2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) | 978-1-6654-3577-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/IPDPSW52791.2021.00152

Facilitating Staging-based Unstructured Mesh Processing to Support Hybrid In-Situ Workflows

Zhe Wang*, Pradeep Subedi*, Matthieu Dorier†, Philip E. Davis*, Manish Parashar‡
*Rutgers University {jay.wang, pradeep.subedi, philip.e.davis}@rutgers.edu
†Argonne National Laboratory mdorier@anl.gov
‡University of Utah manish.parashar@utah.edu

***Abstract*—In-situ and in-transit processing alleviate the gap between the computing and I/O capabilities by scheduling data analytics close to the data source. Hybrid in-situ processing splits data analytics into two stages: the data processing that runs in-situ aims to extract regions of interest, which are then transferred to staging services for further in-transit analytics. To facilitate this type of hybrid in-situ processing, the data staging service needs to support complex intermediate data representations generated/consumed by the in-situ tasks. Unstructured (or irregular) mesh is one such derived data representation that is typically used and bridges simulation data and analytics. However, how staging services efficiently support unstructured mesh transfer and processing remains to be explored. This paper investigates design options for transferring and processing unstructured mesh data using staging services. Using polygonal mesh data as an example, we show that hybrid in-situ workflows with staging-based unstructured mesh processing can effectively support hybrid in-situ workflows, and can significantly decrease data movement overheads.**

***Index Terms*—in-situ, in-transit, data-driven, scientific workflow, unstructured mesh**

I. Introduction

The growing gap between computation speeds and I/O bandwidth has resulted in I/O becoming a key bottleneck for scientific workflows [1], [2]. This gap is the main motivation for data visualization and analytic pipelines to move from a post-hoc (i.e., file-based) paradigm to an in-situ paradigm. The term "in-situ processing" is often used as an umbrella word to describe systems that process data where (or close to where) it is generated, bypassing the file system. We follow a convention [3] that distinguishes the "in-situ" and "in-transit" terms in this work. The term "in-situ analytics" will be used to denote processing that shares the same computing resources with the simulation, in contrast to "in-transit analytics", in which a subset (or derived representation of) the raw data is transferred and processed through separate processors or staging resources, either on the same machine or different computing resources in the same system.[1]

Instead of processing data analytics all-in-situ or all-in-transit, hybrid in-situ processing adopts both in-situ and in-transit analytics in one system to improve workflow efficiency. In particular, in-situ analytics may use various data reduction techniques [6] before in-transit analytics is done on intermediate data products. Since only the reduced data is transferred to staging services, the pressure on data I/O and computing resources is alleviated during the workflow execution. The in-situ detection task may further decrease data movements by only transferring interesting data regions to subsequent tasks. In this paper, we use "data-driven hybrid in-situ workflow" to represent the workflow with the following properties[2]: (1) the workflow contains both in-situ and in-transit analytics steps in a single system; (2) the in-situ analytics is used to reduce the raw data size using data reduction techniques and the detection of interesting data; (3) only interesting data blocks are transferred to the in-transit analytics for further processing.

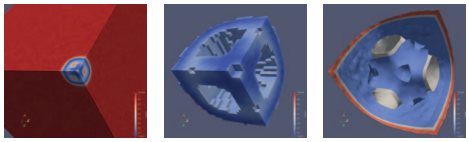
Several challenges pave the way to achieving efficient data-driven hybrid in-situ workflow mentioned above, including how to split a data analytics pipeline into in-situ and in-transit stages [3]; how to express the control flow with flexible trigger conditions [7]–[9]; and how to redistribute data partitions between the simulation and coupled components [10]–[12]. One less explored research challenge lies in how data staging services process and transfer the intermediate data between the in-situ and in-transit steps of the pipeline. Indeed while simulations often rely on structured representations such as Cartesian grids, in-situ data reduction may generate much more complex data representations [6]. Existing staging services [10]–[12] have been designed for structured data and, while they can be used as-is should the in-situ reduction step itself produce structured data, they lack features and efficiency when dealing with unstructured data.

One important such reduced data representation is *unstructured meshes* [13]; for example, the unstructured polygonal mesh is crucial to bridge simulation data, topology-based analytics, and visualization [13]–[15]. Instead of using a bounding box to describe the mesh implicitly, the unstructured data contains an explicit mesh described using multiple arrays of various data types and lengths to describe points, cells, and attributes. How the staging service transfers and processes unstructured meshes efficiently and facilitates data-driven in-situ workflows needs to be explored further. Without enabling custom data representations generated by data reduction in staging-based data service, the benefits of the data-driven hybrid in-situ workflow cannot be fully achieved. In this work, we 1) compare different strategies for transferring derived

[1]Other works [4], [5] use term "tightly-coupled/inline" and "loosely-coupled" to distinguish between these processing modes.

[2]We mainly consider data analytics in this work, but workflow tasks may also include visualization.

978-1-6654-3577-2/21/$31.00 ©2021 IEEE
DOI 10.1109/IPDPSW52791.2021.00152



(a) raw data (b) threshold (c) isosurface

Fig. 1. Different representations of the interesting data regions.

data with unstructured mesh representations; 2) extend a staging service to support various data transfer and processing strategies for unstructured meshes; 3) show that staging-based unstructured mesh processing improves the efficiency of data-driven hybrid in-situ workflows. With the capability of processing unstructured meshes, our experiment shows that the hybrid in-situ processing decreases the accumulated cost of data transfer and processing by up to 52.5% compared with running all analytics in an in-situ manner.

The rest of the paper is organized as follows: Section II discusses related works, and Section III illustrates motivating workflows for this paper. We then present the design and implementation details of our work in Section IV. In Section V, we evaluate our approach and discuss results. Finally, we conclude the paper and discuss the future work in Section VI.

II. Related Work

Hybrid in-situ processing incorporates in-situ and in-transit processing in one system to improve workflow efficiency. Bennett et al. [3], [16] describe several examples of how to decouple analytics into a combination of in-situ and in-transit stages. For example, the tasks that are highly efficient and massively parallel can be scheduled in-situ; and the small scale or serial tasks can be scheduled in-transit. Dreher et al. [16] present a data-flow oriented framework that composes in-situ and in-transit analytics flexibly based on python scripts. Zheng et al. [17] present the placement of analytics based on performance metrics. Furthermore, more recent works [4], [9], [18]–[20] try to model the workflow execution process and decide a proper analytics placement strategy dynamically using context in various scenarios.

Data-driven in-situ reduction plays a significant role in decreasing the data movement by selecting data subset for further processing. Various strategies have been proposed to reduce the data and extract interesting regions with different levels of information loss [6]. Data reduction techniques also aim to extract key geometry features necessary for further analytics or visualizations. Using the mesh decimation [21] or the mesh sampling [22], the reduced data can be a coarser representation of the original representation. The reduced data may also consist of derived representations such as images [23] or abstracted topologies [24] from the original mesh for the convenience of further processing. The decision to transfer reduced data for further processing may be affected by the priority value of data blocks [8], [22] or different analytics requirements from the data fidelity [2], [21].

In-situ unstructured mesh processing extracts, analyses, and visualizes the generalized unstructured mesh by in-situ

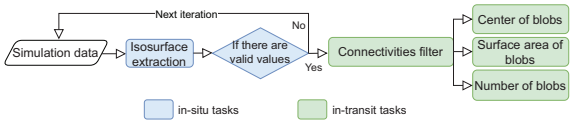


Fig. 2. Data analytics workflow based on mini-simulation.

paradigm. The unstructured data are fundamental representation for geometry-based data processing [15], [25]. Tom et al. [14] presented a parallel solution to extract the unstructured data for Voronoi and Delaunay Tessellation. Jonathan et al. [26] discussed a zero-copy solution for unstructured mesh between in-situ coupled analytics. Besides, multiple [27], [28] frameworks provide the capability to transfer the unstructured grid to in-situ tasks with the zero-copy strategy.

On the basis of the aforementioned related work, our work discusses how to optimize the data transfer mechanism between in-situ and in-transit tasks in hybrid processing, especially for unstructured data, which is commonly used for geometry-based data reduction. We also evaluate how the capabilities for unstructured mesh transfer can facilitate hybrid in-situ processing pipelines.

III. Motivating Workflows

One motivating in-situ workflow that runs in a data-driven fashion is the data analysis workflow developed for HACC (Hardware/Hybrid Accelerated Cosmology Code), a cosmological N-body simulation [29]. In the cosmological simulation, the halo region represents a high-density mass containing a large number of particles in a comparatively small region. One important goal of the simulation is to analyse halo regions' properties, such as concentration estimation and formulation of halos. The HACC workflow contains various analysis tasks with multiple goals in addition to visualization of the simulated data. In particular, analytics such as halo identification aims to detect the emergence of halo regions. The detected halo regions are further analysed by more compute-intensive analytics that finds the region center. The halo detection service finishes in a comparatively short time and runs together with the simulation computation, then the halo center finding tasks applied on the detected regions are run by accelerators or dedicated services that may take a relatively long time.

Inspired by the HACC workflow, we developed a data analytics workflow based on a mini-simulation as a proof of concept. We choose a simulation [30] that shows various Spatio-temporal patterns based on a reaction-diffusion model. One property of this simulation is that different blobs are formed during the progression of the workflow. Similar to the HACC workflow that processes halo regions, analytics in this workflow aim to analyse the topology properties of simulated blobs. As illustrated in Figure 1, the data generated by the mini-simulation is presented with different views. With the help of isosurfaces or threshold extractions, it is possible to detect interesting data regions, such as blob areas with different surface values, and use them as an intermediate representation for further topology analysis. Besides, the reduced data such as

unstructured mesh with a particular isosurface value is usually several orders of magnitude smaller than the original data.

As illustrated in Figure 2, we schedule the isosurface extraction task in-situ and run it in a data-parallel way for different data partitions. After that, only the valid reduced data is transferred to in-transit analytics. Other filters, such as connectivity and blob property analytics, are only applied to the data represented by an unstructured mesh. These tasks are triggered when targeted isosurfaces exist in a particular data partition. In Section V-B, we show how a staging service with unstructured mesh processing can facilitate the data-driven hybrid in-situ workflow presented in Figure 2.

IV. Design and implementation

A. Strategies for transferring unstructured mesh data

The general strategy to transfer a customized object is marshaling the data into a binary string and unmarshaling it back at the receiver end. If the data object is organized through regular mesh such as the Cartesian grid, the mesh information is usually a fixed-length array that describes a multi-dimensional bounding box. However, unstructured meshes use an explicit representation to describe the cells and points, which vary with the data complexity. This section presents several strategies to transfer the unstructured mesh with explicit mesh representation. We use the polygonal mesh of VTK (Visualization ToolKit) [13] as an example for convenience of the description, but the strategies described in this section are not limited to VTK polygonal data.

Single-segment: The straightforward strategy to transfer the customized object is to marshal the object into an array that is continuous in memory; after the transfer, the data can be unmarshalled back to the original data representation at the receiver. In this case, there is only one registered memory segment for RDMA in one RPC call. We use the *MarshalDataObject* function in the *vtkCommunicator* class to marshal VTK polygonal data into a VTK char array. When the data is transferred to the receiver, we use the *UnMarshalDataObject* function to reconstruct the VTK polygonal object.

Multi-segment: The alternative method uses multiple segments to transfer one data object, leveraging the fact that the unstructured mesh can be decomposed into several subcomponents. In this case, the intact polygonal data structure can be decomposed into an array of cells, an array of points, and one or multiple arrays of attributes. These arrays are registered for RDMA operation separately. In particular, the Mercury communication library[3] used in this work provides the capability to register several memory segments for RDMA data transfer in one RPC call. The data transfer time may benefit from the small size of a single registered memory segment for RDMA with this strategy.

Multi-segment-optimization: Multi-segment strategy can be further improved by several optimization methods. Suppose there is an available API to access the memory address of the sub-components of the unstructured data model, we can register memory space for RDMA operation by copying the memory addresses to save the data marshaling time. However, if the unstructured data model [26] mix points and cell connectives, or when there is also no flexible mechanism to access the memory addresses of sub-components, in these cases, we need to iterate all cells to split the cell topology array and point coordinates array. One optimization strategy is to overlap the process of data extraction and data transfer for sub-components by non-blocking RDMA calls. However, it requires multiple RPCs to finish one data object transfer.

The evaluation between different strategies to transfer unstructured data is discussed in Section V-A. Moreover, in order to support different unstructured mesh data transfer strategies, we also extended the data model of our staging service, which is discussed in Section IV-B in detail.

B. Data model extension of the staging service

The original data object in our previous work [9] contains an object descriptor and a pointer to the raw data array. In particular, the object descriptor includes the necessary meta-information such as the variable identifier and the spatial bounding box of the raw data. This data model is efficient to represent a structured mesh, such as a Cartesian grid, but it is inconvenient to support unstructured mesh data generated by in-situ analytics. As discussed in Section IV-A, the unstructured mesh data may be decomposed into multiple sub-components that are transferred to the receiver separately; therefore, the staging service should also provide a data model containing multiple arrays with various types and lengths.

For the extension of the data model, we reuse the object descriptor as the identifier of the object. Instead of using an internal raw data array, we introduce an interface class that defines the data object's necessary virtual functions, such as data put, get and delete. The interface class contains the object descriptor with the meta-information, and it can be implemented by different data models as needed. For example, the concrete implementation can be a map that contains various data arrays in one object to support the Multi-segment data transfer strategy discussed in Section IV-A. Besides, we can extend the implementation with VTK smart pointer that contains a general *vtkDataSet*, or implement a more flexible customized data representation without modifying the staging service code in large amounts.

V. Evaluation

All experiments in this evaluation were performed on the Cori supercomputer's Haswell partition[4]. **The corresponding code[5] is publicly available**.

A. Different strategies to transfer unstructured mesh data

In this experiment, we aim to compare different strategies for transferring unstructured mesh data discussed in Section IV-A. We generate polygonal datasets with different sizes then transfer these data to the staging service using different

[3]https://mochi.readthedocs.io/en/latest/

[4]The detailed characteristics of the Cori system are described in [31].

[5]https://github.com/wangzhezhe/Gorilla/tree/master/example/gssimulation

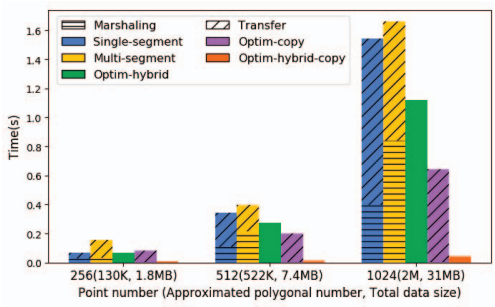


Fig. 3. Comparison between different strategies to transfer a polygonal mesh sphere. The x axis represents the point number in each horizontal and vertical line of the sphere. The values in parenthesis represent the approximated polygonal number and associated total data size, respectively.

strategies. In particular, the datasets used in the experiment are the sphere geometry created by the VTK library, and the number of polygons is controlled by the number of points in the vertical and the horizontal dimensions of the sphere. As illustrated in Figure 3, the x axis labels the polygonal data size. For example, when there are 256 points in every horizontal and vertical line, there are around 130K polygons on the sphere geometry, and the total size for this data is 1.8MB in the format of the VTK file. The y axis represents the average time spent on different stages to transfer the data object.

As Figure 3 shows, the distinctions between different strategies become obvious only when the number of polygons becomes comparatively large. If we use the Single-segment strategy as the baseline, the naive Multi-segment strategy takes more time to marshal the data, but there is a benefit for transferring data with several discontinuous segments in one RPC. The long data marshaling time is caused by using the deep copy strategy to extract and register sub-components for RDMA operation. Furthermore, we can overlap the sub-components extraction and the transfer process. For example, the Optim-hybrid method that adopts non-blocking RPC can mix the data extraction and transfer. This method shows 32.7% improvements compared with the original Multi-segment. In addition, since the *vtkPolyData* provides flexible mechanisms to access the memory addresses of its sub-components, we can get the memory address of sub-arrays directly, then transfer them with RDMA operations. With the optimization that avoids using the deep memory copy to extract the sub-arrays from the *vtkPolyData* (Optim-copy in Figure 3), the data marshaling time becomes trivial. We can save more than half of the data transfer time compared with the original Multi-segment way in total. Finally, the benefits can be accumulated if we adopt optimizations for both copy and non-blocking RPC. For example, with 1024 points, the Optim-hybrid-copy saves around 1.5s compared with the original Multi-segment method (around 0.5s comes from Optim-hybrid and around 1s comes from Optim-copy); the total data transfer time can be reduced by around 97.47 percentage.

B. Data-driven workflow with unstructured mesh processing

This experiment aims to evaluate how the staging service with unstructured mesh processing can decrease the overhead

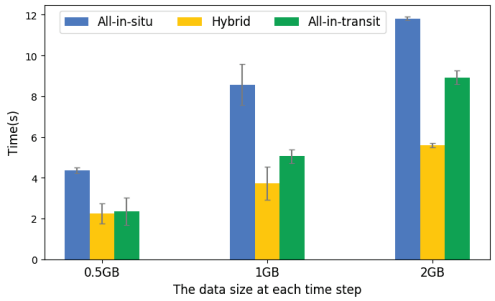


Fig. 4. Overhead of data processing in the simulation.

of a data-driven hybrid in-situ workflow. We use 32 MPI processes to run the simulation for 20 time steps, and there are 4 MPI processes for staging services (the architecture is described in [9]). Then we use the data transfer techniques and associated data model extension discussed in Section IV to execute the hybrid in-situ workflow presented in Figure 2. In addition, we also run analytics using an all-in-situ paradigm and an all-in-transit paradigm for the sake of comparison.

As illustrated in Figure 4, the x axis represents the size of the data generated at each time step, and the y axis represents the accumulated in-situ processing overhead on the simulation run time. In particular, running all analytics in-situ takes the longest time for all test scenarios. This is because the connectivity filter and subsequent blob analytics task need to aggregate all data in one process, but the simulation cannot progress during this processing. In contrast, the all-in-transit strategy shows comparatively better performance than all-in-situ in this evaluation. This is caused by the overlap between data generation and processing. Furthermore, with the hybrid strategy that contains both in-situ and in-transit tasks, we continue to decrease the overhead by transferring the unstructured mesh data extracted based on isosurfaces. For example, when 2GB of data is produced at each step, compared with the all-in-situ and all-in-transit method, the hybrid method saves up to 52.5% and 32.5% respectively.

It is worth noting that the time spent on processing the data generated by the last time step needs to be considered (less than one second in this experiment) when we calculate the critical execution path. Besides, we may also need to consider how the simulation might be impacted by the use of separate resources to run the data staging service. If it takes more time to reduce the data in-situ than it takes to transfer the original data to the staging service, the hybrid method becomes potentially disadvantageous.

VI. Conclusion and Future Work

In this work, we compared multiple strategies to transfer unstructured mesh data and process it in a hybrid in-situ workflow. Our evaluation showed the benefits and discussed the limitations of using optimizations in the hybrid in-situ processing. As future work, we plan to use system and performance metrics to identify how to split more complex analytics pipelines into in-situ and in-transit stages.

ACKNOWLEDGEMENT

The research presented in this paper was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program. This research was also supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This work was also supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

REFERENCES

[1] Manish Parashar. Addressing the petascale data challenge using in-situ analytics. In *Proceedings of the 2nd International Workshop on Petascal Data Analytics: Challenges and Opportunities*, PDAC '11, page 35–36, New York, NY, USA, 2011. Association for Computing Machinery.

[2] James Kress, Randy Michael Churchill, Scott Klasky, Mark Kim, Hank Childs, and David Pugmire. Preparing for in situ processing on upcoming leading-edge supercomputers. *Supercomputing frontiers and innovations*, 3(4), 2016.

[3] J. C. Bennett, H. Abbasi, P. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pebay, D. Thompson, H. Yu, F. Zhang, and J. Chen. 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*, pages 1–9, 2012.

[4] James Kress, Matthew Larsen, Jong Choi, Mark Kim, Matthew Wolf, Norbert Podhorszki, Scott Klasky, Hank Childs, and David Pugmire. Comparing the efficiency of in situ visualization paradigms at scale. In *International Conference on High Performance Computing*, pages 99–117. Springer, 2019.

[5] Hank Childs, Sean D Ahern, James Ahrens, Andrew C Bauer, Janine Bennett, E Wes Bethel, Peer-Timo Bremer, Eric Brugger, Joseph Cottam, Matthieu Dorier, et al. A terminology for in situ visualization and analysis systems. *The International Journal of High Performance Computing Applications*, page 1094342020935991, 2020.

[6] Shaomeng Li, Nicole Marsaglia, Christoph Garth, Jonathan Woodring, John Clyne, and Hank Childs. Data reduction techniques for simulation, visualization and data analysis. In *Computer Graphics Forum*, volume 37, pages 422–447. Wiley Online Library, 2018.

[7] Matthew Larsen, Amy Woods, Nicole Marsaglia, Ayan Biswas, Soumya Dutta, Cyrus Harrison, and Hank Childs. A flexible system for in situ triggers. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, ISAV '18, pages 1–6, New York, NY, USA, 2018. ACM.

[8] M. Dorier, R. Sisneros, L. B. Gomez, T. Peterka, L. Orf, L. Rahmani, G. Antoniu, and L. Bougé. Adaptive performance-constrained in situ visualization of atmospheric simulations. In *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 269–278, 2016.

[9] Z. Wang, P. Subedi, M. Dorier, P. E. Davis, and M. Parashar. Staging based task execution for data-driven, in-situ scientific workflows. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 209–220, 2020.

[10] Jay F Lofstead, Scott Klasky, Karsten Schwan, Norbert Podhorszki, and Chen Jin. Flexible io and integration for scientific codes through the adaptable io system (adios). In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*, pages 15–24, 2008.

[11] Ciprian Docan, Manish Parashar, and Scott Klasky. Dataspaces: an interaction and coordination framework for coupled simulation workflows. *Cluster Computing*, 15(2):163–181, 2012.

[12] Hasan Abbasi, Matthew Wolf, Greg Eisenhauer, Scott Klasky, Karsten Schwan, and Fang Zheng. Datastager: scalable data staging services for petascale applications. *Cluster Computing*, 13(3):277–290, 2010.

[13] W Schroeder, K Martin, and B Lorensen. Vtk textbook, 2006.

[14] Tom Peterka, Dmitriy Morozov, and Carolyn Phillips. High-Performance Computation of Distributed-Memory Parallel 3D Voronoi and Delaunay Tessellation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 997–1007. IEEE Press, 2014.

[15] Dmitriy Morozov and Tom Peterka. Efficient Delaunay Tessellation through K-D Tree Decomposition. In *Proceedings of SC16*. IEEE Press, 2016.

[16] M. Dreher and B. Raffin. A flexible framework for asynchronous in situ and in transit analytics for scientific simulations. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 277–286, 2014.

[17] Fang Zheng and Hongbo Zou. Flexio: Location-flexible execution of in situ data analytics for large scale scientific applications.

[18] Tu Mai Anh Do, Loıc Pottier, Stephen Thomas, Rafael Ferreira da Silva, Michel A Cuendet, Harel Weinstein, Trilce Estrada, Michela Taufer, and Ewa Deelman. A novel metric to evaluate in situ workflows.

[19] Guillaume Aupy, Brice Goglin, Valentin Honoré, and Bruno Raffin. Modeling high-throughput applications for in situ analytics. *The International Journal of High Performance Computing Applications*, 33(6):1185–1200, 2019.

[20] Tong Jin, Fan Zhang, Qian Sun, Melissa Romanus, Hoang Bui, and Manish Parashar. Towards autonomic data management for staging-based coupled scientific workflows. *Journal of Parallel and Distributed Computing*, 146:35 – 51, 2020.

[21] Tao Lu, Eric Suchyta, Dave Pugmire, Jong Choi, Scott Klasky, Qing Liu, Norbert Podhorszki, Mark Ainsworth, and Matthew Wolf. Canopus: A paradigm shift towards elastic extreme-scale data analytics on hpc storage. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 58–69. IEEE, 2017.

[22] Ayan Biswas, Soumya Dutta, Jesus Pulido, and James Ahrens. In situ data-driven adaptive sampling for large-scale simulation data summarization. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, pages 13–18, 2018.

[23] James Ahrens, Sébastien Jourdain, Patrick OLeary, John Patchett, David H Rogers, and Mark Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 424–434. IEEE, 2014.

[24] Christian Heine, Heike Leitte, Mario Hlawitschka, Federico Iuricich, Leila De Floriani, Gerik Scheuermann, Hans Hagen, and Christoph Garth. A survey of topology-based methods in visualization. In *Computer Graphics Forum*, volume 35, pages 643–667. Wiley Online Library, 2016.

[25] Peer-Timo Bremer, Gunther Weber, Julien Tierny, Valerio Pascucci, Marc Day, and John Bell. Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 17(9):1307–1324, 2010.

[26] Jonathan Woodring, James Ahrens, Timothy J Tautges, Tom Peterka, Venkatram Vishwanath, and Berk Geveci. On-demand unstructured mesh translation for reducing memory pressure during in situ analysis. In *Proceedings of the 8th International Workshop on Ultrascale Visualization*, pages 1–8, 2013.

[27] Matthew Larsen, Eric Brugger, Hank Childs, Jim Eliot, Kevin Griffin, and Cyrus Harrison. Strawman: A batch in situ visualization and analysis infrastructure for multi-physics simulation codes. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, pages 30–35, 2015.

[28] Utkarsh Ayachit, Andrew Bauer, Earl PN Duque, Greg Eisenhauer, Nicola Ferrier, Junmin Gu, Kenneth E Jansen, Burlen Loring, Zarija Lukic, Suresh Menon, et al. Performance analysis, design considerations, and applications of extreme-scale in situ infrastructures. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 921–932. IEEE, 2016.

[29] S. Habib, V. Morozov, N. Frontiere, H. Finkel, A. Pope, and K. Heitmann. Hacc: Extreme scaling and performance across diverse architectures. In *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–10, 2013.

[30] Arjen Doelman, Tasso J Kaper, and Paul A Zegeling. Pattern formation in the one-dimensional gray-scott model. *Nonlinearity*, 10(2):523, 1997.

[31] NERSC. The Cori System. https://docs.nersc.gov/systems/cori/.