12-2021

Lossy Compression and Its Application on Large Scale Scientific Datasets

Tasmia Reza
treza@g.clemson.edu

11-2021

Lossy compression and its application on large scale scientific datasets

Tasmia Reza

Lossy compression and its application on large scale scientific datasets

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
Tasmia Reza
December 2021

Accepted by:
Jon Calhoun, Committee Chair
Walt Ligon
Ulf Schiller

Abstract

High Performance Computing (HPC) applications are always expanding in data size and computational complexity. It is becoming necessary to consider fault tolerance and system recovery to reduce computation and resource cost in HPC systems. The computation of modern large scale HPC applications are facing bottleneck due to computation complexities, increased runtime and large data storage requirements. These issues can not be ignored in current supercomputing era. Data compression is one of the effective ways to address data storage issue. Among data compression, the lossy compression is much more feasible and efficient than the traditional lossless compression due to low I/O bandwidth of large applications. The goal of this work is to observe and find the optimal lossy compression configuration which has the minimal user controlled error with maximum compression ratio. For this purpose two large scale application have been experimented with various parameters of well known compression method called SZ. The first application is a quantum chemistry based HPC application NWChem. The second application is the vascular blood flow simulation data generated by parallel lattice Boltzmann code for fluid flow simulations with complex geometries called HemeLB. SZ compressor is integrated in the applications' code for testing the correctness and scalability and give a comparative picture of the performance change. Lastly the statistical methods are tested to pre-determine the data distortion for different error bounds.

Dedication

To my family and friends.

Acknowledgments

I would like to start in the name of Allah, the most gracious and the most merciful as he has given me the opportunity to complete this degree. I would like to sincerely thank my advisor Dr. Jon C. Calhoun for his constant support and guidance throughout my journey here at Clemson. I have gotten the opportunity to research in the Computer engineering field and find direction for my future career. I would also like to extend my gratitude towards my committee members Dr. Walt Ligon, Dr. Melissa Smith and Dr. Ulf Schiller for their valuable advice in this work. Dr. Franck Cappallo, Dr. Sheng Di and Dr. Kristopher Keipert at Argonne National Laboratory helped me a lot to conduct research of the NWChem part of this thesis. I have learned a great deal from their team while working as a visiting student at Argonne Laboratory over the summer 2019. I would also like my labmates who have made my journey here at Clemson easy and enjoyable.

Last but not least, I would like to thank my parents, my sister and my friends here and back at home for always guiding me through thick and thin. I would not have come this far without their help.

Table of Contents

List of Tables

[None]

List of Figures

Chapter 1

Introduction

Large-scale high-performance computing (HPC) applications are facing a performance challenge of low I/O bandwidth and coupled with large volumes of data that needs to be stored for scientific analysis and visualization. Future HPC systems are expected to experience failures more frequently than current systems [61]. Checkpoint-restart is a well known technique of saving computational progress at a fixed interval to later recover the application state in case of unplanned failures [9]. In order to recover from the more frequent failures, applications will need to rely increasingly on checkpoint-restart [19]. Increased reliance on checkpoint-restart places additional strain on the system and increasing the computational cost for running applications especially for simulations with a high-level of computation between checkpoints.

To reduce the volume of data stored and increase the effective memory bandwidth, researchers and practitioners have begun integrating lossless and lossy data compression into HPC applications [7, 35]. Lossless compression preserves the accuracy of the data but it fails to perform significantly in certain cases. For example, it faces more challenge of floating point data compression as the least significant bits of

the mantissa tend to be very poorly correlated [22]. The compression ratio achieved is really poor for lossless compression.

Recent work shows that lossy compression is able to reduce data volumes by order-of-magnitudes more than lossless compression [15]. Lossy compression achieves large reductions in data volume by allowing a user controllable level of inaccuracy into the data when compressing.

Although lossy compression is an attractive solution to this problem, establishing performance models and methodologies on how best to integrate lossy compression into HPC applications remains an area of active study [7, 62, 34]. Data compression has shown only modest progress for scientific data as floating point numbers make efficient use of the available bits ultimately causing much lower compression rates [30].

There are various HPC applications such as Blast2 [11], Sedov [60], BlastBS [66], Eddy [65], Vortex [20], BrioWu [5], GALLEX [52], MacLaurin [8] etc. They have been used by researchers to test the effectiveness of lossy compression by varying different parameters of the application and the compressors.

This motivated this research to study the lossy compression and its contribution to improve high-performance computing (HPC) applications such as NWChem and HemeLB.

For NWChem application, we explore checkpointing of NWChem [64], an open-source HPC computational chemistry code with extensive capabilities for large scale simulations. Unlike prior work, that focuses on checkpointing at the iteration boundary, we checkpoint at the sub-iteration level. NWChem iteratively converges to a solution and therefore makes a good candidate to determine the impact of restarting from a lossy compressed checkpoint. Data compression is done by a state-of-the-art HPC lossy compressor with user controlled error bound type and error bound.

Another HPC application we looked into is the HemeLB which is a large par-

allel lattice-Boltzmann simulation framework that creates segmented angiographic data from patients which is helpful in medical data analysis [26]. Its code is a parallelised lattice-Boltzmann application which is optimised for sparse geometries such as vascular networks which can generate complex and voluminous data structures. For example, HemeLB can create a load-balanced domain decomposition at runtime which allows it to run simulations at varying core counts for same simulation domain data. The File I/O operations are paralleled using MPI-IO by a group of reading processes and they can be adjusted in size by using a compile-time parameter. These characteristics makes it optimal for studying the effects lossy data compression in the application simulation. The lossy compression has the possibility to speed up the simulation process and reduce the data size to a large magnitude. The data will be analyzed to predict certain flow features based on the structure of the arteries. This has the potential to reduce the need for costly simulations in the long run.

There is the question of finding the optimal lossy error bound that allows reasonable data deviation after lossy compression. This balance is unique for each large-scale application due to the nature of the application as well as data point variations. Usually it is determined by multiple experimental run which ultimately costs time and storage space. In this part of the research, we have tried to make this task easier to predetermine the lossy error bounds with respect to user allow data deviation. The statistical metrics such as mean, median, variance etc can be used to run the experiments where the data distortion can be set beforehand. This work can reduce the burden of extra experimental run and while keeping the lossy error bound within user's limit.

This thesis makes the following contributions:

- Contribution from NWChem application

  - implements checkpoint restart at the sub-iteration level of the NWChem application;
  - evaluates results from experimental runs to find a balance between user induced error and lossy compression performance; and
  - quantifies the performance of sub-iteration level lossy compressed checkpoint restart for NWChem.

- Contribution from HemeLB blood flow simulation application
  - apply in situ lossy compressor SZ inside blood flow simulation analysis script;
  - Look at compression ratio and data distortion in separate variable levels;
  - discuss the cause of the varying results for different variables.

- Contribution from statistical methods
  - establish relation between lossy error and data distortion from lossy compression for different statistical methods;
  - discuss the different outcomes for different satistical methods;
  - describe its applicability in scientific research community.

The rest of the paper is as follows: Chapter 2 discusses the background of this thesis. Chapter 3 shows how we instrument NWChem with lossy data compression and the performance results. Chapter 4 describes the effects of lossy compression on Hemelb data. Chapter 5 shows how the lossy error bounds can be determined for user allowed data distortion. Finally, chapter 6 states our conclusions from the research conducted in the thesis.

Chapter 2

Background

2.1 High-Performance Computing (HPC)

High-performance computation (HPC) is implemented in a number of science and engineering disciplines such as climate, physics, cosmology, environmental modeling, device and semiconductor simulation, seismology, finance, social science etc [18]. The supercomputers are playing an important role in HPC systems and applications. Supercomputers contain fastest high-performance system to perform large scale, complex computations. Multiple numbers of processing units, large size of RAM memories, faster connecting between multiple nodes, larger I/O throughput, remote access to clusters, larger energy consumption etc. are the features that make them effective and valuable than other computing devices. Its advantage is not only performing large scale and complex calculations but also solving the problems faster which can be done on servers or clusters of PCs [13]. Supercomputers perform cutting edge high-performance computing which are crucial in scientific and technological advancement. The significance of supercomputing in present and future scientific applications has made researchers invested in making it more efficient. The primary

performance bottleneck for many scientific computing codes is the speed and storage of large scale complex computation.

2.2 Compression for HPC Data

Scientific high-performance data sizes are growing with advancement of technology. Increasing complexity of scientific simulations as well as larger computing processors and storage spaces are greatly contributing in the expansion [43]. This is bringing progress along with some strains on computing units and data storage availability. Storing such data uncompressed results in large files that are slow to read from and write to disk, often causing I/O bottlenecks in simulation, data processing, and visualization that stall the application. With disk performance lagging increasingly behind the frequent doubling in CPU speed, this problem is expected to become even more urgent over the coming years [43]. Data compression is a technique which is being used to tackle this problem. In this method datasets are compressed and uncompressed to reduce the data storage that is needed to be transferred between memory locations or file systems. This ultimately boosts the I/O performance at the cost of excess computation cycles and additional compression algorithms. Data compression techniques are classified into two categories such as lossless and lossy. Lossless compression reduces data size without loss in data fidelity. The lossless compression reaches its own limitations due to the binding of preserving data accuracy [44]. However, for numerical HPC data, the compression ratios vary between 1-4×. For example, a common lossless compressor like GZIP can perform only modest reduction of floating-point data. Other known lossless compressions are LZ77 [67], Huffman encoding [31], FPC [6] and Fpzip [43].

2.3 Lossy Compression

Lossy compression achieves higher compression ratios compared to lossless compression by allowing inaccuracies into the data when compressing [59]. The decompressed data usually cannot be recovered or reconstructed exactly as the original data. Much higher rate of compression than lossless methods is achieved at the cost of the defect in data. Some applications can tolerate the data error without compromising their computation goals. For example in many image or video transmission the increased speed of transfer is desirable over accuracy. On the other hand, some scientific applications are sensitive over accuracy as their computation depends on the precision of the data. The optimal balance of data accuracy and I/O bottleneck becomes a crucial issue for some scientists. For example, climate scientists have resisted to apply lossy compression algorithms on their files [3]. The data storage is such as crucial issue that lossy compression has become vital whereas the data accuracy has to be studied for the whole process. It is useful to consider the complexity of the compression algorithm, the storage required for the compression process, the speed improvement of the compression for the processing units, the compression ratio as well as the accuracy of the decompressed data. Modern HPC lossy compressors such as SZ [15] and ZFP [42] allow the user to bound the type and magnitude of error introduced into the data. The error bounding metrics can either apply to each value point-wise or be a property over the full data set. Key to successful use of lossy compression algorithms is setting the compressor's error bound [7, 50]. Determining the error bound and error bounding type for various applications remains an open question.

2.3.1 Lossy compression - SZ

SZ is a lossy compressor developed at the Argonne National Laboratory. It is an error-bounded in-situ data compressor which significantly reduces the data sizes within user defined bounds[15]. It is used to compress different types of data such as single-precision, double-precision as well as different sizes of arrays up to five dimensions. It supports three programming languages: Fortran, C and Java.

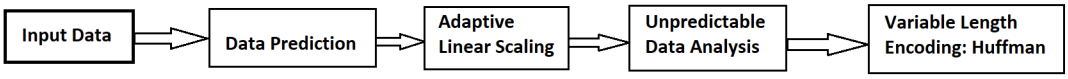

Figure 2.1: SZ compression algorithm

In Figure 2.1 the SZ data compression steps are shown. The first step is data prediction which is done using Lorenzo prediction by default [41]. In 2D data, it is performed by comparing one data point with three neighboring data points where it makes the process easier and faster to compute than using all the data points. Examples of well-known space-filling curves [15] are Peano curve [53], Moore curve [49], Hilbert curve [28] and Lebesque curve (or Z-order curve) [40].The SZ lossy compressor adopts a data prediction method which is a selection of either a 1-layer Lorenzo predictor [33] or linear regression method to predict each data point by its neighboring values in the multidimensional space [63]. The Lorenzo predictor estimates the scalar value of a sample on the corner of an n-dimensional cube from the scalar values of the others $2^n - 1$ corners. It estimates the value of a scalar field at one corner of a cube based on the values at the other corners. In Figure 2.2 the data prediction based on neighborhood points is depicted.

The next step is adaptive quantization. Quantization is the process of mapping

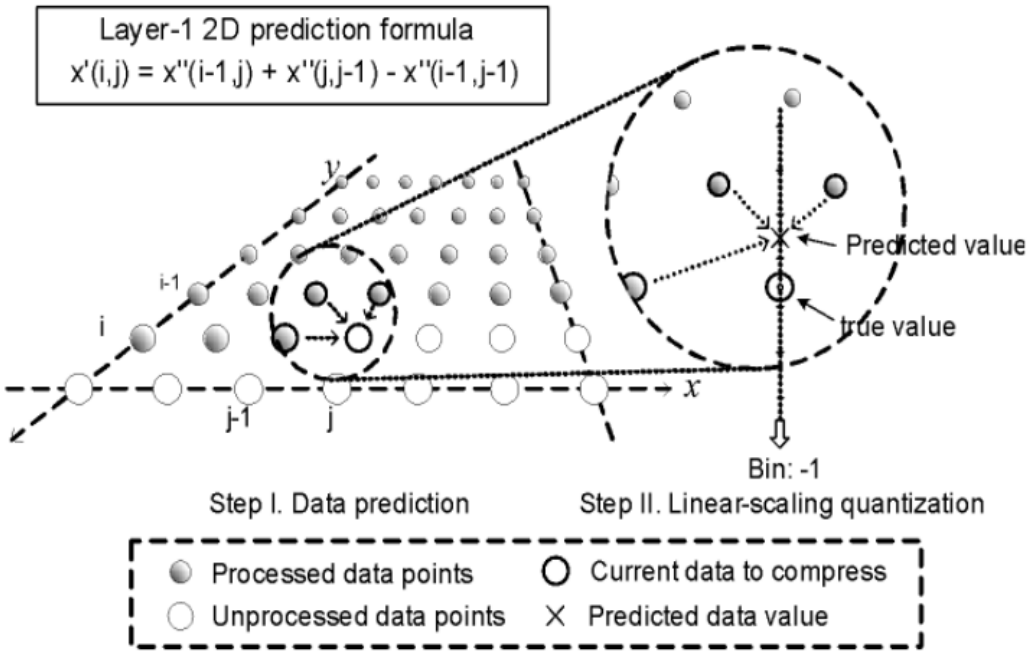

Figure 2.2: SZ curve fitting model

input values to a smaller output value using method such as truncation. In this SZ step linear scaling is adding error bound with quantization. Thus the number of values mapped together is reduced in size within the user defined error bound. It is done where each floating point data value is converted to an integer number in terms of the formula,

$$quantization = (predicted\ value - true\ value)/2\varepsilon$$

where ε refers to the userspecified error bound (i.e., linear-scaling quantization) [63].

The following step is the unpredictable data analysis. In here the data which could not be compressed in the previous step is getting compressed. It is done using a multi-step process to reduce the number of mantissa bits required to represent

each floating point data [15]. At first all the unpredictable data are mapped to a smaller range by letting all the values minus the median value of the range. Then the value is truncated by disregarding the insignificant mantissa part based on the pre determined error bounds. Finally the leading-zero based floating-point compression method is performed to further reduce the storage size. It is done using the XOR operation for the consecutive normalized values and compress each by using a leading zero count followed by the remaining significant bits.

The final SZ compression step is lossless pass of zstd. A Huffman encoding algorithm customized for integer code numbers is then applied to the quantization codes generated by Step two. A dictionary encoder such as Gzip [14] or Zstd [12] is used to significantly reduce the Huffman-encoded bytes generated from Step three.

SZ contains various types of compression error bounds such as absolute error bound (ABS), relative error bound (REF), point-wise relative error bound (PW_REL), peak signal-to-noise ratio(PSNR), absErrorBound and relBoundRatio (ABS_AND_REL) etc. Absolute error ϵ means the decompressed data must be between [x - ϵ] and [x + ϵ].

Where, x = original data

Relative error bound takes into account the range size (maximum value - minimum value). For example, if the relative error bound ratio is set to ϵ and the difference between the maximum and minimum value in the data set is δ. So, the global value range size is δ and the error bound will actually be δ*ϵ in the relative error bound mode.

Point-wise relative error bound controls the compression errors based on a relative error ratio in comparison with each data point's value. If the point-wise relative error bound is ϵ then the real compression error bound for each data point will be equal to ϵ*(individual data value).

2.4 NWChem

North**W**est **Chem**istry (NWChem) [64] is an open-source computational chemistry software package that provides a comprehensive range of methods used to address molecular simulation problems. NWChem project had the goal to create molecular modeling software that provides 10 to 100 times the effective capability than ones available on conventional supercomputers [37]. This makes NWChem algorithm parallel scalable both in the size of the computational resource as well as in the molecular system model. The algorithms must distribute data across the total system memory and not limiting the the functional problem size by the effective memory of any single computational node. NWChem represents tradeoff between computational cost and accuracy. It is an improvement from the petascale's relatively small molecular systems which can saturate the computational throughput and memory bandwidth. Utilizing a common computational framework, diverse theoretical descriptions can be used to provide the best solution for a given scientific problem. This paper provides an overview of NWChem focusing primarily on the core theoretical modules provided by the code and their parallel performance.

2.5 HemeLB data

Cerebrovascular diseases such as brain aneurysms can greatly impact a person's health and wellbeing. Studies show that 13 % of strokes are caused by subarachnoid hemorrhage[54], bleeding in the brain due to the ruptured blood vessels. Medical professionals are increasingly relying on non-surgical or pre-surgical detection as it reduces complications and side effects for patients. For asymptomatic brain aneurysms, the detection of unruptured blood vessels is complicated and surgical treatments can

lead to issues such as neurological deficits and mortality. Recent scientific advances have made it possible to simulate a patients' blood flow in order to predict quantitative rupture risk. Thus, helping with the aneurysm detection. HemeLB[46] is a lattice-Boltzman code that is able to simulate an arterial blood flow simulation. HemeLB works with complex domain geometries that are mapped to the unique arterial structure of patients. Using high-performance parallel processing and leveraging the time dependent nature of the body's cardiac cycle researchers are able to quantify the risks of rupture. To determine the likelihood of rupture, 100s of GB of data must be logged and analyzed. The analyzed simulation data is preprocessed before being fed into a machine learning model to predict the likelihood of rupture.

2.6 Statistical Method

Statistics plays an important in data analysis by helping us to make decisions based on the information or data available. The statistical methods can be used to collect, organize, analyze, and interpret numerical information from populations or samples [4]. For data classification, specially quantitative data analysis summary statistics is a way to look at it. Summary statistics summarizes and provides information about user's sample data [25]. It organizes and provides detailed information about the values in your data set. For example, it can the mean of the data and the skewedness of the data. Summary statistics fall into three main categories such as the measure of location, measure of spread as well as the graphs or charts. The measure for the data can be determined by mean, median, variance, range, quartiles, skewed etc.

Chapter 3

NWChem

3.0.1 NWChem

In this chapter, we focus on a many-body method called coupled-cluster singles and doubles (CCSD)[56]. CCSD is a widely used iterative method that serves as a precursor to the "gold standard" method of computational chemistry, CCSD(T)[57]. Specifically, this work targets a variation of the CCSD algorithm that was implemented with the Tensor Contraction Engine (TCE)[2, 29]. The TCE is an automatic code generation module that takes equations expressed in a high-level domain specific language as input, and produces corresponding high-performance parallel FORTRAN code. The TCE has been successfully used to implement dozens of many-body methods, and today approximately 2/3 of the lines of code in NWChem are machine-generated.

The majority of the runtime for the CCSD method is attributed to the iteratively solving of a set of tensors called $\hat{T}_1$ and $\hat{T}_2$ cluster amplitudes that encode singly and doubly excited terms within a many-body coupled-cluster wavefunction. [We limit the scope of this work to $\hat{T}_2$ amplitudes, as they're more computationally

demanding to compute and consume more storage than the $\hat{T}_1$ cluster amplitudes.] The cluster amplitudes are generally converged within 10-30 CCSD iterations.

The CCSD method formally scales as $O(n^6)$ with respect to the molecular system size, so iteration times can quickly become intractable for even moderately sized systems. Users typically scale their problem size to target iteration times on the order of minutes to hours. To guard against system and software faults, NWChem provides infrastructure for checkpointing CCSD computations by storing cluster amplitudes upon the completion of a given CCSD iteration. But especially for an exascale class system, losing minutes or hours of computation time due to a fault will waste an excessive amount of computer resources. In this work, we attempt to mitigate against this loss by checkpointing at the sub-iteration level. The TCE implementation of the $\hat{T}_2$ amplitude equation is composed of an operation tree with approximately 20 intermediate terms, with each term representing a tensor contraction between a set of 2 and 4-dimensional tensors[29]. We have implemented a framework for checkpointing application state at the boundaries of the intermediate term computations. Secondly, we have investigated the potential of lossy compression of the intermediate terms to reduce checkpointing overhead. While a previous work investigated compression for fully-computed $\hat{T}_2$ cluster amplitudes within NWChem CCSD[30], only lossless compressors were used. Motivated by recent progress in reduced and mixed-precision iterative refinement within CCSD and other many-body methods[55], we extend the previous work to assess the impact of lossy compression on the accuracy of converged $\hat{T}_2$ amplitudes, and to investigate how restarting from a lossy checkpoint state affects the number of iterations required for convergence.

cat

3.1 Sub-iteration Checkpointing of NWChem

NWChem is a long running HPC application which requires multiple iterations to converge to a solution. Previous work on checkpointing NWChem focus on the coupled-cluster singles and doubles (CCSD) computation and checkpoints at a per-iteration granularity [30]. However the per-iteration time can be significant; sometimes consuming hours or even days. The high per-iteration cost makes iteration level checkpoint-restart expensive and inefficient because the the potentially large overhead when restarting.

To address this large overhead when restarting, this work elects to checkpoint at a finer granularity. We target checkpointing at a sub-iteration level. The iteration computes the $\hat{T}_2$ tensor. This tensor's construction in broken into the calculation of 24 intermediate sub-tensors. We modify the code of NWChem to checkpoint each sub-tensor individually. Thus, we are able to recover at a sub-tensor granularity. When checkpointing each sub-tense we employ the SZ lossy compressor to reduce the size of each sub-tensor. This configuration allows us to select individual error bounds and error bounding types for each sub-tensor.

3.2 Related Work

Previous works on integrating lossy checkpointing into HPC applications have shown reductions in the I/O fraction of HPC application [7], required compression levels to improve performance [34], and have modeled checkpointing when extra iterations are required to restore convergence [62]. The integration of lossy compression in HPC workflows and applications requires specific selection of error bounds to get minimum errors in simulation results. Trial and error is an effective way to

establish the correct lossy compression parameters for checkpoint-restart or in-line computation [51, 38, 58]. Incorporating domain knowledge allows for establishing methodologies and heuristics for using lossy compressed data for analytics [3, 21, 50]. Different error bounding metrics have an impact on floating-point truncation error [17] as well as the distribution of compression error [45]. Other researchers have worked to find methodologies for selecting error tolerances for lossy checkpoint-restart on HPC simulations [7, 62]. This work reduces NWChem's checkpointing size by using differenced checkpoint and cutoff techniques to increase the effectiveness of Lempel-Liv (gzip)[30]. This has dramatically increased the compression ratios than standard compression techniques.

3.3 Experimental Results

All of our experiments are run on the Bebop Cluster operated by the Laboratory Computing Resource Center (LCRC) at Argonne National Laboratory. Bebop nodes consist of Intel Xeon E5-2695V4 CPUs with 128GB DDR4 RAM. We test with the 6.8.1 release of NWChem and the 2.1.5 version of the SZ lossy compressor. We evaluate lossy checkpointing of the sub-iterations of NWChem using a simulation of water molecules that converges in 17 iterations. We simulate a restart from a lossy compressed checkpoint at iteration 10.

During each run of NWChem we record the compression time, decompression time, compression ratio, energy deviation from a lossless simulation, and number of iterations to converge. Any runs whose energy deviation differs from the lossless reference simulation greater than $1e{-}5$ is considered an unusable result due to violating conservation of energy [27]. We explore compressing sub-tensors individually and all together.

3.3.1 Lossy Checkpointing Individual Sub-Tensors

We first explore the impact of each sub-tensor to the computation. The size of each of the sub-tensors ranges from 100 to 10,000 elements. For our experiments, we use two types of error bounding with SZ which are absolute (ABS) and relative (REL) with various error bounds ranging from $1e{-}1$ to $1e{-}10$.

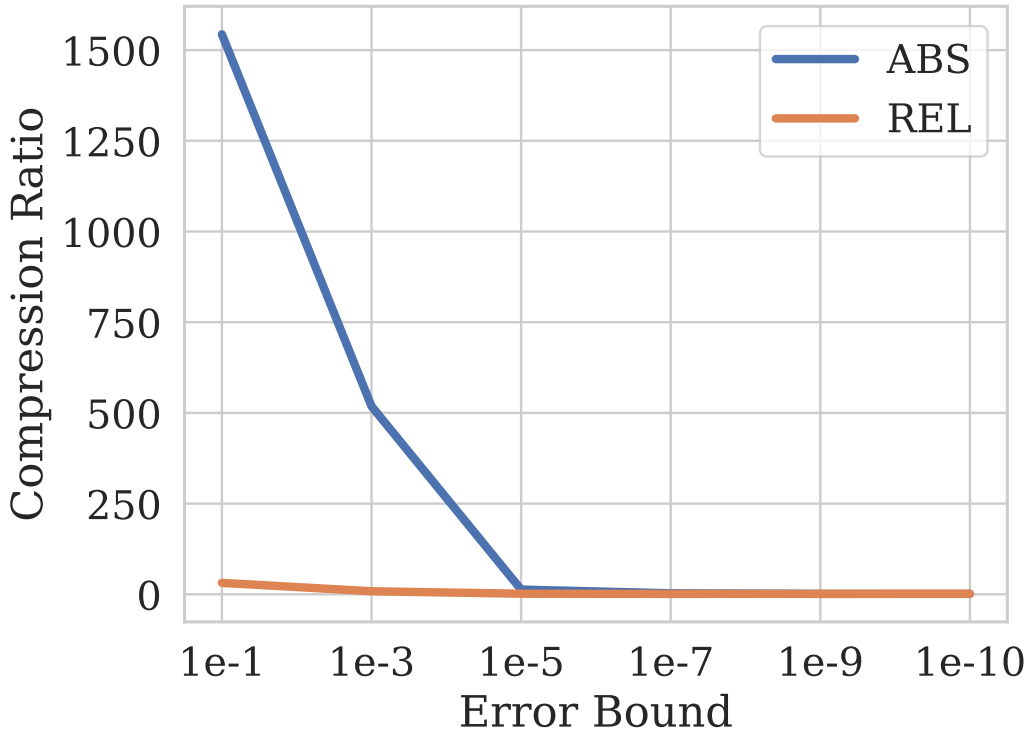

Figure 3.1: Average compression ratio of compressing sub-tensors of T2 individually.

Figure 3.1 shows the average compression ratio for compressing the sub-tensors. In the figure, we see significantly higher compression ratios for absolute error bounds than for relative error bounds for equivalent error bounds. Compression ratios for relative error bounding remain near 1-2× across all the error bounds. As we demand more accurate data from the compressor the compression ratio tends toward 1 indi-

cating that the data does not compress. From Figure 3.1, we see that absolute error bounds of $1e{-}5$ reduces the data set size efficiently all other configurations yield little if any compression.

Figure 3.2 shows the average compression bandwidth for different error bounds for both absolute and relative error bounding types for the sub-tensors. In this figure, we see the compression bandwidth for all configurations that use absolute error bounding yield higher compression bandwidth compared to the equivalent configuration using relative error bounding. Moreover, as the error bound increases, the compression bandwidth increases for both error bounding types. As the error bounds enforce more accurate data, the compression bandwidth approaches zero indicating that compression yields unacceptable performance.

Figure 3.3 plots the average decompression bandwidth for both error bounds. As with Figure 3.2, we average across sub-tensors with a corresponding error bound and error bounding type. From Figure 3.3, we see similar behaviour to compression bandwidth. The major difference is that the decompression bandwidth is lower than the compression bandwidth for similar high error bounds. As the error bound better preserves the data, the decompression bandwidth approaches zero which can lead to increased overhead for NWChem simulations with large quantifies of data.

Figure 3.4 shows the average deviation in energy between a run of NWChem that does not restart from a lossy compressed checkpoint and one that does. From the figure, we see that the deviation in energy is very minor (approximately $1e{-}9$ for all configurations and is well below the level of acceptability of $1e{-}5$). Therefore, each simulation proceeds valid data for the computational scientist — i.e., conversation of energy between the simulations. Thus, we are able to lossy compress checkpoints each sub-tensor of $\hat{T}_2$ and successfully restart.

Even if the simulation does not deviate from the expected energy value, the

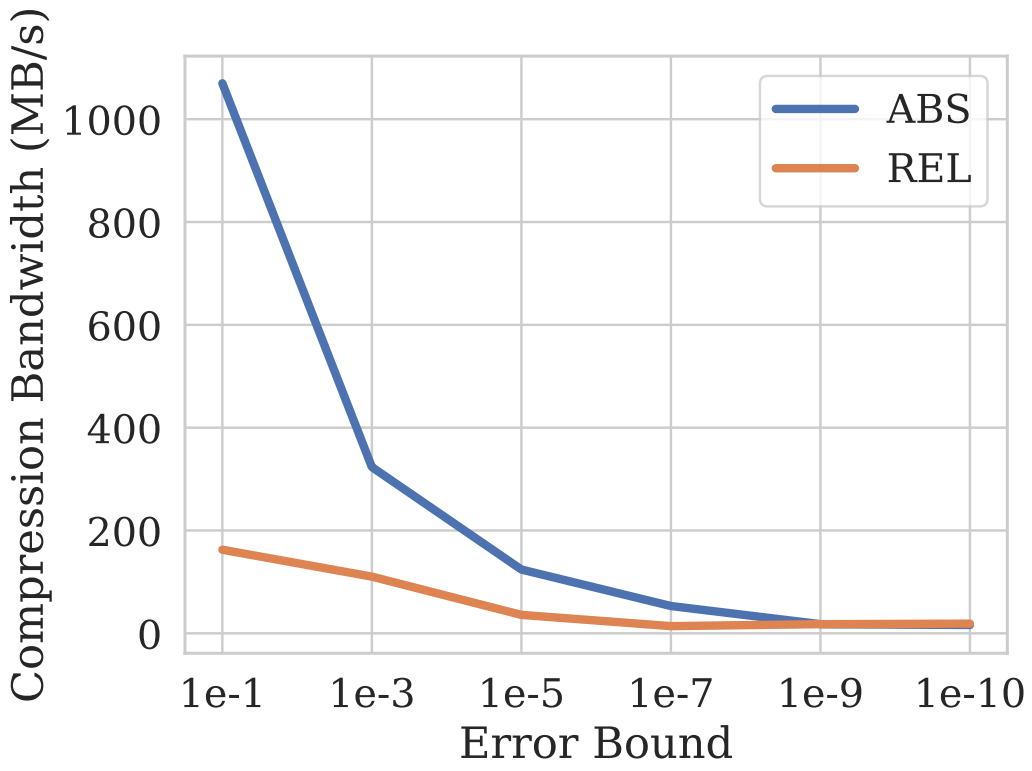

Figure 3.2: Average compression bandwidth of compressing sub-tensors of T2 individually.

number of iterations required to achieve the computational result may increase. Investigating the number of extra iterations reveals that at most one extra iteration is required for all experiments that restart from a single sub-tensor. On average we require 0.66 extra iterations. Thus, we are able to restart NWChem from lossy checkpoints with little impact to the total number of iterations.

3.3.2 Lossy Checkpointing Multiple Sub-Tensors

We now focus on the impact of compressing multiple sub-tensors at the same time. To highlight the worst case scenario, we restart from a checkpoint in which all the sub-tensors are lossy compressed. We do not show plots for compress bandwidth,

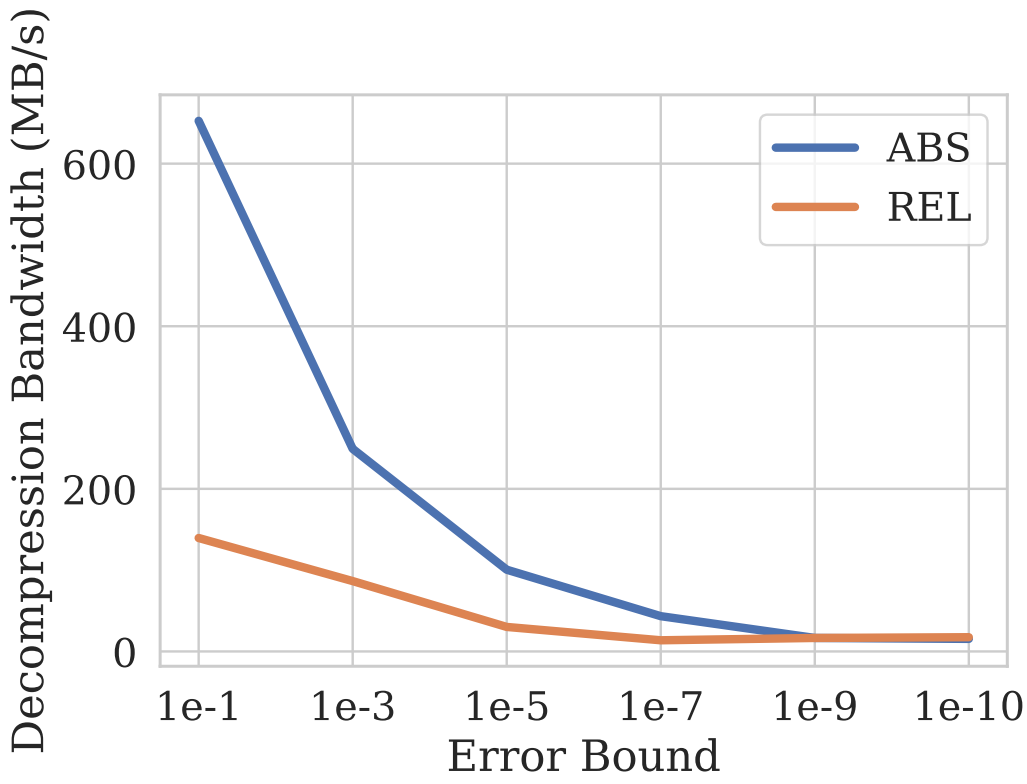

Figure 3.3: Average decompression bandwidth of compressing sub-tensors of T2 individually.

decompression bandwidth, and compression ratio as they are equivalent to those shown in Section 3.3.1. This is due to how we checkpoint each sub-tensor individually.

In Figure 3.5 we plot the energy deviation for various absolute and relative error bounds ranging from $1e{-}1$ to $1e{-}10$. Comparing to Figure 3.4, we see that the deviation is slightly higher indicating that there is more deviations in the simulation. This increase is due to be restarting from a lossy checkpoint all the sub-tensors lossy compressed. Even though the magnitude of the deviation is larger, the magnitude is well within our simulation accuracy bound of $1e{-}5$. This shows that sub-iteration checkpointing is feasible to enable restarting when failure strikes and not impact the accuracy of an NWChem simulation.

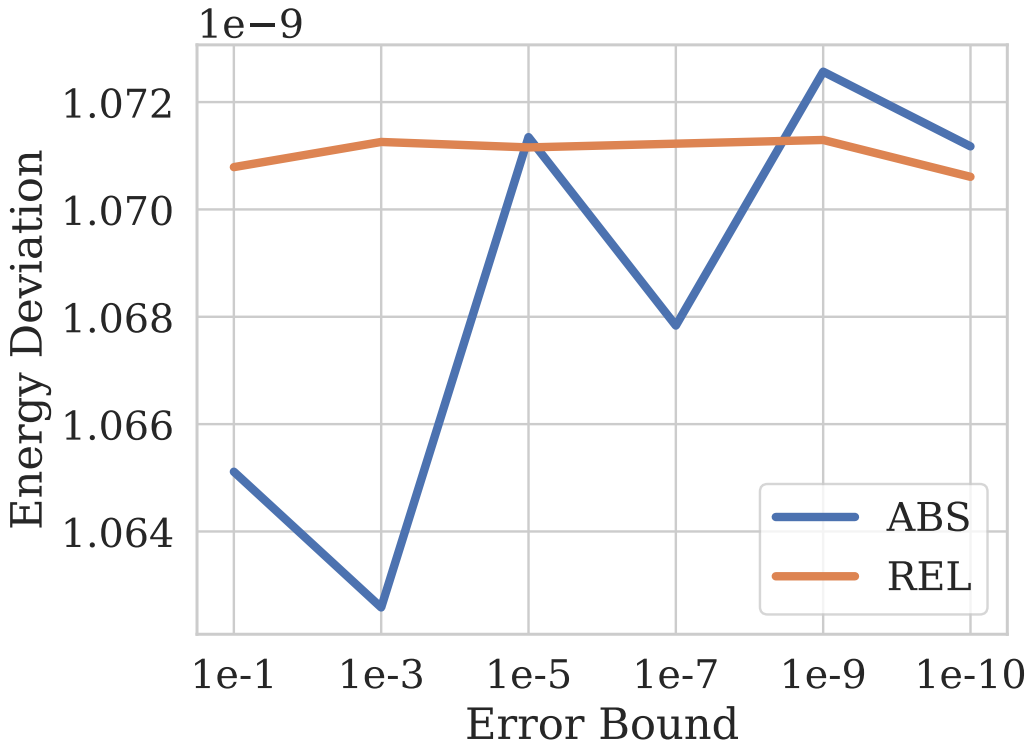

Figure 3.4: Average energy deviation of compressing sub-tensors of T2 individually.

3.4 Conclusion

Checkpointing scientific application becomes more important as the failure rate on large scale systems increase. The NWChem application's per iteration time can be hours or days. In this thesis, we explored lossy checkpointing of sub-iterations of NWChem. We explored the applicability of lossy checkpointing at this granularity by evaluating the compression bandwidth, decompression bandwidth and compression ratio for number of sub-tensors. Our results show that absolute error yields better performance than relative error for error bounds on the range $1e{-}1$ to $1e{-}5$. For all the experiments, the number of extra iterations increased by at most 1 compared to lossless run. The energy deviations were remarkably lower making the sub-tensor

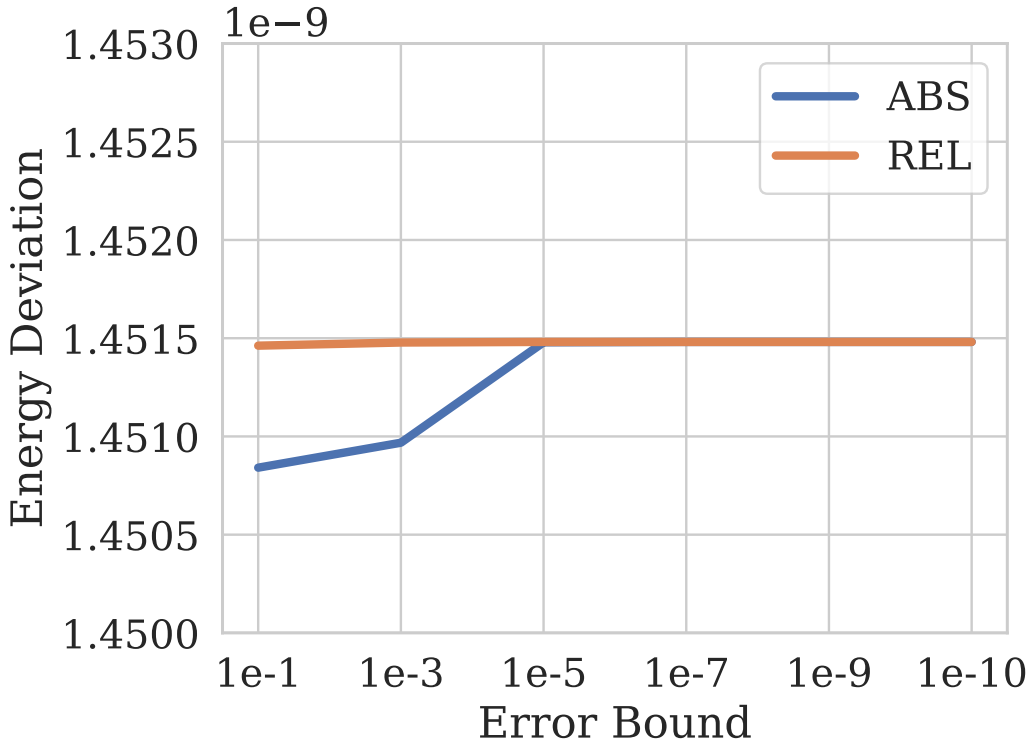

Figure 3.5: Energy Deviation for compressing all sub-tensors simultaneously.

level lossy checkpointing acceptable in NWChem application.

Chapter 4

Vascular Blood Flow Simulation

The vascular blood flow simulation data determines the likelihood of rupture. The HemeLB code is generated using the lattice Boltzmann model which creates considerable amount of data in 4D space-time [46]. For example, 100s of GB of data needs to be generated and analyzed for simulation. HemeLB leverages finite element method to constitute vascular blood vessel geometries. The features that we looked into are inter alia, wall shear stress (WSS), oscillatory shear index (OSI), vorticity, flow impingement regions etc. All of these features are indicators of rupture risks [47]. The early detection of rupture is complicated due to variability of aneurysm geometries and the complex flow patterns and the limited understanding of the relevant mechanisms.

4.1 Data Structure

The HemeLB simulation data have been collected from Professor Schiller's Research Group's storage at Palmetto supercomputer cluster. A typical simulation output generates the flow field as a 3D vector field which is written in every 100 time

steps. The vector field is written in the form (id, x, y, z, vx, vy, vz) where id is a voxel, x, y, and z are the coordinates of the voxel, and vx, vy, vz are the components of the velocity vector [1]. Some data variables are 1D such as pressure and some are 3D such as velocity. Simultaneously, the wall shear stresses for each flow configuration are written to a separate file in the form (id, x, y, z, wss). This generates geometries in the range of 100 to 200 million voxels such that an estimated 500 GB of data is generated per aneurysm geometry per cardiac cycle. The variables that we looked into are velocity, pressure, helicity, vorticity, gamma and QCriterion.

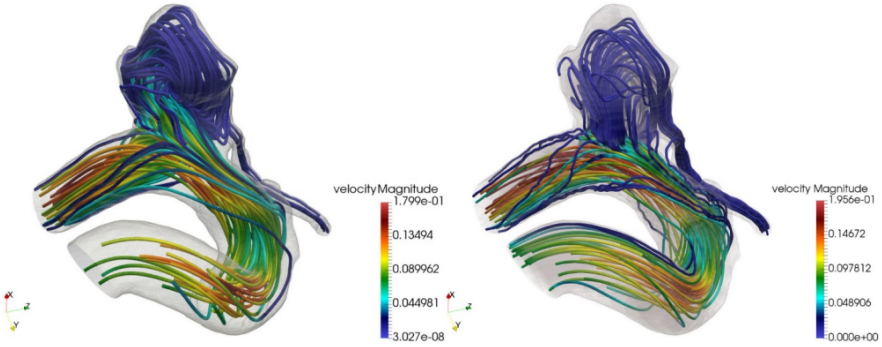

Figure 4.1: Visualization of velocity streamline of blood flow through an aneurysm affected artery (without a stent-mesh flow diverter) [1]

An visual representation of velocity of blood flow simulation is presented in figure 4.1.

4.2 Lossy Compression (SZ) application on data variables

The files generated for the variables are large sized data and need much longer to generate from simulation. They are generated using software analysis. Inside the simulation script each variable is computed separately and they ultimately combined

to generate output values for each variable. This gives us the idea to compress the variables separately inside the simulation script.

SZ is the lossy compressor that has been used to perform the data compression. There are various advantages of using this compressor among which are multiple error bounding modes (ABS, REL, PSNR etc), high compression ratio (10x - 100x), supporting various I/O data formats and most importantly multiple application programming interfaces (API) such as C interface and FORTRAN interface. We have written a C wrapper of SZ to compress and then decompress the data in-situ. The analysis script is modified to call the SZ lossy compressor inside the script for each separate variables, then compress/decompress the data array where it simulates reading in lossy compressed data file. The goal is to see the impact of compression on each variable as we ran the simulation over the decompressed data.

4.3 Related Work

There are various HPC simulation datasets which use compression specially lossy compression to efficiently run their applications. For example, European climate model ECHAM uses bit-oriented file format standard that uses GRIB2, APEX and MAFISC [3]. There are other similar applications such as CESM climate simulation [32] , XGC1 [10], CODAR [39], GAMESS [24] etc. Since such simulation datasets are critical, several factors are needed to be considered to find the optimal compression algorithm for specific applications. Some variables might be larger in size whereas some variables might be smaller which effects their significance in the whole computation. This work [3] emphasize the importance of customizing lossy compression based on variable-by-variable operation to get the best output data which is closest to the original data. The gain of storage and I/O time savings is much intriguing to

invest in lossy compression application in large scale HPC datasets.

4.4 Experimental Results

Lossy compression causes error in data in application which can ultimately hinder the data analysis. From the data generation from this HemeLB code, the output gives summary of the analysis. The experiment is at first run without using compression with the original data. Then the experiment is repeated for varying error bounds ranging from 1e-2 to 1e-11 with an absolute error bound. The data distortion is calculated using the difference between the absolute and lossy compressed results for the variables. The compression ratio is also generated for each compression run for all the differing variables. From the SZ lossy compressed run, we have looked into the compression ratio to evaluate the compression performance and the data distortion to evaluate the error from the compression. The results are shown in the following sections.

4.4.1 Compression Ratio

In Figure 4.2 the compression ratio for all the seven variables are presented. The x-axis represents the error bound and the y-axis is the average compression ratio for all the variables. Some observations from the plot are:

All the variables show similar trends for compression ratio across varying error bounds. It shows that the compression ratio is much lower (≈1-3) for lowest error bound 1e-10 and much higher (50-1000) for highest error bounds from 1e-4. It clearly indicates that compression is much lower for tighter error bounds. The velocity and pressure shows almost same trends i.e., their compression is identical. All the other fours variables follow similar trends where the QCriterion has highest compression

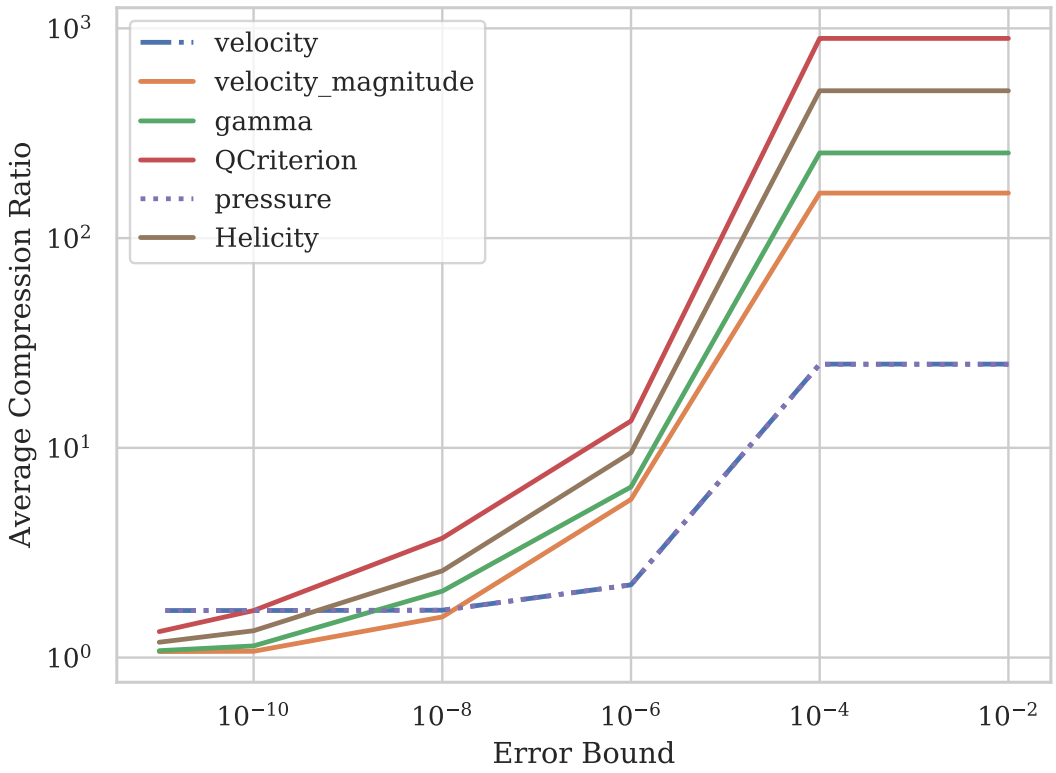

Figure 4.2: Compression Ratio

ratio and the velocity magnitude has the lowest. For error bounds higher 1e-4 than the velocity and pressure has higher compression ratio whereas for error bounds lower than 1e-8 they shows lowest compression. For all the trends it gets constant across 1e-4 to 1e-2. It is reasonable to comprehend that from 1e-4 to higher error bounds, the maximum compression is achieved.

4.4.2 Data Distortion

Data distortion represents the difference between the original experimental run and all the other error bounds ranging from 1e-10 up to 1e-2. The x-axis presents the error bounds and the y-axis presents the data deviation for each variables. In the

output files, each variable is represented by minimum, maximum and mean of their representative variables. All these trends are presented from figure 4.3 to 4.9. The plots are shown to observe which variable have data distortion that stays within the user fixed SZ error bounds.

4.4.2.1 Gamma

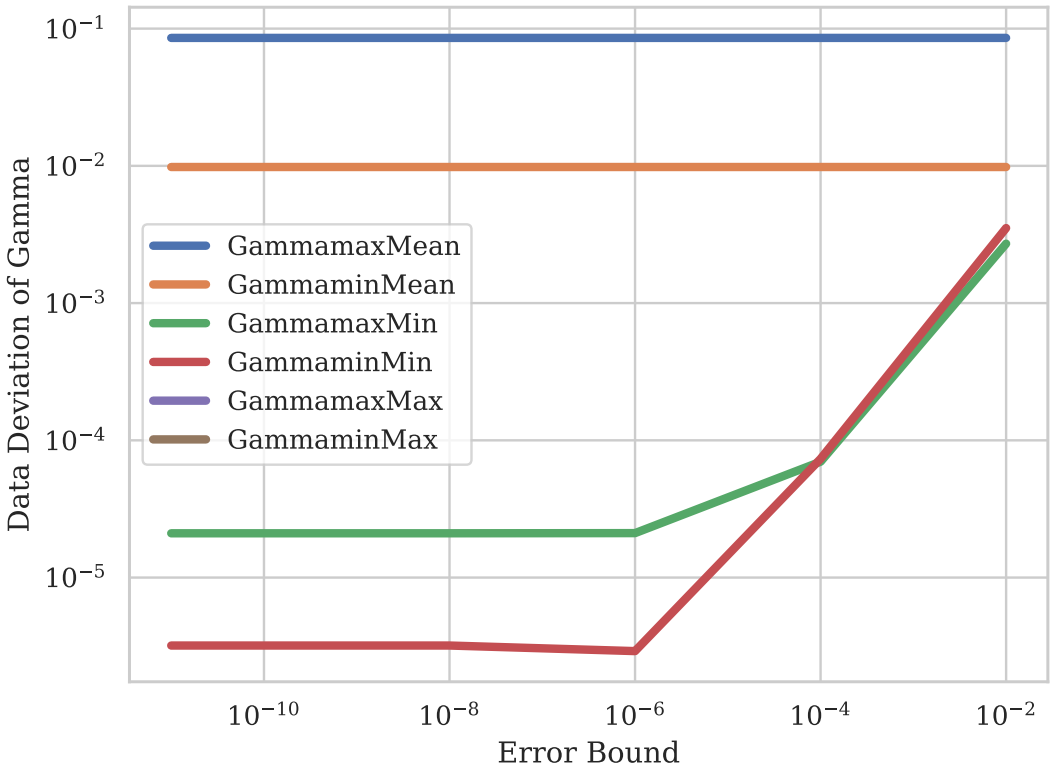

Figure 4.3: Gamma

From Figure 4.3 the six trends for the GammamaxMean, GammaminMean, GammamaxMin, GammaminMin, GammamaxMax and GammaminMax sub variables are observed. It can be noticed that both the GammamaxMean and GammaminMean values attain the highest data deviation. The deviation for them stay constant across all the error bounds. GammamaxMean shows the highest data devi-

ation. The min trends show the lowest data deviation. Six of these values comprise the gamma variable. It is seen that for error bounds higher than 1e-4 the GammamaxMin and GammaminMin gets the data distortion within the error bound. The GammamaxMean and GammaminMean do not get desired results i.e., they get the data distortion much higher than the error bounds.

4.4.2.2 Helicity

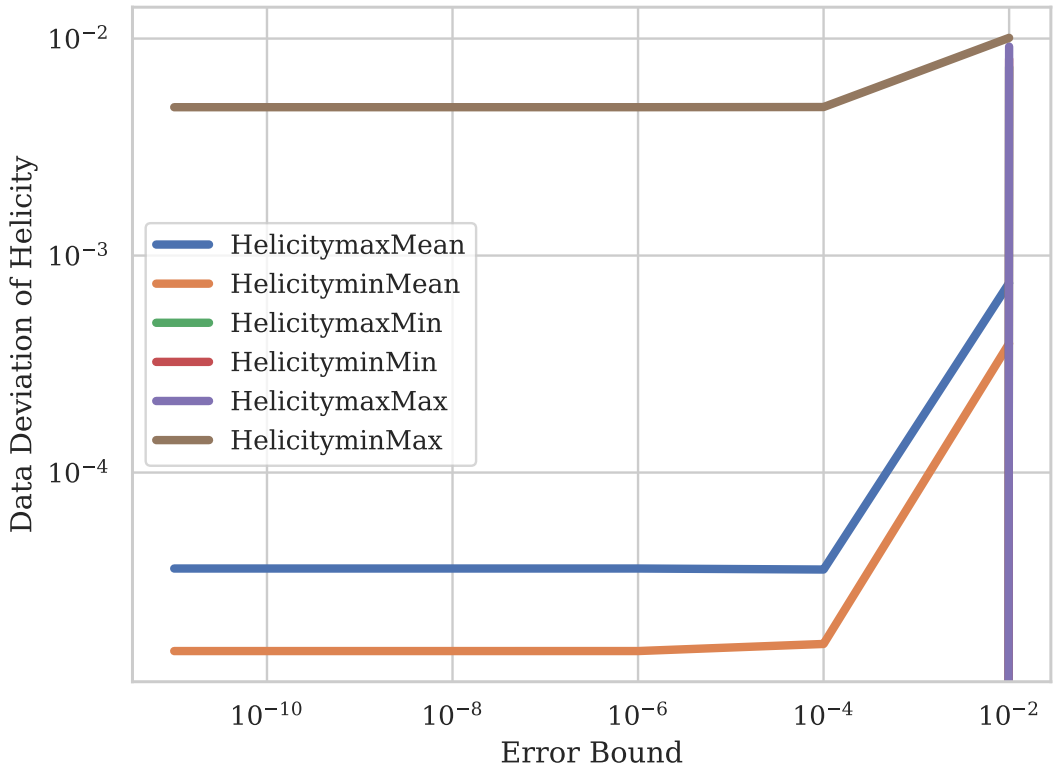

Figure 4.4: Helicity

From Figure 4.4 the six trends for the HelicitymaxMean, HelicityminMean, HelicitymaxMin, HelicityminMin, HelicitymaxMax and HelicityminMax sub variables are observed. It can be noticed that all the trends have the data deviation within the error bound for only the 1e-2 error bound. Only the HelicitymaxMean and He-

licityminMean has the data deviation less that error bound from error bounds higher than 1e-4.

4.4.2.3 OVI

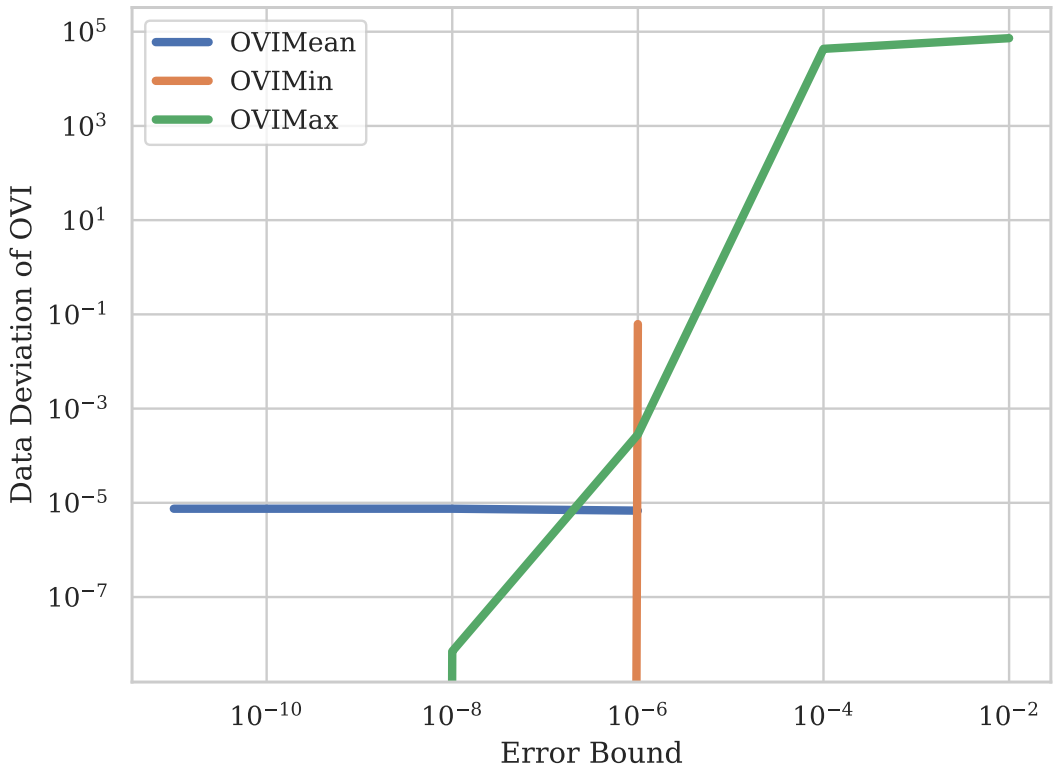

Figure 4.5: OVI

From Figure 4.5 the three trends for the OVIMean, OVIMin and OVIMax sub variables are observed. It can be noticed that all three of these trends have higher data deviation than the error bounds. For OVIMAX the data deviation exceeds much higher than the error bound. Here the lossy compression is inefficient as the error in the data is in much higher magnitude than the compressor error bound.

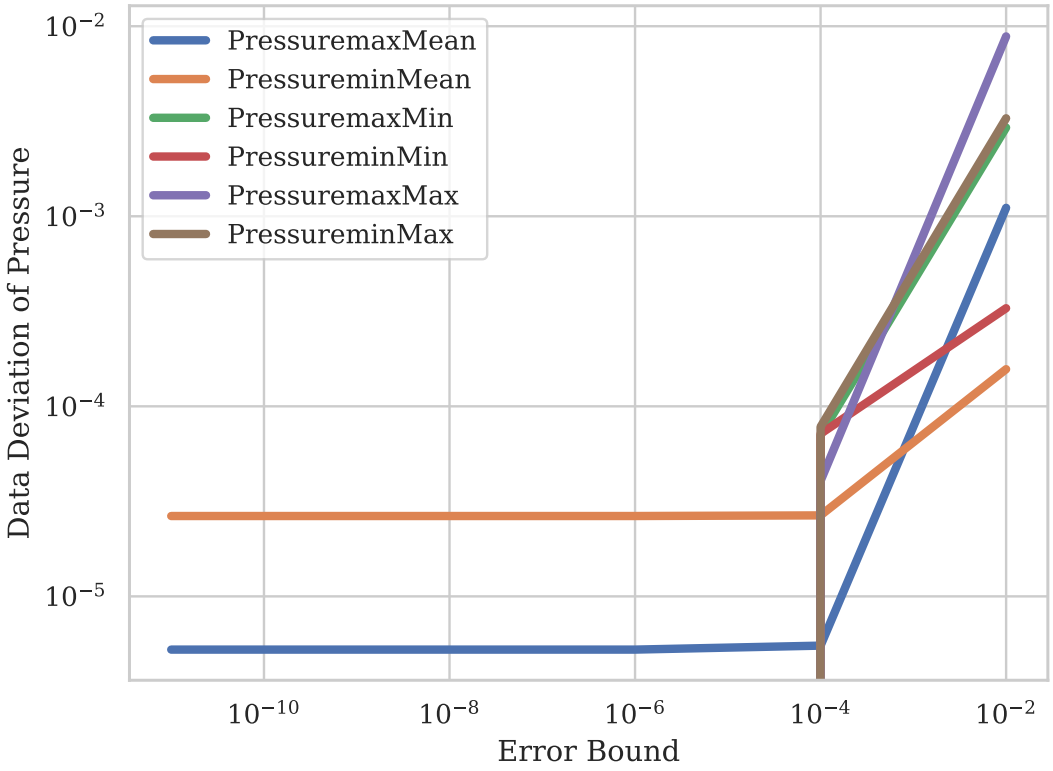

Figure 4.6: Pressure

4.4.2.4 Pressure

From Figure 4.6 the six trends for the PressuremaxMean, PressureminMean, PressuremaxMin, PressureminMin, PressuremaxMax and PressureminMax sub variables are observed. The plot shows all the trends have the data deviation within the error bound for only the 1e-2 error bound. PressuremaxMin and PressureminMax shows acceptable data deviation from error bound 1e-4 and higher.

4.4.2.5 QCriterion

From Figure 4.7 the two trends for the QCriterionmaxMean, QCriterionminMean sub variables are observed. It can be noticed that both the trends QCriterion-

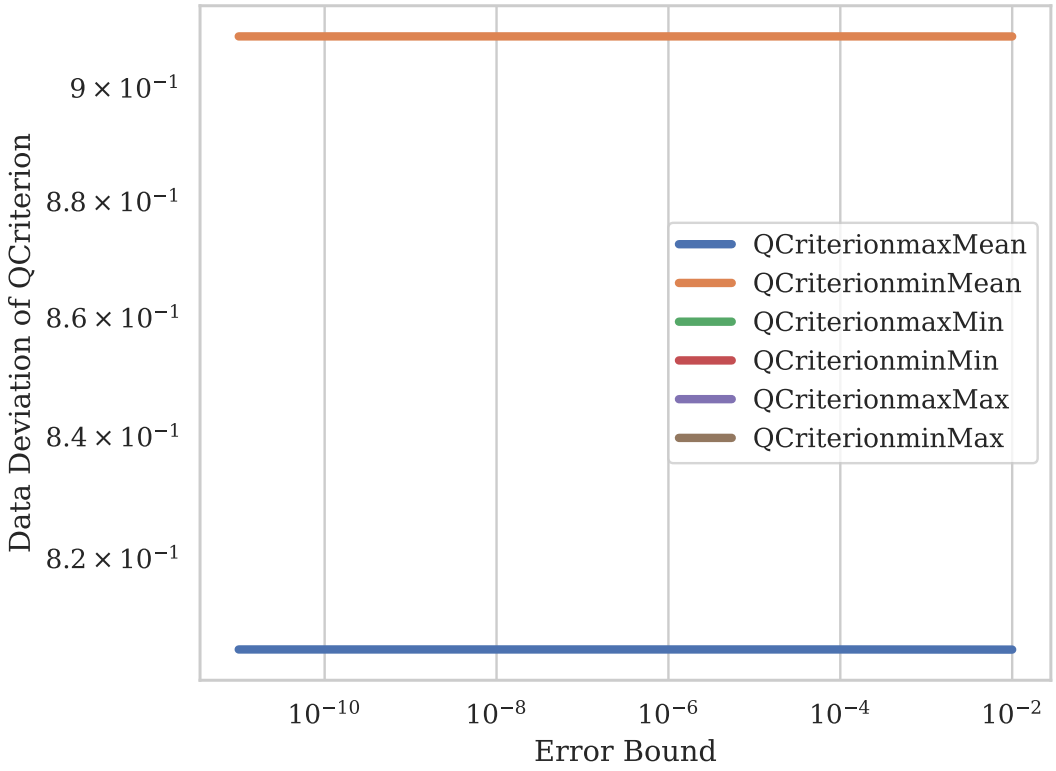

Figure 4.7: QCriterion

maxMean and QCriterionminMean shows much higher data distortions than the SZ error bounds. It can be clearly stated that these sub variables are not compatible for lossy compression.

4.4.2.6 Velocity Magnitude

From Figure 4.8 the six trends for the VelocityMagnitudemaxMean, VelocityMagnitudeminMean, VelocityMagnitudemaxMin, VelocityMagnitudeminMin, VelocityMagnitudemaxMax and VelocityMagnitudeminMax sub variables are observed. All trends show data distortions lower than the error bounds for 1e-4 and higher error bounds. VelocityMagnitudemaxMax shows data distortion within desired limits for 1e-6 and higher error bounds. VelocityMagnitudemaxMin and VelocityMagni-

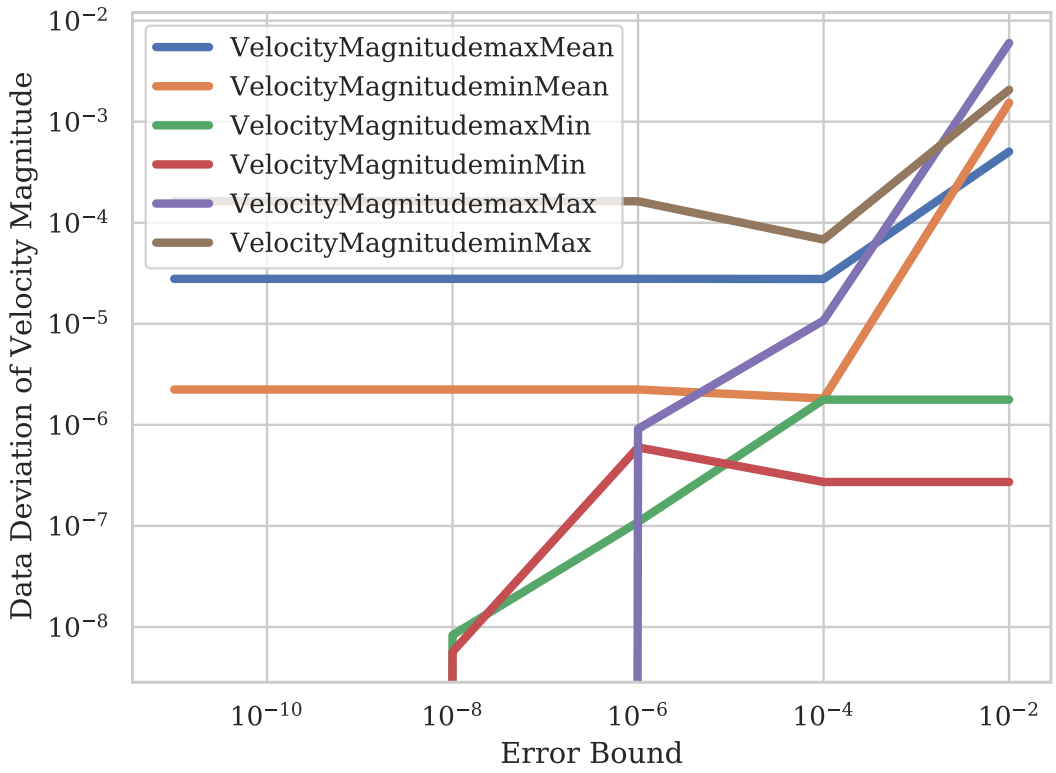

Figure 4.8: Velocity Magnitude

tudeminMin also show data distortion within desired limits for 1e-8 and higher error bounds.

4.4.2.7 Vorticity Magnitude

From Figure 4.9 the six trends for the VorticityMagnitudemaxMean, VorticityMagnitudeminMean, VorticityMagnitudemaxMin, VorticityMagnitudeminMin, VorticityMagnitudemaxMax and VorticityMagnitudeminMax sub variables are observed. VorticityMagnitudemaxMin and VorticityMagnitudeminMin shows data distortion within limits for the error bounds 1e-4 and higher. VorticityMagnitudeminMean and VorticityMagnitudeminMax shows acceptable data dostortion for error bounds 1e-2 and higher.

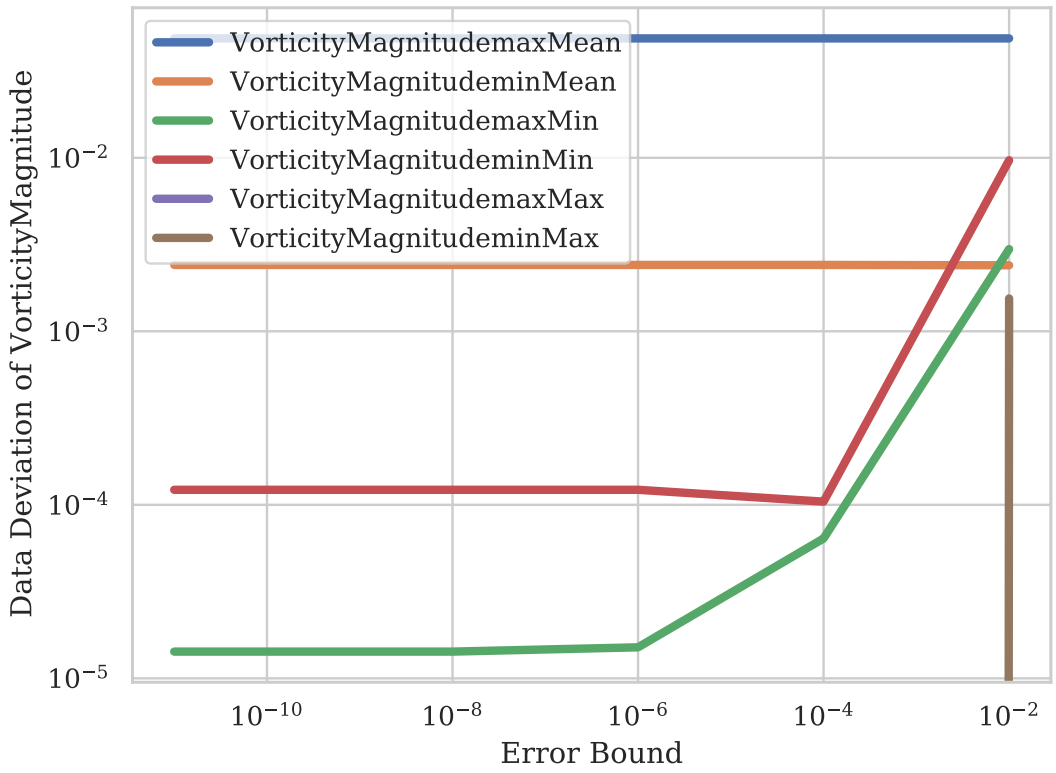

Figure 4.9: Vorticity Magnitude

4.5 Conclusion

The compression ratio plot shows that the compression is suitable for a certain error bound range. The compression increases from error bound range between 1e-6 to 1e-4. Also the compression ratios get constant after 1e-2 and higher error bound. For most observable, deviations seem to steeply increase in the error bound range 10e-6 to 10e-4, which is also the range where the compression ratio increases more steeply. The min and max values seem to be more affected than the mean values which is reasonable. The vast and random data deviation for different lossy compressed error bounds clearly indicates that the seven variables' construction from the sub variables are complex. It is difficult to directly decide the proper error bounds for the user

allowed data deviation. It also proves that not all the variables are suitable for lossy compression. For example, QCriterion performs much worse for lossy compression so it might not be cost effective to lossy compress it. The prediction for suitable error bounds is harder to predict. There is advantage of choosing which variable should be compressed as they all are compressed separately inside the simulation scripts.

Chapter 5

Statistical Methods

5.1 Types of statistical methods

It is observed from Chapter 3 and specially from Chapter 4 that lossy compression improves the experiments better by reducing I/O bottleneck, runtime and data storage space. This comes up with the issue of finding the optimal lossy compressed error bound that don't overrun the benefits by making it is difficult considering the large scale of HPC applications and complexity of the computations. In Chapter 4 the data distortion plots show that various variables shows large data distortion for smaller lossy error bounds. It actually harms the overall simulation as there is in order of magnitude data distortion for slightest lossy error bound difference. The experimental cost could have been lower if the data distortions for different error bounds could be predicted before the experimental run.

Statistics deal with collection, presentation, analysis and usage of data to make decisions and solve problems [48]. Many aspects of science and engineering requires working with large scaled data. Statistics plays an important rule in working with new designs, improving existing systems which can ultimately improve scientific

works and industrial productions. Statistical methods can be used to research how data distortion is estimated from lossy error bounds. These methods are combined of mathematical formulas, models and techniques to analyze different types of datasets. They helps researchers and users to assess based on the outputs.

It this chapter, we have experimented with various statistical methods to see if the data distortion for their computation can be predicted for lossy compression induced error bounds. These methods are presents in the following sections.

5.1.1 Mean

The arithmetic mean in statistics indicate the amount which is the sum of all values divided by the total number of values. It commonly measures the central tendency[23]. It is often used to find the middle point in a dataset organized in increasing pattern. Mean is also identified by average. The formula of mean is,

$$\overline{x} = (\sum_{i=1}^{n} x_i)/n$$

where, x_i = value of ith element in the dataset

n = Number of total elements

Most high-performance computing are performed on supercomputers where the large-scale data analysis causes storage data overhead and slower I/O. Lossy compression is implemented in high-performance computing applications with complex geometries. It gives the advantage of faster high-performance computing while using less resources such as processing and memory storage. In recent works, lossy compression has shown promising results by compressing the data with acceptable distortions ultimately making the computation faster and economically sustainable. One of the characteristics of current state-of-the-art lossy compressors is performing compression

within fixed error bounds[16]. The error bounds are user controlled because different applications have different properties where users have different tolerance for error in their data. The error bound in the compressors works as a boundary where the worst possible error bound is the user defined error bound. In this section lossy compression and its impact on data accuracy has been experimented using the following formula. The goal is to see the change in the mean value where same error is induced for every element in the dataset.

For this research we have set all the error bounds across all the data points as same user defined error bound. It is set this way because this is the highest error bound possible in any given dataset. Generally the errors stay below and upto the given error bound. It ensures that these lossy compressed results are the worst possible outcome, thus the users get an idea what error they can expect from their data.

The lossy compressed formula of mean is,

$$\overline{x_\varepsilon} = (\sum_{i=1}^{n}(x_i + \varepsilon))/n$$

where, ε = the error bound of lossy compression

This equation can be rearranged as,

$$\overline{x_\varepsilon} = (\sum_{i=1}^{n} x_i)/n + n\varepsilon/n$$

or,

$$\overline{x_\varepsilon} = \overline{x} + \varepsilon$$

Now, Data distortion is the difference between the lossy compressed mean and original mean. If we know the maximum data distortion that is allowed by the user

then we can determine the highest error bound upto which the data distortion can be within limit.

We look into the deviation by calculating the difference between the lossy mean and the original mean to assess the effect of lossy compression.

We can write this equation,

$$\tau = \overline{x_\varepsilon} - \overline{x}$$

Where, τ = Data distortion

or,

$$\tau = \varepsilon + \overline{x} - \overline{x}$$

or,

$$\tau = \varepsilon$$

Which means that for average or mean calculation, the error bound is exactly the same as the data distortion.

In Figure 5.1 the data distortion and their corresponding lossy error bounds are plotted. The x axis shows the error bounds and the y axis shows the data distortion. There are two trends for two experiments run on the same dataset. The blue trends shows the results for first experiment. In that part, the users select the data distortion and based on that the error bounds are determined. In the next experiment the data distortion is computed based on the previously found error bounds. That is the red dotted points on the graph. It is observed that both of trends show exactly the same results. It complies with out findings that for mean computation, the data distortion is exactly the same as the lossy compressed error.

The target is to determine the suitable lossy compression error bound which

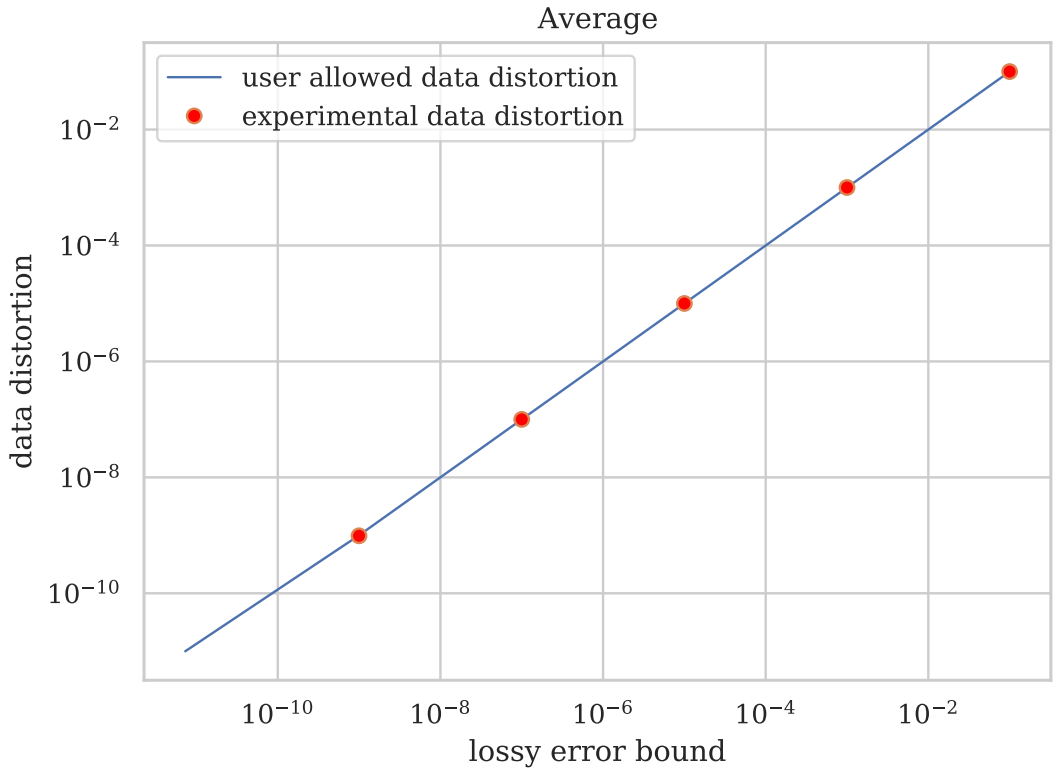

Figure 5.1: Average

limits the data distortion within the user defined range. The lossy compressed mean shows the data distortion after the experimental run. There is chance of additional computation to pin point the error bound for a particular dataset. This calculation will save time and storage space by determining the error bound before the experiment which will reduce chance of multiple experimental run.

5.1.2 Root Mean Square (RMS)

The next statistical method that we looked into is root mean square (RMS). It can be said as the square root of the arithmetic mean of the squares of all the values

[36]. It is also addressed as quadratic mean. The formula of RMS is,

$$R.M.S. = \sqrt{(\sum_{i=1}^{n} x_i^2)/n}$$

Where, x_i = value of ith element in the dataset

n = Number of total elements

We construct the formula for error introduced in the input dataset. The errors are considered such that they can be induced from lossy compression. The lossy compressed formula of R.M.S. is,

$$R.M.S._{\varepsilon} = \sqrt{(\sum_{i=1}^{n}(x_i + \varepsilon)^2)/n}$$

where, ε = the error bound of lossy compression

This equation can be rearranged as,

$$R.M.S._{\varepsilon} = \sqrt{(\sum_{i=1}^{n} x_i^2)/n + 2\varepsilon((\sum_{i=1}^{n} x_i)/n) + \varepsilon^2}$$

or,

$$R.M.S._{\varepsilon} = \sqrt{(\sum_{i=1}^{n} x_i^2)/n + 2\varepsilon\overline{x} + \varepsilon^2}$$

Where, $\overline{x}$ = Mean of the input dataset

Now, Data distortion is the difference between the lossy compressed R.M.S. and original R.M.S.

We can write this equation,

$$\tau = R.M.S._{\varepsilon} - R.M.S.$$

Where, τ = Data distortion

Again,

$$\tau = \sqrt{(\sum_{i=1}^{n} x_i^2)/n + 2\varepsilon\overline{x} + \varepsilon^2} - \sqrt{(\sum_{i=1}^{n} x_i^2)/n}$$

The simplified version of the above equation can be written as,

$$\varepsilon^2 + 2\varepsilon\overline{x} = \tau^2 + 2\tau R.M.S.$$

This is a quadratic equation. Solving it for ε gives the value of the error bound,

$$\varepsilon = \frac{-(2\overline{x}) \pm \sqrt{(2\overline{x})^2 + 4(\tau^2 + 2\tau R.M.S.)}}{2}$$

Since we are looking at only positive error bounds, we only consider the following equation for error bound determination,

$$\varepsilon = \frac{-(2\overline{x}) + \sqrt{(2\overline{x})^2 + 4(\tau^2 + 2\tau R.M.S.)}}{2}$$

The data distortion and their corresponding lossy error bounds are plotted like previous one for mean.

In Figure 5.2 the x axis shows the error bounds and the y axis shows the data distortion for root mean square. There are two trends for two experiments run on the same dataset. The blue trends shows the results for first experiment. In that part, the

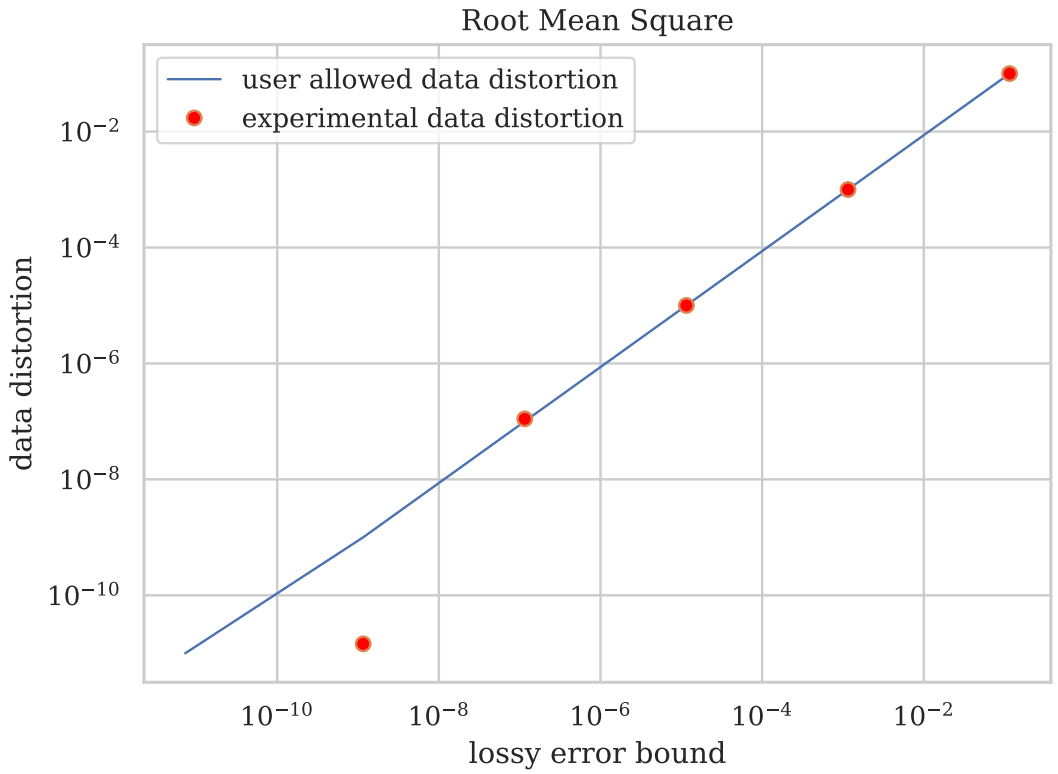

Figure 5.2: Root Mean Square

users select the data distortion and based on that the error bounds are determined. In the next experiment the data distortion is computed based on the previously found error bounds. That is the red dotted points on the graph. It is observed that both of trends show exactly the same results for lossy error bounds higher than 1e-8. For error bounds between 1e-10 and 1e-8 the data distortion is lower than what was allowed. Which means that the distortion prediction works and it keeps the distortion lower than what expected. It is in accordance with our findings for RMS computation.

5.1.3 Variance

Variance is the measure of variability from the average value of the dataset [23]. It is calculated by using the following formula:

$$Variance = (\sum_{i=1}^{n} |x_i - \overline{x}|^2)/n$$

Where, x_i = value of ith element in the dataset

$\overline{x}$ = average value in the dataset

n = Number of total elements

From the section of mean, the error bound is user defined and fixed across all the data points. The lossy compressed equation for variance is,

$$Variance_\varepsilon = (\sum_{i=1}^{n} |(x_i)_\varepsilon - \overline{x_\varepsilon}|^2)/n$$

where, ε = the error bound of lossy compression

$(x_i)_\varepsilon = x_i + \varepsilon$

$\overline{x_\varepsilon} = \overline{x} + \varepsilon$

The equation is rearranged as,

$$Variance_\varepsilon = (\sum_{i=1}^{n} |(x_i + \varepsilon) - (\overline{x} + \varepsilon)|^2)/n$$

or,

$$Variance_\varepsilon = (\sum_{i=1}^{n} |x_i - \overline{x}|^2)/n$$

It can be showed that,

$$Variance = Variance_{\varepsilon}$$

This clearly indicates that any amount of fixed valued error induction in the dataset will not change the variance value. That is because if we shift all the values in the dataset array by the same value (i.e. error bound) then the spread in the dataset is the same. Thus the variance is not changed. If the values added to each index is different the variance will change.

Since we are experimenting with the lossy compression where all the induced error values remains same across specific experiment then for our evaluation the error bounds have no effect on the variance data distortion.

5.1.4 Standard Deviation

The standard deviation is the average amount of variability in a given data set. It determines the distance of each value from the mean of the dataset [23].

Higher standard deviation means the data points are distributed far from the mean. Similarly a low standard deviation means that more data points are aggregated closer to the mean value.

$$\sigma = \sqrt{(\sum_{i=1}^{n} |x_i - \overline{x}|^2)/n}$$

where, x_i = value of ith element in the dataset

$\overline{x}$ = average value in the dataset

n = Number of total elements

From the section of mean, the error bound is user defined and fixed across all the data points. The lossy compressed equation for standard deviation is,

$$\sigma_\varepsilon = \sqrt{(\sum_{i=1}^{n} |(x_i)_\varepsilon - \overline{x_\varepsilon}|^2)/n}$$

where, ε = the error bound of lossy compression

$(x_i)_\varepsilon = x_i + \varepsilon$

$\overline{x_\varepsilon} = \overline{x} + \varepsilon$

The equation is rearranged as,

$$\sigma_\varepsilon = \sqrt{(\sum_{i=1}^{n} |(x_i + \varepsilon) - (\overline{x} + \varepsilon)|^2)/n}$$

or,

$$\sigma_\varepsilon = \sqrt{(\sum_{i=1}^{n} |x_i - \overline{x}|^2)/n}$$

It can be showed that,

$$\sigma = \sigma_\varepsilon$$

5.1.5 Median

The median is the middle number in a data set. The data points are organized in ascending order and then the middle number is the median. The formula for finding median is following,

$$Median = ((n+1)/2)^{th}\, Term$$

when is n is odd

$$Median = ((n/2)^{th}\, Term + ((n/2) + 1)^{th}\, Term)/2$$

when n is even

where, n = number of elements in the dataset

From the section of mean, the error bound is user defined and fixed across all the data points. The lossy compressed equation is,

$$Median = ((n + 1)/2)^{th}\, Term + \varepsilon$$

when is n is odd

$$Median = (((n/2)^{th}\, Term + \varepsilon) + (((n/2) + 1)^{th}\, Term) + \varepsilon)/2$$

when n is even

where, ε = the error bound of lossy compression

It can be seen that for median the lossy compression the error bound is directly related to the data distortion since all the data values are getting induced the same error.

5.1.6 Interquartile Range

The interquartile range is a statistical method which indicates the region containing the bulk of the points in a dataset. The interquartile range formula is following:

$$IQR = Q_3 - Q_1$$

where, IQR = Interquartile Range

Q_3 = Third quartile

Q_1 = First quartile

Q_1 and Q_3 can be determined from the following formulas,

$$Q_1 = ((n+1)/4)^{th}\, Term$$

$$Q_3 = (3(n+1)/4^{th})\, Term$$

For the lossy compression the error bound is user defined and fixed across all the data points. The lossy compressed equation is the same as the lossless equation as the error bounds cancel out for Q_1 and Q_3,

5.2 Conclusion

The work in this chapter presents that some statistical methods predict the data distortion for lossy error bounds. It depends on the structure of the statistical mathematical formula. For example, the data distortion and error bound relations are shown for the mean and root mean square. This is useful for scientists and researchers lossy compressing mean and root mean square computations. The variance and standard deviation doesn't show any effect for any errors. That is because they

look at the distribution of the data and exactly same error for each data value keeps the distribution same. The median and quarantile depends on the specific position of a data value in the ascendingly organized dataset. The data distortion will be exactly the same as the lossy error. This work concludes that there is merit in investigating different statistical methods to find the lossy errors that will keep the data distortion within bounds.

Chapter 6

Conclusions and Discussion

6.1 Contribution

The goal of this research is to investigate the lossy compression performance of HPC application simulations. The part of this work is to study methods to predict the data distortion from lossy compression beforehand. Statistical methods have been used to create model and investigate the pre detection of data distortion. It was showed that lossy compression can improve the HPC applications but at the same time there are some challenges that threatened the gain from the lossy compression. Chapter 3 showed promising compression gain for lossy compression on sub-iteration level of NWChem. The compression bandwidth, decompression bandwidth and compression ratio for number of sub-tensors were examined to look at the results. The absolute error bounds performed significantly better for error bounds range from 1e-1 to 1e-5. This gave good results considering the significantly lower increase of extra iteration only upto 1 and lower energy deviation in sub-tensor level checkpoint restart.

The HemeLB blood flow simulation data gave variety of results for lossy compression. The compression ratio and the data distortion were studied. It was observed

that the compression ratio is better in the error bound range of 10e-6 to 10e-4. The lossy compression for different variables do not give predicted results always. The construction for different variables are complex in nature which makes the lossy compression analysis difficult. Another advantage of separate variable compression is that the user can choose the suitable variables for improved results.

The last portion of the work looks into the statistical methods for lossy error bound prediction based on user defined data distortion. Some variables can be used to predict the error bound while for other variables the relation between error bound and data distortion was not established. It can seen that the relation depends on the mathematical formula of the methods. This work can be explored in future studies that would be beneficial for scientists and researchers working with large scale HPC applications.

Bibliography

[1] https://matdat18.wordpress.ncsu.edu/files/2018/02/Schiller.pdf.

[2] Alexander A Auer, Gerald Baumgartner, David E Bernholdt, Alina Bibireata, Venkatesh Choppella, Daniel Cociorva, Xiaoyang Gao, Robert Harrison, Sriram Krishnamoorthy, Sandhya Krishnan, et al. Automatic code generation for many-body electronic structure methods: the tensor contraction engine. *Molecular Physics*, 104(2):211–228, 2006.

[3] Allison H. Baker, Haiying Xu, John M. Dennis, Michael N. Levy, Doug Nychka, Sheri A. Mickelson, Jim Edwards, Mariana Vertenstein, and Al Wegener. A methodology for evaluating the impact of data compression on climate simulation data. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '14, pages 203–214, New York, NY, USA, 2014. ACM.

[4] Charles Henry Brase and Corrinne Pellillo Brase. *Understandable statistics*. Cengage Learning, 2014.

[5] Moysey Brio and Cheng Chin Wu. An upwind differencing scheme for the equations of ideal magnetohydrodynamics. *Journal of computational physics*, 75(2):400–422, 1988.

[6] M. Burtscher and P. Ratanaworabhan. High throughput compression of double-precision floating-point data. In *Data Compression Conference, 2007. DCC '07*, pages 293–302, March 2007.

[7] Jon Calhoun, Franck Cappello, Luke N Olson, Marc Snir, and William D Gropp. Exploring the feasibility of lossy compression for pde simulations. *The International Journal of High Performance Computing Applications*, 33(2):397–410, 2019.

[8] ASC FLASH Center. Flash user's guide. *Chicago: University of Chicago*, 2005.

[9] Hua Chai, Jun He, and Christophe Lombard. Checkpoint and restart, April 12 2018. US Patent App. 15/840,235.

[10] RM Churchill, Choong Seock Chang, S Ku, and Julien Dominski. Pedestal and edge electrostatic turbulence characteristics from an xgc1 gyrokinetic simulation. *Plasma Physics and Controlled Fusion*, 59(10):105014, 2017.

[11] Phillip Colella and Paul R Woodward. The piecewise parabolic method (ppm) for gas-dynamical simulations. *Journal of computational physics*, 54(1):174–201, 1984.

[12] Yann Collet and Murray Kucherawy. Zstandard Compression and the application/zstd Media Type. RFC 8478, October 2018.

[13] National Research Council et al. *Getting up to speed: The future of supercomputing*. National Academies Press, 2005.

[14] P. Deutsch. Gzip file format specification version 4.3. Technical report, United States, 1996.

[15] Sheng Di and Franck Cappello. Fast error-bounded lossy hpc data compression with sz. In *2016 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2016, Chicago, IL, USA, May 23-27, 2016*, pages 730–739, 2016.

[16] Sheng Di, Dingwen Tao, Xin Liang, and Franck Cappello. Efficient lossy compression for scientific data based on pointwise relative error bound. *IEEE Transactions on Parallel and Distributed Systems*, 30(2):331–345, 2019.

[17] James Diffenderfer, Alyson Fox, Jeffrey Hittinger, Geoffrey Sanders, and Peter Lindstrom. Error analysis of zfp compression for floating-point data. *SIAM Journal on Scientific Computing*, 02 2019.

[18] Jack Dongarra. Trends in high-performance computing. In *Handbook of Nature-Inspired and Innovative Computing*, pages 511–520. Springer, 2006.

[19] Elmootazbellah Nabil Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys (CSUR)*, 34(3):375–408, 2002.

[20] P Fisher. Nek5000 user guide, 2010.

[21] Ian Foster, Mark Ainsworth, Bryce Allen, Julie Bessac, Franck Cappello, Jong Youl Choi, Emil Constantinescu, Philip E. Davis, Sheng Di, Wendy Di, Hanqi Guo, Scott Klasky, Kerstin Kleese Van Dam, Tahsin Kurc, Qing Liu, Abid Malik, Kshitij Mehta, Klaus Mueller, Todd Munson, George Ostouchov, Manish Parashar, Tom Peterka, Line Pouchard, Dingwen Tao, Ozan Tugluk, Stefan Wild, Matthew Wolf, Justin M. Wozniak, Wei Xu, and Shinjae Yoo. Computing just what you need: Online data analysis and reduction at extreme scales.

In Francisco F. Rivera, Tomás F. Pena, and José C. Cabaleiro, editors, *Euro-Par 2017: Parallel Processing*, pages 3–19, Cham, 2017. Springer International Publishing.

[22] Nathaniel Fout and Kwan-Liu Ma. An adaptive prediction-based approach to lossless compression of floating-point volume data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2295–2304, 2012.

[23] D. Freedman, R.L. Pisani, R. Purves, and A. Adhikari. *Statistics*. Norton, 1991.

[24] Alexander Gaenko, Albert DeFusco, Sergey A Varganov, Todd J Martinez, and Mark S Gordon. Interfacing the ab initio multiple spawning method with electronic structure methods in gamess: Photodecay of trans-azomethane. *The Journal of Physical Chemistry A*, 118(46):10902–10908, 2014.

[25] Stephanie Glen. "summary statistics: Definition and examples" from statisticshowto.com: Elementary statistics for the rest of us! `https://www.statisticshowto.com/summary-statistic`.

[26] Derek Groen, James Hetherington, Hywel B Carver, Rupert W Nash, Miguel O Bernabeu, and Peter V Coveney. Analysing and modelling the performance of the hemelb lattice-boltzmann simulation environment. *Journal of Computational Science*, 4(5):412–422, 2013.

[27] Trygve Helgaker, Torgeir A. Ruden, Poul Jørgensen, Jeppe Olsen, and Wim Klopper. A priori calculation of molecular properties to chemical accuracy. *Journal of Physical Organic Chemistry*, 17(11):913–933, 2004.

[28] David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes*, pages 1–2. Springer, 1935.

[29] So Hirata. Tensor contraction engine: Abstraction and automated parallel implementation of configuration-interaction, coupled-cluster, and many-body perturbation theories. *The Journal of Physical Chemistry A*, 107(46):9887–9897, 2003.

[30] S. Hogan, J. R. Hammond, and A. A. Chien. An evaluation of difference and threshold techniques for efficient checkpoints. In *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*, pages 1–6, June 2012.

[31] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[32] James W Hurrell, Marika M Holland, Peter R Gent, Steven Ghan, Jennifer E Kay, Paul J Kushner, J-F Lamarque, William G Large, D Lawrence, Keith Lindsay, et al. The community earth system model: a framework for collaborative research. *Bulletin of the American Meteorological Society*, 94(9):1339–1360, 2013.

[33] Lawrence Ibarria, Peter Lindstrom, Jarek Rossignac, and Andrzej Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. In *Computer Graphics Forum*, volume 22, pages 343–348. Wiley Online Library, 2003.

[34] Dewan Ibtesham, Kurt B. Ferreira, and Dorian C. Arnold. A checkpoint compression study for high-performance computing systems. *IJHPCA*, 29(4):387–402, 2015.

[35] Tanzima Zerin Islam, Kathryn Mohror, Saurabh Bagchi, Adam Moody, Bronis R. de Supinski, and Rudolf Eigenmann. McrEngine: a scalable checkpointing system using data-aware aggregation and compression. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 17:1–17:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[36] A.R. Jones. *Probability, Statistics and Other Frightening Stuff*. Working Guides to Estimating & Forecasting. Taylor & Francis, 2018.

[37] Ricky A Kendall, Edoardo Aprà, David E Bernholdt, Eric J Bylaska, Michel Dupuis, George I Fann, Robert J Harrison, Jialin Ju, Jeffrey A Nichols, Jarek Nieplocha, et al. High performance computational chemistry: An overview of nwchem a distributed parallel application. *Computer Physics Communications*, 128(1-2):260–283, 2000.

[38] Daniel Laney, Steven Langer, Christopher Weber, Peter Lindstrom, and Al Wegener. Assessing the effects of data compression in simulations using physically motivated metrics. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 76:1–76:12, New York, NY, USA, 2013. ACM.

[39] Kenneth Laws, Jeffrey D Paduan, and John Vesecky. Estimation and assessment of errors related to antenna pattern distortion in codar seasonde high-frequency radar ocean current measurements. *Journal of Atmospheric and Oceanic Technology*, 27(6):1029–1043, 2010.

[40] Henri Lebesgue. Sur l'intégration et la recherche des fonctions primitives, 1904.

[41] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. Error-controlled lossy compression optimized for

high compression ratios of scientific datasets. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 438–447, 2018.

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

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

[44] Peter Lindstrom. Error distributions of lossy floating-point compressors. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2017.

[45] Peter Lindstrom. Error distributions of lossy floating-point compressors. *Joint Statistical Meetings 2017*, pages 2574–2589, October 2017.

[46] Marco D Mazzeo and Peter V Coveney. Hemelb: A high performance parallel lattice-boltzmann code for large scale fluid flow in complex geometries. *Computer Physics Communications*, 178(12):894–914, 2008.

[47] H Meng, VM Tutino, J Xiang, and A Siddiqui. High wss or low wss? complex interactions of hemodynamics with intracranial aneurysm initiation, growth, and rupture: toward a unifying hypothesis. *American Journal of Neuroradiology*, 35(7):1254–1262, 2014.

[48] Douglas C Montgomery, George C Runger, and Norma F Hubele. *Engineering statistics*. John Wiley & Sons, 2009.

[49] Eliakim Hastings Moore. Errata:"on certain crinkly curves"[trans. amer. math. soc. 1 (1900), no. 1, 72–90; 1500526]. *Transactions of the American Mathematical Society*, 1(4):507, 1900.

[50] Joseph Nardi, Noah Feldman, Andrew Poppick, Allison Baker, and Dorit Hammerling. Statistical analysis of compressed climate data. Technical report, NCAR, 2018.

[51] Xiang Ni, Tanzima Islam, Kathryn Mohror, Adam Moody, and Laxmikant V Kale. Lossy compression for checkpointing: Fallible or feasible? In *Poster Session of the 2014 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, Washington, DC, USA, 2014. IEEE Computer Society.

[52] A Obabko. Simulation of gallium experiment, 2005.

[53] Giuseppe Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[54] Marina Piccinelli, Alessandro Veneziani, David A Steinman, Andrea Remuzzi, and Luca Antiga. A framework for geometric analysis of vascular structures: application to cerebral aneurysms. *IEEE transactions on medical imaging*, 28(8):1141–1155, 2009.

[55] Pavel Pokhilko, Evgeny Epifanovsky, and Anna I Krylov. Double precision is not needed for many-body calculations: Emergent conventional wisdom. *Journal of chemical theory and computation*, 14(8):4088–4096, 2018.

[56] George D Purvis III and Rodney J Bartlett. A full coupled-cluster singles and doubles model: The inclusion of disconnected triples. *The Journal of Chemical Physics*, 76(4):1910–1918, 1982.

[57] Krishnan Raghavachari, Gary W Trucks, John A Pople, and Martin Head-Gordon. A fifth-order perturbation comparison of electron correlation theories. *Chemical Physics Letters*, 157(6):479–483, 1989.

[58] Naoto Sasaki, Kento Sato, Toshio Endo, and Satoshi Matsuoka. Exploration of lossy compression for application-level checkpoint/restart. In *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium*, IPDPS '15, pages 914–922, Washington, DC, USA, 2015. IEEE Computer Society.

[59] Khalid Sayood. *Introduction to data compression*. Newnes, 2012.

[60] Leonid Ivanovich Sedov and AG Volkovets. *Similarity and dimensional methods in mechanics*. CRC press, 2018.

[61] Marc Snir, Robert W Wisniewski, Jacob A Abraham, Sarita V Adve, Saurabh Bagchi, Pavan Balaji, Jim Belak, Pradip Bose, Franck Cappello, Bill Carlson, Andrew A Chien, Paul Coteus, Nathan A DeBardeleben, Pedro C Diniz, Christian Engelmann, Mattan Erez, Saverio Fazzari, Al Geist, Rinku Gupta, Fred Johnson, Sriram Krishnamoorthy, Sven Leyffer, Dean Liberty, Subhasish Mitra, Todd Munson, Rob Schreiber, Jon Stearley, and Eric Van Hensbergen. Addressing failures in exascale computing. *International Journal of High Performance Computing Applications*, 28(2):127 – 171, May 2014.

[62] Dingwen Tao, Sheng Di, Xin Liang, Zizhong Chen, and Franck Cappello. Improving performance of iterative methods by lossy checkponting. In *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '18, pages 52–65, New York, NY, USA, 2018. ACM.

[63] Robert Underwood, Sheng Di, Jon C Calhoun, and Franck Cappello. Fraz: A generic high-fidelity fixed-ratio lossy compression framework for scientific floating-point data. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 567–577. IEEE, 2020.

[64] Marat Valiev, Eric J Bylaska, Niranjan Govind, Karol Kowalski, Tjerk P Straatsma, Hubertus JJ Van Dam, Dunyou Wang, Jarek Nieplocha, Edoardo Apra, Theresa L Windus, et al. Nwchem: A comprehensive and scalable open-source solution for large scale molecular simulations. *Computer Physics Communications*, 181(9):1477–1489, 2010.

[65] Owen Walsh. Eddy solutions of the navier-stokes equations. In *The Navier-Stokes Equations II—Theory and Numerical Methods*, pages 306–309. Springer, 1992.

[66] Andrew L Zachary, Andrea Malagoli, and Phillip Colella. A higher-order godunov method for multidimensional ideal magnetohydrodynamics. *SIAM Journal on Scientific Computing*, 15(2):263–284, 1994.

[67] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.