Productivity Challenges for Integrating Simulation and Analysis

Ron A. Oldfield Nathan Fabian Ken Moreland David Rogers

October 8, 2013

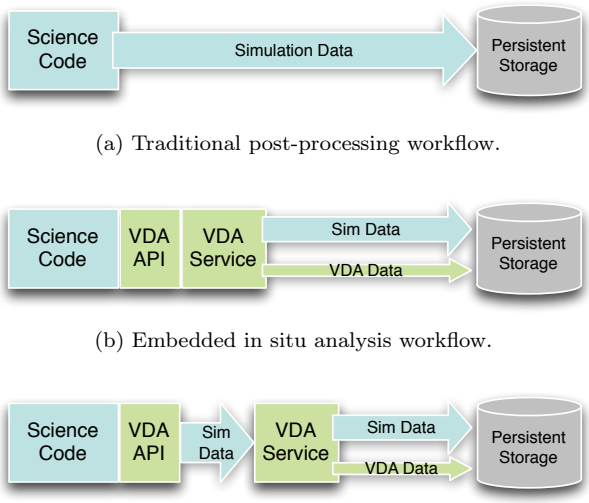
1 Introduction

I/O research over the last several decades has identified a substantial and growing gap between the rate an application can generate data and the rate the storage system can consume it [4, 5]. At Exascale, with a projected storage system rate of 60 TB/s [1], the I/O system throughput will be less than 1% of the generating capacity of a large-scale simulation. These trends will have a direct impact on the way computational scientists integrate analysis and visualization into their application workflow. The result is an evolution away from application workflows that consist of sequences of independent simulation and analysis steps that store transient results in a storage system, to integrated approaches that perform these steps concurrently.

Figure 1 shows three types of application workflows used to integrate simulation and analysis. A traditional post-processing workflow involves storing simulation results to disk and later retrieving them for visualization and data analysis. At at Exascale, post-processing approaches will not be able to capture the volume or granularity of data necessary for analysis of these extreme-scale simulations.

In situ analysis embeds analysis into the simulation code, either through code-specific operations, or by incorporating a separate library. In this case, analysis executes on the same compute resources as the simulation, making it easy to deploy and allowing for dual use (if implemented) of key application data structures. Depending on the complexity of the analysis code, however, in situ approaches may have substantial memory, computation, and communication requirements and could create stability, scalability, and resilience issues for highly-tuned production codes.

In transit analysis offloads the analysis operation onto a separate partition of compute resources, using the high-speed network rather than the file system for communication. This approach requires a thin client library to aggregate and communicate important data structures to the VDA service, effectively creating a parallel pipeline that allows overlap of simulation and analysis. Although this approach is less intrusive than in situ approaches, it requires additional compute resources and is more complicated to coordinate the allocation and execution of an application and a service.



(a) Traditional post-processing workflow.

(b) Embedded in situ analysis workflow.

(c) Service-oriented in transit workflow.

Figure 1: Traditional and emerging workflow diagrams showing the flow of information from simulation to persistent storage.

Use cases for integrated simulation and analysis are numerous, including real-time fragment detection for a shock physics code [7], a Quantum Monte Carlo (QMC) code coupled with a a service to generate *observables* in a different coordinate system, the Pixie3D magnetohydrodynamics (MHD) code coupled to PixiePlot and ParaView [**?**], and perhaps most compelling, the CASL code for simulating light water reactors that couples numerous physics and analysis codes [8].

In the remainder of this paper, we outline the key challenges associated with supporting integrated simulation and analysis in a production environment. Then we propose an R&D path forward that addresses these challenges in a way that complements existing ASCR efforts.

2 Productivity Challenges

Although a number of ad hoc approaches exist for integrating simulation and analysis [9, 3, 7], there are a number of challenges that hinder their use as an effective solution in a production HPC environment. The following list identifies a few of these challenges.

Inadequate Runtime Environment: One of the fundamental challenges that hinder the potential value of application workflows is the runtime environment. For decades, capability-class systems have used a batch-scheduled model for execution that does not allow dynamic expansion or contraction of an allocation, sharing a node between executables, control over topological placement, or connecting to existing jobs – all features that could be utilized for integrating analysis and simulation.

Incompatible Configuration Environment: Another productivity issue hindering integrated simulation and analysis is the complexity of the build and configure environment. Since it is often the case that the analysis and simulation come from two different code basese, there are often mismatches in configuration tools (e.g., CMake, autotools), required versions of third-party libraries (TPLs), supported compilers, and conflicts in compiler optimizations.

Mechanisms for Sharing Data: Sharing data structures between the simulation and analysis code is another challenge for integration. If the simulation treats analysis as a "black box" there must be a duplicate copy of the data being analyzed. For in situ methods, this doubles the required memory. Even for in transit, there is a substantial memory burden to serialize data into a buffer before transport to the analysis service.

Portable Inter-Application Communication: Every HPC system has an implementation of MPI for intra-application communication, but there are no portable interfaces for the type of intra-application communication that is required for in transit analysis. Researchers generally implement their own API (e.g., Nessie [6], Mercury [10]) on top of the proprietary APIs like uGNI (Cray), DCMF (IBM), or InfiniBand. A portable, standard, API would accelerate and broaden the use of in transit approaches.

3 Addressing Challenges

Recently awarded ASCR projects such as Hobbes and ARGO are directly addressing some of the OS and runtime issues identified in the previous section; however, the goal of those projects is to develop runtimes that support application composition. There is still a need for complimentary projects that push the boundaries of the application-composition model by developing libraries and applications that leverage expansion/contraction for dynamic load balancing, inter-node interfaces for data sharing, and intra-application communication interfaces that allow for RPC, pipeline, and event-based programming models. While developing the supporting OS/R is in the scope of the existing ASCR projects, the tools to evaluate and enhance the productivity of these tools is not.

Although there has been some good progress on meta-configuration tools like the TriBITS tool used for Trilinos [2], there is still much work to do to simplify this process and make it a widely used and productive tool. For example, modularization to configure and build precisely what is required by the simulation or analysis code is still a challenge. Given the complexity of some of the analysis workflows and code-coupling projects, there is a clear benefit to using an in transit approach where components are connected through a network, not a library. This allows the independent configuration of the workflow components, as long as the networking protocol for connecting the executables is supported by both.

Finally, to address the data-sharing issue, there needs to be a concerted effort to use a common set of parallel data structures useful for both analysis and simulation codes. In cases where there is a near exact match in the data layout, this will enable direct use of data structures within the simulation, saving memory and performance. In cases where there is not an exact match, the data structure descriptions will be sufficient to enable minimal-copy adaptors to convert from one structure to the other. Both approaches will greatly simplify the integration of in situ and in transit approaches. In addition to addressing commonalities in data-structure design, there needs to be mechanisms for perhaps versioning data structures to enable the asynchronous execution of in situ analysis and simulation computation. We encourage an exploration of methods that minimize duplication of data, e.g., copy-on-write.

References

[1] S. Ahern, A. Shoshani, K.-L. Ma, et al. Scientific discovery at the exascale. Report from the DOE ASCR 2011 Workshop on Exascale Data Management, Analysis, and Visualization, Feb. 2011.

[2] R. A. Bartlett, M. A. Heroux, and J. M. Willenbring. TriBITS lifecycle model. Technical Report SAND2012-0561, Sandia National Laboratories, February 2012.

[3] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. W. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pébay, D. Thompson, H. Yu, F. Zhang, and J. Chen. Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In *Proceedings of SC2012: High Performance Networking and Computing*, Salt Lake City, UT, Nov. 2012. ACM Press.

[4] G. A. Gibson, J. S. Vitter, and J. Wilkes. Strategic directions in storage I/O issues in large-scale computing. *ACM Computing Surveys*, 28(4):779–793, Dec. 1996.

[5] J. L. Hennessy and D. A. Patterson. *Computer architecture (2nd ed.): a quantitative approach.* Morgan Kaufmann Publishers Inc., 1996.

[6] J. Lofstead, R. Oldfield, T. Kordenbrock, and C. Reiss. Extending scalability of collective I/O through Nessie and staging. In *Proceedings of the 6th Parallel Data Storage Workshop*, PDSW '11, pages 7–12, Seattle, WA, Nov. 2011.

[7] K. Moreland, R. Oldfield, P. Marion, S. Joudain, N. Podhorszki, V. Vishwanath, N. Fabian, C. Docan, M. Parashar, M. Hereld, M. E. Papka, and S. Klasky. Examples of in transit visualization. In *Proceedings of the 2nd International Workshop on Petascale Data Analytics: Challenges and Opportunities*, PDAC '11, pages 1–6, Seattle, WA, Nov. 2011.

[8] R. Pawlowski, R. Bartlett, N. Belcourt, R. Hooper, and R. Schmidt. A theory manual for multiphysics code coupling in LIME. Technical Report SAND2011-2195, Sandia National Laboratories, March 2011.

[9] D. Rogers, K. Moreland, R. Oldfield, and N. Fabian. Data co-processing for extreme scale analysis. Technical Report SAND2013-1122, Sandia National Laboratories, March 2013. Level II ASC Milestone 4745.

[10] J. Soumagne, D. Kimpe, J. Zounmevo, M. Chaarawi, Q. Koziol, A. Afsahi, and R. Ross. Mercury: Enabling remote procedure call for high-performance computing. In *Proceedings of the IEEE International Conference on Cluster Computing*, Indianapolis, Indiana, Sept. 2013.