DOE Award #: DE-SC0005309

Name of the Recipient: Northwestern University

Project Title: Damsel: A Data Model Storage Library for Exascale Science

PI: Alok Choudhary

Date of the report: July 11, 2014

Project Final Report/Accomplishments

Project Goal

The goal of this project is to enable exascale computational science applications to interact conveniently and efficiently with storage through abstractions that match their data models. The project consists of three major activities: (1) identifying major data model motifs in computational science applications and developing representative benchmarks; (2) developing a data model storage library, called Damsel, that supports these motifs, provides efficient storage data layouts, incorporates optimizations to enable exascale operation, and is tolerant to failures; and (3) productizing Damsel and working with computational scientists to encourage adoption of this library by the scientific community.

Background

Computational science applications have been described as having one of seven motifs (the “seven dwarfs”), each having a particular pattern of computation and communication. From a storage and I/O perspective, these applications can also be grouped into a number of data model motifs describing the way data is organized and accessed during simulation, analysis, and visualization. Major storage data models developed in the 1990s, such as Network Common Data Format (netCDF) and Hierarchical Data Format (HDF) projects, created support for more complex data models. Development of both netCDF and HDF5 was influenced by multi-dimensional dataset storage requirements, but their access models and formats were designed with sequential storage in mind (e.g., a POSIX I/O model). Although these and other high-level I/O libraries have had a beneficial impact on large parallel applications, they do not always attain a high percentage of peak I/O performance due to fundamental design limitations, and they do not address the full range of current and future computational science data models.

It is well recognized that a different approach, one that leverages the lessons and best practices learned from previous approaches, is needed to achieve the scalability required from high-level I/O and storage libraries to fulfill the promise of exascale systems. As the International Exascale Software Project (IESP) report observes, “The purpose of I/O by an application can be a very important source of information that can help scalable I/O performance when hundreds of thousands (to millions) of cores simultaneously access the I/O system.” In other words, the high-level view of the data model is overlooked rather than exploited. Also, the data layout used in these codes and how that layout interacts with I/O software used to save the data to or read the data from storage systems are highly relevant. Arguably, the model of building “verticals” with customized interfaces and formats for the data model motifs of computational science is the next important step in enabling usable and high performance exascale I/O, and this model represents the underlying approach of our project.

Technical Progress

Northwestern University, along with team members from our collaborating institutions, Argonne National Laboratory and HDF5, contributed to the development of a C programming interface for Damsel. The interface, like the underlying data model, borrows many ideas from the mesh library MOAB. This MOAB heritage gives us confidence that the Damsel API will be a good fit for unstructured mesh applications. We have built on the MOAB ideas, allowing for algorithmic descriptions of regularly structured models. Such algorithmic descriptions should allow us to efficiently implement structured grid workloads.

In order to exercise and evaluate our programming interface, Northwestern University and our collaborators have worked on test cases and application drivers. We have developed a prototype for an astrophysics application named FLASH developed at University Chicago. The data model used by FLASH is a block-based semi-structured AMR (adaptive mesh refinement) grid. Our goal is to build a use case for FLASH using the current Damsel API functions. From the study of the FLASH source codes, we have captured the essential features of data model used by FLASH, such as the hierarchy relationship among blocks. We are in the process of incorporating these features into Damsel to support a block-based AMR data model. The current version of Damsel is using HDF5 to store data in files and the implementation using Parallel-NetCDF interface is under development. While the goal of Damsel is to provide an I/O library for the sophisticated data models in use by applications, the completed Parallel netCDF interface will allow us to evaluate Damsel's behavior for the less sophisticated regular structured grid models widely used for application I/O today.

We are also working on developing MPI-IO optimizations. Any improvement at the MPI-IO level will benefit all MPI applications and high-level libraries build on top of MPI-IO, such as PnetCDF, HDF5, and Damsel. Our recent optimizations include a pipelining strategy that overlaps the communication and I/O phases in MPI collective I/O, an I/O aggregator location adjustment, and a new method to enable reusing MPI-IO hints across multiple files. These optimizations have been demonstrated successfully to improve performance for a climate simulation application named GCRM on a Cray XE6 parallel machine using up to 16K processes. The above work is presented in a paper submitted to the Supercomputing conference 2012.

Northwestern University hosted a 2-day meeting to resolve and finalize data model API issues in December 2011. Participants were able to come up with a basic list of API functions based on Damsel data model. We participated in the "Damsel Boot Camp" meeting at Argonne National Lab in March 2012; this meeting served as a next step from a design mindset to an implementation mindset. Attendees to the boot camp gained familiarity with the HDF5-developed prototype and the necessary software dependencies, began creating additional test cases (e.g. FLASH), and developed a schedule for future software milestones.

Data Models

Damsel library is designed to be very flexible that allows users to describe an arbitrary data model, such as unstructured grids consisting of vertices and edges. We define several commonly used objects: dimensions, units, entities, entity sets, tag, and handles. Entities are the basic topological objects. The types of an entity can be Vertex, Edge, Triangle, Quadrilateral, Tetrahedron, Pyramid, Prism, Septahedron, Hexahedron, Polygon, or Polyhedron. Each topology entity has a prescribed number, arrangement, and local (within-entity) numbering of corner vertices to form the entity. Damsel uses a 0-based numbering system, where vertices are numbered from 0..N-1, N being the number of corner vertices in the entity. The relative placement of vertices defining an entity is described by the entity's canonical numbering. Topological dimension is the number of independent directions in the local space of an entity. Each entity type in Damsel has a topological dimension 0 for vertex, 1 for edge, 2 for Triangle, Quadrilateral, Polygon and 3 for Tetrahedron, Pyramid, Prism, Septahedron, Hexahedron, Polyhedron. A topology is defined as a set where if any two members of a set intersect, the intersection is also a member of the set. In the context of Damsel entity types, this means that entities of lower topological dimension bound entities of higher dimension. For prescribed-topology entities, only specific adjacency relationships are allowed, and are described by the canonical numbering. For example, tetrahedron entities are bounded by four triangles, 6 edges, and 4 vertices. If a tetrahedron has vertices numbered 0 to 3, its faces will be composed of vertices (0, 1, 2), (0, 1, 3), (1, 2, 3), (2, 0, 3).

An entity set is an arbitrary (i.e., application-defined) collection of handles, where handles refer to entities, sets, tags, dimensions, units, or the interface itself. Entity sets are referenced using handles; the handle type used to refer to entity sets is the same data type as that used for other handles (i.e. for entities, tags, dimensions, units, and interface). A tag is a piece of data that can take on a distinct value for each entity, entity set, tag, dimension, unit, or the root set. A tag has specified name, size type, size (in bytes), optional data type, optional default value, and optional storage type. After a tag is created, it is referred to using a handle returned from the tag creation function. The tag handle is stored using the same data type as is used for other handles.

Once created, individual entities are referenced using an entity handle. The data type used for entity is a 64-bit integer. An entity handle supports increment, decrement, and range operations. Specifically, a collection of contiguous entity handles can be specified as a start handle and end handle, or as a start handle and number of entities. If two handles returned by a given Damsel instance are the same, they refer to the same entity. An entity of any type except Polyhedron is described by its connectivity, which is a list of vertex handles, which form the entity. For example, its 8 vertex handles describe a hexahedron entity.

Programming model

A typical Damsel programming flow for creating a file consists of the following steps: 1) initialize the Damsel environment, 2) create a Damsel model, 3) create Damsel containers, 4) add tags to the handles in a container, 5) create a collection of containers, 6) map the model to a file, 7) commit I/O to the file, 8) close the model, and 9) exit the Damsel environment.

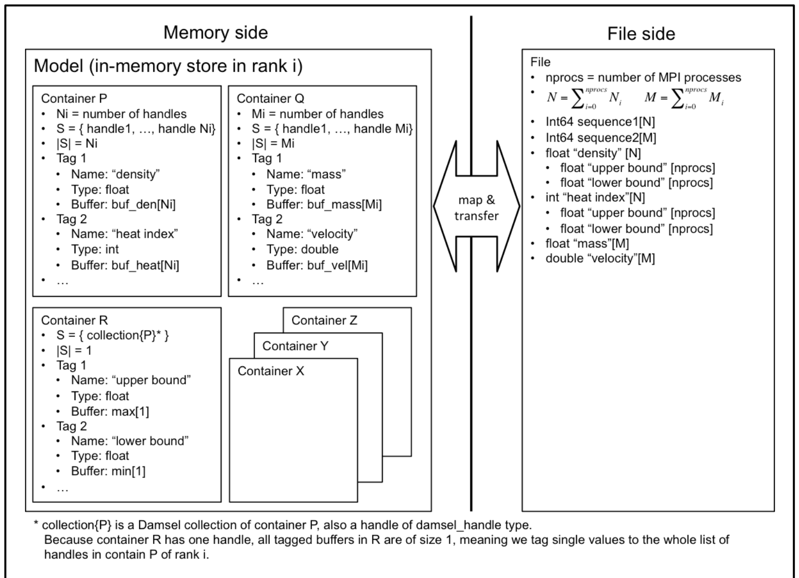

Figure 1. Damsel programming model.

The Damsel environment is initialized and terminated by the two APIs respectively: DMSLlib_init() and DMSLlib_finalize(). All Damsel APIs must be called in between these two environment functions. There are file store and memory store that defined the data in memory and data layout to be mapped to the file. A Damsel model is an in-memory store that describes the data model from the application perspective. One Damsel file contains one model. It is common practice to attach file name and file access hints right after a model is created. The programming model is illustrated in Figure 1.

A Damsel container is a holder of a set of Damsel handles. A Damsel handle is an integral number that refers to an entity or a collection of entities. Entities are application objects, such as vertices and edges of a grid. The set of Damsel handles in a container can be of a list of contiguous integral values or noncontiguous, independent values. Containers aren't really part of the Damsel data model, i.e., they are not stored in the files. Instead, they are used to group handles, so we can pass into other Damsel APIs. Damsel library collects containers from all processes and integrate them into a single view data model in file. Damsel containers are like MPI fileviews that are defined independently by each MPI process. They are used to tell the I/O library (Damsel and MPI-IO) how the local defined data should be mapped to the files. The per-process information (containers/fileviews) will not appear in the file contents. The container constructor APIs are used to create a container with handles of a sequence of contiguous integral values and a vector of independent values.

Once a container is created, one can tag the handle set in a container with a name, I/O buffer, and data type of data in the buffer. A tag can be considered as a solution variable, for example, associated to all vertices of a grid. A tag is represented by a name that must be uniquely defined in a model, a data type, and an I/O buffer pointing to the memory space where the data is stored. We give each buffer element in memory an identifier, or "handle". For example, if the buffer were an array of 5 integers one would assign each of those array elements a handle. In this case, a handle can be a "memory address".

A container can also contain other containers. This is achieved by first "converting" a Damsel container to a Damsel collection. A Damsel collection is of type Damsel handle, so a set of collection handles can be used to create a container. Thus containers can form a hierarchical structure. However, a tag can only be created for the top-level handles of a container.

Once a model is defined, the corresponding file-side store must be mapped and created. In Damsel, data transfer from memory to a file is done in a either blocking or nonblocking fashion. For nonblocking, once the transfer begins, I/O buffers registered in the model are expected not to be changed, until the transfer is completed. To free up the memory used in model construction, the close APIs are used to close models and containers. A program may release containers as soon as they are passed to Damsel and need not maintain a reference until after I/O completes.

Implementation

The implementation of Damsel library is programmed in C and built on top of HDF5. Files created through Damsel can be examined by the utility h5dump. Parallel file access is using HDF5 hyperslab features and MPI collective I/O. We also created a Damsel I/O driver for Parallel netCDF library, so the regular parallel netCDF programs can make use of Damsel. The source codes were nightly built across many machine platforms through the service named "NMI Metronome" to ensure the portability of the library. The machines include various Linux OS versions: Redhat, Ubuntu, MacOS, and Debian. Single writer test program can write 10 M vertices data in less than 10 seconds.

We developed an I/O optimization called subfiling. Subfiling is a mechanism to partition a Damsel file into multiple partitioned files (subfiles) internally, making the damsel data appear as a single file to users. Currently user's intent of using subfiling is conveyed through the damsel trait. Once this information is given to the damsel library, all tags defined in the program will be partitioned and stored into subfiles. Figure 2 illustrates a high-level view of the subfiling mechanism. In contrary to a normal case where a damsel model has only one attached damsel file, subfiling creates multiple attached damsel files to a damsel model. Regardless of whether subfiling is enabled or not, all files generated are in the damsel file format. The default damsel file format comprises of two types of datasets: one for storing metadata and the other for storing actual data. If no subfiling is used, all datasets are stored in a single file specified by the user.

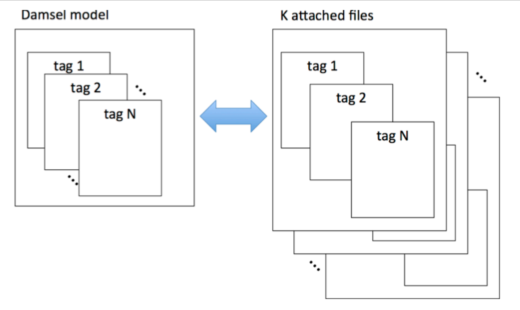

Figure 2. Subfiling in Damsel.

Example programs

We have developed a set of example programs to introduce users the typical programming flows of using Damsel APIs. The programs include parallel writes and reads for both structured and unstructured data models. The sequence of Damsel function calls are accompany with comments to describe the purpose and expected results from the calls. The output files generated from the parallel write programs can be examined by the HDF5 utility "h5dump". The files are used as the input files to the parallel read programs. The example programs are available on line in the URL below:

http://cucis.ece.northwestern.edu/projects/DAMSEL/parallel_write.html and http://cucis.ece.northwestern.edu/projects/DAMSEL/parallel_read.html

Use Case Study

FLASH: FLASH is a modular, parallel multi-physics application, developed at University of Chicago. FLASH uses a structured AMR grid, i.e. the problem domain is hierarchically partitioned into blocks of equal sizes (in array elements). Each block in AMR tree is a 2D/3D mesh on a node/leaf. These blocks are ordered in the Morton space-filling curve, also shown below. A block's info includes its tree level, parent/children, neighbors, coordinates, bounding box. Block cells store the solution data. Figure 3 shows the data model of an AMR tree used in FLASH and the entities defined in Damsel. Mapping to Damsel data model is described in details in the case study page, http://cucis.ece.northwestern.edu/projects/DAMSEL/damsel_usecase_flash_detail.html

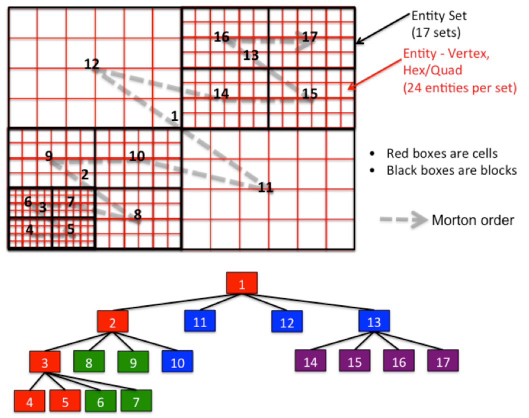

Figure 3: Representing FLASH AMR Grid using Damsel Data Model. Each block in FLASH is represented as an entity set and each cell as an entity.

GCRM: Global Cloud Resolving Model (GCRM) is supported by the DOE SciDAC program as one of the major climate simulation application frameworks. The geodesic grid structure used in GCRM is shown in Figure 4. The data model can be described in the following two ways using DAMSEL API.

Approach 1: The base entity is a vertex, and 6 vertices are used to create a polygon (hexagon). For the hexagons at interfaces, we also create edges because some solution variables are stored on the edges of hexagons at the "interface". For the hexagons at "layers", we just use vertices. Figure 4(d) illustrates "interfaces" and "layers". If a solution variable is stored as "cell-centered", it will define as a tag on the hexagon entity. If a solution variable is edge-centered, it will be defined as a tag on edge entity, and corner-variables as a tag on vertex entity. We create two containers for layers and interfaces, respectively. The only distinction between the two is that we define explicit edges for the vertices of hexagons in interface. In the code example we considered the 9 polygons as shown in Figure 4(c), with 6 polygons on the interfaces and 3 on the layers. We create 13*3 vertices, first 13 vertices are used to create 3 polygons (interface) and 13 edges, and then next vertices are used to create 3 polygons at layer. Then last 13 vertices are used to create 3 polygons and 13 edges. Each polygon at interface will be tagged with cell-centered variable. Each vertex at layer will be tagged with corner variables. Each edge at an interface will be tagged with edge-centered variable.

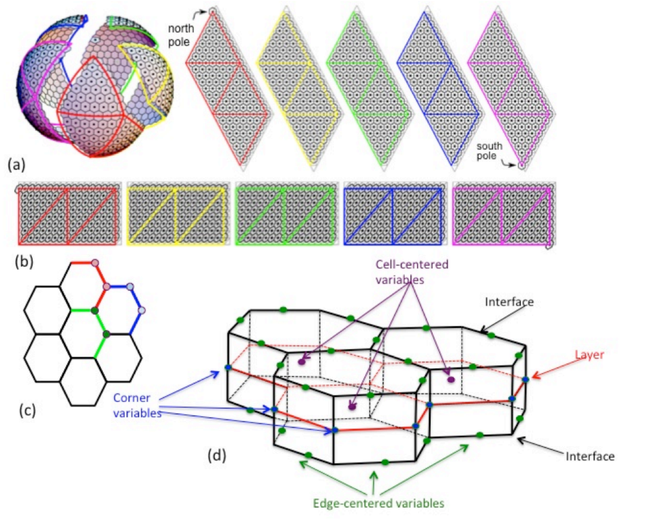

Figure 4. The geodesic grid structure. (a) and (b) illustrate the generation of grids from applying hexagons on the surface of the Earth. (c) and (d) describe the connectivity of individual grid cells and the entities of cells. Image courtesy Dave Randall, Bruce Palmer, Karen Schuchardt, PNNL.

Approach 2: We define cells, corners and edges (as shown in the figure) as Vertex entities in DAMSEL and treat the geodesic grid as an unstructured grid. In GCRM, there are two types of variables -

grid and non-grid (named field variables). We tag each vertex entity with corresponding grid/non-grid variables. There are three containers/collections for 1) cells at layers, 2) corners at layers, and 3) edges at interfaces and we represent each as a sequence of vertex handles.

API References

Damsel Library functions consist of the following categories.

- Damsel Environment functions
- Damsel Data Model functions
- Damsel Container functions
- Damsel Tag functions
- Damsel Entity function
- Damsel Collection function
- Damsel Datatype function
- Damsel trait function

The details of APIs can be found in http://cucis.ece.northwestern.edu/projects/DAMSEL/damsel_api.html

Collaboration and outreach (application domain scientists: FLASH, GCRM)

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/DAMSEL, contains description of Damsel library, which includes the programming model and C application programming interface reference guide. Many case studies are also available, including several fundamental data entities to real production applications. It also includes the I/O optimizations developed is named "subfiling", example codes, and references to the studied cases.

The internal software development repository is running on a SVN server at Argonne National Lab. and its trac/wiki page is https://trac.mcs.anl.gov/projects/damsel. This platform provides an easy and secure platform for collaborative software development from parties at remote locations. Registered users will be able to access all development history and internal user discussion.

Software Release

The source codes of Damsel version 1.0.0 and instruction documents for building the library are openly available to the public from the project web page.

Presentations

1. Alok Choudhary, "Big Data + Big Compute = An Extreme Scale Marriage for Smarter Science?" Plenary Talk at the Supercomputing Conference, Nov 21, 2013.
2. Alok Choudhary, "Discovering Knowledge from Massive Networks and Science Data - Next Frontier for HPC", keynote at the Department of Energy Computational Science Graduate Fellowship Annual Conference, July 26, 2012.
3. Alok Choudhary, "Discovering Knowledge from Massive Social Networks and Science Data – Next Frontier for HPC", Keynote in the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 2012.
4. Alok Choudhary, Wei-keng Liao, Saba Sehrish, Seung Son, Quincey Koziol, Ben Clifford, Rob Ross, Rob Latham, Tim Tautges, Nagiza Samatova, Drew Bayouka, Sriram Lakshminarasimhan, Xiaocheng Zou, and Zhenhuan Gomg, “Damsel: A Data Model Storage Library for Exascale Science”, Poster and a short technical paper at the DoE ASCR meeting, April 2012.
5. 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.

Publications

1. Md Mostofa Ali Patwary, Diana Palsetia, Ankit Agrawal, Wei-keng Liao, Fredrik Manne, and Alok Choudhary. Scalable Parallel OPTICS Data Clustering Using Graph Algorithmic Techniques. In the International Conference for High Performance Computing, Networking, Storage and Analysis, November 2013.
2. William Hendrix, Diana Palsetia, Md. Mostofa Ali Patwary, Ankit Agrawal, Wei-keng Liao, and Alok Choudhary. A Scalable Algorithm for Single-Linkage Hierarchical Clustering on Distributed-Memory Architectures. In the Symposium on Large-Scale Data Analysis and Visualization, October 2013.
3. Saba Sehrish, Seung Woo Son, Wei-keng Liao, Alok Choudhary, and Karen Schuchardt. Improving Collective I/O Performance by Pipelining Request Aggregation and File Access. In the 20th EuroMPI Conference, September 2013.
4. Seung Woo Son, Saba Sehrish, Wei-keng Liao, Ron Oldfield, and Alok Choudhary. Dynamic File Striping and Data Layout Transformation on Parallel System with Fluctuating I/O Workload. In the Workshop on Interfaces and Architectures for Scientific Data Storage, September 2013.
5. Bharath Pattabiraman, Stefan Umbreit, Wei-keng Liao, Alok Choudhary, Vassiliki Kalogera, Gokhan Memik, and Frederic Rasio. A Parallel Monte Carlo Code for Simulating Collisional N-body Systems. The Astrophysical Journal Supplement, IOP Publishing, February 2013.
6. Rob Latham, Chris Daley, Wei-keng Liao, Kui Gao, Rob Ross, Anshu Dubey, and Alok Choudhary. A Case Study for Scientific I/O: Improving the FLASH Astrophysics Code. Computer and Scientific Discovery, 5, March 2012.
7. Arifa Nisar, Wei-keng Liao, and Alok Choudhary. Delegation-based I/O Mechanism for High Performance Computing Systems. IEEE Transactions on Parallel and Distributed Systems, vol. 23,no. 2, pp. 271–279, February 2012.
8. S. Lakshminarasimhan, J. Jenkins, I. Arkatkar, Z. Gong, H. Kolla, S.-H. Ku, S. Ethier, J. Chen, C. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. ISABELA-QA: Query-driven analytics with ISABELA- compressed extreme-scale scientific data. In Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis (SC11), Seattle, WA, November 2011.
9. Md Mostofa Ali Patwary, Diana Palsetia, Ankit Agrawal, Wei-keng Liao, Fredrik Manne, and Alok Choudhary. A New Scalable Parallel DBSCAN Algorithm Using the Disjoint-Set Data Structure. In the International Conference for High Performance Computing, Networking, Storage and Analysis, November 2012.

10. Seong Jo Kim, Seung Woo Son, Wei-keng Liao, Mahmut Kandemir, Rajeev Thakur, and Alok Choudhary. IOPin: Runtime Profiling and Optimization of Parallel I/O in HPC Systems. In 7th Parallel Data Storage Workshop, held in conjunction with the International Conference for High Performance Computing, Networking, Storage and Analysis, November 2012.
11. Sriram Lakshminarasimhan, Prabhat Kumar, Wei-keng Liao, Alok Choudhary, Vipin Kumar, and and Nagiza F. Samatova. On the Path to Sustainable, Scalable, and Energy-efficient Data Analytics: Challenges, Promises, and Future Directions. In the 2012 International Green Computing Conference, June 2012.
12. Lalith Polepeddi, Ankit Agrawal, and Alok Choudhary. Poll: A Citation-Text-Based System for Identifying High-Impact Contributions of an Article. In the Workshop on Data Mining in Networks, held in conjunction with the IEEE International Conference on Data Mining, December 2011.
13. Ankit Agrawal and Alok Choudhary. Identifying HotSpots in Lung Cancer Data Using Association Rule Mining. In the Workshop on Biological Data Mining and its Applications in Healthcare: Prediction, Extremes, and Impacts, held in conjunction with the IEEE International Conference on Data Mining, December 2011.
14. Kathy Lee, Diana Palsetia, Md. Mostofa Ali Patwary, Ankit Agrawal, Alok Choudhary, and Ramanathan Narayanan. Twitter Trending Topic Classification. In the Workshop on Optimization Based Methods for Emerging Data Mining Problems, held in conjunction with the IEEE International Conference on Data Mining, December 2011.
15. Yu Cheng, Kunpeng Zhang, Yusheng Xie, Ankit Agrawal, Wei-keng Liao, and Alok Choudhary. Learning to Group Web Text Incorporating Prior Information. In the Workshop on Optimization Based Methods for Emerging Data Mining Problems, held in conjunction with the IEEE International Conference on Data Mining, December 2011.
16. Kui Gao, Chen Jin, Alok Choudhary, and Wei-keng Liao. Supporting Computational Data Model Representation with High-performance I/O in Parallel netCDF. In the IEEE International Conference on High Performance Computing, December 2011.
17. William Hendrix, Isaac Tetteh, Ankit Agrawal, Fredrick Semazzi, Wei-keng Liao, and Alok Choudhary. Community Dynamics and Analysis of Decadal Trends in Climate Data. In the Workshop on Knowledge Discovery from Climate Data: Prediction, Extremes, and Impacts, held in conjunction with the IEEE International Conference on Data Mining, December 2011.
18. Kunpeng Zhang, Yu Cheng, Yusheng Xie, Ankit Agrawal, Diana Palsetia, Kathy Lee, Wei-keng Liao, and Alok Choudhary. SES: Sentiment Elicitation System for Social Media Data. In the Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction, held in conjunction with the IEEE International Conference on Data Mining, December 2011.
19. Yuhong Zhang, Md. Mostofa Ali Patwary, Sanchit Misra, Ankit Agrawal, Wei-keng Liao, and Alok Choudhary. Enhancing Parallelism of Pairwise Statistical Significance Estimation for Local Sequence Alignment. In the Workshop on Hybrid Multicore Computing, held in conjunction with the IEEE International Conference on High Performance Computing, December 2011.
20. Chen Jin, Saba Sehrish, Wei-keng Liao, Alok Choudhary, and Karen Schuchardt. Improving the Average Response Time in Collective I/O. In the 18th EuroMPI Conference, September 2011.
21. Kunpeng Zhang, Yu Cheng, Wei-keng Liao, and Alok Choudhary. Mining Millions of Reviews: A Technique to Rank Products Based on Importance of Reviews. In the International Conference on Electronic Commerce, August 2011.
22. 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.