In Situ Workflow Management for Extreme-scale Turbulent Flow Simulation

John Wu, Lawrence Berkeley National Laboratory

High-fidelity computer simulations of turbulent flows are widely used in studying important applications such as combustion, wind power, nuclear power and fusion tokomak. These simulations produce massive amounts of data that need to be thoroughly analyzed. So far, the majority of these data records are stored in data files on rotating disks to be analyzed after the simulations are complete. However, a couple of recent developments have made such post-processing approach impractical on the expected extreme-scale computers. First, the increase in computing power has far outpaced the increase in the speed of disk storage systems; therefore, we are able to produce data much faster than could be stored or retrieved from disks. At the same time, the simulations have become more realistic, which typically produce more data than the previous generations of simulations. An emerging strategy is to analyze the data to extract the desired information while the necessary data is available in memory, an approach generally known as *in situ processing*. Next, we briefly describe two use cases involving turbulent flow simulations and then provide a likely scenario on the expected extreme-scale computer where more complete in situ processing is needed.

One common tool to study large turbulent flow is to turn the large volume of data into movies; therefore a common in situ operation is to generate visualization of the turbulent flow [7]. Early implementation of such visualization often have hard-coded visualization algorithm in the simulation programs. The recent development of tools such as ADIOS [5] allows the simulation program to be separated from the visualization tools, which increase the flexibility, for example, allowing different types of visualization algorithms to be used and allowing different set of computing resources to be used for movie making so that the simulation software could proceed without waiting for the visualization operations. This flexibility offered by these new in situ processing systems also makes it possible for additional operations to be performed while data is in memory [2, 6].

Another common use of in situ processing is to extract features from data, so that the important characteristics of the data are preserved while a relatively small amount of data records are stored permanently. These features are typically application dependent. For example, from an atmosphere simulation, one might extract storms; from ocean simulations, one might extract eddies; from a fusion plasma simulation, one might extract blobs. This type of feature extract could be extreme useful for industrial uses of turbulent flow simulations, where the features to be extracted are much more well understood than in a typical scientific exploration where the features to be extracted could be ill-defined or depending on the new insight gleaned from the recent analysis of the data.

The in situ analysis could also be used to generate auxiliary data that could speed up the common data analysis operations. For example, database indexes could significantly accelerate many types of data access operations, however, generating such indexes could be time consuming because they require all data records to be read into memory in order to build an index. Generating such an index and storing it on disk along with the original data will ensure the indexes are available together with data, therefore makes it more convenient to access the large data files [4]. In situ process could also be used to compress the data [3], compute statistics about the data, and derive useful byproducts.

In short, in situ data processing is a power strategy to mitigate the ever-widening gap between the compute power and storage speed. We anticipate it will be widely used in large-scale turbulent flow simulations. As the in situ processing systems evolve in the next decade or so, we foresee that the in situ process will become more common, which effectively turn the parallel computing paradigm from the current single-program-multiple-data (SIMD) paradigm of parallel computing will evolve into multiple-program-multiple-data (MIMD).

Currently, a majority of the large simulation codes are SIMD programs. The use of in situ processing allows a second program to be used with the main simulation code. In the next few years, we expect more analysis programs to be used with the main simulation code, which make whole parallel job into a MIMD program.

Another reason for our prediction is that the upcoming computing hardware will be more heterogeneous. The extreme scale computer is likely to be composed of a large number of CPU cores with likely even more accelerators; and the code to be run on the CPUs and the accelerators are likely to be different. Therefore, a user program might have to have multiple components for the variety of the computing elements in the system.

To effectively manage the multiple programs, we anticipate the in situ data processing engine to turn into distributed workflow management systems that will take charge of the data management and task management on the extreme scale computers. This workflow management system will handle data management tasks and task scheduling tasks. Among these tasks, we next describe two that might require more attention from the research community than the others: load balancing and code coupling.

As simulations become more complex, one strategy to capture more physics and more fluid dynamics is to use adaptive meshes [1]. Because these mesh structures are dynamically generated, the amount of work could change dramatically if the work is statically assigned. The work assignment will have to be dynamically adjusted. In the upcoming computers, we expect the task of dynamic distributing the work to become more complex because of the more complex configuration of both memory system and the compute system. The existing load balancing generally assume the cost of the dividing the work to be captured by the graph partitioning problem, which assume the cost of data movement to be the same from one processor to another. However, in the anticipated exascale systems, the cost of moving one byte of data from one place to another is likely to vary dramatically based on the characteristics of the memory involved. For example, the data transfer from a GPU to its host CPU is like to be very different from going from one CPU to another CPU, or from fast memory to slow memory. New methods are needed to provide load balancing on the new computer platforms.

A MIMD program will like to have multiple components dealing with different physical processes or different type of domains of a simulation. In any case, it is important for the information to be able to transfer from one component to another. Current generation of the in situ processing systems primarily transfer data from the main simulation program to the in situ analysis procedures. We anticipate the data movement patent to become more complex, with more bidirectional data flows among the various components of a MIMD program. The coupling of these components clearly require rigorous mathematics to ensure correctness, however, this type of math support is likely to be problem dependent and provided by simulation libraries such as Chombo or BoxLib. The workflow engine will be primarily be responsible for data management tasks such as moving the data among the sources and destinations, and cleaning up the memory after the data objects are no longer needed.

[1] M. J. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics," *Journal of computational Physics,* vol. 82, pp. 64-84, 1989.

[2] J. H. Chen, A. Choudhary, B. De Supinski, M. DeVries, E. Hawkes, S. Klasky, W. Liao, K. Ma, J. Mellor-Crummey, and N. Podhorszki, "Terascale direct numerical simulations of turbulent combustion using S3D," *Computational Science & Discovery,* vol. 2, p. 015001, 2009.

[3] J. Jenkins, I. Arkatkar, S. Lakshminarasimhan, D. A. Boyuka II, E. R. Schendel, N. Shah, S. Ethier, C.-S. Chang, J. Chen, and H. Kolla, "ALACRITY: Analytics-driven lossless data compression for rapid in-situ indexing, storing, and querying," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems X*: Springer, 2013, pp. 95-114.

[4] J. Kim, H. Abbasi, L. Chacón, C. Docan, S. Klasky, Q. Liu, N. Podhorszki, A. Shoshani, and K. Wu, "Parallel In Situ Indexing for Data-intensive Computing," in *LDAV*, 2011, pp. 65-72.

[5] G. F. Lofstead, Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J. Y. Choi, S. Klasky, R. Tchoua, R. A. Oldfield, and others, "Hello ADIOS: The Challenges and Lessons of Developing Leadership Class I/O Frameworks.," Sandia National Laboratories2012.

[6] L. Wu, K. Wu, A. Sim, and A. Stathopoulos, "Real-Time Outlier Detection Algorithm for Finding Blob-Filaments in Plasma," in *SC14*, 2014.

[7] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma, "In situ visualization for large-scale combustion simulations," *IEEE Computer Graphics and Applications,* pp. 45-57, 2010.