Analyzing the Performance and Accuracy of Lossy Checkpointing on Sub-iteration of NWChem

Tasmia Reza*, Kristopher Keipert†, Sheng Di†, Xin Liang‡, Jon C. Calhoun*, Franck Cappello†
*Holcombe Department of Electrical and Computer Engineering,
Clemson University, Clemson, South Carolina 29634
†Argonne National Laboratory, Lemont, USA
‡ University of California, Riverside, CA, USA
treza@g.clemson.edu, kkeipert@anl.gov, sdi1@anl.gov,
xlian007@ucr.edu, jonccal@clemson.edu, cappello@mcs.anl.gov

***Abstract*—Future exascale systems are expected to be characterized by more frequent failures than current petascale systems. This places increased importance on the application to minimize the amount of time wasted due to recompution when recovering from a checkpoint. Typically HPC applications checkpoint at iteration boundaries; however, for applications that have a high per-iteration cost, checkpointing inside the iteration limits the amount of re-computation. This paper analyzes the performance and accuracy of using lossy compressed checkpointing in the computational chemistry application NWChem. Our results indicate that lossy compression is an effective tool for reducing the sub-iteration checkpoint size. Moreover, compression error tolerances that yield acceptable deviation in accuracy and iteration count are quantified.**

***Index Terms*—lossy data compression, checkpoint-restart, NWChem, coupled-cluster singles and doubles**

I. 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 [1]. 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 [2]. In order to recover from the more frequent failures, applications will need to rely increasingly on checkpoint-restart [3]. 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 [4], [5]. Recent work shows that lossy compression is able to reduce data volumes by order-of-magnitudes more than lossless compression [6]. 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 [4], [7], [8]. 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 [9].

There are various HPC applications that have been used to test the effectiveness of lossy compression by varying different parameters of the application and the compressors. For this work, we explore checkpointing of NWChem [10], 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 [11], 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 SZ [6], a state-of-the-art HPC lossy compressor with user controlled error bound type and error bound.

This paper makes the following contributions:

- 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.

The rest of the paper is as follows: Section II discusses data compression and the NWChem application. Section III describes the how we instrument NWChem with lossy data compression. Section IV presents the performance results we obtain. Finally, Section VI states our conclusions and our future work.

II. Background

A. Compression for HPC Data

Data compression techniques are classified into two categories such as lossless and lossy. Lossless compression [12]–[15] reduces data size without loss in data fidelity. However, for numerical HPC data, the compression ratios vary between

1-4×. Lossy compression achieves higher compression ratios compared to lossless compression, by an order-of-magnitude, by allowing inaccuracies into the data when compressing. Modern HPC lossy compressors such as SZ [6] and ZFP [16] 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 [4], [17]. Determining the error bound and error bounding type for various applications remains an open question.

B. NWChem

NWChem [10] is a computational chemistry software package that provides a comprehensive range of methods used to address molecular simulation problems. NWChem development is focused on providing highly scalable algorithms that can efficiently utilize high-performance distributed-memory supercomputers. The methods implemented in NWChem represent tradeoffs between computational cost and accuracy, where the most demanding methods can easily saturate the computational throughput and memory bandwidth of today's petascale systems for relatively small molecular systems.

In this work, we focus on a many-body method called coupled-cluster singles and doubles (CCSD) [18]. CCSD is a widely used iterative method that serves as a precursor to the "gold standard" method of computational chemistry, CCSD(T) [19]. Specifically, this work targets a variation of the CCSD algorithm that was implemented with the Tensor Contraction Engine (TCE) [20], [21]. 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 [21]. 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 [11], only lossless compressors were used. Motivated by recent progress in reduced and mixed-precision iterative refinement within CCSD and other many-body methods [22], 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

III. 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 [11]. 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.

IV. 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

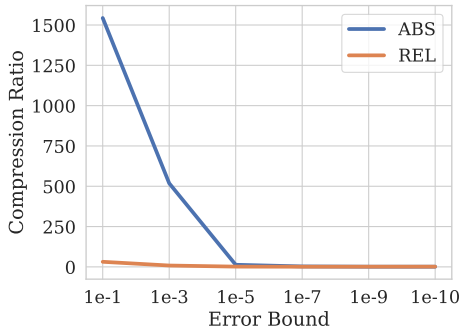

Fig. 1: Average compression ratio of compressing sub-tensors of T2 individually.

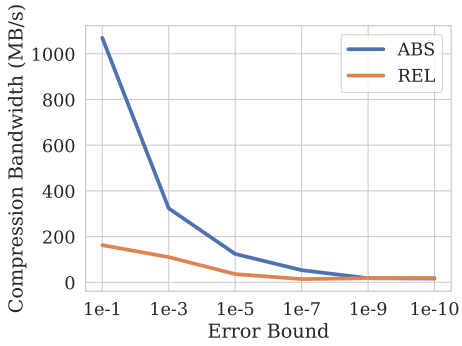

Fig. 2: Average compression bandwidth of compressing sub-tensors of T2 individually.

an unusable result due to violating conservation of energy [23]. We explore compressing sub-tensors individually and all together.

A. 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 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$\times$ across all the error bounds. As we demand more accurate data from the compressor the compression ratio tends toward 1 indicating that the data does not compress. From Figure 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 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 plots the average decompression bandwidth for both error bounds. As with Figure 2, we average across sub-tensors with a corresponding error bound and error bounding type. From Figure 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,

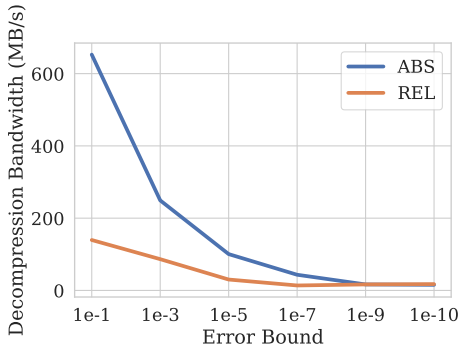

Fig. 3: Average decompression bandwidth of compressing sub-tensors of T2 individually.

the decompression bandwidth approaches zero which can lead to increased overhead for NWChem simulations with large quantifies of data.

Figure 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 $\tilde{T}_2$ and successfully restart.

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

B. 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

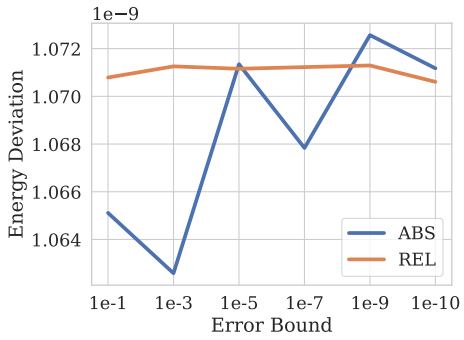

Fig. 4: Average energy deviation of compressing sub-tensors of T2 individually.

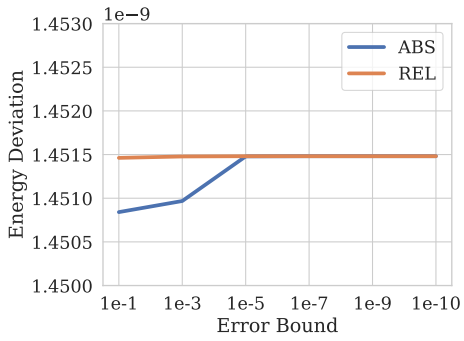

Fig. 5: Energy Deviation for compressing all sub-tensors simultaneously.

are lossy compressed. We do not show plots for compress bandwidth, decompression bandwidth, and compression ratio as they are equivalent to those shown in Section IV-A. This is due to how we checkpoint each sub-tensor individually.

In Figure 5 we plot the energy deviation for various absolute and relative error bounds ranging from $1e{-}1$ to $1e{-}10$. Comparing to Figure 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.

V. Related Work

Previous works on integrating lossy checkpointing into HPC applications have shown reductions in the I/O fraction of HPC application [4], required compression levels to improve performance [8], and have modeled checkpointing when extra iterations are required to restore convergence [7]. 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 [24]–[26]. Incorporating domain knowledge allows for establishing methodologies and heuristics for using lossy compressed data for analytics [17], [27], [28]. Different error bounding metrics have an impact on floating-point truncation error [29] as well as the distribution of compression error [30]. Other researchers have worked to find methodologies for selecting error tolerances for lossy checkpoint-restart on HPC simulations [4], [7]. This work reduces NWChem's checkpointing size by using differenced checkpoint and cutoff techniques to increase the effectiveness of Lempel-Liv (gzip) [9]. This has dramatically increased the compression ratios than standard compression techniques.

VI. Conclusion and Future Work

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 paper, we explore lossy checkpointing of sub-iterations of NWChem. We explore 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 level lossy checkpointing acceptable in NWChem application.

Future work focus on studying the impact of restarting multiple times from lossy compressed checkpoints and evaluating the resultant energy deviation and increase in number of iterations. In addition, we plan to derive and verify a performance model for lossy checkpointing NWChem to understand the trade-off between the number of extra iterations upon restart and time saved by lossy compressed checkpointing.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. SHF-1910197 and Grant No. SHF-1619253. This work is supported by the US Department of Energy under subaward No. 9F-60179. 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 supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

REFERENCES

[1] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, A. A. Chien, P. Coteus, N. A. DeBardeleben, P. C. Diniz, C. Engelmann, M. Erez, S. Fazzari, A. Geist, R. Gupta, F. Johnson, S. Krishnamoorthy, S. Leyffer, D. Liberty, S. Mitra, T. Munson, R. Schreiber, J. Stearley, and E. V. Hensbergen, "Addressing failures in exascale computing," *International Journal of High Performance Computing Applications*, vol. 28, no. 2, pp. 127 – 171, May 2014.

[2] H. Chai, J. He, and C. Lombard, "Checkpoint and restart," Apr. 12 2018, uS Patent App. 15/840,235.

[3] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Computing Surveys (CSUR)*, vol. 34, no. 3, pp. 375–408, 2002.

[4] J. Calhoun, F. Cappello, L. N. Olson, M. Snir, and W. D. Gropp, "Exploring the feasibility of lossy compression for pde simulations," *The International Journal of High Performance Computing Applications*, vol. 33, no. 2, pp. 397–410, 2019.

[5] T. Z. Islam, K. Mohror, S. Bagchi, A. Moody, B. R. de Supinski, and R. 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*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 17:1–17:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389020

[6] S. Di and F. 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*, 2016, pp. 730–739. [Online]. Available: https://doi.org/10.1109/IPDPS.2016.11

[7] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Improving performance of iterative methods by lossy checkponting," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: ACM, 2018, pp. 52–65. [Online]. Available: http://doi.acm.org/10.1145/3208040.3208050

[8] D. Ibtesham, K. B. Ferreira, and D. C. Arnold, "A checkpoint compression study for high-performance computing systems," *IJHPCA*, vol. 29, no. 4, pp. 387–402, 2015. [Online]. Available: http://dx.doi.org/10.1177/1094342015570921

[9] 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)*, June 2012, pp. 1–6.

[10] M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski, T. P. Straatsma, H. J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T. L. Windus *et al.*, "Nwchem: A comprehensive and scalable open-source solution for large scale molecular simulations," *Computer Physics Communications*, vol. 181, no. 9, pp. 1477–1489, 2010.

[11] 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)*, June 2012, pp. 1–6.

[12] P. Deutsch, "Gzip file format specification version 4.3," United States, Tech. Rep., 1996.

[13] Y. Collet and M. Kucherawy, "Zstandard Compression and the application/zstd Media Type," RFC 8478, Oct. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8478.txt

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

[15] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 5, pp. 1245–1250, 2006. [Online]. Available: http://dx.doi.org/10.1109/tvcg.2006.143

[16] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, Dec 2014.

[17] J. Nardi, N. Feldman, A. Poppick, A. Baker, and D. Hammerling, "Statistical analysis of compressed climate data," NCAR, Tech. Rep., 2018.

[18] G. D. Purvis III and R. J. Bartlett, "A full coupled-cluster singles and doubles model: The inclusion of disconnected triples," *The Journal of Chemical Physics*, vol. 76, no. 4, pp. 1910–1918, 1982.

[19] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon, "A fifth-order perturbation comparison of electron correlation theories," *Chemical Physics Letters*, vol. 157, no. 6, pp. 479–483, 1989.

[20] A. A. Auer, G. Baumgartner, D. E. Bernholdt, A. Bibireata, V. Choppella, D. Cociorva, X. Gao, R. Harrison, S. Krishnamoorthy, S. Krishnan *et al.*, "Automatic code generation for many-body electronic structure methods: the tensor contraction engine," *Molecular Physics*, vol. 104, no. 2, pp. 211–228, 2006.

[21] S. 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*, vol. 107, no. 46, pp. 9887–9897, 2003.

[22] P. Pokhilko, E. Epifanovsky, and A. I. Krylov, "Double precision is not needed for many-body calculations: Emergent conventional wisdom," *Journal of chemical theory and computation*, vol. 14, no. 8, pp. 4088–4096, 2018.

[23] T. Helgaker, T. A. Ruden, P. Jrgensen, J. Olsen, and W. Klopper, "A priori calculation of molecular properties to chemical accuracy," *Journal of Physical Organic Chemistry*, vol. 17, no. 11, pp. 913–933, 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/poc.841

[24] X. Ni, T. Islam, K. Mohror, A. Moody, and L. 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*, ser. SC '14. Washington, DC, USA: IEEE Computer Society, 2014.

[25] D. Laney, S. Langer, C. Weber, P. Lindstrom, and A. 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*, ser. SC '13. New York, NY, USA: ACM, 2013, pp. 76:1–76:12. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503283

[26] N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, "Exploration of lossy compression for application-level checkpoint/restart," in *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 914–922. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2015.67

[27] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. 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*, ser. HPDC '14. New York, NY, USA: ACM, 2014, pp. 203–214. [Online]. Available: http://doi.acm.org/10.1145/2600212.2600217

[28] I. Foster, M. Ainsworth, B. Allen, J. Bessac, F. Cappello, J. Y. Choi, E. Constantinescu, P. E. Davis, S. Di, W. Di, H. Guo, S. Klasky, K. K. Van Dam, T. Kurc, Q. Liu, A. Malik, K. Mehta, K. Mueller, T. Munson, G. Ostouchov, M. Parashar, T. Peterka, L. Pouchard, D. Tao, O. Tugluk, S. Wild, M. Wolf, J. M. Wozniak, W. Xu, and S. Yoo, "Computing just what you need: Online data analysis and reduction at extreme scales," in *Euro-Par 2017: Parallel Processing*, F. F. Rivera, T. F. Pena, and J. C. Cabaleiro, Eds. Cham: Springer International Publishing, 2017, pp. 3–19.

[29] J. Diffenderfer, A. Fox, J. Hittinger, G. Sanders, and P. Lindstrom, "Error analysis of zfp compression for floating-point data," *SIAM Journal on Scientific Computing*, 02 2019.

[30] P. Lindstrom, "Error distributions of lossy floating-point compressors," *Joint Statistical Meetings 2017*, pp. 2574–2589, October 2017.