DuoModel: Leveraging Reduced Model for Data Reduction and Re-Computation on HPC Storage

Huizhang Luo, Qing Liu, Zhenbo Qiao, Jinzhen Wang, Mengxiao Wang, and Hong Jiang

Abstract—High-performance computing (HPC) applications generate large amounts of floating-point data that need to be stored and analyzed efficiently to extract the insights and advance knowledge discovery. With the growing disparities between compute and I/O, optimizing the storage stack alone may not suffice to cure the I/O problem. There has been a strong push in the HPC communities to perform data reduction before data is transmitted to storage in order to lower the I/O cost. However, as of now, neither lossless nor lossy compressors can achieve the adequate reduction ratio that is desired by applications. This paper proposes DuoModel, a new approach that leverages the similarity between the full and reduced application models, and further improve the data reduction ratio. DouModel further improves the compression ratio of state-of-the-art compressors via compressing the differences (termed as delta) between the data products of the two models. For data analytics, the high fidelity data can be re-computed by launching the reduced model and applying the compressed delta. Our evaluations confirm that DuoModel can further push the limit of data reduction while the high fidelity of data is maintained.

Index Terms—High-performance computing, data reduction, re-computation

1 Introduction

Scientific applications produce vast amounts of floating-point data that capture the microscopic physical phenomena in high fidelity. For a single production run, such as that of a fusion simulation, it can generate more than 1 TB of data in one snapshot. In a time evolution that consists of thousands of steps, the total analysis output can easily reach PBs for one run. Such enormous data volume poses unprecedented pressure on HPC storage systems, even for the largest parallel file systems, such as Spider,[1] and this drives computer system researchers, applied mathematicians, and application scientists to co-design new software/hardware solutions that can sustain data coming out of exascale applications. Very recently, data reduction is recognized as a critical step in an exascale application workflow, prior to data in motion and at rest. The goal of data reduction is to reduce the volume and velocity of data being moved, through either lossy or lossless compression. In particular, lossless compression can reconstruct the original data from the compressed data with no bit-level changes. For HPC floating-point data, it was shown that lossless compressors typically achieve a reduction of less than 4X [1], and the general-purpose deduplication [2] can achieve only 20 to 30 percent reduction, thus being fairly ineffective in reducing HPC data volume. In contrast, lossy compression, such as FPZIP [3], ISABELA [4], ZFP [5], and SZ [6], leverages the general tolerance of reduced accuracy in applications, and uses approximations and partial data discarding to compress the content, resulting in a much higher compression ratio with user-prescribed error bound. Despite the recent success in lossy compression, the reduction ratio still has a long way to go to meet the application requirements on next-generation systems.

This paper complements state-of-the-art lossy compressors and explores strategies that can further push the limit of compression ratio—we believe that data need to be pre-conditioned prior to compression such that they match the design philosophies of a compressor. This work builds upon two key observations: 1) on future HPC systems, compute resources will be increasingly cheaper, as compared to storage [7]. Therefore, intermediate data products should be computed as much as possible, instead of being stored and analyzed later; 2) modern lossy compressors, e.g., ZFP, SZ, rely on the local smoothness within a dataset to compress data. The degree of local smoothness is, however, governed by the underlying laws of physics and the mathematical properties of a target problem. As such, compression should be done in a way that is physics- and math-aware so that potentially a more accurate prediction can be done. In this work, this is achieved by co-running a shadow application, termed as *reduced model*. A reduced model mimics the original application, termed as *full model*, but with reduced degrees of freedom, e.g., reduced resolution, and quantities. There are two advantages in this design: 1) a reduced model can be executed with significantly low complexities, resulting in low CPU and memory overhead, and 2) in principle, it does not require substantial code changes for an application, meanwhile still capturing the physics being studied. We reveal the high correlation between the data generated from a full model and a reduced model (Section 2), and design DuoModel to further improves the compression ratio by compressing the difference (termed as *delta*) between the data products of the two models. The delta can be stored on persistent storage and retrieved later for data analytics. To reconstruct the original data, an analytics pipeline can first run the reduced model, and apply the delta. The proposed scheme works for most scientific applications that have reduced model and full model versions, but not works for the chaotic system, e.g., turbulent systems. The major contributions of this paper are as follows:

- We illustrate the high similarity between the full and reduced models, and develop DuoModel to exploit this characteristic for efficient compression;
- We evaluate the reduction ratio and overhead of DuoModel, and study the cost of re-computing the original data.

The remainder of this paper is organized as follows. Section 2 provides the background and motivation. Section 3 presents the designs and implementations of DuoModel, and Section 4 discusses the evaluation results, along with conclusions in Section 5.

Our experiments for evaluating the full and reduced models, including scientific applications and scripts for evaluation, are publicly available at https://github.com/luohuizhang/Recomputing.

2 Background and Motivation

2.1 Full Model versus Reduced Model

Compared to a reduced model [8], a full model has higher fidelity calculation and data products, since a full model captures the underlying physics with higher degrees of freedom, a higher spatial resolution of grid, or a higher precision (e.g., using double precision). In addition, the stability of math solvers often requires a

1. https://www.olcf.ornl.gov/olcf-resources/data-visualization-resources/spider/

- *H. Luo, Q. Liu, Z. Qiao, and J. Wang are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102. E-mail: {huizhang.luo, qing.liu, zq38, jw447}@njit.edu.*
- *M. Wang and H. Jiang are with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019. E-mail: mengxiao.wang@mavs.uta.edu, hong.jiang@uta.edu.*

Manuscript received 29 Mar. 2018; revised 1 May 2018; accepted 3 May 2018. Date of publication 20 July 2018; date of current version 10 Sept. 2018.
(Corresponding author: Qing Liu.)
Recommended for acceptance by G. Grider.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/LCOS.2018.2855118

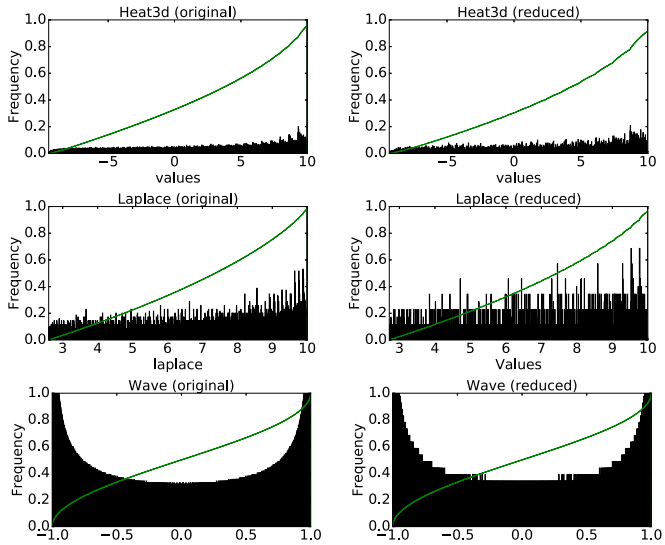

Fig. 1. Data features. The black bars and green curve show the PDF and CDF of data values, respectively.

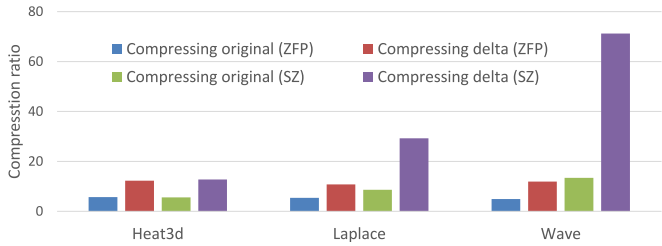

Fig. 2. Compressing delta versus original data. The number of cores used in the full and reduced models are 64 and 16, respectively.

finer time evolution, due to the Courant-Friedrichs-Lewy (CFL) condition [9]. A reduced model can be solved with lower computational cost and coarser granularity than a full model. And the problem size and the amount of output data will grow exponentially as the granularity of a reduced model becomes finer.

To demonstrate the generality of similarity between the full and reduced models, we use three classical partial differential equations (PDEs), including Heat3d equation, 2D Laplace equation, and 1D Wave equation [10], [11]. All these PDEs are solved in parallel, using Message Passing Interface (MPI) for inter-processor communication.

2.2 Motivation

The idea of this paper is motivated by three observations. First, advanced floating-point compressors, such as ZFP and SZ, assume the local smoothness to compress data. This may not be valid for all applications, and will result in sub-par performance. We believe data need to be pre-conditioned prior to compression so that they are more compressible. Second, there have been growing disparities between compute and I/O. On future HPC systems, it is commonly recognized that there will be an abundance of compute resources, as compared to storage resources. This architectural trend motivates us to trade compute for I/O, in order to mitigate the slow storage bottleneck. Third, HPC simulations typically use a high spatiotemporal resolution and high degrees of freedom to improve fidelity. However, we notice that there exists a high correlation between high and low fidelity. For example, Fig. 1 shows the probability density function (PDF) and cumulative distribution function (CDF) of data generated by the three classical PDEs with constant boundary conditions. It is evident that data from the full and reduced models are highly similar.

The high correlations inspire us to compress the delta, which are more compressible due to the improved smoothness, between the

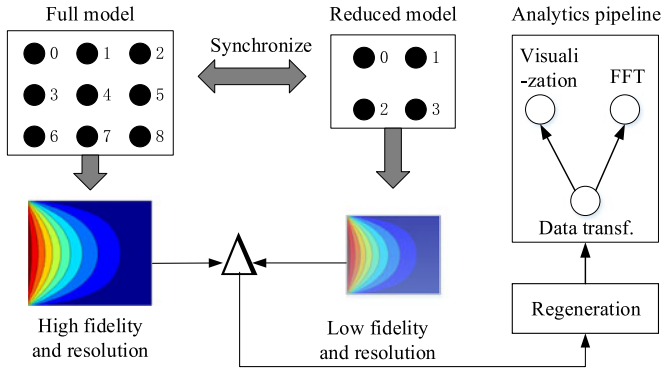

Fig. 3. The schematic of using DuoModel.

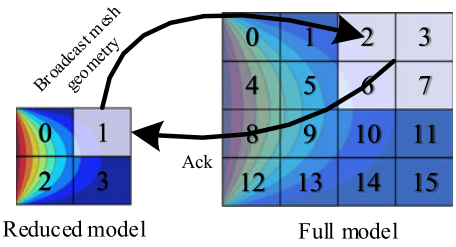

Fig. 4. Mapping of sub-domains. The sub-domains 2, 3, 6, and 7 of the full model are mapped to sub-domain 1 of the reduced model.

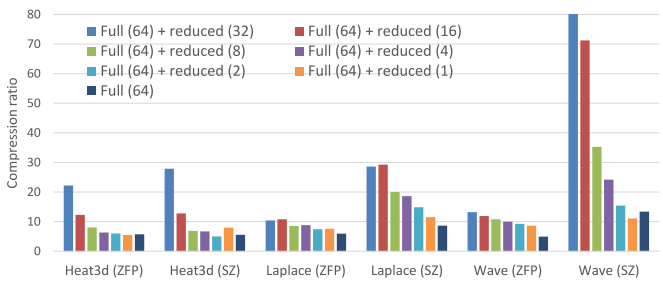

Fig. 5. Compressing delta versus original data. The numbers in the bracket indicate the number of cores allocated to either the full model or the reduced model.

data products of the full and reduced models. To this end, we run the two models synchronously, then calculate and store the delta (with details in Fig. 3). For both ZFP and SZ, the compression ratios are further improved, as compared to compressing the original data directly, shown in Fig. 2. If data need to be retrieved and analyzed post-run, we can launch the reduced model as a part of the analytic pipeline, apply the delta and re-generate the high-fidelity data.

3 Design and Implementation

3.1 Approach Overview

The goal of DuoModel is to pre-condition data and further push the limit of compression ratio. We take advantage of the intrinsic correlation between the data of full and reduced models, as well as the increasing compute capability of modern HPC systems. We output the compressed delta between the two models, and if needed, re-compute the high-fidelity data by applying the delta to the reduced model.

As shown in Fig. 3, the proposed approach consists of four steps:

Step 1. The full and reduced models are allocated compute resources, and run synchronously. The amount of compute resources allocated to the reduced model depends on the resource availability and the goal of the reduction ratio.

Step 2. An MPI processor in the reduced model sends the intermediate data product to the associated processors, i.e., those

TABLE 1
Configurations of the PDE Applications

	Heat3d			Laplace			Wave		
	Problem size	Sub_domain	Time Step	Problem size	Sub_domain	Time Step	Problem size	Sub_domain	Time Step
Full (64)	64*128*64	4*4*4	.34789E-07	192*192	8*8	6.64258E-06	40960	64	0.0000125
Reduced (32)	64*64*32	2*4*4	3.18034E-06	96*192	4*8	6.64258E-06	20480	32	0.000025
Reduced (16)	32*64*32	2*4*2	3.18034E-06	96*96	4*4	2.60308E-05	10240	16	0.00005
Reduced (8)	16*32*16	2*2*2	2.14335E-05	48*96	2*4	2.60308E-05	5120	8	0.0001
Reduced (4)	8*16*8	1*2*2	0.000125	48*48	2*2	0.0001	2560	4	0.0002
Reduced (2)	8*8*8	1*2*1	0.000125	24*48	1*2	0.0001	1280	2	0.0004
Reduced (1)	4*4*4	1*1*1	0.0005787	24*24	1*1	0.0003698	640	1	0.0008

calculate the same physical sub-domain, in the full model. The processors in the full model receive the data, and calculate the delta.

Step 3. The delta are subsequently compressed using lossy compressors, and then written to persistent storage.

Step 4. If the high fidelity data need to be retrieved by an analytics pipeline, we run the reduced model, and decompress and apply the delta to the reduced model output.

3.2 Implementation

In this section, we discuss the technical details of DuoModel. To make DuoModel a realistic solution in a production HPC environment, the following questions need to be addressed:

- How to enable the communication between the two models in an HPC environment?
- The reduced model may deviate significantly from the full model after a number of iterations. How to synchronize the two models?
- How to establish a mapping between the sub-domains of the two models, and how to calculate the delta?

3.2.1 DuoModel Launching and Communication

We launch the two models and run them concurrently via the option ':' of the $aprun$ command, a job execution command used on Titan,[2] to load the executables of the full and reduced models into compute nodes. This allows the two models to be launched concurrently, with the caveat that they share the same MPI communicator, MPI_COMM_WORLD. We further use MPI_Comm_split() to split the communicator into two new sub-communicators, based upon the processor coordinates, and feed them to the full and reduce models, respectively, to keep the intra-model communications intact. The synchronization between the full and reduced models is achieved via the MPI_COMM_WORLD communicator.

3.2.2 Synchronizing the Full and Reduced Models

To achieve a meaningful correlation of data between the full and reduced models, their executions should be synchronized with regard to the simulated physical time. For both Heat3d and Wave, the synchronization can be achieved by keeping the physical elapse time identical between the two models. Note that to maintain the stability of calculation as dictated by CFL, a time step of the reduced model is longer than that in the full model. Therefore, we keep the reduced model idle until the full model reaches the same physical time. For the Laplace equation, since it aims to calculate the final stable state of a system, we calculate the delta at the stable state in the end.

To synchronize the two models, we need to identify the mapping between sub-domains of the two models, as illustrated in Fig. 4. Recognizing that in general, the local mesh geometry that a processor performs calculation on can be fairly irregular, we adopt the following approach to establishing the mapping in order to have a broad applicability: First, we record the local mesh geometry that an MPI processor belongs to. For example, for a uniform grid we record the hyperslabs, and for a unstructured mesh we record the set of triangles. Then, each processor in the reduced model broadcasts their local mesh geometries to the full model. The processors in the full model receive the mesh geometry and check for the overlap with its own mesh geometry. If there is an overlap, it sends an acknowledgement back to the corresponding processors in the reduced model. Fig. 4 shows an example of mapping of the sub-domains in the full and reduced models in a uniform mesh.

3.2.3 Calculating Delta

To calculate the delta, we perform coarse-to-fine prolongation using piece-wise linear interpolation on the reduced model data, similar to the prolongation transfer operation in multigrid. Intuitively, data in the reduced model can be regarded as a uniform sampling of the full model data. We estimate the new grid points using the linear interpolation. Lastly, the delta is calculated as $DELTA = D' - D$. Note that during post-processing, if the high fidelity data (D) are requested, the low fidelity data are re-computed and similarly prolongated to D'. Then the delta is decompressed, and applied to D', i.e., $D = D' - DELTA$.

4 Performance Evaluation

4.1 System Setup

We evaluate the effectiveness of the proposed DuoModel on Titan at Oak Ridge National Laboratory. For the preliminary results in this paper, we use eight compute nodes. On Titan, each compute node has a 16-Core 2 GHz AMD processor and 32 GB of RAM, and the operating system is Cray Linux environment. The full models are always allocated 64 cores, and we vary the number of cores allocated to the reduce models as a proxy for the degree of fidelity. The details of configurations for the applications are listed in Table 1. For ZFP and SZ, the evaluations are done under the same relative error bound of 10^{-4}.

4.2 Compressing Delta versus Original Data

Fig. 5 illustrates the compression ratios of the original data versus deltas under different scales of reduced models. As the compute resources allocated to the reduced model increases, the compression ratio of delta increases for both ZFP and SZ. The reason is that, as the fidelity of a reduced model increases, the delta is expected to approach zero, which is in general more compressible. In particular, for the Wave equation, the compression ratio of delta using SZ is over 70X using additional 25 percent processors for the reduced model—a 5X improvement over compressing the original data directly. This clearly demonstrates the performance benefit of pre-conditioning data prior to compression. We point out that running the reduced model is not free and does require additional compute resources. However, with the abundance of compute resources on future systems, those resources pre-allocated for

2. https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/

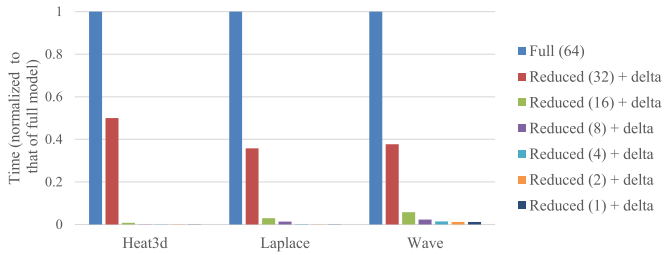

Fig. 6. Total time to generate the original data.

resilience purposes, and those that are not used (e.g., those CPUs unused for a GPU code) can be used for the reduced model, trading compute for I/O is possible.

4.3 Understanding Re-Computing Cost

The total re-compute time consists of three parts, running the reduced model, recompute the high resolution data based on linear interpolation, and applying the delta. Fig. 6 shows the total time of re-computing the original data, normalized against the time of running the full model directly. Note that the total time collected here is the wall-clock time multiplied by the number of cores used. It is clear that DuoModel requires a very small fraction of time to re-compute the original data. In particular, the reduced models of Heat3d, Laplace, and Wave equation with 16 processors incur 0.8, 2.9, and 5.7 percent re-compute cost, respectively. The main reason is that reduced models take much less time and resources than the full models.

5 Conclusion

This paper provides DuoModel, a new approach that leverages the similarity between the full and reduced application models to further reduce data on HPC storage. The proposed approaches are based on three observations. First, state-of-the-art scientific data compressors can not achieve further reduction routinely. Second, as computing is getting cheaper, we can re-compute the full data to reduce storage cost. Third, and the most importantly, scientific simulations' output data have similarity between the full and reduced models. It is easy to characterize the delta data, defined as the difference between the two models. Compressing the delta can achieve much higher compression ratios than the original data. Thus, we store the compressed delta rather than the original data to reduce the amount of data. When the original data are needed, they can be re-computed by launching the reduced model and applying the compressed delta. By this way, we reduced the storage cost while the high fidelity of data is maintained.

Acknowledgments

The authors wish to acknowledge the support from the US National Science Foundation under Grant No. CCF-1704504, CCF-1812861, and CCF-1629625, NetApp research grant, and NJIT research startup fund.

References

[1] T. Lu, Q. Liu, X. He, H. Luo, E. Suchyta, J. Choi, N. Podhorszki, S. Klasky, M. Wolf, T. Liu, and Z. Qiao, "Understanding and modeling lossy compression schemes on HPC scientific data," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2018, pp. 1–10.

[2] D. Meister, J. Kaiser, A. Brinkmann, T. Cortes, M. Kuhn, and J. Kunkel, "A study on data deduplication in HPC storage systems," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2012, pp. 1–11.

[3] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1245–1250, Sep./Oct. 2006.

[4] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data," in *Proc. Eur. Conf. Parallel Process.*, 2011, pp. 366–379.

[5] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 2674–2683, Dec. 2014.

[6] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2016, pp. 730–739.

[7] I. Foster, et al., "Computing just what you need: Online data analysis and reduction at extreme scales," in *Proc. Eur. Conf. Parallel Process.*, 2017, pp. 3–19.

[8] P. Benner, S. Gugercin, and K. Willcox, "A survey of projection-based model reduction methods for parametric dynamical systems," *SIAM Rev.*, vol. 57, no. 4, pp. 483–531, 2015.

[9] R. Courant, K. Friedrichs, and H. Lewy, "Courant-Friedrichs-Lewy condition," 2018. [Online]. Available: https://www.cfd-online.com/Wiki/CourantFriedrichsLewy_condition

[10] Dournac, "MPI parallelization for numerically solving the 3D heat equation," 2018. [Online]. Available: https://dournac.org/info/parallel_heat3d

[11] Jburkardt, "C source codes for laplace and wave equations using MPI," 2018. [Online]. Available: https://people.sc.fsu.edu/~jburkardt/c_src/c_src.html