Low-Cost Post Hoc Reconstruction of HPC Simulations at Full Resolution

Ayman Z. Yousef*
Duke University

Erik W. Draeger†
Lawrence Livermore National Laboratory.

Amanda Randles‡
Duke University

Abstract

High performance computing has played a pivotal and ongoing role in the field of computational fluid dynamics, enabling the simulation of increasingly larger-scale models. However, this rapid growth in model and simulation size has outpaced the capabilities of input/output (I/O) operations. Consequently, the conventional approach of saving and outputting data to persistent storage for analysis has become increasingly challenging, limiting the benefits of these advanced models. To address this challenge, we present a method for effectively handling massive-scale simulation data, ensuring its persistence at full spatial and temporal resolution for flexible *post hoc* analysis. We employ an *in situ* approach that captures interprocess communicated data, compressing cached data to a fraction of the overall simulation domain. We successfully reconstruct subdomains at full spatial and temporal resolution during post-processing through communication-free rerun using the cached halos. We detail the storage requirements of the new approach and demonstrate the substantial reductions in computational resources required to precisely recapitulate data within a local region of interest.

1 Introduction

Computational fluid dynamics (CFD) has become more prevalent in several industries, ranging from manufacturing to the automotive industry [18] to more biologically adjacent areas such as the study of cardiovascular disease [14]. The space has evolved rapidly with the emergence of large compute clusters, gaining popularity due to the advantages of more computationally grueling numerical simulations over physical testing. Simulations composed of trillions of elements can now be achieved in a few hours using high performance computing (HPC) systems [3]. This surge in model size requires a reexamination of traditional approaches to input/output (I/O) and post-processing data analysis, as writing full system checkpoints at a frequency high enough to capture temporal dynamics is quickly becoming prohibitive in terms of both wall time and available storage resources. New approaches are needed to enable accurate data analysis of large, high-resolution CFD simulations.

Many proposed post-processing techniques are based on data reduction. Downsampling, in which a subset of data is saved, is commonly used for cost savings [5]. For CFD simulations, temporal and spatial downsampling has the potential to remove essential details, for example, in the case of high Reynolds number flows. In general, coarsening of high-resolution data has the potential to introduce unintended consequences and potentially weaken the predictive accuracy of the original simulation. Similarly, limiting the storage of high-resolution data to a subset of the simulation domain requires precise foreknowledge of all regions of interest and prevents the investigation of phenomena that are not well understood at the outset.

In situ techniques have emerged as an alternative, where data analysis is performed during generation to avoid downsampling errors. Despite progress in loosely coupled/tightly coupled *in situ* methods, comprehensive data exploration remains challenging, requiring analysis to be run at the same scale as the associated simulation. As a result, the total compute cost of these routines can easily exceed that of the initial simulation, particularly when multiple exploratory simulations are required [8]. Interactive data exploration of simulations on a large number of nodes is similarly impractical, as compute nodes would be kept idle during visualization and decision-making. The need for real-time interaction would also create the possibility of missing transient events, necessitating repeated runs at scale.

In this study, we introduce a novel *post hoc* analysis routine (Figure 1) designed to accommodate evolving analysis requirements. This approach reduces the overall computational cost of analysis whilst maintaining the native resolution of investigated simulations. Our method saves sequences of communicated halo cells to the file system, enabling subsequent *post hoc* reconstruction and analysis of subdomains at their original resolution at a modest computational cost. This enables large-scale simulations to be exactly regenerated in local subsets on demand and at full fidelity. As spatial and temporal domains of interest are identified, high-resolution dynamics within those regions can be extracted and stored for subsequent use with pre-existing analysis and visualization frameworks.

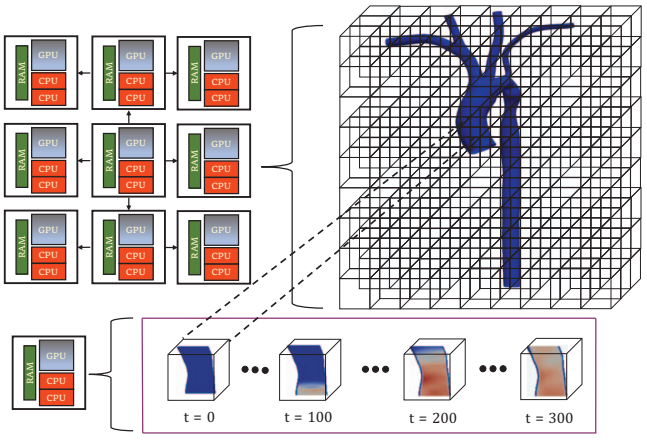

Figure 1: Traditional large-scale CFD simulations require communication across processes to capture the fluid flow in complex geometries. Shown here, we present a method that permits communication-free subdomain reconstruction at a lower resource cost to enable analysis of regions of interest.

This study offers the following contributions:

- Introduce a methodology that exhibits comparable flexibility to traditional *post hoc* workflows, allowing the preliminary analysis to inform the subsequent analysis with reduced computational expenses compared to entirely *in situ* approaches.
- Illustrate the ability of the proposed method to enable lossless, full-resolution *post hoc* analysis.
- Conduct experimental evaluations to compare the integrity of the data from the reconstructed data and the overall resource costs of simulation and analysis with conventional techniques.

*e-mail: ayman.yousef@duke.edu
†e-mail: draeger1@llnl.gov
‡e-mail:amanda.randles@duke.edu

2023 IEEE 13th Symposium on Large Data Analysis and Visualization (LDAV) | 979-8-3503-3018-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/LDAV60332.2023.00009

2832-6512/23/$31.00 ©2023 IEEE
DOI 10.1109/LDAV60332.2023.00009

2 Related Work

In response to increasing computational demands, there has been a growing interest in compression-based techniques. These techniques are commonly applied to 3D scientific data sets that encompass a wide range of values, although they often introduce some degree of loss. The integration of machine learning for accelerated data compression has emerged as a promising area of research, particularly in time-based systems, as evidenced by studies such as [10, 22].

In situ-based techniques have continuously evolved and are preferred due to their inherent advantages and benefits of working directly with data in place, as highlighted in works like [2, 11]. These techniques offer improved I/O costs compared to compression-based methods but present certain limitations. One drawback is their reliance on re-running simulations on equivalent systems, which poses a significant challenge for large-scale simulations that require access to limited HPC resources.

Furthermore, in the case of *in-transit* analysis routines, the overall cost to the main simulation can become prohibitive or lead to suboptimal utilization of HPC hardware, as discussed in [12]. Moreover, *in situ* analysis routines often contend for time and resources with the primary simulation, resulting in less-than-ideal runtime. The ideal approach is to perform analysis on a machine that is best suited for the workload, is cost-effective, and is widely accessible. Currently, there are no other methods that offer the same level of flexibility and enable analysis with full temporal and spatial resolution.

3 Approach

3.1 Stencil Codes

Stencils are a common pattern in scientific and engineering applications [7, 16]. In these patterns, each time step involves a series of sweeps over the domain, where points are updated by applying a function to neighboring elements within the stencil's shape. The arithmetic intensity of stencils tends to be low and the performance is memory bound [4]. However, due to the independent nature of point updates, stencil codes can be effectively parallelized to achieve significant speed-ups, as observed in [9]. The spatial domain is divided among MPI processes (called "domain decomposition") in parallel implementations, allowing them to evolve local elements concurrently. During domain decomposition, overlapping regions between adjacent tasks called "halo regions" or "halos" are defined. These regions establish the relationships between tasks. After local updates, synchronization with neighboring processes takes place, during which data residing in these halo regions are exchanged to ensure accurate domain updates for each rank.

3.2 Lattice Boltzmann Method and Halo Replay

With the focus of our study surrounding stencil algorithms and models capable of simulations at scale, we use HARVEY [15], a massively parallel hemodynamics application that implements the lattice Boltzmann method (LBM). LBM is a numerical scheme that approximates the Navier-Stokes equations and exhibits near-ideal scaling [21]. This scaling, alongside algorithmic structure and parallelism-based efficiencies, maintains its popularity in several industries [17].

HARVEY is used predominantly to generate personalized patient-specific blood flow simulations derived from medical imaging data, with the objective of supporting clinical decision-making [6, 19, 20]. HARVEY is specialized for simulations involving sparse, complex vascular geometries, which at high resolution can result in systems with trillions of fluid points [13]. The most common scheme is the D3Q19 lattice framework, characterized by a cubic spatial domain and a set of 19 double-precision discrete velocities per lattice point [1]. Fluid data are represented by a distribution array derived from hemodynamic data. Distribution arrays are updated via streaming and colliding (LBM particle distribution functions) at every time step, operations local to a neighborhood. To accurately perform these routines, constant communication with neighboring tasks is required to update arrays. The only communication needed outside of streaming/colliding occurs during the initialization process/load balancing section, further justifying the decision to apply our *post hoc* methodology to HARVEY.

The Halo Replay method, introduced in [2], exploited this necessary data transfer within HARVEY by caching halos in memory for the on-demand reconstruction of fluid points. This accumulation of halos, termed a "halo buffer", allowed a "child task" behind the primary simulation to advance independently of communication amongst other MPI processes. Previous work was focused on *in situ* analysis with the Replay method, with the halo buffers strictly available and accessible at run-time by maintaining the halo buffer in memory. While advantageous, this approach falls victim to the same disadvantages as other similar *in situ* techniques: having to re-launch simulations on the same resource set as expensive, initial simulations, and noninteractivity inherent to certain *in situ* analysis routines. [2]. Therefore, we introduce our new *post hoc* framework that further extends Halo Replay, Post Hoc Replay.

By caching and writing to storage the same halo buffers necessary in Halo Replay, we can deterministically replicate our simulations at full spatiotemporal resolution without the restrictions presented above (Figure 2). The prospect of having halos on hand also allows the unique prospect of launching simulations that encompass subdomains of prior runs/simulations free from any communication. The following is a brief overview of Post Hoc Replay:

- An initial simulation is run, during which the initial setup process writes to disk communication-dependent parameters and a checkpoint consisting of the full distribution array.
- As the simulation progresses, halos are appended to a pre-allocated halo buffer. At the end of the run, the halo buffer is written to disk. This routine is only performed by pre-declared processes, encapsulating subdomains of the overall simulation.
- To reconstruct these subdomains, the initialization files (which includes the full checkpoint) and halo buffer(s) are loaded in. The simulation then advances as normal, substituting communication for loading of corresponding halo buffer values.

Because all data is kept intact and stored at full machine precision, we can deterministically reconstruct simulations. We now delve into the contributions to overhead and storage.

3.3 Estimating the Storage Costs of Halo Data

While the Post Hoc Replay method requires storing data from every time step, the large surface-to-volume ratio of typical HPC simulations means that the total storage cost of a simulation using Post Hoc Replay may be modestly higher than that from traditional checkpointing. The flexible regeneration of intermediate temporal data in specific regions of interest also allows users to checkpoint far less frequently to reduce total storage if needed.

To illustrate the relative storage cost of the halos as a function of system size and parallel decomposition, we present a simple model problem of a dense block of fluid points on a uniform grid $n_x \times n_y \times n_z$ distributed across a Cartesian logical process grid of $p_x \times p_y \times p_z$ MPI tasks. On a single task, the number of fluid points in the halo region can be calculated by multiplying the area of each face of the rectangular domain by the thickness of the halo region:

$$N_{fluid}^{loc} = 2d_h \left[\left(\frac{n_x}{p_x}\right)\left(\frac{n_y}{p_y}\right) + \left(\frac{n_x}{p_x}\right)\left(\frac{n_z}{p_z}\right) + \left(\frac{n_y}{p_y}\right)\left(\frac{n_z}{p_z}\right) \right] \tag{1}$$

where d_h is the thickness of the halo region in grid points, typically $d_h = 1$ for the applications being discussed here. For simplicity, we assume that the fluid grid is an exact multiple of the process

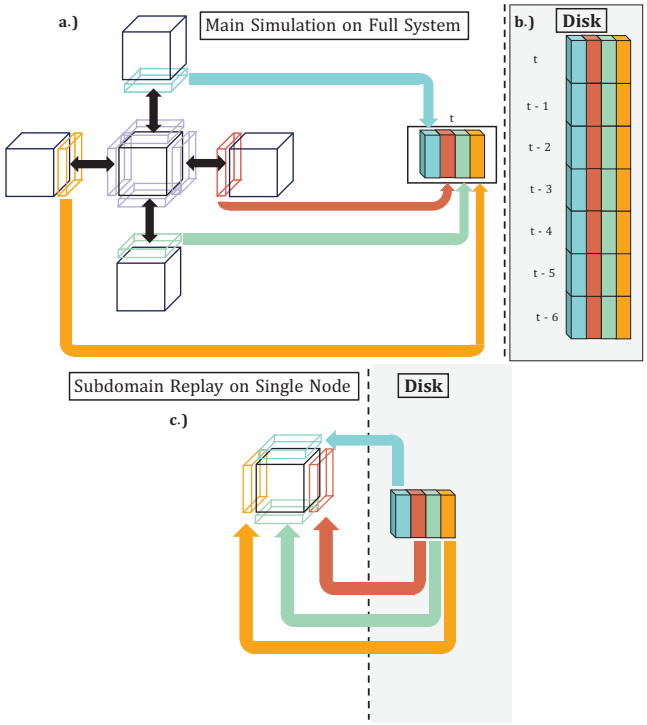

Figure 2: Outline of Post Hoc Replay. a.) Halo data generated from initial simulations are stored within memory as the simulation progresses. b.) Accumulation of these halos is saved to disk and later loaded in. c.) In place of inter-node communication, at each time step the corresponding portion of the halo region from neighboring tasks is copied into the appropriate MPI buffers.

grid. The total fraction of fluid points in the halo region can thus be written as

$$f_{halo} = \frac{p_x p_y p_z N^{loc}_{fluid}}{n_x n_y n_z} = 2d_h \left(\frac{p_x}{n_x} + \frac{p_y}{n_y} + \frac{p_z}{n_z} \right) \quad (2)$$

if we assume $\frac{p_x}{n_x} \approx \frac{p_z}{n_z} \approx \frac{p_z}{n_z}$ we can simplify this as

$$f_{halo} \cong 6d_h \frac{p_x}{n_x} \quad (3)$$

From this, we can generalize Eq. 3 for any parallel decomposition that results in mostly cubic local domains as

$$f_{halo} \approx 6d_h \left(\frac{N_{fluid}}{N_{procs}} \right)^{-1/3} \quad (4)$$

In other words, the relative cost of a halo buffer compared with a full checkpoint can be estimated using only the average number of fluid points per task. This estimation provides a simple guideline for calculating expected storage costs and adjusting checkpoint frequency to compensate. For example, when each task has 10 million fluid points, f_{halo} is 0.0278, indicating a halo buffer is 35.9x smaller than a full checkpoint. This analysis should not overshadow the fact that halos must be saved from every time step, making halo storage requirements comparable to standalone checkpointing in many cases. One should thus balance the advantages of the *post hoc* analysis capabilities against computational requirements and storage limits.

4 Results/Discussion

4.1 Simulation Details

The input geometries used consisted of increasing-sized cubes for detailed performance analysis and an aorta (referenced in Figure 1) to provide a biologically relevant example for validation. The composition of the cubic inputs used are summarized in Table 1. Input sizes were chosen in order to linearly scale the number of fluid points. The flow was simulated through the cubic geometries for a total of 10,000 timesteps. Full simulation checkpoints were written every 500 timesteps, representing a typical standalone analysis setup. For simplicity, halo buffers were cached to local memory every time step and written to disk at the end of the simulation.

Table 1: Cubic fluid simulation parameters

Label	Size (μm)	# of Fluid Points	# of Ranks
C320	320 x 320 x 320	32,157,432	128
C400	400 x 400 x 400	63,044,792	256
C500	500 x 500 x 500	123.505.992	512
C630	630 x 630 x 630	246,587,017	1024
C750	750 x 750 x 750	418,508,992	2048

4.2 Hardware/Software

The simulations were performed on two different computing systems: Oak Ridge National Laboratory's Summit and the Duke Compute Cluster. The Duke Compute Cluster is an Intel-based system consisting of 60 nodes, each equipped with 40 physical cores. Summit is an IBM-based supercomputer with a peak performance of approximately 200 petaFLOPS. Summit comprises 4,600 IBM Power System AC922 nodes, each node housing two IBM POWER9 processors and six NVIDIA Tesla V100 accelerators. Additionally, the nodes are equipped with 512 GB of DDR4 memory, 96 GB of High Bandwidth Memory, and 6.4 TB of non-volatile memory.

4.3 Validation of Post Hoc Replay

In our validation experiments, we compared the lattice Boltzmann distribution arrays of the original full simulation against those reconstructed using the Post Hoc Replay approach. Figure 3 presents the results of this analysis for the aortic geometry. In all cases, the Post Hoc Replay results agreed exactly with the original simulation results to machine precision. This result is a confirmation of the expected lossless nature of the algorithm.

4.4 Analysis of Storage Cost Across Routines

To assess the relative storage costs of our method, we compared the size of halo buffers produced by Post Hoc Replay across all ranks and a full simulation checkpoint, each for a single time step. We observe in Figure 4 the storage requirements for the halo buffers and checkpoints for different system sizes. The size of a halo buffer on a single task at a single time step is shown, along with the size of the halo buffers of all tasks at a single time step and the size of a full checkpoint of all fluid data on all tasks. All data is saved in double precision. While full checkpoints are significantly larger than halo buffers across all input sizes, halo data from all time steps must be stored to enable fully deterministic reconstruction. Even with proper checkpointing, improper domain decomposition could cause cause storage costs of halo data to exceed that of conventional checkpoint writing. The frequency of full checkpoints will vary significantly by use case. For some systems, frequent full checkpoints will be needed to later identify precise regions of interest for Post Hoc Replay analysis. For other systems, where the region of interest is known *a priori*, only infrequent defensive checkpoints may be needed to enable restart in the case of hardware failure or simulations that require multiple jobs to achieve a given total runtime. Similarly,

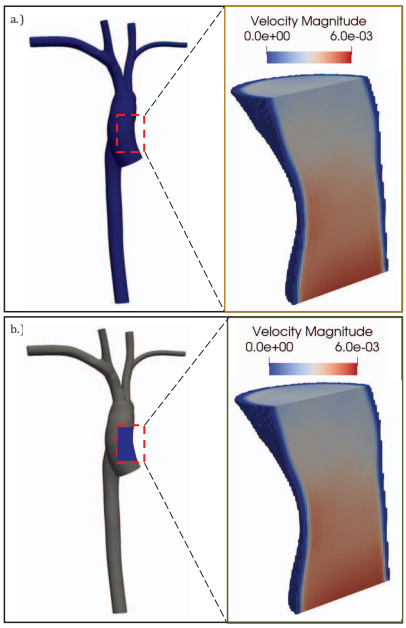

Figure 3: We showcase an example of subdomain reconstruction. a.) Simulation domain on MPI task 4, loaded from a checkpoint at t = 500. b.) Simulation domain of MPI task 4, loaded from a checkpoint at t = 0 and advanced to t = 500 using halo replay on a single task. All values agree within machine precision.

for systems with a well-identified region of interest, only the halo buffers for the task(s) with data in that region need to be stored. Otherwise, the best practice is to store all halo buffers to enable full replay capabilities.

The flexibility Post Hoc Replay provides is further manifested when considering the relative ease with which our method integrates with checkpointing. Because checkpoints can capture the complete simulation state at a specific timestep, our *post hoc* routine allows us to co-opt a checkpoint as the starting point for a simulation within a localized region and advance the simulation forward in time, provided halo buffers are properly captured for that region. Thus, this characteristic provides a cost-effective method for spanning a simulation between sparse checkpoints or capture specific "windows". Examining a particular region of interest with a much finer-time granularity can pay massive dividends when considering questions like cancer metastasis and wound healing.

4.5 Computational Cost Reduction with Post Hoc Replay

The ability of Post Hoc Replay to fully reconstruct subsets of a simulation at significantly lower computational cost is shown in Figure 5 for the cubic geometries described in Table 1. The computational resources required to fully reconstruct all 10,000 time steps of a single task's spatial domain are shown in the lower panel along with the corresponding resource requirements of the full simulation. The ratio of the two values is shown in the upper panel and corresponds to a reduction in required resource requirements to fully regenerate a given task's data. For larger simulations such as C750, the reduction is considerable at three orders of magnitude. The fact that the reduction exceeds the ratio of MPI tasks in each run is due to the additional efficiency gained from the communication-free Post Hoc Replay. Similarly, we measured the total overhead introduced to the primary simulation by both halo buffer writing for a single rank across all timesteps and full simulation checkpoint writing every 500 timesteps. Figure 5 shows that checkpoint writing more significantly affects the runtime of the initial simulation in all cases, further

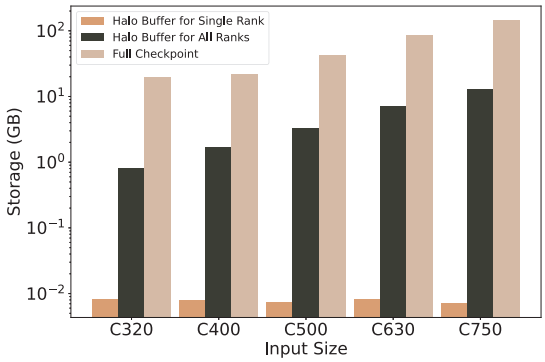

Figure 4: Storage required for halo buffers and checkpoints for one time step across all input sizes. Checkpoints exceed halo buffer storage sizes across all inputs, illustrating the inherent compression characteristic of our method

characterizing Post Hoc Replay as the more efficient methodology. However, the efficiency of this method is not yet fully realized. For Post Hoc Replay, regarding both the generation and utilization of halo buffers, read/write operations predominantly dominate runtime.

Lower computational costs combined with confidence in the fidelity of Post Hoc Replay enable more flexible deployment of traditional analysis routines. *In situ* visualization, use of visualization engines, and statistical analysis routines can all be deployed on subdomains of interest. This separation of the traditional HPC simulation workflow into large-scale data generation and modest-scale data analysis pieces and the greater ease of access to smaller-scale compute resources can be instrumental in reducing the overall time to the solution of scientific questions.

5 Summary and Conclusions

We have demonstrated the viability of the Post Hoc Replay method for enabling fully deterministic reconstruction of simulation data within a subdomain with no loss in accuracy and at a significantly reduced computational cost compared with the original simulation, particularly when localized regions of interest can be identified. Storage resource requirements were estimated and shown to be manageable. Although demonstrated on a stencil-based CFD approach, the method is easily generalizable to any with discrete domain decomposition and short-range interactions between neighboring tasks. This method enables a new paradigm in resource-separated HPC simulation and analysis, which is expected to be of broad interest given the common HPC resource constraints facing many researchers.

There are several avenues of improvement with regard to implementation. The prospect of saving halo buffers across all ranks for all timesteps can become more storage-intensive depending on the use case. The overhead associated with writing and reading the halo buffers is similarly non-negligible. To avoid a prohibitive memory footprint and imbalanced use of hardware resources, we are exploring a more robust implementation that can make use of advanced file systems features, such as burst buffers or fast local storage, to efficiently stage these data from memory out to the file system throughout the simulation with a minimal performance impact.

Acknowledgments

An award of computer time was provided by the INCITE program. This research also used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

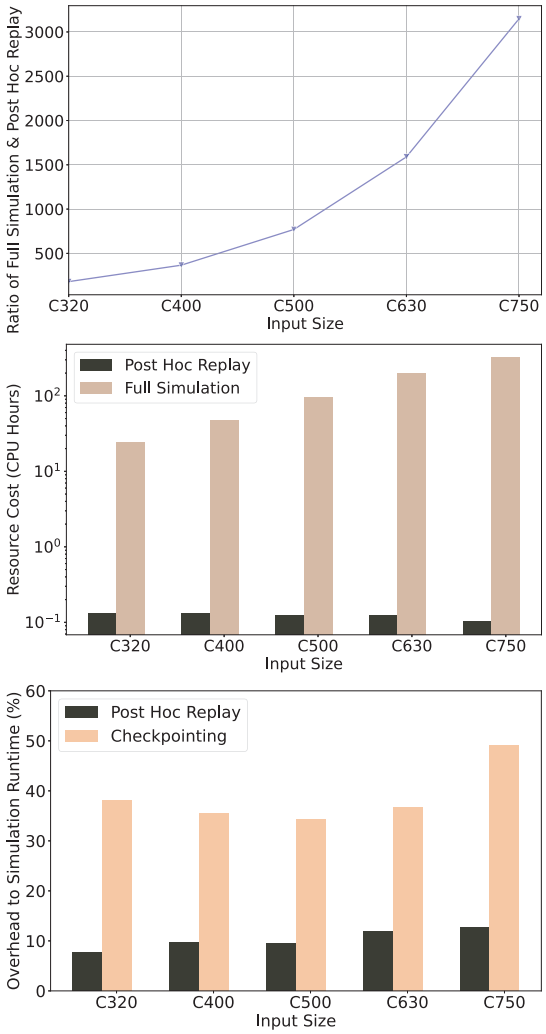

Figure 5: (Middle) Comparison of simulation costs in Summit CPU-hours for both full simulation and Post Hoc Replay. (Top) Ratio of computational resources needed to run the full simulation compared with those required to reconstruct all time steps on a single MPI task using Post Hoc Replay. (Bottom) Overhead to initial simulation introduced by halo buffer writing and checkpoint writing. Halo buffers spanned the entire simulation while checkpoints were written every 500 timesteps.

REFERENCES

[1] J. Ames, D. F. Puleri, P. Balogh, J. Gounley, E. W. Draeger, and A. Randles. Multi-GPU immersed boundary method hemodynamics simulations. *Journal of computational science*, 44:101153, 2020.

[2] J. Ames, S. Rizzi, J. Insley, S. Patel, B. Hernaandez, E. W. Draeger, and A. Randles. Low-Overhead In Situ Visualization Using Halo Replay. *2019 IEEE 9th Symposium on Large Data Analysis and Visualization (LDAV)*, 00:16–26, 2019.

[3] M. M. Bhatti, M. Marin, A. Zeeshan, and S. I. Abdelsalam. Recent trends in computational fluid dynamics. *Frontiers in Physics*, 8:593111, 2020.

[4] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, K. Yelick, C. R. D. Nersc, L. Berkeley, and C. S. Division. Stencil Computation Optimization and Auto-tuning on State-of-the-Art Multicore Architectures. *2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2008*, (November), 2008.

[5] A. M. Dunton, L. Jofre, G. Iaccarino, and A. Doostan. Pass-efficient methods for compression of high-dimensional turbulent flow data. *arXiv*, 2019.

[6] J. Gounley, E. W. Draeger, and A. Randles. Numerical simulation of a compound capsule in a constricted microchannel. *Procedia Computer Science*, 108:175–184, 2017.

[7] T. Gysi, C. Osuna, O. Fuhrer, M. Bianco, and T. C. Schulthess. Stella: A domain-specific tool for structured grid methods in weather and climate models. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1–12, 2015.

[8] J. Kress, M. Larsen, J. Choi, M. Kim, M. Wolf, N. Podhorszki, S. Klasky, H. Childs, and D. Pugmire. Comparing the efficiency of in situ visualization paradigms at scale. vol. 11501 LNCS, pp. 99—117, 2019.

[9] S. Krishnamoorthy, M. Baskaran, U. Bondhugula, J. Ramanujam, A. Rountev, and P. Sadayappan. Effective automatic parallelization of stencil computations. *ACM SIGPLAN Notices*, 42(6):235—244, 2007.

[10] S. Madireddy, J. H. Park, S. Lee, P. Balaprakash, S. Yoo, W.-k. Liao, C. D. Hauck, M. P. Laiu, and R. Archibald. In situ compression artifact removal in scientific data using deep transfer learning and experience replay. *Machine Learning: Science and Technology*, 2(2):025010, 2021.

[11] N. Marsaglia, M. Mathai, S. Fields, and H. Childs. Automatic in situ camera placement for isosurfaces of large-scale scientific simulations. 2022.

[12] S. Ramesh, H. Childs, and A. Malony. Serviz: A shared in situ visualization service. In *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 277–290. IEEE Computer Society, 2022.

[13] A. Randles, E. W. Draeger, and P. E. Bailey. Massively Parallel Simulations of Hemodynamics in the Primary Large Arteries of the Human Vasculature. *Journal of Computational Science*, 9:70—75, 2015.

[14] A. Randles, D. H. Frakes, and J. A. Leopold. Computational fluid dynamics and additive manufacturing to diagnose and treat cardiovascular disease. *Trends in biotechnology*, 35(11):1049–1061, 2017.

[15] A. P. Randles, V. Kale, J. Hammond, W. Gropp, and E. Kaxiras. Performance Analysis of the Lattice Boltzmann Model Beyond Navier-Stokes. pp. 1063—1074, 2013.

[16] E. Raut, J. Anderson, M. Araya-Polo, and J. Meng. Evaluation of distributed tasks in stencil-based application on gpus. In *2021 IEEE/ACM 6th International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)*, pp. 45–52. IEEE, 2021.

[17] K. V. Sharma, R. Straka, and F. W. Tavares. Lattice boltzmann methods for industrial applications. *Industrial & Engineering Chemistry Research*, 58(36):16205–16234, 2019.

[18] Q. Si, R. Lu, C. Shen, S. Xia, G. Sheng, and J. Yuan. An intelligent CFD-based optimization system for fluid machinery: Automotive electronic pump case application. *Applied Sciences*, 10(1):366, 2020.

[19] M. Vardhan, J. Gounley, S. J. Chen, A. M. Kahn, J. A. Leopold, and A. Randles. The importance of side branches in modeling 3D hemodynamics from angiograms for patients with coronary artery disease. *Scientific reports*, 9(1):1–10, 2019.

[20] M. Vardhan, J. Gounley, S. J. Chen, P. Nair, W. Wei, L. Hegele, J. Kusner, A. M. Kahn, D. Frakes, J. A. Leopold, et al. Evaluation of intracoronary hemodynamics identifies perturbations in vorticity. *Frontiers in Systems Biology*, 2(1), 2022.

[21] S. Williams, L. Oliker, J. Carter, and J. Shalf. Extracting ultra-scale Lattice Boltzmann performance via hierarchical and distributed autotuning. *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1—12, 2011.

[22] X. Yu, Y. Peng, F. Li, S. Wang, X. Shen, H. Mai, and Y. Xie. Two-level data compression using machine learning in time series database. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1333–1344. IEEE, 2020.