DOE Award #: DE- FG02-08ER25848

Name of the Recipient: Northwestern University

Project Title: Active Storage with Analytics Capabilities and I/O Runtime System for Petascale Systems

PI: Alok Choudhary

Date of the report: March 18, 2015

Project Final Report/Accomplishments

Project Goal

Computational scientists must understand results from experimental, observational and computational simulation generated data to gain insights and perform knowledge discovery. As systems approach the petascale range, problems that were unimaginable a few years ago are within reach. With the increasing volume and complexity of data produced by ultra-scale simulations and high-throughput experiments, understanding the science is largely hampered by the lack of comprehensive I/O, storage, acceleration of data manipulation, analysis, and mining tools. Scientists require techniques, tools and infrastructure to facilitate better understanding of their data, in particular the ability to effectively perform complex data analysis, statistical analysis and knowledge discovery.

The goal of this work is to enable more effective analysis of scientific datasets through the integration of enhancements in the I/O stack, from active storage support at the file system layer to MPI-IO and high-level I/O library layers. We propose to provide software components to accelerate data analytics, mining, I/O, and knowledge discovery for large-scale scientific applications, thereby increasing productivity of both scientists and the systems. Our approaches include 1) design the interfaces in high-level I/O libraries, such as parallel netCDF, for applications to activate data mining operations at the lower I/O layers; 2) Enhance MPI-IO runtime systems to incorporate the functionality developed as a part of the runtime system design; 4) Develop parallel data mining programs as part of runtime library for server-side file system in PVFS file system; 3) Prototype an active storage cluster, which will utilize multicore CPUs, GPUs, and FPGAs to carry out the data mining workload.

Background

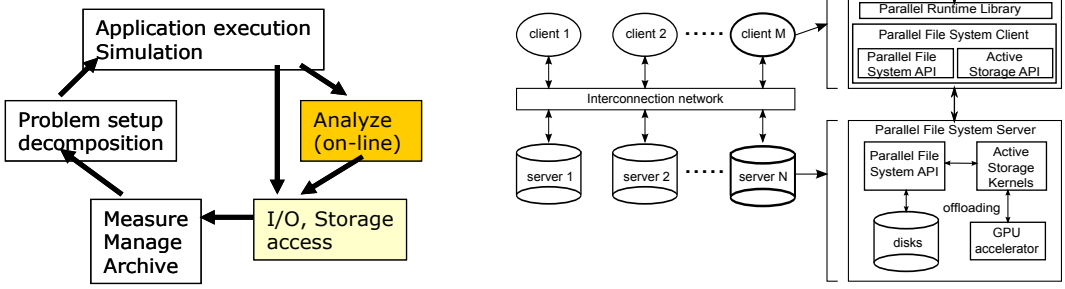
One common characteristic of most applications is that they are all very data intensive. In many cases, scaling the simulation from the computation perspective is achieved for tens of thousands of processors, but performing I/O and subsequent analyses are a major bottleneck and all scientists pointed that out as a major hurdle to effectively utilizing petascale systems and accelerating discoveries. For example, in applications such as climate modeling, combustion and astrophysics simulations, dataset sizes range between 100TB-10PBs and the required compute performance is 100+ Teraops.

Our proposal asks the following fundamental questions: “What if parallel I/O runtime interfaces are developed in which a user can perform efficient and scalable I/O, and at the same time, the user can perform scalable analysis, mining, subset operations, and transformations as part of the process of performing I/O (Figure 1(a))?” Furthermore, “Can this runtime exploit specific architecture features and hardware in the form of active storage so that these functions can be performed concurrently with application execution?”

Technical Progress and Accomplishments:

Active storage system – We have developed an active storage system software for parallel I/O. This system enables data analytic tasks within the context of parallel file systems through three key features:

- Enhanced runtime interface that uses predefined data analytics kernels in parallel file systems: We



(a) (b)

Figure 1: (a) The proposed I/O runtime enhancements with appropriate interfaces to enable on-line, in-place analysis and data transformations as a part of the I/O runtime. Furthermore, it is proposed that special nodes in large systems, using the concept of active storage be used. (b) The architecture of our active storage system. Each file server is an active storage system that is equipped with a GPU and a local disk.

expose the semantics of predefined analysis kernels, such as the data type of data blocks on the disk, to parallel file systems so that execution of embedded kernels is possible on the server.

- File stripe alignment during runtime: In order to allow a file server to perform proper computation on striped files, our system adjusts to misaligned computational units by pulling missing bytes, when needed, from the neighboring servers that hold them.
- Server-to-server communication for aggregation and reduction: In order to perform computation entirely on the server side, servers need to communicate their partial (local) results with other servers to obtain the complete results. To this end, we augmented the storage servers with basic collective MPI communication primitives (e.g., broadcast and allreduce).

To demonstrate the effectiveness of our approach, we built an active storage systemon top of a parallel file system, PVFS, and parallel runtime system library, MPICH2. Figure 1(b) illustrates the high-level view of our active storage system in our storage deployment architecture. Our active storage approach utilizes a processing capability within the storage nodes in order to avoid large data transfers. Computational capabilities, including optional GPU accelerators, within the storage infrastructure are used to reduce the burden on computationally intensive kernels. The active storage nodes are located on the same communication network as the client (compute) nodes. On the client side, we use MPI for communication.

To provide easy use of our active storage system equipped with these analytic kernels, we enhanced the MPI-IO interface and functionality to both enable analytics and utilize active storage nodes for performing the data analysis. We chose MPI for our prototyping for two reasons. First, MPI is a widely used interface, especially in science and engineering applications, and numerous parallel applications are already written in MPI. Therefore, it would provide an easy migration path for those applications to effectively utilize our approach. Second, MPI provides a hint mechanism by which user-defined information can be easily transferred to intermediate runtime libraries, thereby making incorporating data analysis kernels easier.

In our design, the analysis application on the client nodes uses normal MPI and MPI- IO calls to perform its I/O and computation/communication. For our active storage-based application, the client invokes our enhanced MPI-IO calls to initiate both data read and computation, and the corresponding functions and code are executed on the active storage nodes, which may use any available hardware acceleration functions. An advantage of this design is that it facilitates a runtime decision within the MPI-IO

implementation on where to execute analysis – the functions could just read and then analyze on compute nodes, if the cost were lower. The results of active storage operations then are returned to the client in the original file read buffer. We note that higher-level libraries, such as Parallel netCDF can be easily extended to use this infrastructure. Currently, users are required to develop their own functions (e.g., analytics, mining, statistical, subsetting, searching, etc.), develop their own accelerators using the GPUs, and embed them with our active storage APIs so that they can be called from their applications.

Our system demonstrates a better way of enabling an active storage system to carry out data analytic computations as part of I/O operations. The experimental results obtained with a set of data analytic kernels, including data-mining kernels, demonstrate that our system can improve the overall performance by 50.9%, on average. We show that the compute-intensive kernels of the k-means data clustering algorithm can be offloaded to the file servers that are equipped with local GPU accelerators, leading to 58.4% performance improvement when the algorithm became compute-intensive. Overall, we show that our approach consistently outperforms the traditional storage model with a wide variety of data set sizes, number of nodes, and computational loads.

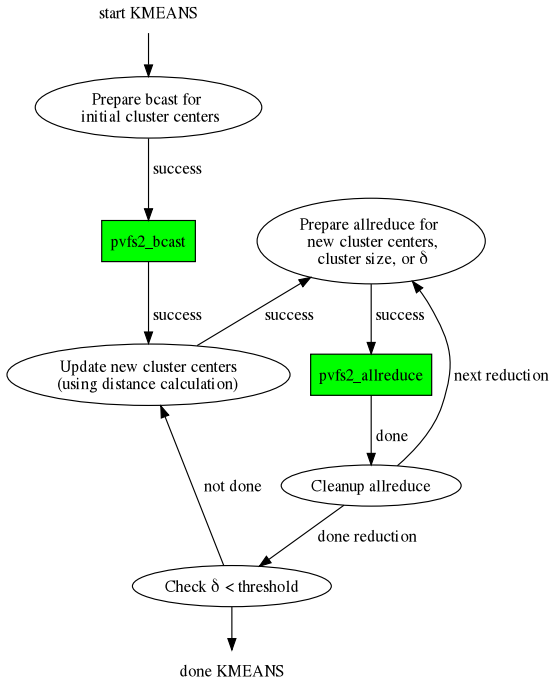


Figure 2. The k-means clustering kernel implemented within the PVFS server. It uses pvfs_bcast and pvfs2_allreduce for server-server broadcasting and allreduction.

Enhanced Runtime Interface in Parallel File Systems – Each analysis function developed for our active storage node consists of two versions: traditional C/C++ codes and accelerator-specific codes. The C/C++ codes are executed on the normal storage server, whereas the accelerator-specific codes are executed on the accelerator if available and required based on performance considerations.

Since we use MPI-IO as our parallel runtime library, we can explore several options for exposing these new functions. One option is to use the hints mechanism in MPI-IO and define new hints (key/value pairs) that specify the data analysis function that gets executed on the server. Another option is to use the user-defined data representations in MPI-IO. The MPI-IO interface allows users to define their own data representation and provide data conversion functions. MPI will internally call those functions when reading the data. While this method supports both predefined and user-defined operations, it is restricted to MPI-based environments; hence, server codes need to be rewritten in MPI or be interfaced with MPI-based analysis functions. A third option is to define extended versions of MPI-IO read calls that take an additional argument specifying the operation to be performed. In this paper, we explore the combination of both the extended version of MPI read calls and the hints mechanism. While the former provides a way to specify the operation to be called on the server, the latter allows us to pass application-specific semantics that are required to complete the specified kernels.

Server-to-Server Communication – Inter-server communication already exists in PVFS to handle various server operations such as small I/O and metadata operation. Using the server-to-server communication primitives in PVFS, we implemented two collective primitives: broadcast and allreduce. We chose these two mainly because they are used in one of our benchmark routines, k-means. While several viable algorithms exist, our implementation uses the recursive distance-doubling algorithm for

both collective operations. These two operations each are built as separate state machines in PVFS, so any embedded kernel that needs to use collective operations can use these facilities without having to re-implement them. For instance, the k-means clustering program is composed of multi-phased operations: cyclic data updates and potentially large computation until convergence. When it is parallelized, the compute-intensive portion, which is calculating minimum distance, can be executed in parallel. However, a reduce operation must be used as new cluster centers obtained in each server need to be updated. We note that, while implementing these two collective primitives mainly to ease embedding more complicated data analysis kernels, we believe they can be used for other purposes whenever server tasks can benefit by being cast in terms of collective operations. Figure 2 shows the k-means clustering kernel we implemented using two collective primitives.

Parallel netCDF – Our work has focused on design of data analytics functions in Parallel netCDF library (PnetCDF) that can communicate with the active storage servers for offloading the analysis tasks to be run on the server-side GPUs or FPGAs. We constructed a set of data statistics functions in PnetCDF. Since data stored in netCDF file format is well defined by its associated metadata, these statistics functions can utilize this information to calculate the data partitioning among multiple client processes as well as the I/O servers that store the related file stripes. Metadata is also pushed down to the MPI-IO library layer using MPI primitive data types in order to identify the partitioning boundaries of file stripes. We used a parallel K-means data clustering application in our experiments and we were able to execute the K-means clustering using PnetCDF functions.

CUDA Module –In addition to the common statistic functions, such as sum, max, min, average operations, we have implemented three data mining applications to run on GPUs: K-means, fuzzy K-means, and principal component analysis. These programs are written in CUDA and run on NVIDIA Geforce 8800GT GPUs. Our performance evaluations show the speedups as high as 150 times of the performance using the state-of-the-art CPUs. We have constructed a data mining library in CUDA that can be linked by the server-side file system, so the I/O servers equipped with GPU coprocessors can perform the data mining tasks close to where the data resides. Our hybrid MPI and GPU programs for the above three data mining applications have been further extended their parallelism on GPU clusters for very large data sets. We have carried out our experiments on the TeraGrid machine, Lincoln, a large scale GPU cluster at NCSA.

FPGA Module – In addition to k-means, we have also implemented a principal component analysis application for network intrusion detection on a Xilinx Virtex-II Pro FPGA. The results achieve a throughput of up to 24.72 Gbps, which satisfy the needs of Gigabit network connections.

I/O Delegation Subsystem – We have developed various optimizations for the I/O delegate and caching system. They are software components at the MPI-IO layer where certain I/O tasks, such as file caching and consistency control are delegated to a small set of compute nodes, collectively termed as I/O delegate nodes. This layer is implemented at the bottom layer of ROMIO where it intercepts all the system I/O requests initiated by ROMIO and redirects them to delegate nodes. We incorporated a static file domain assignment approach from our earlier work on MPI-IO file domain for further performance improvement. We also explored the opportunity of using more than one delegate processes in a multi-core compute node machine. This work can greatly benefit the performance of MPI independent I/O whose optimizations have been considered difficult by the parallel I/O community and almost none exists. We conducted our experiments using the FLASH and S3D I/O kernels on supercomputers using Lustre parallel file system. Testing and development were performed on several parallel machines: Abe at NCSA and Franklin at NERSC. Our experiments show that using MPI independent I/O functions in the two application kernels can outperform the same kernels using the collective ones. The observed improvement ranges from 2.5 times to 15 times better I/O bandwidth.

In order to simulate the file system server-side active storage functionality, we use the idea and implementation of the I/O delegation software component to run the data analytics operations at the user

space. The I/O delegation is our earlier work that uses MPI dynamic process management to allocate a set of separate compute nodes to handle the I/O requests from the application processes. It allows us to exercise the data analytics interface design and software mechanism to activate the computation on the servers. The advantage of using this I/O delegation is the flexibility to pass the high-level metadata among the I/O software layers. It is critical for a data analytics operation to understand the structure of the data and this information is usually lost when data reaches at the file systems.

We have worked with the PVFS file system team at Argonne National Laboratories to extend the development on I/O delegation platform to the PVFS file servers. The immediate focus will be to build the library so the file servers can link and run the data analytics functions. The next step is to identify the group of servers that hold the required data and participate the operations, so that the proper communication group, such as MPI communicator, can be created. We will use the file system striping configuration information and low-level system file metadata status to achieve this goal.

Personel:

Professors: Alok Choudhary and Wei-keng Liao

Postdoc: Kui Gao

Graduate Students: Prabhat Kumar, Afrifa Nisar, and Berkin Ozisikyilmaz,

Collaborators at Argonne National Laboratories: Robert Ross, Rajeev Thakur, Sam Lang, and Seung Woo Son

Relationships to Other Projects

We collaborated with the Geodesic Grid I/O team led by Karen Schuchardt at Pacific Northwest National Laboratory to improve the parallel I/O performance of the Global Cloud Resolving Model (GCRM) framework. GCRM is supported by the DOE SciDAC program as one of the major climate simulation application frameworks. The geodesic grids used by the GCRM covers the entire earth surface with clouds with the dimensions of longitude, latitude, and altitude. A 4-Km grid resolution run will contain 42M horizontal cells and generate about 0.3 TB data for each snapshot, assuming 100 vertical layers and a modest number of 3D variables. A use case study has been created to describe the data model and layout for geodesic grids, which includes source codes, illustrative figures, input data, and expected outputs. The URL is: http://cucis.ece.northwestern.edu/projects/DAMSEL/GCRM_write.html

We collaborated with Dr. Anshu Dubey and Christopher Daley, application scientists from the ASC / Alliances Center for Astrophysical Thermonuclear Flashes at the University of Chicago. The FLASH code is to study the surfaces of compact stars such as neutron stars and white dwarf stars, and in the interior of white dwarfs. FLASH code uses an AMR-based domain decomposition method to partition the data. I/O has long been a performance bottleneck for FLASH in production runs. A use case study has been created to describe the data model and layout for Adaptive Mesh Refinement grids. The URL is: http://cucis.ece.northwestern.edu/projects/DAMSEL/damsel_usecase_flash_detail.html

Web Access

The project web page, http://cucis.ece.northwestern.edu/projects/FASTOS/, contains description of the developed framework.

Presentations:

- Alok Choudhary, "Discovering Knowledge from Massive Social Networks and Science Data - Next Frontier for HPC", keynote at the International Conference on High Performance Computing, Bangalore, India December 2011.
- Alok Choudhary, "Developing Scalable and Power-Efficient Data Mining Kernels", in the Understanding Climate Change Workshop, University of Minnesota, August 2011.

Publications:

1. B. Ozisikyilmaz, G. Memik, and A. Choudhary. Machine Learning Models to Predict Performance of Computer System Design Alternatives. In Proceedings of the International Conference on Parallel Processing (ICPP), September 2008.
2. B. Ozisikyilmaz, G. Memik, and A. Choudhary. Efficient System Design Space Exploration Using Machine Learning Techniques. In Proceedings of The Design Automation Conference (DAC), June 2008.
3. Arifa Nisar, Wei-keng Liao, and Alok Choudhary. Scaling Parallel I/O Performance through I/O Delegate and Caching System. In the Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, Texas, November 2008.
4. Wei-keng Liao and Alok Choudhary. Dynamically Adapting File Domain Partitioning Methods for Collec- tive I/O Based on Underlying Parallel File System Locking Protocols. In the Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, Texas, November 2008.
5. Alok Choudhary, Wei-keng Liao, Kui Gao, Arifa Nisar, Robert Ross, Rajeev Thakur, and Robert Latham. Scalable I/O and Analytics. In the Journal of Physics: Conference Series, Volume 180, Number 012048 (10 pp), August 2009.
6. Kui Gao, Wei-keng Liao, Arifa Nisar, Alok Choudhary, Robert Ross, and Robert Latham. Using Subfiling to Improve Programming Flexibility and Performance of Parallel Shared-file I/O. In the Proceedings of the International Conference on Parallel Processing, Vienna, Austria, September 2009.
7. Nithin Nakka, Alok Choudhary, Wei-keng Liao, Lee Ward, Ruth Klundt, and Marlow Weston. Detailed Anal- ysis of I/O Traces for Large Scale Applications. In the Proceedings of the International Conference on High Performance Computing, Cochin, India, December 2009.
8. Seung Woo Son, Samuel Lang, Philip Carns, Robert Ross, Rajeev Thakur, Berkin Ozisikyilmaz, Prabhat Ku- mar, Wei-keng Liao, and Alok Choudhary. Enabling Active Storage on Parallel I/O Software Stacks. In the Proceedings of the 26th IEEE Symposium on Massive Storage Systems and Technologies, Incline Village, Nevada, May 2010.
9. Seong Jo Kim, Yuanrui Zhang, Seung Woo Son, Ramya Prabhakar, Mahmut Kandemir, Christina M Patrick, Wei-keng Liao, and Alok Choudhary. Automated Tracing of I/O Stack. In the Proceedings of the 17th European MPI Users Group conference (EuroMPI), Stuttgart, Germany, September 2010.
10. Prabhat Kumar, Berkin Ozisikyilmaz, Wei-keng Liao, Gokhan Memik, and Alok Choudhary. High Performance Data Mining Using R on Heterogeneous Platforms. In Workshop on Multithreaded Architectures and Applications, in conjunction with the International Parallel and Distributed Processing Symposium, May 2011.
11. Chen Jin, Saba Sehrish, Wei-keng Liao, Alok Choudhary, and Karen Schuchardt. Improving the Average Response Time in Collective I/O. In the Proceedings of the 18th European MPI Users Group conference (EuroMPI), Santorini, Greece, September 2011.
12. Kui Gao, Chen Jin, Alok Choudhary, and Wei-keng Liao. Supporting Computational Data Model Rep- resentation with High-performance I/O in Parallel netCDF. In the Proceedings of the High Performance Computing Conference, Bengaluru, India, December 2011.
13. Arifa Nisar, Wei-keng Liao, and Alok Choudhary. Delegation-Based I/O Mechanism for High Performance Computing Systems. In the IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 2, pp. 271- 279, Feb. 2012. DOI. 10.1109/TPDS.2011.166