Accelerating In-Transit Co-Processing for Scientific Simulations Using Region-Based Data-Driven Adaptive Compression

Marcus Wallden

Graduate School of Information Science and Technology, Osaka University

1. Introduction

Processing capabilities of supercomputers are improving at a rapid pace. However, I/O bandwidth is advancing at a much slower rate. This is one of the most pressing matters facing large-scale scientific simulations. Simulations can generate many tera- or petabytes of data, meaning that it is difficult to save all important data to permanent storage as a result of limited storage capacity or time-consuming I/O operations [1-3]. In-transit co-processing can instead be used to analyze and visualize data for each time step while it is generated.

In-transit co-processing is typically performed on a separate group of nodes, which we call *transit nodes*. Utilizing a separate node group means that co-processing can be performed asynchronously during the simulation stage, resulting in a less strict time limitation compared to other solutions. The data transfers required to perform in-transit co-processing can be accelerated by compressing the simulation data. However, using a lossy compression method could lower the accuracy and detail of regions of interest in the data. It could instead be preferable to selectively compress regions based on their contribution to the simulated phenomenon. Reducing regions which are not of interest would result in less time-consuming data transfers, without any significant loss in data quality.

We have developed an approach, called ATCO, to accelerate in-transit co-processing for large-scale simulations which utilize multivariate and temporal data sets in the form of structured rectilinear grids. In these grids, lightweight computations are adaptively performed in-situ to accelerate the simulation, data reduction, data transfers and co-processing. The importance of *blocks*, convex and contiguous data regions which make up the simulation data, are determined by utilizing various importance metrics and filters. Such information can then be used to identify important regions, down-sample, reduce, compress or simply remove parts of the data based on user-defined constraints. By this approach, we strive to reduce the data size and the in-transit data transfer time by utilizing multiple reduction and compression methods based on the entropy of the data. The loss of detail in regions of interest can thus be kept at a minimum.

2. Related Work

Determining and analyzing regions of interest in three-dimensional (3D) data sets has been an important topic in many different fields of research, and has as such been explored in many related works [4-8]. However, such research has generally been limited in scope or in its ability to perform multiple analyses to determine block importance. Our approach improves upon this related work by being able to efficiently handle multiple importance analyses and by adaptively utilizing multiple different compression and reduction strategies.

Close to our work is a paper by Dorier et al. [4], in which various importance metrics were used to dynamically reduce unimportant data. Their approach specifically targeted in-situ visualization and supported elementary data reduction. In addition, they utilized load balancing based on a random distribution to balance the load among the available compute nodes. However, their approach only supported the use of a single reduction method.

3. Adaptive In-Transit Co-Processing

Our approach consists of three stages: the in-situ stage,

where importance is calculated and which compression method to use is determined for each block, the *distribution stage*, where generated simulation data is compressed, load balanced and transferred to transit nodes, and the *in-transit* stage, where data is decompressed and restructured on the transit nodes.

A key issue when calculating the importance of individual blocks is data locality. The simulation region is typically allocated in contiguous memory space. However, a block makes up a 3D subset of the simulated region, which leads to low cache hit rates, especially when using small block sizes. Utilizing multiple importance metrics or advanced data access patterns further complicate this issue. Our solution is to dynamically allocate a separate buffer for a block when it is analyzed. Relevant *filters*, which calculate importance metrics and determine which compression methods to use based on user-defined conditions, can then be applied in sequence for each block, leading to low memory overhead and high cache hit rates. This approach also ensures that the generated simulation data of each block only needs to be allocated and copied once, minimizing the overhead introduced by this step. Operating on data allocated on a per-block basis leads to a much higher data locality, meaning that multiple importance calculations can be performed at a lower computational cost.

Which importance metrics and filters to use depends on the used simulation. Similarly, which compression methods to use for which importance values depends on the needs of the researcher. We consider four compression and reduction methods: run-length encoding (RLE), LZ77 [9], Homogeneous (a block's data is reduced to a single value) and Skip (the block is not transferred to any transit node).

The generated simulation data varies throughout the simulation, meaning that the compressed data size is constantly changing. It follows that the need for compression and data reduction also changes throughout the simulation. It could be preferable to adaptively change filter parameters and the used compression methods based on some criteria. The criteria could be based on the memory usage, execution time or the remaining allocated time on a compute cluster. Rudimentary use of an adaptive condition has been explored in some related work [4]. However, only one condition was used, which limits its application.

We support the use of multiple filters and importance metrics, and a key issue is how to adaptively modify the conditions without affecting the intended flow of analysis. Our solution is to utilize an adaptive *condition window*, by which filter conditions can vary. The value of the condition window can slide one interval of 0.05 between 0.0 and 1.0 between each time step, based on input to the program. Filter conditions can have a defined lower and upper bound. Let c_l and c_h be the lower and upper bounds, respectively, and w *be* the current value of the condition window. The current value of the filter condition can then be calculated using the equation

$$c_l + (c_h - c_l) \cdot w. \quad (1)$$

This method ensures that the filter condition values adaptively can be changes in a controlled manner, thus retaining the intended flow of the analysis.

4. Double-Planar Richtmyer-Meshkov Instability

To evaluate the proposed approach we devised a double-planar case of the Richtmyer-Meshkov Instability using the simulation program CNS3D.

Initially, all fluids are at rest, other than the shocked incoming air. The shock wave passes from left to right, causing the membranes separating the two fluids to rupture. The membranes are supported on a fine wire mesh with a grid spacing of 0.4 cm. The initial boundary conditions and initial setup are illustrated in Fig. 1. Small initial perturbations increase greatly in size, leading to high Reynolds number turbulent mixing of the two gases. The simulation ran for 15,653 time steps on 32

compute nodes at a resolution of 2401 × 601 × 1201, for a total of 3,228 node hours. Out of all time steps, 80 were analyzed and used for testing purposes. In total, four transit nodes were used for co-processing. A visualization of three time steps of the mass fraction are shown in Fig. 2.

5. Results and Discussion

For testing purposes we devised a filter pipeline, consisting of three filters. Blocks which had a range of 0 were set to use Homogeneous compression. The Homogeneous compression method is lossy. However, since the range of the affected blocks was zero, all information could be retained. Blocks which consisted of more than 90% distinct values were set to use no compression, since lossless compression methods would not be able to compress such data as effectively. All other blocks were set to use RLE compression. The average execution times are displayed in Fig. 3. Using the lossless

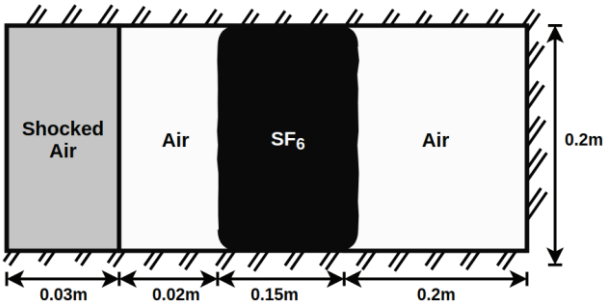

Fig. 1: Initial and boundary condition of the double-planar problem.

Fig. 2: Visualization of the mass fraction of the Richtmyer-Meshkov Instability simulation at recorded time steps 1, 40 and 80.

RLE method reduced the data transfer time by 35.2% as compared to using no compression. However, the decompression process significantly increased the co-processing time on the transit nodes. Compared to the RLE method, ATCO sped up the decompression process by a factor of 2.0. The speedup compared to using no compression was measured at 1.05. As for the data distribution stage, ATCO achieved a speedup of 1.14 compared to using the RLE method, and a factor of 1.77 compared to using no compression. Interestingly, ATCO was able to achieve better performance in all aspects (compressed data size, compression, transfer and decompression times) compared to the other lossless compression methods used for comparison. This is a result of the fact that ATCO scales with the compression methods which are used in its pipeline; in this case RLE and Homogeneous. This behavior should extend to other compression methods as well.

6. Conclusion

We devised and evaluated a novel approach to perform in-transit co-processing in which the simulation data is analyzed to determine the importance of all regions of data. Such information is then used to simultaneously utilize multiple compression and reduction methods to accelerate the in-transit co-processing while minimizing loss of detail in regions of interest. The approach was able to calculate the importance of regions of data expediently, even in cases where multiple importance metrics were in use. Our approach was able to achieve better performance in all aspects (compressed data size as well as compression, data transfer and decompression times) in all evaluated tests than the methods used for comparison. Compared to a run length encoding compression, our approach achieved speedups of 1.14 and 2 when performing data transfers and data decompression, respectively. The excellent scalability and performance make the approach suitable to be used in tandem with many large-scale simulations which utilize in-transit co-processing to analyze the generated data. In future work

we plan to extend the approach to work better with in-situ workflows.

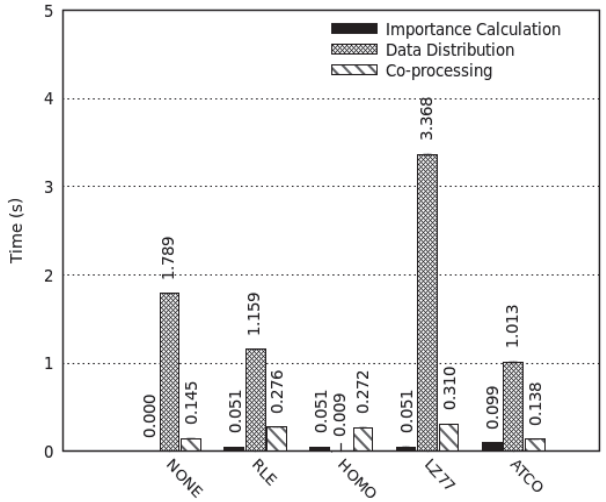

Fig. 3: Average execution times of the three main stages.

References

[1]. Rivi M, Calori L, Muscianisi G, Slavnić V. In-situ Visualization: State-of-the-art and Some Use Cases. 2012.

[2]. Yu H, Wang C, Grout RW, Chen JH, Ma K. In Situ Visualization for Large-Scale Combustion Simulations. IEEE Computer Graphics and Applications 2010; 30(3): 45-57.

[3]. Bennett JC, Abbasi H, Bremer P, et al. Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.; 2012: 1-9.

[4]. Dorier M, Sisneros R, Gomez LB, et al. Adaptive Performance-Constrained In Situ Visualization of Atmospheric Simulations. In: 2016 IEEE International Conference on Cluster Computing (CLUSTER). 2016: 269-278.

[5]. Wang C, Yu H, Ma K. Importance-Driven Time-Varying Data Visualization. *IEEE Transactions on Visualization and Computer Graphics* 2008; 14(6): 1547-1554.

[6]. Nouanesengsy B, Woodring J, Patchett J, Myers K, Ahrens J. ADR Visualization: A Generalized Framework for Ranking Large-Scale Scientific Data Using Analysis-Driven Refinement. In: 2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV). 2014: 43-50.

[7]. Berger MJ, Oliger J. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics* 1984; 53(3): 484-512.

[8]. Shimokawabe T, Onodera N. A High-Productivity Framework for Adaptive Mesh Refinement on Multiple GPUs. In: Computational Science – ICCS 2019. 2019: 281-294.

[9]. Ziv J, Lempel A. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory 1977; 23(3): 337-343.