ICEE: Wide-area In Transit Data Processing Framework For Near Real-Time Scientific Applications

Jong Y. Choi*, Kesheng Wu†, Jacky C. Wu†, Alex Sim†,
Qing G. Liu*, Matthew Wolf‡, CS Chang§, Scott Klasky*

* Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA
† Lawrence Berkeley National Laboratory, Berkeley, California, USA
‡ Georgia Institute of Technology, Atlanta, Georgia, USA
§ Princeton Plasma Physics Laboratory, Princeton, New Jersey, USA

***Abstract*—Modern science often requires a large group of scientists to pool their resources together to make progress. These collaborations produce huge amounts of data that needs to be shared across geographically distant locations. In some of these cases, immediate feedback is needed to control a central experimental facility, while the collaborators are oceans apart. To enable this type of distributed collaboration, we explore the design options for building an in transit data analysis framework over wide-area network. We implement this framework based on Adaptive I/O System, ADIOS, and demonstrate the effectiveness of our implementation with a set of diagnostic data from the KSTAR experiment. Our tests show that we can complete a large range of tasks in less than a second over a wide-area network spanning across the US. To best utilize the compute resources and meet the time constraints for the system, we also carefully study the performance characteristics of the key components of the ICEE system.**

I. Introduction

With increased automation and faster data acquisition systems, large-scale scientific experiments are generating ever increasing amount of data. At the same time, more and more science projects, such as, the Large Hadron Collider (LHC) and International Thermonuclear Experimental Reactor (ITER), are international collaborations with many thousands of scientists from all continents of the world. In all these experiments, a team of scientists have to be present to monitor the progress of the on-going data collection, adjust the control settings, and prevent catastrophic events. Allowing some of these real-time monitoring and data analysis tasks to be performed remotely could significantly reduce the burden of travel and encourage more scientists to provide valuable input. We believe that the networking and data analysis technologies are fast enough for us to design and construct a real-time remote data processing framework over wide-area networks.

There are a number of challenges in building a remote data analysis framework for a large experimental facility. Large experiments nowadays can easily generate a few tens of terabytes a day or petabytes a year. The datasets often contain a number of different but related measurements. The data records may be generated at different time and with different intervals, which makes it difficult to correlate the measurements. Furthermore, a typical analysis task may involve different types of measurements. In this work, we address the challenges of minimizing the amount of data retrieved across wide-area network and reduce the amount of time spent on input and output to disk storage systems by conducting as much of the computation in memory as possible.

The research presented in this paper is originally motivated by the collaborative fusion research conducted at Korea Superconducting Tokamak Advanced Research (KSTAR) facility. KSTAR is a magnetic confinement fusion device operates in pulses, known as *shots*. During a shot, many diagnostic instruments collect gigabytes of information that needs to be stored, analyzed and shared among scientists [1]. As in most pulse based experiments, there are three types of analyses: *in-shot-analysis*, *between-shot-analysis* and *post-run-analysis*. A shot in KSTAR can last from a few seconds to 300 seconds for a full-mode production. Due to the short duration of each shot, the analyses performed during a shot are aimed to provide safety critical functions such as monitoring the health of an experiment to make sure there is no instability occurred. If an instability is observed, the experiment needs to be aborted in milliseconds to avoid damage to the experimental facility. By and large, the challenges come from the amount of data produced in a long pulse (typically a few terabytes) and the near real time requirement. Between two consecutive shots, there is also a relatively longer period of time for *between-shot-analysis* (See Fig. 1). KSTAR is planning to provide such large-scale experimental data in near real-time to external collaborators remotely located in other countries, including research labs in USA.

A significant amount of work have been done to address post-run-analysis challenges. In this work we, instead, focus on the more challenging first two cases in KSTAR data analyses due to their stringent QoS requirements, and present a system to support on-line stream-based wide-area data analysis. Our system, however, is not limited to KSTAR only and can be generalized to many remote collaboration cases where near real-time reponse is a must. More specifically, we consider a system where large-scale data is generated by experiments (e.g., KSTAR) or numeric simulations in one site and consumed by analytics in near real-time, running in multiple remote sites collaboratively (See Fig. 2).

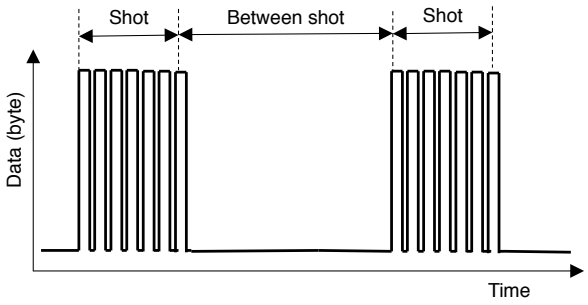


Fig. 1: An example of shots and data streams in fusion experiments. Three types of analyses can happen: in-shot-analysis, between- shot-analysis and post-run-analysis.

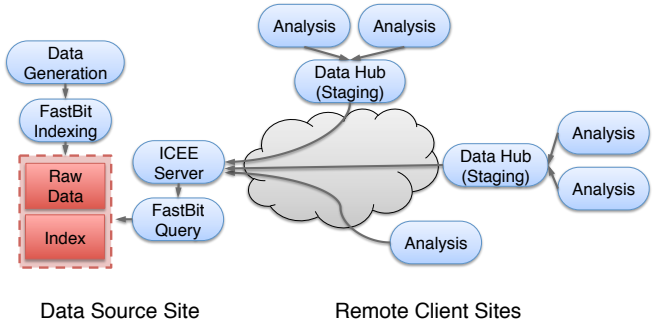


Fig. 2: An overview of ICEE system.

In building such a near real-time system, we need to first address two challenges. First, disk I/O needs to be avoided as much as possible. A common pratice to transfer data across a wide-area network is to transfer them as files. Namely, simulations or experiments write data to a local storage system which are subsequently read and transfered to remote locations by file transfer tools (such as scp, GridFTP, etc). On the remote side, the received data is then saved as files and then consumed by analytic applications. This naive solution involves four rounds of I/O operations and can easily result in poor scalability and data readiness. In contrast, this work designed a solution to process, analyze, summarize and reduce the data before it reaches the relatively slow disk storage system, a process known as in transit processing (or in-flight analysis). In our previous work, known as ADIOS [2], we developed various solutions supporting such scenarios and observed much improved performance. We leverage our IO solution, ADIOS, in order to increase the data handling capability for collaborative scientific applications [3].

Secondly, we need to keep in mind that data can go onto long-haul wide-area networks, whose bandwidth is relatively limited and unpredictable. As a result, it is imperative to minimize the amount of data being moved and improve the efficiency of collaborative data processing. The approach we take is to integrate indexing data structures, which is often much more smaller than raw data, with primary data and use them to locate target data records quickly when must. For example, a workflow that computes the sizes of magnetic islands might take input data from a high-resolution imaging device, however only certain pixels from an image would be part of the magnetic islands. Filtering the data records before they are transferred will reduce the amount of data moved and therefore improve the overall system response time. An indexing structure could be used to perform this filtering operation efficiently. Furthermore, a set of data analysis algorithms depending on sparse matrix operations can benefit from indexing data in preparing sparse input from a large volume of dense data. For this end, we employed a cutting-edge indexing solution, FastBit, in our system [4].

However, index building is an expensive operation and can be inappropriate in a certain scenario. Then, various questions may be asked: When should we index? Is it beneficial if we transfer index to a remote site, instead of keeping in a server? We will address such questions in this paper.

In short, in this paper, we propose a solution based on two key technologies: I) wide-area in transit (or in-flight) data processing to avoid expensive file I/Os and II) indexing and querying to search data efficiently and provide efficient network data transfer by moving partial data selectively. More specifically, our system, named ICEE (also our project name in collaboration with KSTAR[1]), is designed to support wide-area in transit data processing for stream-based scientific applications. Our system consists of the three major features, as follows:

- Stream data processing – supports stream-based IO to process pulse data;
- In transit processing – provides remote memory-to-memory mapping between data source (data generator) and client (data consumer);
- Indexing and querying – FastBit.

In the following sections, we discuss background and related work and present our ICEE system in details in Section III. The evaluation of our system and experimental results will be shown in Section IV, followed by our conclusion in Section V.

II. Background and Related Work

We start with introducing our driver application in fusion science and discuss related work.

A. KSTAR fusion experiment

The KSTAR (Korea Superconducting Tokamak Advanced Research) studies nuclear fusion to develop next-generation energy sources[2]. It was the first fully superconducting fusion tokamak in the world. The design and experimental results of the KSTAR reactor will serve as a precursor to the ITER (International Thermonuclear Experimental Reactor) project in building next-generation superconducting tokamak in Cadarache, France. Currently, it is the fusion device with the longest pulse, up to 300 seconds long [1]. This long-duration pulse gives scientists more opportunity to study the fusion plasma and also creates significant data management challenges.

[1]International Collaboration Framework for Extreme Scale Experiments (ICEE), https://sdm.lbl.gov/icee/

[2]More information about KSTAR can be found at http://kstar.nfri.re.kr/.

During the 300-second pulse period, there are always chances for the tokamak plasma to go into dangerous disruption events, which could exert excessive stress on the tokamak structure and/or damage the plasma-facing wall. In order to minimize such a disruption event, a couple dozen diagnostics systems measuring plasma rotation breaking, magnetic island growth in the core, resistive wall mode activity, decline of neutron emissivity from predicted value, and so on, need to work together for monitoring of the precursor phenomena. When an imminent disruption event is predicted to be probable, an automated preventive action is triggered within milliseconds. Collapse of the edge plasma by the so-called, "edge localized mode instability", is another danger that must be suppressed by an early detection using several diagnostics systems, such as microwave imaging or fast visible camera. Besides the predictor-preventer purposes, a real-time data analysis accompanied with real-time experimental control could maximize the multi-million dollar experimental pulse. Furthermore, post-processing data analysis and physics research can be used for the next experimental pulse. For this purpose, large amount of streaming data will have to be distributed quickly and analyzed efficiently by remote scientists.

The particular test data used in our study came from SXR (called Soft X-ray Array), as a part of DAQ (Data Acquisition Systems) in KSTAR. The SXR has 64 channels running with 8 digitizers (max 500KHz) [5], [6].

B. ADIOS

ADIOS has been developed to provide an efficient I/O solution in many large-scale scientific applications, such as combustion simulation (S3D), climate modeling (GEOS-5), Gyrokinetic Toroidal Code (GTC), plasma fusion simulation code (XGC), etc. ADIOS is a modular system that provides I/O abstractions and services that allow scientists to build their scientific applications as a series of services, or modules [2].

The ADIOS API is used by applications to interface to the services as well as write and read large volumes of data very efficiently on tens of thousands of compute cores. It implements several high performance I/O modules and allows an application developer to employ the most appropriate I/O method for his/her application. In addition, applications can make use of data staging and in situ and in transit processing capabilities to filter and transform data in order to reduce network and staging overheads.

C. FastBit

FastBit is a software package for indexing and querying large scientific data sets [4]. It contains a number of state-of-art bitmap indexing methods [7], [8]. These bitmap indexing techniques are well-suited for typical scientific applications where the data is read-only and the queries typically touch upon numerous data records. For example, a typical query on a set of data from space weather modeling might be to find all high-energy particles within a region in space, which could include millions of particles [9]. When analyzing a combustion simulation, a typical query might be to find regions of ignition, which might include millions of mesh points [10]. In these cases, other commonly used indexing techniques such as B-trees or hashing are much slower than bitmap indexes. To work with large scientific data sets, we have also developed a way to parallelize the indexing and querying functions [11], [12].

III. ICEE System

To address the challenges mentioned above, we designed a system, named ICEE, to support wide-area scientific data processing. It has two main features: I) efficient data access capability by supporting in transit (or in-flight) data analysis with ADIOS, and II) data reduction using indexing and querying by FastBit. In the following, we describe our prototype system, ICEE, in details.

A. Architecture

Our system consists of three main components: data acquisition, data server, called ICEE server, and remote clients connecting through wide-area networks. We provide programming interfaces (APIs) for clients to query data hosted by an ICEE server (See Fig. 2).

Data Acquisition: This is an interface between data sensors, which generate raw experimental data, and ICEE server, and manages the raw data in memory for the ICEE server.

The user data can be either pushed to or pulled by the ICEE system. Data may be pushed to ICEE for common activities, such as indexing commonly used variables, which requires a significant amount of computer time. It is important to prepare the indexes before they are needed. Commonly used indexes will typically be built as soon as the data is available, while indexes on infrequently used variables may be specifically requested by a user.

ICEE server: The ICEE server is in charge of providing data to the remote users. It supports common data related activities, such as indexing and processing queries.

In order to minimize disk I/O overheads, operations are performed in memory as much as possible; i.e., index and raw data will be residing mainly in memory and provide as a data stream without accessing files. Data in memory is separated by logical time steps (or called as shots in fusion experiments) but shares similar structures between time steps. This is a natural format of many scientific data, including fusion. Our initial prototype has only a single server, but we plan to expand the number of servers to provide more computing power for the widely distributed user community.

Remote clients: Clients will be distributed over wide-area networks and access remote data by connecting to an ICEE server directly or via data hub which will be delegated by multiple clients. Data hub can act as an aggregator for local clients or parallel data analysis by providing in-memory staging services. Details are discussed in our previous work [13].

Clients can request data with queries to access a portion of data selectively. Server will exploit index data to extract a portion of satisfactory data set.

All communications between clients and servers are a form of Remote Procedure Calls (RPCs), provided by a set of APIs we developed. APIs are based on ADIOS and EVPath, which will be discussed in details in the next.

We develop a new connection module (or plug-in) for ICEE system to help a ICEE server and remote clients be

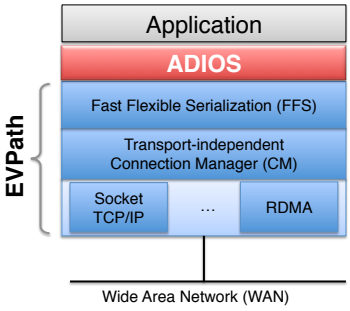


Fig. 3: Software stack of the ICEE system.

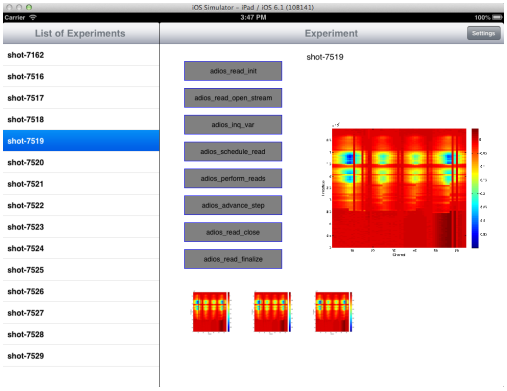


Fig. 4: Remote monitoring application on a table device, showing results from the analysis workflow on ICEE system.

connected easily and exchange data. Our module is integrated with ADIOS and thus shares the same ADIOS APIs (or same interfaces) but supports wide-area data exchanges under the hood. All operations are transparent to users.

B. *Software Components*

Our ICEE system includes the following main components.

In transit interface: ADIOS has been developed to support data-intensive scientific applications which suffer from I/O bottleneck problems, by providing a flexible and efficient I/O solution optimized in various HPC environments. One of the most attractive ADIOS features is its adaptability in which users can choose a platform-specific or problem-specific I/O modules (or called plug-ins) without re-writing applications and, instead, simply specify parameters in a configurable input. In other word, ADIOS provides an uniform I/O interface but help users choose an optimized or problem-specific I/O solution transparently.

We extended our I/O solution, ADIOS, by developing a new add-on interface, specially designed for our ICEE project to support transparent connections between a data server (or provider) and a client (or consumer) through wide-area networks.

By the virtue of ADIOS's uniform interface design, an application can choose to read data from files in a local disk or streams delivered through wide area networks. Switching between local files and stream data is transparent to users. This will greatly help developing process. One can start developing and debugging an application by using locally saved files first and easily convert the application to handle streaming data from remote servers, which is ideal for our ICEE project.

Connection module: To connect geographically remote processes to communicate in time-critical fashion, the choice of efficient connection methods is important. In addition, we should take into account of heterogeneous network environments to support various types of collaborations. Especially, motivated by the KSTAR application, we assume clients (data consumer) are located over Internet in different places and operated in non-uniform computing and network environments. To support such various environments and provide cutting-edge high-performance network functions, we use EVPath [14], [15].

EVPath, which is a library to build event overlay networks, consists of a set of core sub-libraries; CM for connection manager, FFS [16] for fast flexible data serialization, and CoD (C-on-Demand) for data filtering and transformation. Fig. 3 shows the software stack of ADIOS and EVPath.

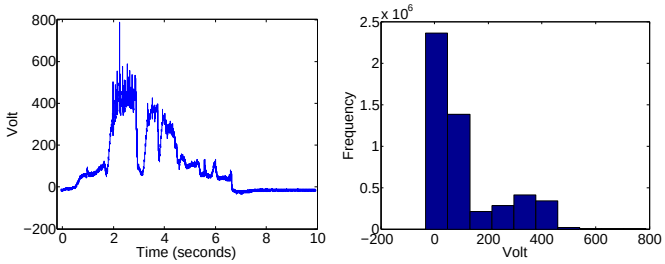
Mobile clients: The workflow and data analysis on ICEE system can be monitored and controlled from portable devices. The portability and computing power of modern mobile devices is increasing [17], [18]. These mobile devices are finding their way in sciences as diverse as space exploration [19] and epidemiology [20]. They are moving into our daily activities [21]. Currently, due to their technical limitations, even in the mobile friendly science applications, they are primarily used for the social networking services (SNS) and system status reporting [22]. We see a potential for mobile computers to replace more functionality of laptops and workstations [23], and producing a mobile extension of the science workspace [24]. In this work, we explore the option of supporting tablet computers as clients of the ICEE system. As the first step, an application is prototyped to monitor the ICEE workflow and data analysis remotely, and access the results from the data analysis on a tablet device, as shown in Fig. 4. Mobile tablet computing support in ICEE system opens up a new horizon in collaborative scientific computing environments, and will enable researchers to access quickly and easily to the workflows and the results, like on their workstations, and to control the analysis and workflow interactively.

C. *Performance Model*

We present a performance model of our ICEE system. This performance will enable us to anticipate the time needed by various steps of the in transit workflow and allow us to better schedule the tasks to meet the time constraints. The key elements of the ICEE system to be discussed here network communication and indexing/querying operated by FastBit. We present element-wise performance models in the ICEE system. Those models can be used to estimate projected performances in various scenarios. Regression results and estimated performance will be discussed in details in Section IV.

Communication time: We assume times, T_c, for sending data of size x (in bytes) is linearly proportional to the size x of data:

$$T_c = \alpha_c x + r_c \quad (1)$$



(a) Time-series channel data (b) Histogram

Fig. 5: An example of Soft X-ray Array (SXR) channel data (a) and its histogram (b) showing number distributions.

where the coefficient α_c represents average transfer time (second/byte) between two nodes and its reciprocal represents average network bandwidth (bytes per second). Residual coefficient r_c presents network latency.

Since we send data with extra header information padded, actual data size to send and receive is slightly larger than real data size x. However, this effect remains constant due to fixed header sizes and can be negligible for large x.

Indexing and Querying: Times for both indexing and querying, denoted by T_i and T_q respectively, for data of size x bytes follow also a linear model. They can be represented by:

$$T_i = \alpha_i x + r_i \quad (2)$$

$$T_q = \alpha_q x + r_q \quad (3)$$

In earlier analysis [7], we have shown the time to create and operate on a FastBit compressed index is a linear function of the number of words (therefore also bytes). When constructing an index, the time needed is proportional to the total number of bytes needed for storing the index. In a given application where the index size is closely related to the raw data size, therefore, it is fine to assume the index construction time is proportional to the raw data size. When answering a query, the time is expected to be proportional to the number of records in the answer. Since the number of answers in an average query is proportional to the number of data records in the raw data size, it is fine to state the query processing time as a linear function of the raw data size.

IV. Performance Results

In this section we present an evaluation of our prototype system. Based on our performance models shown in (1) to (3), we carefully examine the performance of the key components of ICEE, and discuss projected performances on various scenarios. This performance characteristics will be important for composing more complex workflows in the future.

Before presenting experiment results in the next, we briefly discuss our experimental environments and data sets. To simulate wide-area data analysis, we employed two clusters running in geographically separated locations: Sith, located in Oak Ridge National Lab (ORNL), and Carver, located in Lawrence Berkeley National Laboratory (LBNL). Two locations are connected by 10Gbps/100Gbps backbones operated by Energy Sciences Network (ESnet). Sith is a Linux cluster comprised with 40 compute nodes. Each node contains four AMD Opteron Processors (2.3GHz with 8 cores), and 64 GB of memory.

TABLE I: List of data sets

Name	Type	Duration	Signal Length
Set A	Real-world	10 seconds	5,000,000
Set B	Simulated	100 seconds	50,000,000
Set C	Simulated	1,000 seconds	500,000,000

We used a real-world experimental data set measured from KSTAR tokamak, called Soft X-ray Array (SXR). SXR is a time-series data measured by a camera array with 64 channels. Each channel can generate measurement data (as integer numbers) per 2 micro seconds (μs). As of current technology, a fusion reaction can last about 10 seconds per shot and a SXR data set contains 5,000,000 integer numbers per channel. An example is shown in Fig. 5. KSTAR tokamak is expected to increase the reaction time up to about 300 seconds in future.

In addition to this real-world experimental data we have (hereafter we call Set A), we prepared two larger data sets in order to study our system performance to fit the future KSTAR tokamak performances. Based on Set A, we simulated x10 and x100 longer experiments, 100 seconds and 1,000 seconds respectively, and generated data sets, called Set B and Set C. We use those 3 data sets in total for performance projection in the next. They are summarized in Table I.

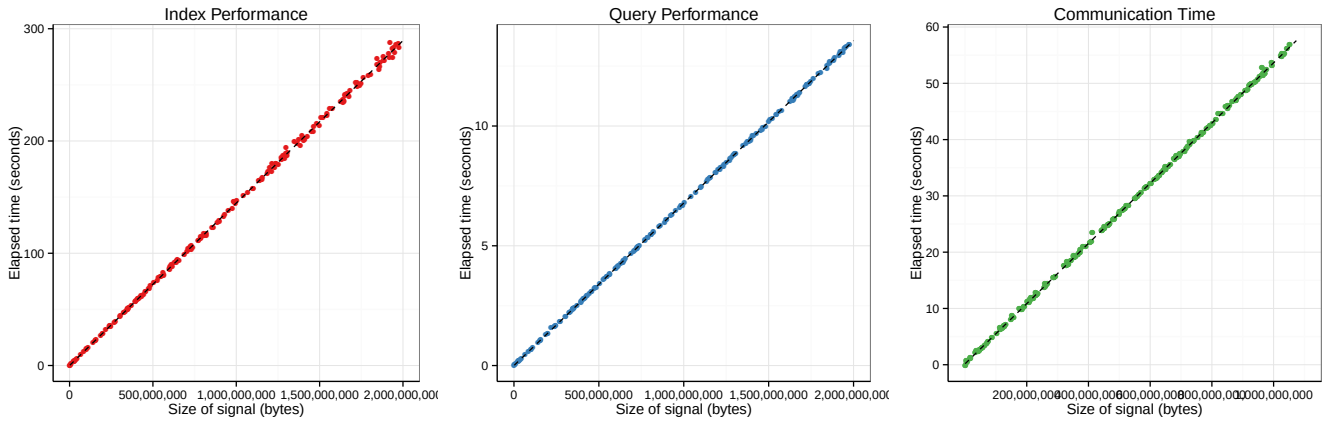
A. Regression fitting

We run experiments to perform linear regression analysis of our system performance.

First, we measured performance of indexing and querying implemented by FastBit in a single process configuration (We didn't explore FastBit's parallel indexing options in this paper. Distributed parallel schemes are discussed in the previous work [11]). Fitting results are shown in Fig. 6a and 6b. We observed both indexing and querying follow a well-fitted linear model in which elapsed times grow proportionally with respect to the length of data.

From this result shown in Fig. 6a, we can compute FastBit indexing speed (or rate), represented by the number of processed integers per second to build an index data. FastBit can process about 1.7 million integers per second, which is over x3.4 times faster than data generation speed of SXR in KSTAR (0.5 millions/second). It is worth noting that FastBit can perform building an index while a KSTAR experiment is being conducted. I.e., we can build indexes during the shot period.

Next, we measured wide-area communication performances in sending and receiving integers between ORNL and LBNL with TCP/IP Protocol. Due to the security restrictions in both institutions, we used SSH tunneling. A fitting result is shown in Fig. 6c. Overall, we observed an average network transportation rate, $2.139 \times 10^{-0.7}$ seconds for sending one



(a) $\alpha_i = 1.447 \times 10^{-7}$, $r_i = 0.06$ (b) $\alpha_q = 6.782 \times 10^{-9}$, $r_q = 0.001$ (c) $\alpha_c = 5.348 \times 10^{-0.8}$, $r_c = 0.159$

Fig. 6: Linear regression analysis of ICEE system.

32-bit integer, which corresponds to 142.67 Mbps network bandwidth.

B. Performance projection

Based on our observed regression results, we project performances of different scenarios.

During a shot: This is a period data is constantly generated and accumulated into a memory buffer. Let assume data generator (such as SXR array) is writing data at the rate of g bytes/second into a memory buffer. Then, at time step t, data size of total x (bytes) will be available in the buffer, such as $x = gt$. FastBit will build indices in an additive way, such that it processes only a certain amount of data (let denote Δx) from the buffer and updates to an existing index structure. FastBit query is performed and a small fraction of the full data in the buffer will be sent back to users through wide-area networks. Let p denote the fraction of queried results.

Then, we define the response time T_{resp} be the sum of all elapsed times (indexing, querying, and communication) which can be computed by the following equations:

$$T_i = \alpha_i \Delta x + r_i \quad (4)$$
$$T_q = \alpha_q x + r_q \quad (5)$$
$$T_c = \alpha_c p x + r_c \quad (6)$$

However, an additive indexing is not always applicable. In such a case, we can rebuild index periodically as data is being generated. Then, T_i can be computed by (2).

Based on the previous fitting results, we computed response times with different indexing strategies, full indexing vs. additive indexing, and also compared them with file transferring times without indexing and query processing, as a base case. Fig. 7 shows simulated results for the 10 second fusion experiment case (Set A) and 1,000 second case (Set C). In these experiments, we assumed $p = 0.1$ (i.e., 10% query results) based on our own experience in image analysis and $g = 2.0 \times 10^6$ (bytes/second) which is the real-world data generation rate in KSTAR's SXR. For simulating additive indexing, we assume 10% delays so that FastBit will read only 2 Mbytes ($\Delta x = 2 \times 10^6$) in Set A or 200 Mbytes in Set C per operation.

Using an additive indexing can provide a significant speed up, compared to the base case using file transfers without FastBit indexing. Meanwhile, full indexing looks expensive, compared to the additive indexing. In Fig. 7b, it took less than 1/3 of the assumed fusion reaction time (1,000 seconds). We think it can be a feasible solution but can not be used for time critical applications.

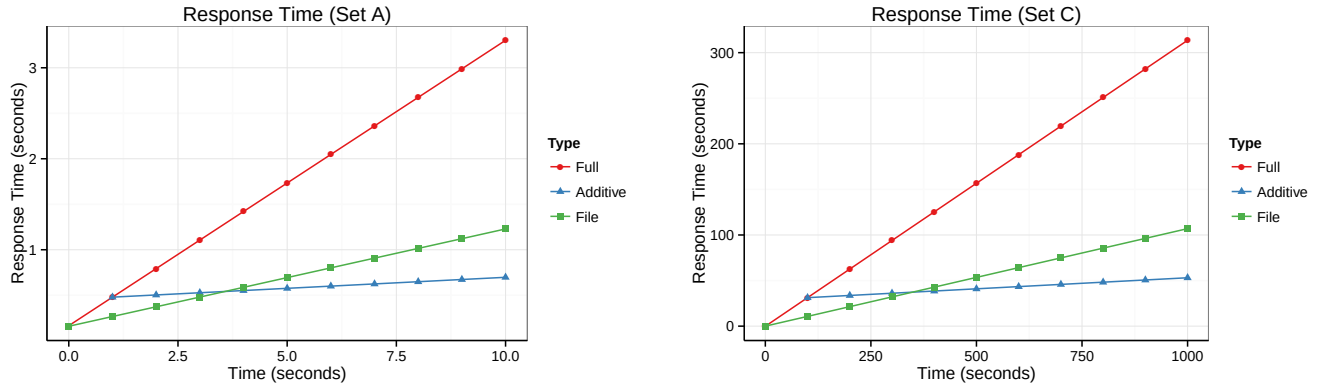
Between shots: Followed by the during shot period, there is a resting period generating no new data and, thus, no indexing operation is necessary. Only querying and network communication will occur. Then, the total response time T_{reap} is the sum of only querying and communication time, estimated by,

$$T_q = \alpha_q M + r_q \quad (7)$$
$$T_c = \alpha_c p M + r_c \quad (8)$$

where M represents the size of data in a buffer. M is a fixed value, corresponding to the maximum length of signals (See Table I).

Fig. 8 shows the estimation of three response times for our data sets (Set A, Set B, and Set C) with different data sizes. We assumed $p = 0.1$ (i.e., 10% query results) here too. As shown in Fig. 8, we can achieve significant performance gains by using FastBit, compared with just sending data (SCP copy) as files. Especially, with Set C, we may achieve maximally 4.2x speedup.

Remote indexing: We explore possibility of distributing index data itself to remote clients so that remote clients can query data locally without need to contact to a data server frequently. This operation will better be performed in a between shots period. To this end, a client can build indices locally after receiving the raw data from a server. Otherwise, a server can distribute indices along with the raw data. The index data structure of FastBit could be larger than the raw data, so a trade-off can be made based on a network speed.



(a) Set A (b) Set C

Fig. 7: Response times during shot period with Set A (a) and Set C (b). Compared two indexing schemes, full indexing vs. additive indexing. Green lines represent only file transferring time without using indexing or query processing.

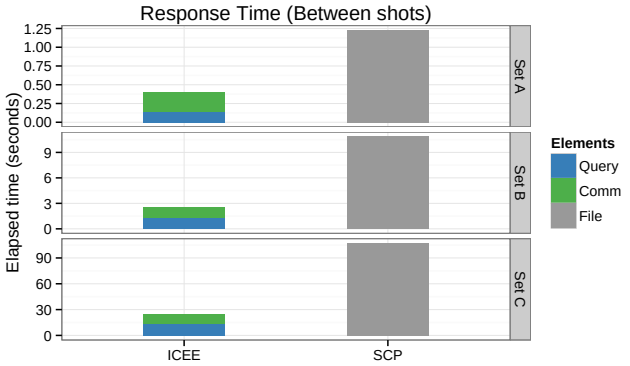


Fig. 8: Estimated response times between shot periods. Assumed 10% ($p = 0.1$) of original data queried. In Set C, we can achieve maximally 4.2x speedup compared to sending data over SCP.

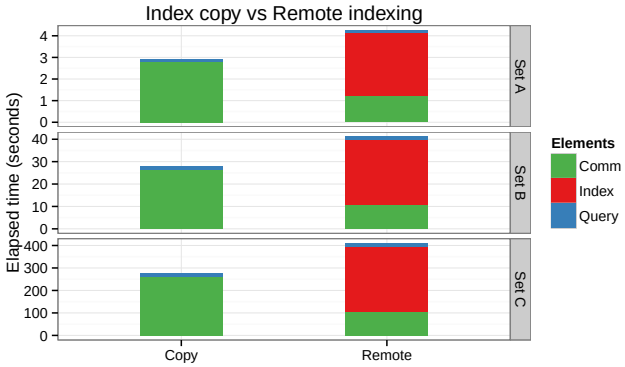


Fig. 9: Comparison of copying index from a server with building index in a local client. In our experiment environment, copying indices through wide-area network works better than building indices locally in a remote client site.

If network speed is good, it is better for a client to receive index data from a server. If not, it is better to receive raw data only and build an index data locally. In our experimental environments, copying index data worked better. Fig. 9 shows our estimation to compare index copying with local indexing building.

V. Conclusion

Large scientific experiments often have participants from around the world, the ability to access and analyze the data in real-time is a necessary feature for increased collaboration and therefore increase productivity. In this paper, we propose a strategy to satisfy this real-time data analysis needs with dynamic subsetting and in transient data processing. We present a prototype system, called ICEE, to support near real-time scientific applications over wide-area networks. Our system is based on two key technologies: I) Adaptive I/O System, ADIOS, to support wide-area in transit (or in-flight) data processing to avoid expensive file I/Os, and II) FastBit to provide indexing and querying features for efficient data searching and provide effective network data transfer by moving partial data selectively.

By using the real fusion experimental data generated from the KSTAR fusion reactor, we provided the performance evaluations of our ICEE system and demonstrated the effectiveness of our implementation. Our test results show that the amount of time needed by major components of the system scales linearly with the number of data records examined. This feature allows the user the option of adjusting the number of data records they want to examine during time-critical analysis steps. For a range of tasks, it is possible to complete them in a few seconds over a wide-area network spanning across the US.

References

[1] G. Lee, M. Kwon, C. Doh, B. Hong, K. Kim, M. Cho, W. Namkung, C. Chang, Y. Kim, J. Kim *et al.*, "Design and construction of the kstar tokamak," *Nuclear Fusion*, vol. 41, no. 10, p. 1515, 2001.

[2] "ADIOS: ADaptable I/O System," https://www.olcf.ornl.gov/center-projects/adios/.

[3] J. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. 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*. ACM, 2008, pp. 15–24.

[4] K. Wu, S. Ahern, E. W. Bethel, J. Chen, H. Childs, E. Cormier-Michel, C. Geddes, J. Gu, H. Hagen, B. Hamann, W. Koegler, J. Lauret, J. Meredith, P. Messmer, E. Otoo, V. Perevoztchikov, A. Poskanzer, Prabhat, O. Rübel, A. Shoshani, A. Sim, K. Stockinger, G. Weber, and W.-M. Zhang, "Fastbit: Interactively searching massive data," in *SciDAC 2009*, 2009, lBNL-2164E.

[5] S. H. Lee, K. B. Chai, S. Jang, W.-H. Ko, J. Kim, D. Seo, J. Lee, I. Bogatu, J.-S. Kim, and W. Choe, "Design and fabrication of a multi-purpose soft x-ray array diagnostic system for kstar," *Review of Scientific Instruments*, vol. 83, no. 10, pp. 10E512–10E512, 2012.

[6] M. García-Muñoz, S. Akaslampolo, O. Asunta, J. Boom, X. Chen, I. Classen, R. Dux, T. Evans, S. Fietz, R. Fisher *et al.*, "Fast-ion redistribution and loss due to edge perturbations in the asdex upgrade, diii-d and kstar tokamaks," in *IAEA Fusion energy conference*, 2012.

[7] K. Wu, E. Otoo, and A. Shoshani, "Optimizing bitmap indices with efficient compression," *ACM Transactions on Database Systems*, vol. 31, pp. 1–38, 2006, http://doi.acm.org/10.1145/1132863.1132864.

[8] K. Wu, A. Shoshani, and K. Stockinger, "Analyses of multi-level and multi-component compressed bitmap indexes," *ACM Transactions on Database Systems*, vol. 35, no. 1, pp. 1–52, 2010, http://doi.acm.org/10.1145/1670243.1670245.

[9] S. Byna, J. Chou, O. Rübel, Prabhat, H. Karimabadi, W. S. Daughton, V. Roytershteyn, E. W. Bethel, M. Howison, K.-J. Hsu, K.-W. Lin, A. Shoshani, A. Uselton, and K. Wu, "Parallel I/O, Analysis, and Visualization of a Trillion Particle Simulation," in *Proceedings of SuperComputing 2012*, Nov. 2012, http://dl.acm.org/citation.cfm?id=2389077.

[10] K. Wu, W. Koegler, J. Chen, and A. Shoshani, "Using bitmap index for interactive exploration of large datasets," in *Proceedings of SSDBM 2003*, Cambridge, MA, USA, 2003, pp. 65–74, a draft appeared as tech report LBNL-52535.

[11] J. Kim, H. Abbasi, L. Chacon, C. Docan, S. Klasky, Q. Liu, N. Podhorszki, A. Shoshani, and K. Wu, "Parallel in situ indexing for data-intensive computing," in *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*. IEEE, 2011, pp. 65–72.

[12] J. Chou, K. Wu, O. Rübel, M. Howison, J. Qiang, Prabhat, B. Austin, E. W. Bethel, R. D. Ryne, and A. Shoshani, "Parallel index and query for large scale data analysis," in *SC11*, 2011.

[13] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, "Datastager: scalable data staging services for petascale applications," *Cluster Computing*, vol. 13, no. 3, pp. 277–290, 2010.

[14] G. Eisenhauer, M. Wolf, H. Abbasi, and K. Schwan, "Event-based systems: opportunities and challenges at exascale," in *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*. ACM, 2009, p. 2.

[15] G. Eisenhauer, K. Schwan, and F. E. Bustamante, "Publish-subscribe for high-performance computing," *Internet Computing, IEEE*, vol. 10, no. 1, pp. 40–47, 2006.

[16] G. Eisenhauer, M. Wolf, H. Abbasi, S. Klasky, and K. Schwan, "A type system for high performance communication and computation," in *e-Science Workshops (eScienceW), 2011 IEEE Seventh International Conference on*. IEEE, 2011, pp. 183–190.

[17] "Mobile Chips Threaten High-Performance Manufacturers," http://www.technologyreview.com/computing/25143/, May 2010.

[18] D. J. Cook and S. K. Das, "Pervasive computing at scale: Transforming the state of the art," in *Pervasive and Mobile Computing*, 2012, pp. 22–35.

[19] A. Sobester, S. Johnston, J. Scanlan, N. O'Brien, E. Hart, C. Crispin, and S. Cox, "High altitude unmanned air system for atmospheric science missions," in *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, 2011.

[20] D. Aanensen, D. Huntley, E. Feil, F. al Own, and B. Spratt, "Epicollect: Linking smartphones to web applications for epidemiology, ecology and community data collection," in *PLoS ONE 4(9): e6968*, 2009.

[21] M. Pelino, C. Mines, and S. Musto, "Forrsights: Mobility dominates enterprise telecom trends in 2011," in *Forrester*, July 22, 2011.

[22] "The NERSC mobile user portal," http://m.nersc.gov/.

[23] "Digital Omnivores: How Tablets, Smartphones and Connected Devices are Changing U.S. Digital Media Consumption Habits," comScore whitepaper, Oct. 10, 2011.

[24] H. Schaffers, T. Brodt, M. Pallot, and W. P. (editors), "The future workspace, perspectives on mobile and collaborative working," in *The MOSAIC Consortium*, 2006.