2015 IEEE 29th International Parallel and Distributed Processing Symposium

Exploration of Lossy Compression for Application-level Checkpoint/Restart

Naoto Sasaki
Dept. of Mathematical and Computing Science
Tokyo Institute of Technology
Tokyo, Japan
Email: sasaki.n.ac@m.titech.ac.jp

Kento Sato
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, USA
Email: kento@llnl.gov

Toshio Endo
Global Scientific Information and Computing Center
Tokyo Institute of Technology
Tokyo, Japan
Email: endo@is.titech.ac.jp

Satoshi Matsuoka
Global Scientific Information and Computing Center
Tokyo Institute of Technology
Tokyo, Japan
Email: matsu@is.titech.ac.jp

***Abstract*—The scale of high performance computing (HPC) systems is exponentially growing, potentially causing prohibitive shrinkage of mean time between failures (MTBF) while the overall increase in the I/O performance of parallel filesystems will be far behind the increase in scale. As such, there have been various attempts to decrease the checkpoint overhead, one of which is to employ compression techniques to the checkpoint files. While most of the existing techniques focus on lossless compression, their compression rates and thus effectiveness remain rather limited. Instead, we propose a lossy compression technique based on wavelet transformation for checkpoints, and explore its impact to application results. Experimental application of our lossy compression technique to a production climate application, NICAM, shows that the overall checkpoint time including compression is reduced by 81%, while relative error remains fairly constant at approximately 1.2% on overall average of all variables of compressed physical quantities compared to original checkpoint without compression.**

***Keywords*-fault tolerance; checkpoint; lossy compression;**

I. Introduction

The scale of HPC systems and the performance are exponentially growing. With the growing performance, application users enjoy the reduced calculation time, and make the problem sizes of the applications larger. While the extreme scale system gives application developers the advantages, this tendency causes several drawbacks. One of the drawbacks is system failures. Because the failure rate increases as systems size grows [1]–[3], MTBF decreases at extreme scale. For example, MTBF of exa-scale supercomputers is projected to decrease to about a few hours [4]. Therefore, applications' users are expected to pay more cost for fault tolerance

To continue its computation even on a failure, HPC applications periodically write checkpoints, which are snapshots of application states, so that applications can roll back to the last checkpoint and restart the computation [5], [6]. However, if we write checkpoints of applications running on supercomputers, the checkpoint time becomes huge overhead [2]. For example, the TSUBAME2.0/2.5 supercomputer has 116 TB memory [7]. If we write an entire memory footprint of the TSUBAME2.0/2.5 supercomputer, it approximately takes 4 hours given 8 GB/s of checkpointing throughput to the parallel file system of TSUBAME2.0/2.5 [8]. If we consider the projected MTBF, a few hours, the straight forward checkpointing cannot solve the problems.

One of approaches to shorten checkpointing time is to reduce the size. Existing works proposed incremental checkpointing [9]–[11], and compression for checkpoints [12], [13]. However, the effectiveness of these approaches are limited in real applications, such as computational fluid dynamics (CFD) applications, since the majority of the memory footprint is frequently updated, and the checkpoint data is floating point values. Thus, the existing approaches does not well-suited for reducing the checkpoint size of real applications

To reduce the size of checkpoint data of real applications, we propose a lossy compression technique, and explore the feasibility. Although lossy compression introduces errors, Our lossy compression techniques based on wavelet transformation can remarkably reduce the checkpoint data of real applications while minimizing the errors. We apply our lossy compression technique to a real climate application, NICAM [14]. The experimental results show that our lossy compression can reduce overall checkpoint time including compression time by 70% with 1.2% of an average relative error, which is much less than simulation errors regularly produced by scientific models and sensors observing input data.

1530-2075/15 $31.00 © 2015 IEEE
DOI 10.1109/IPDPS.2015.67

II. Background

A. Compression of Checkpoint Images

One of approaches to reduce checkpointing overhead is compression. By reducing the checkpoint image size, we can reduce checkpoint overhead. However, compression itself introduces overhead. Therefore, the advantage of compression, i.e., size reduction, must be bigger than the drawback, i.e., checkpoint overhead. Specifically, if we denote compression time, checkpoint time with compression and without compression as C, T_{comp} and T_{orig}, checkpoint compression becomes viable where:

$$C + T_{comp} < T_{orig} \quad (1)$$

In addition, compression time increases as the total checkpoint size increases at extreme scale. The compression algorithm must be not only fast but also scalable to checkpoint size. Most of existing approaches focus on *lossless* compression [15]–[18]. However, if we apply lossless compression to floating point arrays, the compression rate is limited. The lossless compression may not be a viable solution to checkpoint size reduction at extreme scale.

B. Lossy Compression

Another viable approach is *lossy* compression because the technique can remarkably reduce checkpoint size much more than lossless compression. Although lossy compression introduces errors, the errors may be acceptable if we examine processes for developing real scientific applications.

When application developers develop a scientific simulation code, they model the target scientific phenomena, write the code, and run the simulation based on input data observed by sensors. Because scientific models cannot grasp an exact scientific explanation of the target phenomenon, the models can introduce errors. The degree of the errors of several models are known as a few % [19]–[22]. In addition, observed input data itself can also contain errors due to inaccuracy of scientific sensors. Therefore, the simulation introduces errors in practice, and thereby the errors can also be accumulated to the final results as the simulation progresses.

To avoid the errors from being accumulated, data assimilation is used in wide range of computational science areas [20]. Data assimilation is a technique which periodically corrects intermediate results by assimilating the observed real data into the results, which lets us know *errors are inherent to scientific simulations*. Motivated by the facts, we explore whether lossy compression can remarkably reduce checkpoint size while keeping the same ration of errors as what scientific models and sensors produce.

C. Motivation of Wavelet Transformation

Wavelet transformation, one of frequency analysis techniques, is known as an effective method for data compression especially in multi-resolution analysis. In practice, a recent compression algorithm employed in JPEG 2000 uses wavelet transformation in order to achieve higher quality compression than a compression algorithm used in previous version of JPEG, which uses discrete cosine transform [23].

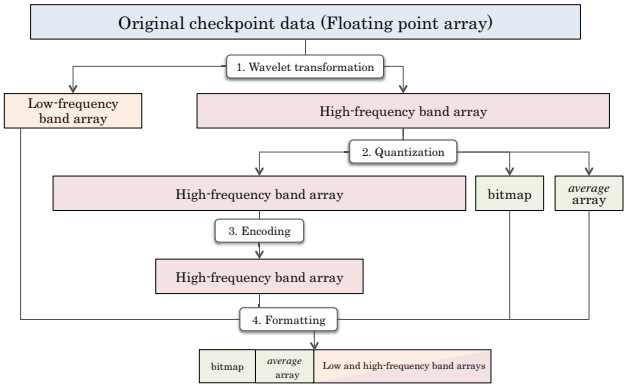

Figure 1: An overview of the proposed compression method

It is known that the compression approach based on wavelet transformation is effective if the target data is *smooth*; the differences between neighborhood values are small. Each value of intermediate data of many scientific applications, such as CFD applications, are smooth because physical quantities, such as pressure, temperatures and velocities, does not spatially changed much. Therefore, we target lossy compression using wavelet transformation.

III. Floating point Lossy Compression

To reduce checkpoint time, we propose a floating point lossy compression (Figure. 1). Our lossy compression targets floating-point arrays of multi-dimensional mesh data, which represents physical quantities such as pressures, temperatures and velocities in scientific applications, such as CFD applications. Because the mesh data accounts for the great majority of checkpoint data in mesh-based scientific applications, we focus on compression of the mesh data represented by floating point arrays.

Our lossy compression algorithm is completed by three steps; *wavelet transformation* (Section III-A), *quantization* (Section III-B) and *encoding* (Section III-C) to floating point arrays, which needs to be checkpointed. Then, we compress the outputted arrays by *gzip* after our lossy compression. Although lossy compression basically introduces errors, our lossy compression can reduce checkpoint size much more than lossless compression.

As we described in Formula 1, total checkpoint time needs to be less than checkpoint time without compression. Therefore, the compression time needs to be scalable to checkpoint size. While time complexity of several existing lossy compression algorithms is $\mathcal{O}(n \log n)$ to checkpoint size, n, our lossy compression is completed with $\mathcal{O}(n)$

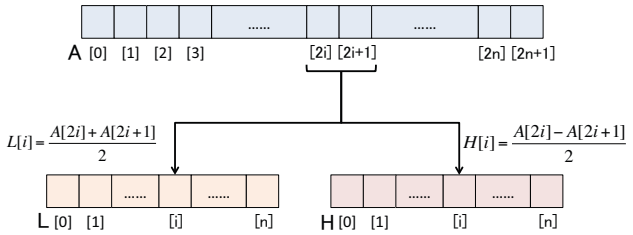

Figure 2: 1D wavelet transformation

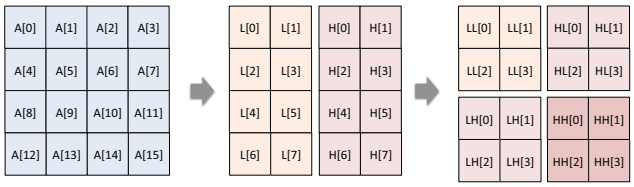

Figure 3: 2D wavelet transformation

A. Haar Wavelet-based Transformation

First, we apply a wavelet transformation based on Haar wavelet transformation to a target array [24]. This transformation is to obtain a well-suited data structure for compressing scientific mesh data while minimizing the errors in *quantization* and *encoding*. Figure. 2 shows the example of the wavelet transformation of an 1D arrays, $A[j]$ $(j = 0 \ldots 2n+1)$. The wavelet transformation divides the array into two sub-band arrays, *low-frequency band array* (L), and *high-frequency band array* (H) where:

$$L[i] = \frac{A[2i] + A[2i+1]}{2} \qquad (i = 0 \ldots n) \qquad (2)$$

$$H[i] = \frac{A[2i] - A[2i+1]}{2} \qquad (i = 0 \ldots n) \qquad (3)$$

Note that two neighboring values can be calculated as $A[2i] = L[i] + H[i]$ and $A[2i+1] = L[i] - H[i]$, thereby this transformation is lossless. In addition, difference between two neighboring values, $A[2i]$ and $A[2i+1]$, is usually small in scientific mesh data. Therefore, the values of H are expected to be small, and are concentrated in a narrow range. This characteristic is an important to minimize errors while reducing the data size in *quantization* and *encoding*.

For a 2D array, we apply the 1D wavelet transformation to each row (x-axis), then each column (y-axis) of 2D array. Thus, after the two transformations, we obtain one low-frequency band region, LL, and three high-frequency band region, LH, HL and HH as shown in Figure. 3. Because difference of neighboring two values in L, H is also small, the values in the three high-frequency band regions also become small, and concentrate in a narrow range. Likewise, for a3D array, we apply the 1D wavelet transformation to z-axis direction addition to x, y-axis. In case of a 3D array, we obtain one low-frequency band region, and seven high-frequency band regions.

B. Quantization

Next, we apply *quantization* to the values in the high-frequency band regions. In the quantization, we transform each value in the high-frequency band regions to limited number of values in order to reduce the variety of values appeared in the high-frequency band regions.; Although this transformation is lossy, and introduce errors, this transformation makes redundant values, which is a important characteristic to achieve high compression rate.

1) Simple Quantization: First, we explain a simple quantization method. In the simple quantization, we divide the values in the high-frequency band regions into n partitions (Step (1) in Figure. 4), then calculate an average of values in each partition, and replace all values with the averages of each partition to which the values belong (Step (2) in Figure. 4). If a value v belongs to an i-th partition, and the average in the partition is $average[i]$, then we change v to $average[i]$. Thus, after the simple quantization, only n kinds of values appear in the high-frequency band regions. Figure. 4 shows an example if n is 4. The value, n, gives us a trade-off between degree of errors and compression rate. However, we can still satisfying compression rate even with large n as described in Section IV-C.

As mentioned in Section III-A, the values in the high-frequency band regions are close to zero, and most of the values usually belong to a few partitions, which makes a *spike* in the distribution as shown in Figure. 4. From our preliminary experiments, we found that our lossy compression introduces undesirable errors with lower compression rate if we apply the quantization to partitions to which a small number of values belong.

2) Proposed Quantization to Reduce an Error: To solve the problem in the simple quantization method, we extend the method to be able to apply the quantization to only partitions in which the *spike* exists. For detection of the spike, we can easily find the spiked partitions by choosing partitions which contain more number of values than the other partitions. Specifically, we divide the values in the high-frequency band regions into d partitions (Step (3) in Figure. 4). We denote $N_{div}[i]$ as the number of the values contained in i-th partition, then we choose partitions such that:

$$N_{div}[i] \geq \frac{N_{total}}{d} \qquad (4)$$

N_{total} is the total number of the values in the high-frequency band regions. Hence, $\frac{N_{total}}{d}$ is equal to the average number of values per a partition (Step (4) in Figure. 4). Then, we apply the *simple quantization* to only the detected partitions (Step (5) in Figure. 4). By using the *spike* detection, we can remarkably reduce errors with comparable compression rate

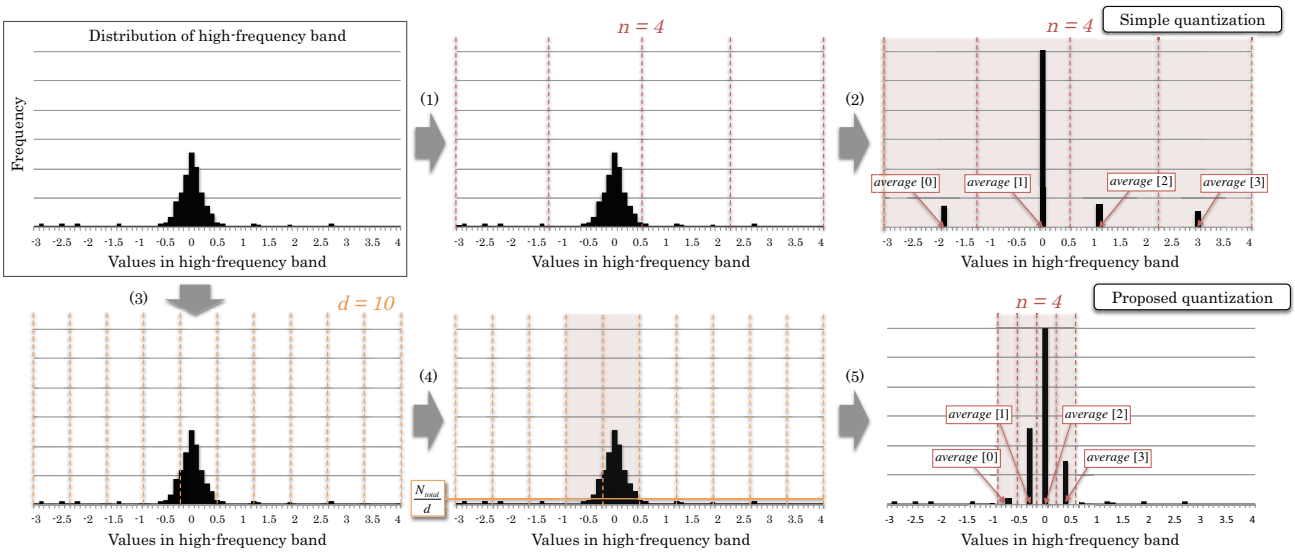

Figure 4: Simple quantization

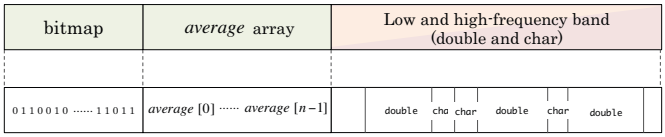

Figure 5: output format

to the simple quantization. Figure. 4 shows an example if d is 10, n is 4.

C. Encoding

After the quantization, the high-frequency band regions to which the quantization is applied consist of only n kinds of floating point values, and these values are stored in the array, $average[i]$ $(i = 0 \ldots n-1)$, which is created in the quantization. To reduce the sizes of the values in the high-frequency band regions, we replace the floating-point values with indexes of the $average$ array, and store them to $char$ variables. The value, n, does not become large number to obtain satisfying errors, 1 byte of the data type is enough to store each index. Because we store the array of the indexes with the $average$ array, this operation is lossless.

D. Output Format

Figure. 5 shows output format when we write the compressed checkpoint to a file system. As described in Section III-B, our proposed lossy compression apply only the part of values in high-frequency band regions. To memorize which values are transformed and encoded, we use *bitmap* for the decompression. To decode the encoded values, we also store the $average$ array, and append to *bitmap*. Finally, we apply gzip to the formatted output.

IV. Evaluation

A. Experimental Conditions and Environment

We apply our approach with lossy compression to target checkpoint data of a real climate application, NICAM [14]. The lossy compression has been widely used in order to output data for visualization of simulations, however, it has been hardly used for checkpointing, in fear of that production of errors in floating point arrays may invalidate the results of simulations. Thus in addition to evaluation of compression time, I/O time and compression rate, we evaluate the effects of lossy compression on simulated results; we include evaluation of relative errors introduced by the lossy compression, by comparing the result data and original data.

Applications usually write checkpoints of intermediate data after certain period of time. To make the intermediate data for checkpointing, we run NICAM for about an hour, which is identical to 720 time steps. In NICAM, a single time step simulate climate for 10 days. In our experiments, the targets of compression data are 3D arrays of pressure, temperature and wind velocity in NICAM. These 3D arrays are double-precision floating point arrays, each of which has size of $1156 \times 82 \times 2$. Among of those, this section mainly describes results for the temperature array because we see the similar results in the other arrays.

We use an in-house cluster as the experimental platform. Each node has a specification shown in Table I, and the nodes share an NFS file system, which is used to store checkpoint images. We evaluate compression rate (cr) as:

$$cr = \frac{cs_{comp}}{cs_{orig}} \times 100 \quad (5)$$

where cs_{orig} is checkpoint size without compression, and

Table I: System specification

Node	
CPU	Intel Core i7-3930K 6 cores 3.20GHz
Memory	DDR3 16GB
Network card	Broadcom bnx2
Shared file system	
File system	Network File System (NFS) v3 1.5TB
RAID	Dell PERC H700 (RAID6)
Disk	Western Digital WD (model:WD2002FAEX)

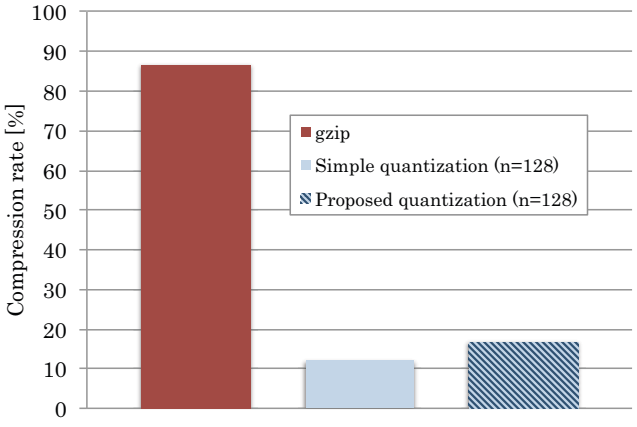

Figure 6: Comparing compression rates of gzip, our lossy compressions with simple quantization and proposed quantization

cs_{comp} is checkpoint size with compression. For evaluation of errors, we use relative error (re_i) expressed as:

$$re_i = \frac{|x_i - \tilde{x}_i|}{\max_j\{x_j\} - \min_j\{x_j\}} \quad (6)$$

where x_i is an original value of an array, $X = \{x_i | i = 0, \ldots, m\}$, , and $\tilde{x}_i$ is an value of an array, which is compressed, then decompressed by the lossy compression to X.

In the experiments, we use several numbers of divisions, *division number*, in the quantization phase (n in Figure. 4), from 2^0 to 2^7. The parameter d is set to be 64 for our proposed quantization in Figure. 4.

B. Lossless V.S. Lossy Compression

Figure. 6 shows compression rates of gzip, our lossy compressions with simple quantization and proposed quantization. For the two lossy compression methods, we set the division number as 128 in this evaluation.

The results show that gzip is apparently insufficient for compressing floating point values; the compression rate is 86.78%. On the other hand, with our lossy compressions, we can remarkably reduce checkpoint size in real applications. From the experimental results, we can see that the effectiveness of lossless compression is limited in real applications, such as CFD applications, since checkpoint data is floating point values. Thus, lossy compression is essential for improving compression rate in real applications.

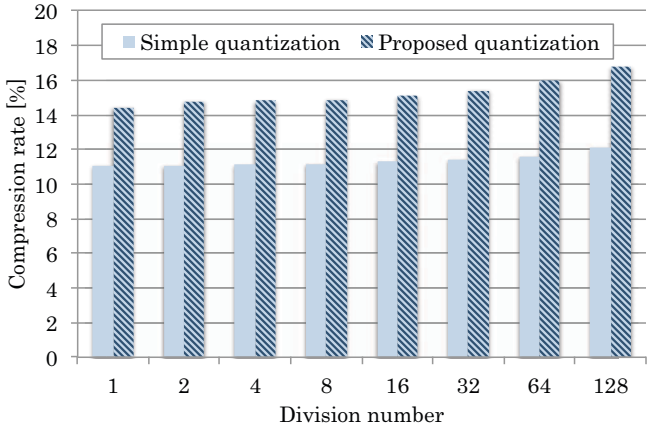

Figure 7: Compression rates under different division number, n, and quantization methods

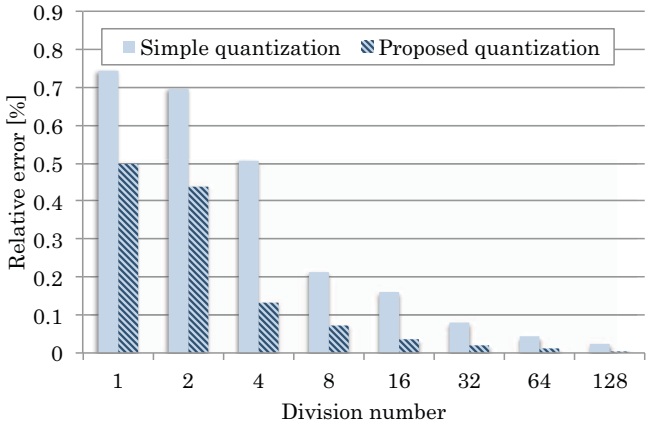

Figure 8: Relative errors under different division number, n, and quantization methods

C. Lossy Compression with Simple and Proposed Quantization

Figure. 7 and Figure. 8 show compression rates and average relative errors respectively, under different division numbers, n, and quantization methods. We compute the average relative errors as $\sum_{i=1}^{m} re_i/m$. These two graphs illustrate a trade-off between compression rates and errors; when we use larger n, relative errors become smaller (better), while compression rates get larger (worse), though the increase is rather gradual. As shown in the figures, our proposed method can reduce more errors while achieving the comparable compression rate to the simple method.

Figure. 7 shows compression rates of a temperature array

in different division numbers and quantization methods. The compression rates tend to increase as the division number (n) increases; it is 11.06% with the simple quantization when $n = 1$, and reaches 12.10% when $n = 128$. With the proposed quantization, the compression rates are larger, which is 14.43% when $n = 1$ and 16.75% when $n = 128$. For other arrays than the temperature array, the measured compression rates are 11% to 13% with simple quantization, and 13% to 29% with proposal quantization.

Figure. 8 shows relative errors of the temperature array in different division numbers and quantization methods. It is natural that the relative errors are reduced with larger n; the average relative error is 0.74% at $n = 1$ with simple quantization, and 0.025% at $n = 128$. With proposal quantization, it is 0.49% at $n = 1$ and 0.0056% at $n = 128$. Also we investigated all the floating point arrays in the application. The average relative errors with simple quantization are in the range of 0.0053% to 14.56%, and the maximum relative errors are 0.048% to 56.84%, which would be intolerable. With the proposed quantization, they are improved. They are 0.0004% to 1.19% in average, and 0.0022% to 5.94% at maximum. We compute the maximum relative errors as $\max_{i=0...m}\{re_i\}$.

As a whole, while the proposed method keeps the compressed size low, the method can significantly reduce the errors as the division number increase. As described in Section II-B, errors in floating point data that are tolerable depending on characteristics of applications and application users' preference. Thus users will need to control the parameter n in order to fulfill their preferences. In future, we will provide more intuitive capability, which can control the errors by specifying a value, such as tolerable degree of errors.

D. Compression Time

As described in Section II-A, one of our goals is to reduce total checkpointing time including compression at large scale. In order to estimate the total checkpointing time of large scale systems based on the results from our in-house cluster, we make the following assumptions. We consider weak scalable cases, where each application process has checkpoint data whose size is constant, 1.5MB. The size is based on checkpoint size of a single array in NICAM. Here the compression time does not depend on parallelism, since compression of checkpoints of each process can be done in a embarrassingly parallel fashion. We obtain the compression time from the actual measurement. Also we obtain the compression rate, 19%, in this case. For the I/O time, we assume that checkpoint images of all processes are stored into the shared parallel file system, whose I/O throughput is 20GB/s. Thus, we can estimate the I/O time as:

$$1.5_{[MB/processes]} \times 0.12 \times P \ / \ 20_{[GB/s]}$$

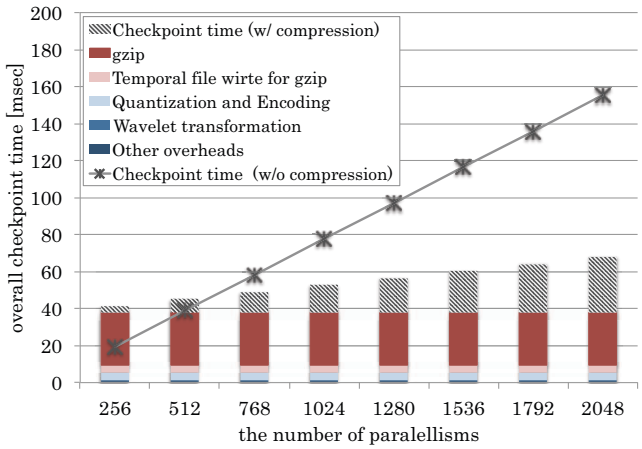

Figure 9: Overall checkpint time in increasing parallelisms

where P denotes the number of parallelisms. Figure. 9 shows the estimated checkpoint costs in increasing parallelisms. For the compression time, we show the detailed breakdown based on the actual measurement. The figure also shows the estimated checkpoint costs without compression.

From the figure, we observe that our approach is superior in the aspect of compression time with larger number of processes, because compression costs get relatively smaller compared to I/O time. The crosspoint is around 768 processes in this case. With 2048 processes, our estimation indicates that we can reduce checkpoint costs by 55%. Because compression time is constant to increasing parallelism, the slop of the total checkpoint time with our proposed method is more flat than one without compression. In this trend, if we scale out the system, the checkpoint costs can be reduced by about 81 (= $\frac{1-0.19}{1} \times 100$)%.

This evaluation is limited to smaller checkpoint sizes (1.5MB/process) due to the available initial input data for NICAM. However, our compression algorithm has computational time complexity, $O(n)$, to checkpoint sizes. Thus, the superiority demonstrated in Figure. 9 is kept with larger checkpoint sizes.

As shown in Figure. 9, most of the compression time is consumed by gzip. The current implementation writes temporary checkpoint data as files, and apply gizp to these files via the file system. This cost will be mostly eliminated by compressing the temporary checkpoint data with *zlib* in memory. Also, we are going to investigate other compression methods that are more appropriate than gzip when combined with our lossy compression. Although our current implementation includes extra overhead, the estimation verifies that our lossy compression method remarkably reduces checkpoint time at extreme scale.

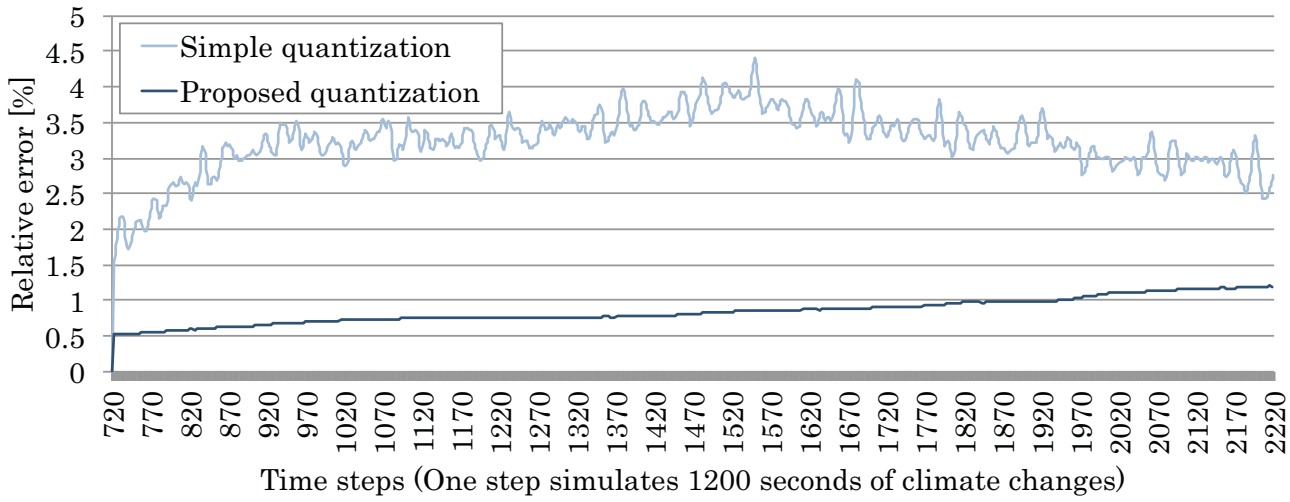

Figure 10: The transition of the relative error with application time steps

E. Feasibility Study of Lossy Compression

In the previous sections, we have evaluated *immediate* errors; we compare original values with decompressed values by the lossy compression. In real simulation runs, application users run the simulations for long time after restarts from failures, and the errors of the successive results may diverge, or may converge. To investigate the successive impacts, we evaluate the errors on each time step by comparing with original values on each step.

First, we run NICAM for 720 time steps, and then make checkpoint images with the lossy compression. After the checkpointing, we decompress the checkpoint, and we rerun for additional 1500 time steps, i.e., 2220 time steps in total, in order to emulate a restart from a failure.

Figure. 10 shows the progress of the relative errors as the time step proceeds. Because the application restarts at a step 720, the x-axis begins from 720, and ends with 2220 steps. It shows the average relative errors of the temperature array. We observe that the proposed quantization exhibits smaller errors than the simple one. In the simple quantization, we also see that the fluctuation of errors are larger, and from around step 1570, the errors start to decrease. Meanwhile, the errors by proposed quantization are milder, and increase slowly throughout steps from 720 to 2220. We also observed the similar tendency with the other arrays.

For both quantization methods, the errors randomly grow up and down while slowly increasing, and the movements resemble to *1D random walk*. If we assume that the errors grow according to an 1D random walk, the expected errors after n steps becomes the order of $\sqrt{n}$, which explains the slow grows of the errors. In practice, scientific models also produce the same degree of errors. Thus, the slow grows of the errors may be acceptable compared to inherent errors to scientific simulations, such as input data errors, sensor errors and model errors. However, we should investigate on many real applications, and the invariants of the physical quantities as future work. In addition, values of the target array can be symmetric, or being obeying the principle of the conservation of energy. If we apply lossy compression to those arrays, the lossy compression can break the consistency. Thus, lossy compression may require users to do data adjustment for the consistency after restart in such applications.

V. Related Work

To restart from failures, applications usually write checkpoints to reliable parallel file systems. However, writing checkpoints to such shared file system incurs huge overhead because parallel file systems are shared by all of compute nodes, and cannot provide enough I/O bandwidth to all the compute nodes. To solve the problem, multi-level checkpointing has been proposed [5], [25]. With multi-level checkpointing, applications can make use of storage hierarchy where the applications write checkpoint to local storage frequently, and to parallel file system less frequently. By optimizing each level of checkpointing intervals using checkpointing models, the application can significantly reduce the checkpointing overhead [25], [26]. However, failure rate is projected to become higher at extreme scale. The existing multi-level checkpointing may not be enough for extreme scale systems.

For further improvement of checkpointing at extreme scale, in-memory checkpointing is one of the approaches [27]–[29]. By directly writing checkpoints to memory in stead of storage sub-systems with an RAID-5 technique, checkpointing time can be improved by one order of magnitude while keeping certain level of reliability. Asynchronous checkpointing also reduce the I/O overhead by overlapping with computations [2]. In addition, utilization

of new storage hierarchy, *burst buffer*, is validated to significantly improve both checkpoint time and storage reliability by storage reliability modeling [30].

Another approach is reducing checkpoint sizes. Our proposed approach is classified in this category. Incremental checkpointing stores only differences with the last checkpoint instead of storing the entire image every time [9]–[11]. If the differences are small, the checkpointing costs are significantly reduced. However, with this approach, the restart costs tend to increase, since the recovery requires several consecutive checkpoint images [9]. In addition, the effects of this approach may be limited in scientific applications because the entire arrays of physical quantities are frequently updated, which results in storing entire arrays. Islam et al. have proposed lossless compression to reduce checkpointing costs [13]. The scheme merges distributed checkpoint images per each variable, and select effective compression methods for each variable. However, compression rate of floating-point arrays can be limited compared to lossly compression in scientific applications . Xiang et al. have studied feasibility of lossy compression for checkpoint data [31]. They applied existing lossy compression [32] to checkpoint data of an N-body cosmology simulation while injecting a varying number of failures. Lossly compression has been becoming feasible for checkpointing in other types of applications. As future work, we'll apply our lossy compression to such applications.

VI. Conclusion

In order to reduce checkpointing time, we have proposed a lossy compression technique based on wavelet transformation. Then, we have applied our approach to a real climate application, NICAM, to evaluate compression times, compression rates, relative errors. The experimental results show that our lossy compression can reduce overall checkpoint time including computation time for compression by 81%, while the introduced relative errors are around 1.2% on overall average of all variables of compressed physical quantities compared to the original values.

Our future work includes improvement of the compression algorithm to reduce compression rates and errors. It is also important to investigate the feasibility in other applications. Finally, we will combine with other efforts to reduce checkpointing costs, such as harnessing storage hierarchy, optimizing checkpoint frequency by checkpointing model for lossy compression.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. (LLNL-CONF-663096). This work is supported by Grant-in-Aid for Scientific Research S 23220003 and JST-CREST, "Software Technology that Deals with Deeper Memory Hierarchy in Postpetascale Era".

References

[1] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramaniam, and R. Sahoo, "Bluegene/l failure analysis and prediction models," in *International Conference on Dependable Systems and Networks(DSN 2006)*, June 2006, pp. 425–434.

[2] K. Sato, N. Maruyama, K. Mohror, A. Moody, T. Gamblin, B. R. de Supinski, and S. Matsuoka, "Design and modeling of a non-blocking checkpointing system," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Salt Lake City, UT, USA: IEEE Computer Society Press, 2012, pp. 19:1–19:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389022

[3] A. Moody, G. Bronevetsky, K. Mohror, and B. De Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2010, pp. 1–11.

[4] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir, "Toward exascale resilience," *Int. J. High Perform. Comput. Appl.*, vol. 23, no. 4, pp. 374–388, Nov. 2009. [Online]. Available: http://dx.doi.org/10.1177/1094342009347767

[5] L. Bautista-Gomez, S. Tsuboi, D. Komatitsch, F. Cappello, N. Maruyama, and S. Matsuoka, "Fti: High performance fault tolerance interface for hybrid systems," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. Seattle, Washington: ACM, 2011, pp. 32:1–32:32. [Online]. Available: http://doi.acm.org/10.1145/2063384.2063427

[6] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive fault tolerance for hpc with xen virtualization," in *Proceedings of the 21st Annual International Conference on Supercomputing*, ser. ICS '07. New York, NY, USA: ACM, 2007, pp. 23–32. [Online]. Available: http://doi.acm.org/10.1145/1274971.1274978

[7] S. Matsuoka, T. Aoki, T. Endo, H. Sato, S. Takizawa, A. Nomura, and K. Sato, *TSUBAME2.0: The First Petascale Supercomputer in Japan and the Greatest Production in the World*. Chapman & Hall/CRC Computational Science, Apr. 2013, vol. 1, ch. 20, pp. 525–556. [Online]. Available: http://www.crcnetbase.com/doi/book/10.1201/b14677

[8] K. Sato, N. Maruyama, K. Mohror, A. Moody, T. Gamblin, B. R. de Supinski, and S. Matsuoka, "Design and modeling of a non-blocking checkpointing system," pp. 19:1–19:10, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389022

[9] N. Naksinehaboon, Y. Liu, C. Leangsuksun, R. Nassar, M. Paun, and S. Scott, "Reliability-aware approach: An incremental checkpoint/restart model in hpc environments," in *8th IEEE International Symposium on Cluster Computing and the Grid, 2008. CCGRID '08*, May 2008, pp. 783–788.

[10] J. S. Plank, J. Xu, and R. H. B. Netzer, "Compressed differences: An algorithm for fast incremental checkpointing," University of Tennessee, Tech. Rep. CS-95-302, August 1995.

[11] J. Sancho, F. Petrini, G. Johnson, and E. Frachtenberg, "On the feasibility of incremental checkpointing for scientific computing," in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings*, April 2004, pp. 58–.

[12] D. Ibtesham, D. Arnold, K. Ferreira, and P. Bridges, "On the viability of checkpoint compression for extreme scale fault tolerance," in *Euro-Par 2011: Parallel Processing Workshops*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7156, pp. 302–311. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29740-3_34

[13] T. Islam, K. Mohror, S. Bagchi, A. Moody, B. De Supinski, and R. Eigenmann, "MCRengine: A scalable checkpointing system using data-aware aggregation and compression," in *2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2012, pp. 1–11.

[14] M. Satoh, T. Matsuno, H. Tomita, H. Miura, T. Nasuno, and S. Iga, "Nonhydrostatic icosahedral atmospheric model (NICAM) for global cloud resolving simulations," *Journal of Computational Physics*, vol. 227, no. 7, pp. 3486 – 3514, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999107000654

[15] T. Bicer, J. Yin, D. Chiu, G. Agrawal, and K. Schuchardt, "Integrating online compression to accelerate large-scale data analytics applications," in *Proceedings of IEEE Parallel and Distributed Processing Symposium (IPDPS), 2013*, May 2013, pp. 1205–1216.

[16] L. A. B. Gomez and F. Cappello, "Improving floating point compression through binary masks," in *IEEE BigData 2013*, Santa Barbara, California, 2013.

[17] 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.

[18] A. Padyana, D. Sudheer, P. K. Baruah, and A. Srinivasan, "Reducing the disk io bandwidth bottleneck through fast floating point compression using accelerators," *International Journal of Advanced Computer Research, vol. 4, no. 1, pp. 134-144, 2014*, 2014.

[19] G. Han, X. Wu, S. Zhang, Z. Liu, and W. Li, "Error covariance estimation for coupled data assimilation using a lorenz atmosphere and a simple pycnocline ocean model," *Journal of Climate*, vol. 26, no. 24, pp. 10 218–10 231, 2014/10/19 2013.

[20] D. Zupanski and M. Zupanski, "Model error estimation employing an ensemble data assimilation approach," *Monthly Weather Review*, vol. 134, no. 5, pp. 1337–1354, 2014/10/19 2006.

[21] J. L. Anderson, "An ensemble adjustment kalman filter for data assimilation," *Monthly Weather Review*, vol. 129, no. 12, pp. 2884–2903, 2014/10/18 2001.

[22] J. S. Whitaker, T. M. Hamill, X. Wei, Y. Song, and Z. Toth, "Ensemble data assimilation with the ncep global forecast system," *Monthly Weather Review*, vol. 136, no. 2, pp. 463–482, 2014/10/18 2008.

[23] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, Jan 1974.

[24] A. Graps, "An introduction to wavelets," *Computational Science Engineering, IEEE*, vol. 2, no. 2, pp. 50–61, Summer 1995.

[25] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski, "Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '10. New Orleans, LA, USA: IEEE Computer Society, Nov. 2010, pp. 1–11. [Online]. Available: http://dx.doi.org/10.1109/sc.2010.18

[26] N. Vaidya, "On checkpoint latency," in *In Proceedings of the Pacific Rim International Symposium on Fault-Tolerant Systems*, 1995, pp. 60–65.

[27] G. Zheng, L. Shi, and L. V. Kale, "FTC-Charm++: An In-Memory Checkpoint-Based Fault Tolerant Runtime for Charm++ and MPI," in *Proceedings of the 2004 IEEE International Conference on Cluster Computing*, ser. CLUSTER '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 93–103. [Online]. Available: http://portal.acm.org/citation.cfm?id=1111712

[28] R. Rajachandrasekar, A. Moody, K. Mohror, and D. K. D. Panda, "A 1 pb/s file system to checkpoint three million mpi tasks," in *Proceedings of the 22Nd International Symposium on High-performance Parallel and Distributed Computing*, ser. HPDC '13. New York, NY, USA: ACM, 2013, pp. 143–154. [Online]. Available: http://doi.acm.org/10.1145/2462902.2462908

[29] K. Sato, A. Moody, K. Mohror, T. Gamblin, B. R. d. Supinski, N. Maruyama, and S. Matsuoka, "Fmi: Fault tolerant messaging interface for fast and transparent recovery," in *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, ser. IPDPS '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1225–1234. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2014.126

[30] K. Sato, K. Mohror, A. Moody, T. Gamblin, B. De Supinski, N. Maruyama, and S. Matsuoka, "A user-level infiniband-based file system and checkpoint strategy for burst buffers," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, May 2014, pp. 21–30.

[31] X. Ni, T. Islam, K. Mohror, A. Moody, and L. V. Kale, "Lossy compression for checkpointing: Fallible or feasible?" in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.

[32] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1245–1250, Sept