Lossy Compression for Checkpointing: Fallible or Feasible?

Xiang Ni*, Tanzima Islam†, Kathryn Mohror†, Adam Moody† and Laxmikant V. Kale*

*Department of Computer Science, University of Illinois at Urbana-Champaign
†Lawrence Livermore National Laboratory
E-mail: xiangni2@illinois.edu, islam3@llnl.gov, kathryn@llnl.gov, moody20@llnl.gov, kale@illinois.edu

I. Introduction

The predominant fault tolerance method for high performance computing (HPC) applications is checkpoint/restart, where an application periodically writes its state in checkpoint files on stable storage, such as a parallel file system (PFS). After a failure, the application reads in the most recent checkpointed state and resumes computation. While this approach is simple, large overheads can occur as applications scale up to hundreds of thousands of processors and higher, because a significant amount of time is required to transfer the large volume of checkpoint files to stable storage.

Compression techniques have been used to reduce checkpoint size. However, due to the randomness of the lower bits in scientific data, the compression ratio achieved by lossless compression is not very high. For example, checkpoints of the PF3D code from Lawrence Livermore National Laboratory (LLNL) have only been compressed by about 15% [1].

On the other hand, lossy compression, which does not perfectly reproduce the data after decompression, can reduce checkpoint size by 3 to $5\times$. However, the key challenge is to guarantee that when restarting from these lossy compressed checkpoints, scientific applications still produce the correct results. The concept of correctness is application dependent; some applications may have less inherent tolerance to loss of precision than others.

In this poster, we investigate the trade-off between compression ratio and loss of precision for applications by injecting failures at different rates, and evaluating the simulation results against correct, expected results. We also inject failures following different distributions to study whether an application is more sensitive to precision loss at earlier or later stages of the simulation.

II. Approach

Our approach requires minimal changes to application code. The pseudo code block (Figure 1 in the poster) explains how adding only one line of code turns on lossy compression for selected data fields. The integer parameter passed to the function `compression_on` specifies the number of bits of significance preserved after compression. A smaller integer value indicates higher lossiness in the compression. We found that this feature is useful for selective precision control. We turn off lossy compression before compressing control variables, which we expect to have the least tolerance of precision loss.

III. Results

A. Experimental Setup

We used the ChaNGa application, an N-body cosmology simulation, in Charm++ [2] for our experiments. For lossy compression of checkpoints, we used fpzip [3], and for lossless compression we used gzip [4]. We ran ChaNGa using the data set *dwf1* in all experiments.

B. Compression

The original, uncompressed checkpoint size of our ChaNGa runs was 700 MB. After applying gzip compression, the checkpoint size was reduced by almost half. The fpzip 32-bit compressor further reduced the checkpoint size to 260 MB. With gzip as a second level compressor after lossy compression, the final checkpoint size was 159 MB, just 22% of the original size.

Although applying the fpzip 16-bit compressor results in even smaller files, we found that ChaNGa would hang after restarting with the reduced checkpoints. Further investigation suggested that the inaccuracy of the position data field caused ChaNGa to hang. To avoid this, we selectively used the fpzip 32-bit compressor for the position data and the fpzip 16-bit compressor for the remaining data. Hence, using the mix of fpzip-16 and fpzip-32 compressors in combination with gzip, we were able to reduce the checkpoint size to 100MB, only 15% of the original size.

C. Effect of failure rate

We injected a varying number of failures using a uniform distribution throughout several one-hour runs of ChaNGa. We examined the dark matter density profile at the end of each run to verify the correctness of the result [5]. In Figure 3 of the poster, we show that when using the fpzip compressor with lossy level 32, no matter how many failures we injected in the run (ranging from 3 to 70), the simulation output was correct, based on dark matter density profile produced by each run. When using a lossy level of 16, as shown in Figure 4, the dark matter density profile matches well when the mean time between failure (MTBF) is larger than 720 seconds. However, the profile differs more drastically for smaller MTBF values, especially when the radius is small.

D. Effect of failure distribution

We injected 10 failures following the Weibull distribution into two ChaNGa runs with fpzip 16-bit lossy compression. For the first run, we used a shape parameter of 0.4, leading to a larger number of failures at the beginning of the run than at then end; for the second, we used a shape parameter of 1.5, resulting in more failures at the end of the run. As shown in the Figure 5 of the poster, when injecting the failures using a shape parameter of 1.5, the dark matter density profile diverges more from the baseline compared to the profile when using a shape parameter of 0.4. Hence, we conclude that ChaNGa is more sensitive to precision loss during the later stage of the simulation.

IV. Future Work

In the future, we plan to extend our study to more simulations that represent different application classes such as linear algebra and chaotic applications. We are also interested in studying the mixture of lossy and lossless compression based on semantic information that is commonly included in checkpoint files written in standard data format, e.g. HDF5. The choice of lossless or lossy compression for each field will depend on the sensitivity of the particular data field to precision loss.

References

[1] D. Laney, S. Langer, C. Weber, P. Lindstrom, and A. Wegener, "Assessing the Effects of Data Compression in Simulations Using Physically Motivated Metrics," *Scientific Programming*, vol. 22, no. 2, pp. 141–155, 2014.

[2] P. Jetley, F. Gioachin, C. Mendes, L. V. Kale, and T. R. Quinn, "Massively Parallel Cosmological Simulations with ChaNGa," in *Proceedings of IEEE International Parallel and Distributed Processing Symposium 2008*, 2008.

[3] 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.

[4] gzip. [Online]. Available: http://www.gzip.org

[5] K. Heitmann, P. M. Ricker, M. S. Warren, and S. Habib, "Robustness of Cosmological Simulations. I. Large-scale Structure," *The Astrophysical Journal Supplement Series*, vol. 160, no. 1, p. 28, 2005.