Stream Processing for Remote Collaborative Data Analysis

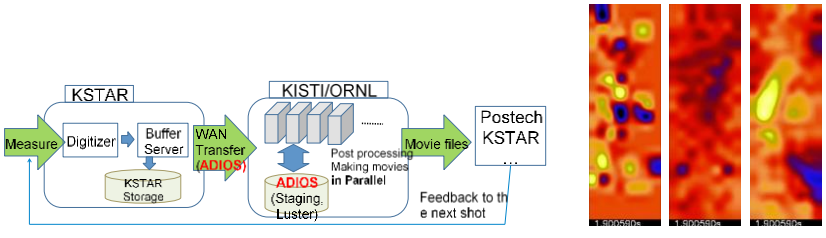
Scott Klasky[146], C. S. Chang[2], Jong Choi[1], Michael Churchill[2], Tahsin Kurc[51], Manish Parashar[3], Alex Sim[7], Matthew Wolf[14], John Wu[7] [1] ORNL, [2] PPPL, [3] Rutgers, [4]GT, [5] SBU, [6]UTK, [7] LBNL

Due to the recent advances in technologies, large-scale science projects – such as, International Thermonuclear Experimental Reactor (ITER), Korea Superconducting Tokamak Advanced Research (KSTAR), and National Spherical Torus Experiment (NSTX) – are generating or going to soon generate huge amounts of data. At the same time, such projects are drawing large international collaborations with scientists from all over the world. In all these experiments, a team of scientists have to be present at the facilities to monitor the progress of the on-going data collection, adjust the control settings, and prevent catastrophic events; while most others access the data remotely. Allowing these remote users to conduct their analysis operations in real time as the data is being collected would enable more scientists to provide feedback to the execution of the shared experiments and increase scientific output. Often there is a strong desire for (near) real-time participation at running the experiments, however there are significant roadblocks to such a remote participation. For example, the computer network might not be fast enough to transfer a sufficient amount of data for a meaningful analysis, the analysis operations might take too long to provide timely feedback from the local resources, or there might not be proper software framework to compose the analysis procedures to conduct the necessary analyses and provide timely feedback. At the present time, the wide-area computer network is fast enough to transmit many gigabytes for a second, and the application scientists often have a variety of data analysis algorithms. What is lacking is a software system that allows scientists to quickly and conveniently compose complex analysis tasks, manage the necessary data movement, execute the specified tasks, and provide timely feedback to the users.

In building such a near real-time system, we need to address a number of critical challenges. First, to reduce the data access latency, we should avoid reading and writing disks. We designed a system to process, analyze, summarize and reduce the data before data reaches to the relatively slow disk storage system, through a process known as in transit processing. Even though the network speed has been steadily increasing over the years, it is still relatively slow compared to the CPUs consuming data, and this trend is only getting worse. We believe it is essential to transfer the minimal amount of necessary data for an analysis operation, and then only work on this sub-chunk of data. The approach we believe the community must take is to integrate indexing data structures to minimize the amount of raw data accessed by the analysis operations, and have this sparse data representation worked on by the analysis and visualization tasks.

ECEI Analysis on KSTAR: The Korea Superconducting Tokamak Advanced Research (KSTAR) is conducting nuclear fusion experiments, which is an important precursor to the ITER project. Currently, it is one of the few long pulse fusion devices, with pulses up to 10s of seconds and targeting up to 300 seconds (expecting pulse lengths of 300-500 seconds in ITER).

Among many diagnostics captured during KSTAR experiments, we focused on a workflow to diagnose high-speed image datasets (0.5 million images per second), called Electron Cyclotron Emission Images (ECEI) [5], which need to be analyzed on two geographically distant locations. The requirement is to connect the ECEI acquisition server in KSTAR to a remote repository, located in Korea Institute of Science and Technology Information (KISTI), and again to another remote analysis site, Postech (120 miles apart from KSTAR) for on-line (or near real-time) processing (fig. 1).



a) b)

Figure 1. KSTAR workflow for Electron Cyclotron Emission Images (ECEI) diagnosis; (a) remote workflow execution diagram for processing ECEI data, and (b) an example output.

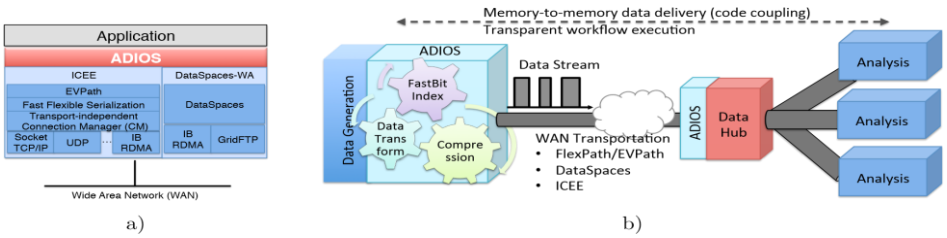


Figure 2. (a) ICEE software stack, and (b) Remote processing as a part of the ADIOS ecosystem.

Method

To address the challenges discussed above, we developed a system, named ICEE, to support remote data stream processing over WAN environments. One of our key design decisions is that we develop ICEE as an extension of our high-performance IO solution, called Adaptive IO System (ADIOS) [4], in order to take full advantage of high-performance I/O functions provided by ADIOS and support additional functions for remote processing on top of it. In short, ICEE is designed to provide flexible I/O services to scientific applications for both local and remote processing. The key advantage of our solution is that data is staged from a data generator (or a group of MPI processes in some cases) and the data can be moved by either files (using gridFTP, etc.) or via socket connections, or RDMA connections. The services (executables) can then run on the same machine (we have implemented this on the DOE LCFs and NERSC) or on different machines, and they can even run on the same node (using different cores, or even time share the cores) or different nodes.

ICEE provides two key features; i) enabling data stream processing and ii) remote communication. First, ICEE provides routines enabling data stream analysis in favor of on-line chunk-wise data processing over batch processing which conventionally used for analyzing data at rest. Secondly, ICEE integrates network functions to send and receive data streams over wide area networks (WANs) by connecting geographically distributed remote processes in a time-critical fashion. To support various cutting edge network environments, ICEE is based on flexible network library, called EVPath [3], under the hood (fig. 2a). With the tight integration with ADIOS, the network operations provided by ICEE is transparent for users in a sense that users can switch between file IO (for storing data in a local area) and network IO (to stream out data) without rewriting applications. Wide-area data steaming was also supported using RDMA by DataSpaces [1], which also provides high-level abstractions for data coupling and scientific workflows. DataSpaces is a part of the ADIOS ecosystem.

In the following, we describe the two main components of ICEE in details; i) stream interface and ii) connection module.

Stream interface: ICEE is an extension of ADIOS, which has been developed to support data-intensive scientific applications suffering from the I/O bottleneck problem. In a recent update, ADIOS added a set of APIs for stream data processing [6]. The newly added stream mode is specifically intended for memory-to-memory data movement between data sources and online analytics, and maintaining the new interface compatible with file I/O in that it can be switched to file mode without code changes. With stream APIs, users can handle streams of data, i.e., a series of data written in multiple times.

More detailed steps on ADIOS/ICEE stream data processing are as follows. A data stream writer creates a unique stream name, and a stream receiver opens the stream with the same name. This will establish a network connection between the writer and the reader via user-specified network transportation method as a parameter. Then, stream writer periodically writes data, and the data is delivered to the reader as it continues working. The receiver receives the data streams in a background process, and is able to read data when ready directly from the memory.

Connection module: To connect geographically remote processes to communicate in timecritical fashion, the choice of efficient connection methods is important. In addition, we should take into account of various network environments to support non-uniform collaborations, as observed in our motivational applications. To support such various environments and provide cutting-edge high-performance network functions, we use an event-driven messaging library, called EVPath [2, 3]. One of the biggest advantages of EVPath is that it provides various high-end network transportation methods, such as Infiniband RDMA, TCP/IP, and UDP.

References

1. Ciprian Docan, Manish Parashar, and Scott Klasky. Dataspaces: an interaction and coordination framework for coupled simulation workflows. *Cluster Computing*, 15(2):163–181, 2012.
2. Greg Eisenhauer, Karsten Schwan, and Fabi´an E Bustamante. Publish-subscribe for highperformance computing. *Internet Computing, IEEE*, 10(1):40–47, 2006.
3. Greg Eisenhauer, Matthew Wolf, Hasan Abbasi, and Karsten Schwan. Event-based systems: opportunities and challenges at exascale. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, page 2. ACM, 2009.
4. Qing Liu, Jeremy Logan, Yuan Tian, Hasan Abbasi, Norbert Podhorszki, Jong Youl Choi,Scott Klasky, Roselyne Tchoua, Jay Lofstead, Ron Oldfield, et al. Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks. *Concurrency and Computation: Practice and Experience*, 26(7):1453–1473, 2014.
5. GS Yun, W Lee, MJ Choi, J Lee, HK Park, B Tobias, CW Domier, NC Luhmann Jr, AJH Donn´e, JH Lee, et al. Two-dimensional visualization of growth and burst of the edge-localized filaments in kstar h-mode plasmas. *Physical review letters*, 107(4):045004, 2011.
6. Fang Zheng, Hongbo Zou, Greg Eisenhauer, Karsten Schwan, Matthew Wolf, Jai Dayal,Tuan-Anh Nguyen, Jianting Cao, Hasan Abbasi, Scott Klasky, et al. Flexio: I/o middleware for location-flexible scientific data analytics. In Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, pages 320–331. IEEE, 2013.