^5×	99.9%	19×

QAOA: quantum approximate optimization algorithm; QFT: quantum Fourier transform; CR: compression ratio.

3.7.1. Exploiting repetition. Most of these data blocks are costly to compute; furthermore, such computation can be considered wasteful since many of the blocks are just repetitions of each other. In order to decrease repetitive computations, the data blocks can be stored on either d disk or memory instead of recalculating these repetitive blocks over and over again. Since the amount of data generated by typical simulations is very large, it is usually not practical to store the data blocks in memory. However, storing them on disk is also not practical because of the excruciatingly high access times for disks. Furthermore, moving huge amounts of data to memory or disk brings bandwidth problems. On the other hand, compressing these blocks may make it possible to fit them into the memory and have significantly smaller bandwidth requirements. In this approach, each unique block is calculated, compressed, and written into the memory only once; and whenever a block is needed again in simulations, it is read from the memory and decompressed. Compared with the original GAMESS infrastructure, where all blocks are generated and consumed by the simulation on the fly and are then deleted from the memory, this approach would achieve a reduction in the block computation costs. Note that all these operations should be done at runtime because blocks are generated and consumed repeatedly during a simulation. Consequently, the timing requirements of such a compression algorithm would be strict.

In order for this approach to be practical, the following condition should be satisfied

$$N \times T_{Cal} > T_{Cal} + T_C + T_W + (N-1) \times (T_R + T_D)$$

where N is the number of times the current block is repetitively consumed during the simulation, T_{Cal} is the block calculation time, T_C is block compression time, T_W is block write time, T_R is block read time, and T_D is block decompression time. The number N can be considered to be between 10 to 30 for most blocks in most typical simulations. Note that T_{Cal} represents the step with the highest time cost and would typically be the largest value in the inequality above.

In Gok et al. (2018), the authors propose SZ-PaSTRI for this approach, which is a new lossy compression algorithm implemented in terms of the SZ compression framework (prediction + quantization + encoding). Significant performance improvements for CRs and speeds are obtained by using SZ-PaSTRI, in that SZ-PaSTRI adopts a very efficient predictor based on the pattern feature of the two-electron integrals data sets. With SZ-PaSTRI, SZ has now three different prediction schemes or three different compression pipelines. The SZ-PaSTRI can keep a high CR (about 10–20) even with very high accuracy requirements, such as 10^{-9} to 10^{-11}. Such high-accuracy requirements are usually considered to be a sufficient condition to achieve acceptable output quality, yet not a necessary condition. Most algorithms that consume two-electron integral data are not analyzed in depth according to their input error propagation to the output; consequently it may be possible to have more relaxed error conditions in the future if such analyses are conducted in detail.

To summarize, in order to avoid recalculations, be able to keep the compressed data blocks in the memory, and reuse them whenever needed, a compression algorithm with very high speed and accuracy, along with a significant CR, is required. Even though lossless compression algorithms always satisfy the accuracy requirement, a lossy compression algorithm may have better opportunities to exploit some other aspects in the data. One such opportunity is self-similarities within a data block, as proposed in the SZ-PaSTRI work (Gok et al., 2018).

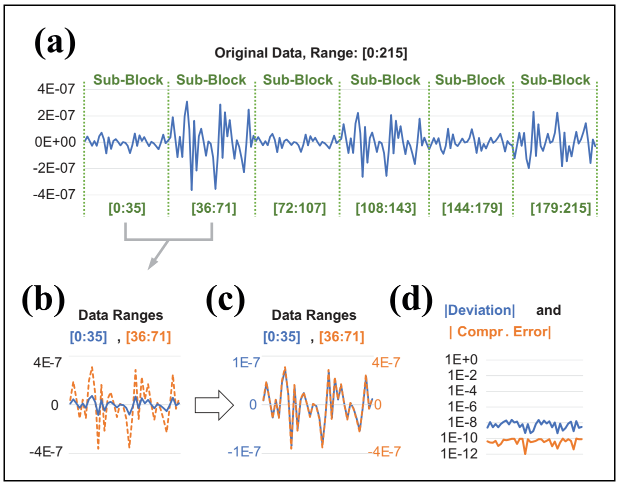

Figure 10. A data block can be represented as a 1-D array where the *x*-axis is an enumeration of four indices and the *y*-axis is the data value. In (a) the first 216 elements of an actual data block are shown. In (b) the overlapped first two subblocks are shown. These are rescaled to match curves are shown in (c). The deviation and absolute compression error between curves are shown in (d), when the error bound is set to be 10-10 (extracted from our prior conference paper (Gok et al., 2018).

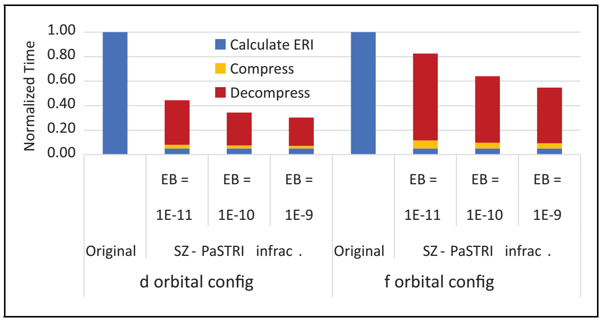

Figure 11. Total computation time needed to obtain integral data for different error bound and orbital configurations. Original represents calculating the data whenever it is needed, whereas SZ-PaSTRI infrastructure represents storing compressed data in the memory. (extracted from our previous conference paper (Gok et al., 2018).

3.7.2. Exploiting self-similarities. Each block in two-electron integral data generated by GAMESS correspond to a scan of full ranges of four indices that correspond to four basis functions. The ranges of such basis functions depend on chemical orbitals, which are represented with letters s, p, d, f, and so on. If the data within a block is enumerated according to these four indices, a self-similarity becomes apparent within that block (Figure 10).

Each block can be divided into subblocks of fixed length, which is defined by the basis function types (Figure 10a). Each subblock can actually be considered as a scaled version of one another, with some minor differences (Figure 10(b) and (c)). In order to represent the whole block, one subblock is chosen to be the *Pattern* and written to the compressed output. With a pattern selected, all subblocks within that block can be represented as a scalar multiplication ($scaledpattern$) of it, requiring only one number to be saved per subblock. Some minor differences are possible between the original data and the scaled pattern, but these differences are far smaller than the original data range and can be encoded by using much fewer bits (Figure 10(d)). SZ-PaSTRI achieves high CRs but also runs fast for both compression and decompression because of the simplistic nature of the scaling algorithm. Additionally, SZ-PaSTRI employs other mathematical and bit-level optimizations to improve both compression speed and ratio even further.

The SZ-PaSTRI algorithm relies on exploiting self-similarities to achieve high CRs, consequently decreasing the pressure on the MS and bandwidth requirements, and relieving the memory bottleneck problem to improve execution speed (Figure 11). Additionally, compared with its

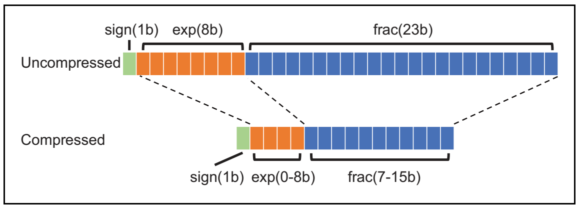

Figure 12. The compressor suggested in Fu et al. (2017). 32-Bit floating-point number is halved in size by adjusting the number of bits used for its exponent and fraction.

parent compressor SZ, exploiting self-similarities allows a much simpler algorithm, resulting in a very fast execution speed for SZ-PaSTRI itself.

Another example that improves overall execution time by leveraging lossy compression can be seen in (Fu et al., 2017). Sunway TaihuLight is one of the world's top supercomputers in terms of its computational capabilities, but its MS and bandwidth are relatively small compared with other supercomputers of similar scale. Consequently, MS and bandwidth become key bottlenecks for many applications on Sunway TaihuLight.

The authors in Fu et al. (2017) ran earthquake simulations that employ floating-point data with large grid sizes and high-frequency support on Sunway TaihuLight. They were focused on memory organization and bandwidth problems since Sunway TaihuLight has a huge number of computing nodes with a relatively small MS. Eventually, they proposed an on-the-fly fast compression algorithm that focuses on speed over anything else. To reach the desired speed, they have chosen a lossy compression algorithm that first employs a lower-resolution version of the original simulation to analyze different blocks in the grid. This analysis focuses on gathering the information about the data value ranges within each block. After obtaining this information, the compression algorithm simply reduces the resolution of the floating-point numbers according to the value range of each block (Figure 12). Each 32-bit floating-point number becomes 16-bit compressed data, thus achieving a 2:1 CR. This CR seems small compared with most other compression algorithms. However, the main requirement for this algorithm was high speed, not high CR, since it is supposed to run on the fly. Eventually, the developers have achieved a 24% performance improvement with almost unnoticeable errors in the output.

In today's computer systems and supercomputers, the most important performance bottleneck is usually the memory system's size and bandwidth. To improve the total system performance, researchers frequently employ different types of compression algorithms to reduce the pressure on the memory systems. A few decades ago, many general-purpose compression algorithms were proposed, usually targeting offline compression and decompression. In contrast, modern systems have serious memory-related bottlenecks, which can be solved by specialized on-the-fly compression algorithms, which have much more strict speed requirements. In order to achieve required execution speeds, lossy algorithms are employed frequently, but the accuracy requirements are usually set very high in order to ensure that the output will not be distorted much. Such accuracy constraints may be relaxed in the future if proper analysis for numerical stability is done for such applications. On the other hand, CR requirements for such compression algorithms is usually not very high, since they only need to relieve the memory system enough to divert the bottleneck to be somewhere else in the system.

4. Toward compressor specialization

The current trend on lossy compression for scientific data is to design generic lossy compressors that can be used for a large diversity of applications and use cases. However, the presentation of the seven use cases reveals that each has a specific and different set of constraints. Table 5 compares the requirements for each use case, based on our interaction with the application developers and users, in terms of speed, ratio, and accuracy. The table presents the relative constraints for different use cases. We evaluate the constraints qualitatively because they quantitatively depend on the specific characteristics of the applications and the system running the applications. For visualization purposes, for instance, the accuracy on error controls is generally low since the cognitive systems of human can tolerate visual distortion to some extent. By comparison, the lossy checkpoiting use case requires fairly high accuracy in general because it needs to control the error propagation (or the impact of the data loss to the execution result) during the simulation after the restart upon the failures.

The table confirms our experience that a single compressor pipeline can serve only a limited set of these use cases. For example, all known results and experiments show that reaching high CRs while keeping high accuracy (reducing footprint on storage) require sophisticated compression pipelines and significant computation at each stage. Compressor software with such characteristics cannot be used for online compression of memory accesses (accelerating execution) because it would slow the application execution instead of accelerating it. A carefully designed hardware version of the same compressor might be fast enough to accelerate the application execution, but currently no results show that this is the case.

The table suggests the need for designing and implementing specialized compression pipelines to serve the different use cases as effectively and efficiently as possible.

5. Related work

We discuss the related work that addresses the use of lossy compression techniques in different domains.

As mentioned previously, since visualization is often the first and foremost step for researchers to understand the characteristics of the scientific data sets, many lossy compressors are designed for visualization. Li et al. (2018b) provide a survey of the lossy compression techniques and

Table 5. Compression constraints for the seven use cases.[a]

Use cases	Examples	Speed	Ratio	Accuracy
Visualization	Climate simulation	High	Very high (100 s)	Low
Reducing data stream intensity	LCLS/APS X-ray data	Very high	Medium (10)	Medium
Reducing footprint on storage	HACC	Slow OK (off-line)	High (10 s)	High for post-analysis
Reducing I/O time	CESM-ATM, HACC	High (online)	Medium (10)	High for post-analysis
Accelerating checkpoint/restart	NWChem	High (online)	Medium (10)	Very high for restart
Reducing footprint in memory	Quantum Circuit Sim.	Medium	Medium (10)	Very high
Accelerating execution	GAMESS	Very high	Low to medium (2–10)	Very high

LCLS: Linear Coherent Light Source; APS: Advanced Photon Source; CESM: Community Earth System Model; HACC: Hardware/Hybrid Accelerated Cosmology Code.
[a]Each constraint can have four qualitative values: very high, high, medium, or low.

their use cases (such as reducing storage cost and I/O time) mainly from the perspective of visualization purpose; in contrast, our article targets more comprehensive use cases (including checkpointing/restart, reducing I/O time, and accelerating execution) in scientific research across different domains.

Some other studies (Son et al., 2014; Welton et al., 2011) discuss the use cases of data compression, but all of them focus only on specific contexts or relatively old compression techniques; in contrast, our article gives a comprehensive discussion of lossy compression use cases with the corresponding constraints identified by users and also describe the up-to-date state-of-the-art lossy compression techniques. Welton et al. (2011), for example, discuss the gap between CPU and network speed and propose to reduce the network traffic by lossless compressors. Son et al. (2014) conducted a survey on the data compression techniques only in the use case of checkpointing. Their survey was also limited to the old compressors such as ISABELA (Lakshminarasimhan et al., 2011) and FPZIP (Lindstrom and Isenburg, 2006). This survey did not cover important lossy compressors such as SZ (Di and Cappello, 2016; Liang et al., 2018; Tao et al., 2017c) and ZFP (Lindstrom, 2014) that have much higher compression performance as confirmed in the recent studies (Liang et al., 2018; Lindstrom, 2014).

6. Conclusion

Limitations in memory and storage space and in bandwidth in supercomputers and scientific instruments, in conjunction with recent progress in lossy compression technology and, in particular, the strict respect of user-set error bounds, have opened opportunities for new use cases of lossy compression for scientific data. In this article, we describe seven identified use cases of lossy compression for scientific research. Three of the use cases, involving visualization, reduction of storage footprint, and reduction of I/O bandwidth, are classic. The other four user cases are more recent. Although overlaps exist between these use cases, our experience is that they have different sets of constraints, as listed in Table 5. While the current trend in lossy compression is to design generic lossy compressors targeting many different use cases, the different sets of some time-conflicting requirements of the use cases suggest that more specialization is needed in compression pipelines. This in turn suggests that, in order to be efficient and effective for a large diversity of use cases, generic lossy compressors need to be adaptable and provide multiple configurations or/and multiple pipelines.

Authors' Note

This research also used computing resources of the Argonne Leadership Computing Facility. Portions of this research were carried out at the Linac Coherent Light Source (LCLS) at the SLAC National Accelerator Laboratory. LCLS is an Office of Science User Facility operated for the US Department of Energy Office of Science by Stanford University. Use of the Linac Coherent Light Source (LCLS), SLAC National Accelerator Laboratory, is supported by the US Department of Energy, Office of Science, Office of Basic Energy Sciences under Contract No. DE-AC02-76SF00515.

Acknowledgements

We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations—the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative. The material was based upon work supported by the US Department of Energy, Office of Science, under contract DE-AC02-06CH11357 and supported by the National Science Foundation under Grant No. 1619253.

ORCID iD

Franck Cappello https://orcid.org/0000-0002-7890-3934

Notes

1. Some compressors such as ISABELA (Lakshminarasimhan et al., 2011) were designed with pointwise error control, but tests have shown that the max error could be much larger than the user-set error bound (Di and Cappello, 2016).
2. In rare cases, ZFP may not respect an error bound if all values within a block cannot be represented by the same exponent.
3. POSIX I/O performance is close to other parallel I/O performance such as MPI-IO (Thakur et al., 1998) when thousands of files are written/read simultaneously on GPFS, as indicated by a recent study (Turner, 2017).

References

Ainsworth M, Tugluk O, Whitney B, et al. (2018) Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science* 19(5–6): 65–76.

Baker AH, Xu H, Hammerling DM, et al. (2017) Toward a multimethod approach: Lossy data compression for climate simulation data. In: *International Conference on High Performance Computing ISC High Performance 2017: High Performance Computing*, Frankfurt, Springer, London, UK, 18–22 June 2017, pp. 30–42.

Balay S, Abhyankar S, Adams M, et al. (2018) *PETSc users manual: Revision 3.10*. Technical report. IL, USA: Argonne National Lab. (ANL).

Ballester-Ripoll R, Lindstrom P and Pajarola R (2018) TTHRESH: Tensor Compression for Multidimensional Visual Data. *ArXiv e-prints*. Available at: https://arxiv.org/abs/1806.05952 (accessed 15 June 2018).

Bautista-Gomez L, Tsuboi S, Komatitsch D, et al. (2011) FTI: high performance fault tolerance interface for hybrid systems. In: *2011 International conference for high performance computing, networking, storage and analysis (SC)*, Washington, DC, 12–18 November 2011, pp. 1–12. Seatle, WA, USA: IEEE.

Boehm C, Hanzich M, de la Puente J, et al. (2016) Wavefield compression for adjoint methods in full-waveform inversion. *Geophysics* 81(6): R385–R397.

Calhoun J, Cappello F, Olson LN, et al. (2018) Exploring the feasibility of lossy compression for PDE simulations. *The International Journal of High Performance Computing Applications* 33(2): 397–410.

Clyne J, Mininni P, Norton A, et al. (2007) Interactive desktop analysis of high resolution simulations: application to turbulent plume dynamics and current sheet formation. *New Journal of Physics* 9(8): 301. Available at: http://stacks.iop.org/1367-2630/9/i=8/a=301.

Deutsch P (1996) *Gzip file format specification version 4.3*. Technical report. RFC Editor, United States.

Di S and Cappello F (2016) Fast error-bounded lossy HPC data compression with SZ. In: *2016 IEEE international parallel and distributed processing symposium*, Chicago, IL, USA, 23–27 May 2016, pp. 730–739. Washington DC, USA: IEEE Computer Society.

Domingos Barbosa, João Paulo Barraca, Dalmiro Maia, et al. (2016) Power monitoring and control for large scale projects: SKA, a case study. In: *Proceedings of the SPIE 9910, Observatory Operations: Strategies, Processes, and Systems VI, 99100J*, Edinburgh, United Kingdom, 15 July 2016. https://doi.org/10.1117/12.2234496.

Engelson V, Fritzson D and Fritzson P (2000) Lossless compression of high-volume numerical data from simulations. In: *Proceedings DCC 2000. Data compression conference*, Snowbird, UT, USA, 28–30 March 2000. Snowbird, UT: IEEE.

Fornek TE (2017) *Advanced photon source upgrade project preliminary design report*. DOI: 10.2172/1423830.

Foster IT, Ainsworth M, Allen B, et al. (2017) Computing just what you need: online data analysis and reduction at extreme scales. In: *Proceedings Euro-Par 2017: Parallel processing—23rd international conference on parallel and distributed computing, Santiago de Compostela*, Spain, 28 August–1 September 2017, pp. 3–19.

Fu H, Yin W, Yang G, et al. (2017) 18.9-Pflops nonlinear earthquake simulation on Sunway TaihuLight. In: *Proceedings of the international conference for high performance computing, networking, storage and analysis (SC17)*, Denver, Colorado, 12–17 November 2017. New York, USA: ACM Press.

Gainaru A, Aupy G, Benoit A, et al. (2015) Scheduling the I/O of HPC applications under congestion. In: *2015 IEEE international parallel and distributed processing symposium (IPDPS)*, pp. 1013–1022. Washington DC, USA: IEEE Computer Society.

Goda K and Kitsuregawa M (2012) The history of storage systems. In: *Proceedings of the IEEE 100 (Special Centennial Issue)*, Belfast, UK, 11 April 2012, pp. 1433–1440. Washington, DC, USA: IEEE Computer Society.

Gok AM, Di S, Alexeev Y, et al. (2018) PaSTRI: error-bounded lossy compression for two-electron integrals in quantum chemistry. In: *2018 IEEE international conference on cluster computing (CLUSTER)*, 10–13 September 2018, pp. 1–11.

Goldschneider JR (1997) *Lossy Compression of Scientific Data Via Wavelets and Vector Quantization*. Seattle: University of Washington.

Habib S (2017) Cosmological simulations with HACC. Available at: https://cpb-us-e1.wpmucdn.com/sites.northwestern.edu/dist/8/307/files/2017/08/habib_hacc_cofi_2017-1wub1ox.pdf (accessed 7 February 2017).

Habib S, Morozov V, Frontiere N, et al. (2016) HACC: extreme scaling and performance across diverse architectures. *Communications of the ACM* 60(1): 97–104.

HDF5 Filter Plugin (2018) Available at: https://support.hdfgroup.org/services/filters.html.

HDF5 Library (2018) Available at: https://www.hdfgroup.org/solutions/hdf5/.

Hurrell JW, Holland MM, Gent PR, et al. (2013) The community earth system model: a framework for collaborative research. *Bulletin of the American Meteorological Society* 94(9): 1339–1360.

Islam TZ, Mohror K, Bagchi S, et al. (2013) MCRENGINE: a scalable checkpointing system using data-aware aggregation and compression. *Scientific Programming* 21(3-4): 149–163.

Kay JE, Baker AH, Hammerling D, et al. (2016) Evaluating lossy data compression on climate simulation data within a large ensemble. *Geoscientific Model Development* 9(12): 4381–4403.

Kukreja N, Hückelheim J, Louboutin M, et al. (2018) Combining checkpointing and data compression for large scale seismic inversion. *CoRR* abs/1810.05268. Available at: http://arxiv.org/abs/1810.05268 (accessed 11 October 2018).

Lakshminarasimhan S, Shah N, Ethier S, et al. (2011) Compressing the incompressible with ISABELA: in-situ reduction of spatio-temporal data. In: *European Conference on Parallel Processing*, Bordeaux, France, 29 August–02 September 2011, pp. 366–379. London, UK: Springer.

LCRC Bebop cluster (2018) Available at: https://www.lcrc.anl.gov/systems/resources/bebop. Online.

Li S, Di S, Liang X, et al. (2018a) Optimizing lossy compression with adjacent snapshots for N-body simulation. In: *2018 IEEE international conference on big data (big data)*, Seatle, USA, 10–13 December 2018, pp. 428–437. Washington DC, USA: IEEE Computer Society.

Li S, Marsaglia N, Garth C, et al. (2018b) Data reduction techniques for simulation, visualization and data analysis. *Computer Graphics Forum* 37: 422–447.

Liang X, Di S, Tao D, et al. (2018) Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In: *2018 IEEE international conference on big data (big data)*, Seatle, USA, 10–13 December 2018, pp. 438–447. Washington DC, USA: IEEE Computer Society.

Lindstrom P (2014) Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics* 20(12): 2674–2683.

Lindstrom P and Isenburg M (2006) Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics* 12(5): 1245–1250.

Marcus G, Ding Y, Emma P, et al. (2015) High fidelity start-to-end numerical particle simulations and performance studies for LCLS-II. In: *Proceedings, 37th international free electron laser conference (FEL 2015)*, Daejeon, Korea, 23–28 August 2015.

Mark SH, Yoon CH, DeMirci H, et al. (2016) Selenium single-wavelength anomalous diffraction de novo phasing using an X-ray-free electron laser. *Nature—Communications* 7: 13388.

McCalpin J (2016) Memory bandwidth and system balance in hpc systems. In: *Invited talk at SC16*. Available at: http://sites.utexas.edu/jdm4372/tag/stream-benchmark/ (accessed 7 October 2017).

Moody A, Bronevetsky G, Mohror K, et al. (2010) Design, modeling, and evaluation of a scalable multi-level checkpointing system. In: *2010 international conference for high performance computing, networking, storage and analysis (SC)*, New Orleans, LA, USA, 13–19 November 2010, pp. 1–11. Washington, DC, USA: IEEE Computer Society.

Nicolaucig A, Ivanov M and Mattavelli M (2003) Lossy compression of tpc data and trajectory tracking efficiency for the alice experiment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 500(1-3): 412–420.

Nielsen MA and Chuang I (2002) *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press.

Offringa A (2016) Compression of interferometric radio-astronomical data. *Astronomy & Astrophysics* 595: A99.

Patauner C (2011) Lossy and lossless data compression of data from high energy physics experiments. PhD Dissertation: CERN-THESIS-2011-211.

Peters SM and Kitaeff VV (2014) The impact of JPEG2000 lossy compression on the scientific quality of radio astronomy imagery. *Astronomy and Computing* 6: 41–51.

Röhrich D and Vestbø A (2006) Efficient tpc data compression by track and cluster modeling. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 566(2): 668–674.

Sasaki N, Sato K, Endo T, et al. (2015) Exploration of lossy compression for application-level checkpoint/restart. In: *2015 IEEE international parallel and distributed processing symposium (IPDPS)*, Hyderabad, India, 25–29 May 2015, pp. 914–922. Washington DC, USA: IEEE Computer Society.

Schmidt MW, Baldridge KK, Boatz JA, et al. (1993) General atomic and molecular electronic structure system. *Journal of Computational Chemistry* 14(11): 1347–1363.

Son S, Chen Z, Hendrix W, et al. (2014) Data compression for the exascale computing era—survey. *Supercomputing Frontiers and Innovations* 1(2): 76–88.

Tao D, Di S, Chen Z, et al. (2017a) Exploration of pattern-matching techniques for lossy compression on cosmology simulation data sets. In: *International conference on high performance computing*, Frankfurt, 18 June 2017, pp. 43–54. London, UK: Springer.

Tao D, Di S, Chen Z, et al. (2017b) In-depth exploration of single-snapshot lossy compression techniques for N-body simulations. In: *2017 IEEE international conference on big data*, Boston, MA, USA, 11–14 December 2017, pp. 486–493. Washington, DC, USA: IEEE Computer Society.

Tao D, Di S, Chen Z, et al. (2017c) Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In: *2017 IEEE international parallel and distributed processing symposium*, Orlando, FL, USA, 29 May–2 June 2017, pp. 1129–1139. Washington, DC, USA: IEEE Computer Society.

Tao D, Di S, Liang X, et al. (2018) Improving performance of iterative methods by lossy checkponting. In: *Proceedings of the 27th international symposium on high-performance parallel and distributed computing*, Tempe, Arizona, 11–15 June 2018, pp. 52–65. New York, USA: ACM Press.

Thakur R, Gropp W and Lusk E (1998) *On implementing MPI-IO portably and with high performance*. Technical report. IL, USA: Argonne National Lab.

Thayer J, Damiani D, Ford C, et al. (2017) Data systems for the Linac coherent light source. *Advanced Structural and Chemical Imaging* 3(1): 3.

Turner A (2017) Parallel I/O performance. Available at: https://www.archer.ac.uk/training/virtual/2017-02-08-Parallel-IO/2017_02_ParallelIO_ARCHERWebinar.pdf (accessed 8 February 2017).

VAPOR (2018) Available at: https://www.vapor.ucar.edu/. Online.

Vohl D, Fluke CJ and Vernardos G (2015) Data compression in the petascale astronomy era: a gerlumph case study. *Astronomy and Computing* 12: 200–211.

Vohl D, Pritchard T, Andreoni I, et al. (2017) Enabling near real-time remote search for fast transient events with lossy data compression. *Publications of the Astronomical Society of Australia* 34: e038.

Wallace GK (1992) The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics* 38(1): xviii–xxxiv.

Wei Z and Ngan KN (2009) Spatio-temporal just noticeable distortion profile for grey scale image/video in DCT domain. *IEEE Transactions on Circuits and Systems for Video Technology* 19(3): 337–346.

Welch B (2005) POSIX IO extensions for HPC. In: *Proceedings of the 4th USENIX conference on file and storage technologies (FAST)*, San Fransisco, 13–16 December 2005. Berkeley, CA: USENIX Conference Department.

Welton B, Kimpe D, Cope J, et al. (2011) Improving I/O forwarding throughput with data compression. In: *2011 IEEE international conference on cluster computing*, Austin, Texas, 26–30 September 2011, pp. 438–445. Washington, DC, USA: IEEE Computer Society.

Wu XC, Di S, Cappello F, et al. (2018a) Amplitude-aware lossy compression for quantum circuit simulation. In: *Proceedings of the 4th international workshop on data reduction for big scientific data (DRBSD-4)*. Arxiv: https://arxiv.org/abs/1811.05140.

Wu XC, Di S, Cappello F, et al. (2018b) Memory-efficient quantum circuit simulation by using lossy data compression. In: *The 3rd international workshop on Post-Moore Era Supercomputing (PMES) in conjunction with IEEE/ACM 29th international conference for High Performance Computing, Networking, Storage and Analysis (SC2018)*. Arxiv: https://arxiv.org/abs/1811.05630.

Yoon CH, DeMirci H, Sierra RG, et al. (2017) Se-SAD serial femtosecond crystallography datasets from selenobiotinyl-streptavidin. *Science—Data* 4: 170055.

Young JW (1974) A first order approximation to the optimum checkpoint interval. *Communications of the ACM* 17(9): 530–531.

Zstandard compressor (2018) Available at: https://facebook.github.io/zstd/.

Author biographies

Franck Cappello received his PhD from the University of Paris XI in 1994 and joined CNRS, the French National Center for Scientific Research. In 2003, he joined INRIA, where he holds the position of permanent senior researcher. He initiated the Grid'5000 project in 2003 and served as Director of Grid'5000 (https://www.grid5000.fr) in its design, implementation, and production phase from 2003 to 2008. Grid'5000 is still in used today and has helped hundreds or researchers for their experiments in parallel and distributed computing and to publish more than 1500 research publications. In 2009, Cappello also became visiting research professor at the University of Illinois. He created with Marc Snir the Joint-Laboratory on Petascale Computing that has developed in 2014 as the Joint Laboratory on Extreme Scale Computing (JLESC: https://jlesc.github.io) gathering seven of the most prominent research and production centers in supercomputing: NCSA, Inria, ANL, BSC, JSC, Riken CCS, and UTK. Over his 10 years tenure as the director of the JLPC and JLESC, Cappello has helped hundreds of researchers and students to share their research and collaborate to explore the frontiers of supercomputing. From 2008, as a member of the executive committee of the International Exascale Software Project, he led the roadmap and strategy efforts for projects related to resilience at the extreme scale. In 2016, Cappello became the director of two Exascale Computing Project (ECP: https://www.exascaleproject.org/) software projects related to resilience and lossy compression of scientific data that will help Exascale applications to run efficiently on Exascale systems. He is an IEEE Fellow and the recipient of the 2018 IEEE TCPP Outstanding Service award.

Sheng Di received his master's degree from Huazhong University of Science and Technology in 2007 and PhD degree from the University of Hong Kong in 2012. He is currently an assistant computer scientist at Argonne National Laboratory. Dr Di is a senior member of IEEE and the recipient of 2018 IEEE-Chicago Distinguished Mentoring Award. His research interest involves resilience on high performance computing (such as silent data corruption, optimization checkpoint model, and in-situ data compression) and broad research topics on cloud computing (including optimization of resource allocation, cloud network topology, and prediction of cloud workload/hostload). He is working on multiple HPC projects, such as in situ error-bounded lossy compression for HPC data, detection of silent data corruption, characterization of failures and faults for HPC systems, and optimization of multilevel checkpoint models.

Sihuan Li is a PhD student in computer science at University of California, Riverside. He is currently a long-term intern student at Argonne National Laboratory. Before those, he also obtained his bachelor's degree in math from Huazhong University of Science and Technology, China. Broadly speaking, his research interests fall into High Performance Computing. Specifically, he mainly study Algorithm Based Fault Tolerance (ABFT), lossy compression and their applications in large-scale scientific simulations. He is an IEEE student member.

Xin Liang received his bachelor's degree from Peking University in 2014 and will receive his PhD degree from University of California, Riverside in 2019. His research interest includes parallel and distributed systems, lossy compression, fault tolerance, high performance computing, large-scale machine learning, big data analysis, and quantum computing.

Ali Murat Gok received his B.S. and M.S. degrees from Bogazici University, Turkey and PhD degree from North-Western University, Evanston, IL in 2005, 2010 and 2018 respectively. He is currently a graduate research participant at Argonne National Laboratory, working on Exascale Computing Project and contributing to EZ (scientific data compression) and CODAR (data analysis and reduction) projects. His research interests include lossy compression, high-performance computing, energy efficient parallel architectures, power efficiency and reliability and hardware characterization.

Dingwen Tao received his bachelor's degree in mathematics from University of Science and Technology of China in 2013 and his PhD degree in computer science from University of California, Riverside in 2018. Currently, he is an assistant professor of computer science in The University of Alabama and leading the High Performance Data Analytics & Computing Lab (HiPDAC) at UA. Prior to this, he worked at Brookhaven National Laboratory, Argonne National Laboratory, and Pacific Northwest National Laboratory. His research interests include high performance computing, parallel and distributed computing, big data analytics, resilience and fault tolerance, scientific data compression and visualization, large-scale machine learning, and numerical simulation. He has published over 20 peer-reviewed high-quality papers in prestigious ACM and IEEE conferences and journals, such as ACM HPDC, PPoPP, SC, IEEE BigData, CLUSTER, IPDPS, MSST, and TPDS.

Chun Hong Yoon leads the Advanced Methods for Analysis Group at the Linac Coherent Light Source, SLAC National Accelerator Laboratory. His research interests are in developing new and efficient tools for large-scale analysis of coherent diffractive imaging experiments at the free-electron lasers.

Xin-Chuan Wu is a PhD student in the Department of Computer Science (Systems Group) at University of Chicago, and working with Prof. Fred Chong. Xin-Chuan received his MS in Computer Science from National Taiwan University and BS in Computer Science from National Chiao Tung University. His research interests include computer architecture, system security, mobile computing, high performance computing, and quantum computing. He worked on several mobile device projects for Android and Linux kernel driver development as a senior software engineer in industry and worked as a research aid at Argonne National Laboratory.

Yuri Alexeev is a Principal Project Specialist at the Argonne National Laboratory. He received his PhD in Physical Chemistry from Iowa State University. His main research interests include computational quantum chemistry, computational biology, molecular dynamics, quantum computing, and high performance computing.

Frederic T Chong is the Seymour Goodman Professor in the Department of Computer Science at the University of Chicago. He is also Lead Principal Investigator for the EPiQC Project (Enabling Practical-scale Quantum Computing), an NSF Expedition in Computing. Chong received his PhD from MIT in 1996 and was a faculty member and Chancellor's fellow at UC Davis from 1997–2005. He was also a Professor of Computer Science, Director of Computer Engineering, and Director of the Greenscale Center for Energy-Efficient Computing at UCSB from 2005–2015. He is a recipient of the NSF CAREER award, the Intel Outstanding Researcher Award, and six best paper awards. His research interests include emerging technologies for computing, quantum computing, multicore and embedded architectures, computer security, and sustainable computing.