4-5-2011

# Object-Based Caching for MPI-IO

Phillip M. Dickens
*Principal Investigator; University of Maine, Orono*, dickens@umcs.maine.edu

**Final Report for Period:** 06/2010 - 12/2010 **Submitted on:** 04/05/2011
**Principal Investigator:** Dickens, Phillip M. **Award ID:** 0702748
**Organization:** University of Maine
**Submitted By:**
Dickens, Phillip - Principal Investigator
**Title:**
Object-Based Caching for MPI-IO

## Project Participants

### Senior Personnel

**Name:** Dickens, Phillip
**Worked for more than 160 Hours:** Yes
**Contribution to Project:**
Dr. Dickens is the project PI and has supervised all project activities.

### Post-doc

### Graduate Student

**Name:** Logan, Jeremy
**Worked for more than 160 Hours:** Yes
**Contribution to Project:**
Jeremy Logan is a full time Research Assistant who is supported through this grant
(Award--
0702748). He was a Ph.D. student working on his doctoral thesis in the area of object-
based
caching for MPI-IO. He received his Ph.D. in January of 2011, and currently holds a post-doctoral position at Oak Ridge National Laboratory.

**Name:** Murphy, Joshua
**Worked for more than 160 Hours:** Yes
**Contribution to Project:**
Joshua Murphy was listed as a graduate student from 6/1/08 - 11/15/08 although he had not completed his undergraduate Capstone Project. For his project, Josh implemented the distributed lock management system for cache objects. He presented his completed project on November 15th, 2008. During the summer of 2008 Josh was funded through a grant from the Maine Space Grant Consortium. Beginning in the fall semester, he was (and continues to be) funded by the department. His participation in the project is now complete.

### Undergraduate Student

### Technician, Programmer

### Other Participant

### Research Experience for Undergraduates

**Name:** Henderson, Julius
**Worked for more than 160 Hours:** Yes

**Contribution to Project:**

Julius worked on the project over the spring and summer of 2008. He worked on the implementation of one-sided communication mechanisms on the SiCortex SC648 supercomputer. This work supported data and meta-data movement between cache managers, which, in turn, supported cache consistency semantics. Julius graduated in the spring of 2010.

**Years of schooling completed:** Other

**Home Institution:** Same as Research Site

**Home Institution if Other:**

**Home Institution Highest Degree Granted(in fields supported by NSF):** Doctoral Degree

**Fiscal year(s) REU Participant supported:** 2008

**REU Funding:** REU supplement

**Name:** Deane, Tristan

**Worked for more than 160 Hours:** Yes

**Contribution to Project:**

Tristan Dean was an undergraduate student supported through the NSF REU program, and implemented multi-threaded support for MPI on the SiCortex SC648. He completed his undergraduate degree in December of 2008 and is no longer participating in the project.

**Years of schooling completed:** Other

**Home Institution:** Same as Research Site

**Home Institution if Other:**

**Home Institution Highest Degree Granted(in fields supported by NSF):** Doctoral Degree

**Fiscal year(s) REU Participant supported:** 2008

**REU Funding:** REU supplement

**Name:** Lamond, William

**Worked for more than 160 Hours:** Yes

**Contribution to Project:**

Will participated in this project during the summers of 2008 and 2009. He worked on the infrastructure of the University of Maine Scientific Grid Portal and in the porting of (MPI) parallel climate change models to the Portal. The goal of this work is to utilize the caching system to improve the I/O performance of such parallel models.

**Years of schooling completed:** Sophomore

**Home Institution:** Same as Research Site

**Home Institution if Other:**

**Home Institution Highest Degree Granted(in fields supported by NSF):** Associate's Degree

**Fiscal year(s) REU Participant supported:** 2009 2008

**REU Funding:** REU supplement

## Organizational Partners

## Other Collaborators or Contacts

I have interacted with climate change researchers within the University of Maine to parallelize their simulation models and provide realtime, remote visualization of current model state. This work is described the July, 2009 issue of Scientific Computing.

## Activities and Findings

**Research and Education Activities: (See PDF version submitted by PI at the end of the report)**

**Findings: (See PDF version submitted by PI at the end of the report)**

**Training and Development:**

Three undergraduates and one graduate student worked on this research over the course of this project. For all of the undergraduate students, this project was their first exposure to computer science research. These students gained valuable experience in high-performance computing, an area in which they had no prior knowledge. They developed an understanding of the need for high-performance computing, the technical challenges that come with designing and implementing parallel/distributed software systems, and learned how to function as a member of a research team. They also learned how their individual contributions fit into the larger context, and benefited significantly from developing, as a team, a larger and significantly more powerful system than they could have developed alone. All three undergraduate students used this research experience as the basis for their Capstone projects.

All of the undergraduates are now either gainfully employed in the IT sector of the economy or attending graduate school. Tristan Dean is working with a large defense contractor in the Boston area. Julius Henderson is now the leader of a software development team at a local hospital. Josh Murphy
is now attending graduate school in Computer Science at the University of Colorado.

Jeremy Logan received his Ph.D. in Computer Science in January of 2011.
He showed tremendous growth in his research skills over
the course of this project, as well as his mentoring skills for the undergraduate students working in the lab.
He is currently in a post-doctoral position at Oak Ridge National Laboratory where he is working with the High Performance I/O group. There is no question that his work on this project prepared him very well for such a research position.

**Outreach Activities:**

## Journal Publications

Dickens, PM; Logan, J, "A high performance implementation of MPI-IO for a Lustre file system environment", CONCURRENCY AND COMPUTATION-PRACTICE & EXPERIENCE, p. 1433, vol. 22, (2010). Published, 10.1002/cpe.149

## Books or Other One-time Publications

Dickens, Phillip M., "Computing Climate Change: Just the Tip of the Iceberg", (2009). JOURNAL/Magazine, Published
Collection: Scientific Computing
Bibliography: July, 2009

## Web/Internet Site

## Other Specific Products

## Contributions

**Contributions within Discipline:**

My principal disciplinary field is Computer and Information Science and Engineering (CISE).

The primary contributions to my discipline are as follows:

1. The overriding contribution of this research is the development of a new model for file data, termed object-based files, which replace the legacy view of a file as a linear sequence of bytes. The object-based file model incorporates information about the file access patterns and is much more closely aligned with the way in which application processes access their data. This new file paradigm has been shown to significantly increase the performance and scalability characteristics of MPI-IO, and addresses many of the challenges associated with parallel I/O.

2. One of the major challenges facing MPI-IO developers is providing high-performance, scalable I/O to MPI applications executing in Atomic mode. This is a notoriously difficult problem, but our research has shown that object-based caching, combined with a lightweight distributed locking mechanism, can provide up to a nine-fold increase in I/O performance (compared to traditional approaches) in this difficult environment.

3. One of the most common file access patterns in scientific applications is overlapping, strided I/O, which is also quite challenging to implement in an efficient manner. However, our approach has been shown to handle this file access pattern quite well, and we have observed up to a four-fold increase in performance compared to native MPI-IO.

4. Having the capability to support cooperating applications can be a very useful tool, particularly with respect to the visualization of scientific data. The translator mechanism developed over the course of this research can provide such functionality, and can do so quite efficiently. Its efficiency stems from the use of interval trees to store and retrieve information about the different object sets.

5. We have identified the reasons behind the poor performance of MPI-IO in a Lustre file system environment. While it has been well documented that MPI-IO performs very poorly in Lustre, the reasons for such performance have not been well understood. This research has identified the fundamental problem as the assumptions upon which most of the parallel I/O optimizations have been based. In particular, the assumption that I/O performance is optimized when performed in large, contiguous blocks simply does not hold in a Lustre environment.

6. Having identified the problem, this research has developed a new user-level library (Y-Lib) that optimizes the performance of MPI-IO in Lustre File Systems. Y-Lib redistributes data in a way that is more closely aligned with the Lustre storage architecture thereby reducing communication overhead and improving performance. We have shown that using this library can result in performance improvements of up to a factor of 30 when compared to unmodified MPI-IO.

**Contributions to Other Disciplines:**

This research is making significant contributions in the general domain of high performance computing. Much of the HPC research utilizes MPI for data-intensive scientific applications executing on large-scale parallel systems.  Improving the performance and scalability characteristics of MPI-IO will directly impact the productivity of such research.

**Contributions to Human Resource Development:**

This research has provided valuable training and experience in high-performance computing technologies for the three undergraduate and one graduate student that have participated in the project. This is especially important for the state of Maine, which needs a well-trained workforce in cutting-edge technologies to attract  new IT business development
in the state.

All three undergraduates have now graduated, and two of them are working in the IT sector, one in the Boston area and one in the state of Maine. The other undergraduate student is now pursuing a doctorate in Computer Science at the University of Colorado.

Jeremy Logan  received his Ph.D. in Computer Science, and is now in a post-doctoral position at Oak Ridge National Laboratory. He is working with the high-performance i/O group at this prestigious lab.

I believe that all of these students benefited significantly from the research experience provided through their participation in this project. I also believe that their work on this project contributed to their succeses after graduation.

**Contributions to Resources for Research and Education:**

While not directly purchased through this grant, its funding has paved the way for other investments for research and education. This includes a 648-node 'green supercomputer' from NSF Grant #0723093 and a cluster of four Tesla Graphics Processing Units (GPUs) and two display walls from the University of Maine System 2010 Strategic Investment Fund.

**Contributions Beyond Science and Engineering:**

The University of Maine has a world-renowned research group in the field of global climate change. Along with climate data collected over decades and tools to access and analyze such data, these researchers have also developed several simulation models to explain and  predict climatic changes. One problem, however, is that these models are largely sequential in nature, which significantly restricts the resolution at which such models can be executed. For example, the University of Maine Ice Sheet Model (UMISM, developed by Jim Fastook), cannot accommodate the new, five-kilometer resolution data that is now available for the Antarctic Ice Sheet. Another problem is that the output data is only visualized after the simulation has completed its execution. This precludes interactive access to the simulation, and powerful techniques such as simulation steering, where simulation activity is dynamically steered to particular areas of interest.

We are working to incorporate the object-based caching system with important climate change models being developed within the University. We believe we can significantly increase the resolution of such simulations and reduce execution time. Such models also present an excellent opportunity to utilize the translator mechanism to perform real-time remote visualization. In particular, the data can be captured between time steps, and pushed to the display wall in the format that will provide the best display speed and resolution. We have performed preliminary research suggesting this approach is workable and, once the translator is tuned for performance, efficient enough to operate in real time.

## Conference Proceedings

Logan, J;Dickens, PM, Using object based files for high performance parallel I/O, "SEP 06-08, 2007", IDAACS 2007: PROCEEDINGS OF

THE 4TH IEEE WORKSHOP ON INTELLIGENT DATA ACQUISITION AND ADVANCED COMPUTING SYSTEMS: TECHNOLOGY AND APPLICATIONS, : 149-154 2007

Dickens, PM;Logan, J, Y-Lib: A User Level Library to Increase the Performance of MPI-IO in a Lustre File System Environment, "JUN 11-13, 2009", HPDC'09: 18TH ACM INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, : 31-37 2009

Logan, J;Dickens, P, Improving I/O Performance through the Dynamic Remapping of Object Sets, "SEP 21-23, 2009", 2009 IEEE INTERNATIONAL WORKSHOP ON INTELLIGENT DATA ACQUISITION AND ADVANCED COMPUTING SYSTEMS: TECHNOLOGY AND APPLICATIONS, : 259-265 2009

Logan, J;Dickens, P, Towards an Understanding of the Performance of MPI-IO in Lustre File Systems, "SEP 29-OCT 01, 2008", 2008 IEEE INTERNATIONAL CONFERENCE ON CLUSTER COMPUTING, : 330-335 2008

Dickens, P;Logan, J, Towards a High Performance Implementation of MPI-IO on the Lustre File System, "NOV 09-14, 2008", ON THE MOVE TO MEANINGFUL INTERNET SYSTEMS: OTM 2008, PART I, 5331: 870-885 Part I 2008

## Categories for which nothing is reported:

Organizational Partners

Activities and Findings: Any Outreach Activities

Any Web/Internet Site

Any Product

**Project Activities and Findings Award**—0702748

**Project Overview**

The goal of this research is to provide high-performance, scalable I/O for large-scale scientific computing applications. The problem is that the I/O requirements of such applications are, in many cases, outpacing the capabilities of even the most powerful file systems available today, and are becoming a significant bottleneck in application performance. The most often cited reasons for such poor performance include the I/O access patterns exhibited by scientific applications (e.g., non-contiguous I/O), poor file system support for parallel I/O optimizations, the high cost of enforcing strict file consistency semantics, and the latency of accessing I/O devices across a network. However, it has been our hypothesis that a more fundamental problem, whose solution would help alleviate all of these challenges, is the legacy view of a file as a linear sequence of bytes. The problem is that application processes rarely access data in a way that matches this file data model, and a large component of the scalability problem is the cost of dynamically translating between the process data model and the file data model at runtime. In fact, the data model used by applications is more accurately defined as an *object model*, where each process maintains a set of perhaps unrelated objects. It has been our belief that aligning these two different data models will significantly enhance the performance of parallel I/O for data-intensive scientific applications.

To address this issue, we have developed a more powerful *object-based* file model, which is much more closely aligned with the application's I/O access patterns. Additionally, we have integrated the support system for this new file model into the ROMIO version of the MPI-IO standard. A critical component of this support system is the *object-based caching system* that provides an interface between MPI applications and object-based files. Object-based files and the caching system are based on MPI file views, or, more precisely, the intersections of these views. The idea is that such intersections, which is what we term *objects*, can identify all of the file regions within which conflicting accesses are possible, and, by extension, all of those regions for which there can be no conflicts. This information is leveraged by the underlying runtime system to significantly increase the parallelism of file accesses and decrease the cost of enforcing file consistency semantics and global cache coherence. This, in turn, provides new opportunities to support the I/O requirements of current and next-generation data-intensive applications.

**Project Successes**

We have been very successful in meeting (and, in fact, exceeding) the goals of this research. We have developed and implemented object-based files and the runtime infrastructure necessary to support this new file model. This includes the object-based caching system, an object-based distributed locking system, and a translator mechanism capable of translating between different object sets. Further, we have integrated a prototype runtime system into the ROMIO version of the MPI-IO standard. We have performed a large number of experimental studies, across a wide range of difficult problems, which show that our object-based system can provide significantly better performance and scalability characteristics than current approaches are able to provide.

We have demonstrated quite clearly that the fundamental hypothesis of this research is true: Moving away from the legacy view of a file as a linear sequence of bytes, and toward a more powerful object-based file model, does, in fact, provide significantly better I/O performance. Further, this new approach does, to a large extent, mitigate all of the issues that are commonly cited as the causes of poor I/O performance. We have also shown that basing objects on MPI file views does provide critical information about which file regions must be locked and which regions do not require locking, which significantly simplifies the design of the locking system and reduces the overhead of locking. These are powerful results.

During the course of this project, we gained significant insight into the impact on performance resulting from a mismatch between the application data model and the file data model. This insight enabled us to diagnose a long-standing problem related to the poor performance of data-intensive MPI applications executing in a Lustre file system environment, the reasons for which had hitherto not been understood. We solved this problem through the development of a high-performance ADIO driver for Lustre file systems, resulting in up to a 1000% increase in performance when compared to current approaches. This is a particularly important contribution because Lustre and MPI-IO are both critical tools for high-performance scientific applications.

Jeremy Logan, the Ph.D. student funded through this grant, completed his Ph.D. requirements in January of this year (2011), and is now in a postdoctoral position at Oak Ridge National Laboratory working with the high-performance I/O team. The three undergraduate students that participated in the project have all now graduated, and are either gainfully employed in the IT sector or pursuing graduate degrees in Computer Science.

**Project Activities**

We have produced excellent results over the term of this project. We have completed the implementation and integration of all components of the object-based caching system and have significantly improved its performance and functionality. This includes the integration of a high-performance distributed locking mechanism with which the cache can enforce strict file consistency semantics. We have also developed Y-Lib, a high-performance ADIO driver for Lustre file systems, which addresses a particular problem that was causing very poor I/O performance in this environment. We have also completed a prototype implementation of what we term the *translator*, which, as will be discussed below, extends the functionality of the caching system and provides a mechanism that can support *cooperating applications*.

We have performed extensive experimentation to evaluate the correctness and performance of the system utilizing the Tungston cluster housed at the National Center for Supercomputing Applications (NCSA), the Ranger and Lonestar clusters housed at the Texas Advanced Computing Center (TACC) and the BigRed cluster located at Indiana University. These experimental studies have demonstrated the power and flexibility of this new approach.

**Object Based Caching System**

We have successfully completed the object-based caching system and its integration into the ROMIO implementation of the MPI-IO standard. We have performed extensive experimentation throughout the development of the caching system, testing for both correctness and performance. Our initial experimentation focused on the benefits of using the caching system to write large checkpoint files. This is a critical issue in high-performance computing centers where large-scale simulations may take weeks or months to complete their execution, while the mean time to failure in the computing system is measured in hours or days. Thus long-running applications periodically save their state to enable them to restart their computation in the event of a failure. It is widely noted, however, that producing such checkpoint files is becoming a dominant cost for such applications.

We utilized the FLASH I/O benchmark, which directly addresses the issue of writing large checkpoint files, to investigate the performance of the caching system with respect to this important issue. The initial experiments utilized the caching system but wrote the checkpoint files to disk as linear files. We observed up to a 40% improvement in performance, when compared to unmodified MPI-IO, when using the caching system. We then extended these experiments to evaluate the benefits of storing checkpoint files as objects rather than as a linear sequence of bytes. We further extended the study by comparing this approach to both HDF5 and PNetCDF. These experiments demonstrated that storing files as objects improved performance by up to a *factor of four* when compared to HDF5, and a *factor of thirty* when compared to PNetCDF. These are impressive results, and show that saving checkpoint files as objects can provide tremendous performance benefits.

All of these experiments were conducted on TeraGrid facilities, primarily on the large-scale computing clusters at the National Center for Advanced Applications (NCSA) and the Texas Advanced Computing Center (TACC).

**High Performance Distributed Locking Mechanism**

We next completed the development of an efficient distributed locking system. One important goal of this research was to provide to MPI applications a high-performance implementation of MPI Atomic mode. Providing such an implementation is a significant challenge, and it is very well documented that applications using Atomic mode generally perform quite poorly. One factor that can lead to such poor performance is the granularity of the locking mechanism. For example, some file systems only support locking at the file level, which serializes access to the file. Another issue is that locking mechanisms are implemented at the file system layer, making it expensive to acquire and manipulate locks. Yet another reason for poor performance is that current techniques tend to suffer from high levels of false sharing, where a file region is locked even though there is no possibility of conflicting I/O operations within that region. This can occur because of the granularity of the locking system (e.g., can only lock at the block level), and/or because there is not enough information to know that conflicting accesses are not possible.

We have made significant progress in addressing all of these issues over the course of this research. We implemented a user level locking system that removes the need for expensive calls to the file system. The granularity of the locking mechanism is at the object level, which significantly reduces (or eliminates all together) the problem of false sharing. The fact that the locking system knows which objects are private and which are shared, and thus only locks the shared objects, can significantly reduce the overhead of locking. The fact that objects are non-overlapping, coupled with the fact that the set of processes that can access a given object is known at object creation time, means that there is no need for a centralized lock manager. This makes the object-based locking system extremely efficient.

We performed three sets of experiments to evaluate the design of our locking system and to track its performance when executing in MPI Atomic mode. All of these experiments were executed on the Ranger cluster at TACC. The first set was designed to compare the performance of our system to that of native MPI-IO using the MPI-Tile-IO benchmark executing in Atomic mode. The performance of the locking system was excellent, and we observed up to a *nine-fold* increase in performance in the case of file writes when compared to native ROMIO. The results were not so dramatic in the case of file reads, although we still observed up to a 35% improvement in performance.

In the second set of experiments, we investigated the performance of our system using an overlapping, strided I/O access pattern. This is a very common access pattern in scientific applications, which can result in significant levels of false sharing. The benchmark varies the size of the file region that must be locked to guarantee the strict file consistency semantics required by MPI Atomic mode. In these experiments, we observed that our system provided up to a *four-fold* increase in performance compared to native MPI-IO. The difference in performance is most likely due to our ability to lock at the object level, coupled with information about the application's access patterns. These two factors significantly reduced the file regions that required locking, resulting in significantly higher performance.

The third set of experiments investigated the performance of the locking mechanism itself. In particular, we looked at the impact on performance as a function of the number of lock managers. We observed that the number of lock managers did, in fact, have a very significant impact on performance, depending on the file access patterns and the number of application processes. Future work will investigate the development of simple models that can guide our choice on the number of lock managers to use as a function of these two parameters.

**Translator Mechanism**

Over the course of this research, we have demonstrated quite clearly that writing an application's data to disk as a set of objects can be significantly faster than writing the data to disk as a linear sequence of bytes. We have also shown that reading an object-based file is similarly efficient when the object set required by the reader matches the object set used to create the file. However, if there is a mismatch in the object sets of the reader and writer, then a translation between the two object sets must be performed. We have developed what we term a *translator mechanism* to carry out this mapping. The key question we have investigated is whether the cost of performing the translation outweighs the benefits of writing the file as an object file.

We conducted two sets of experiments to investigate this issue using the MPI-Tile-Reader/Writer benchmark, which models two applications coordinating their activity to visualize scientific data. A producer application consists of a set of processes that generate a dense two-dimensional set of pixel data that is written to a shared file (MPI-Tile-Writer), and the consuming application consists of a set of processes that read the pixel data from the shared file and display the data on a tiled wallboard (MPI-Tile-Reader). The tiled wallboard consists of a set of individual monitors that together display the entire image. Adjacent monitors (in the horizontal and vertical directions) share a column of pixel data to help blend the individual components of the image into a smoother aggregate image. The object sets of the tile reader and tile writer are different in this application.

The first set of experiments compared the time required to write the pixel data to disk as an object file versus a linear file (using unmodified ROMIO). The difference in performance was quite dramatic, where writing the file as an object file was up to *nine* times faster than writing it as a linear file. The second set of experiments investigated the time required to perform the translation between the two different object sets. We observed that even with the translation costs, our approach was still over *five* times faster than that obtained by unmodified ROMIO. We conclude from these experiments that the benefits of creating object-based files can outweigh the cost of performing a translation, and, in at least some cases, by a significant amount.

The translator mechanism also opens the door to the dynamic remapping of object streams, which we believe will be a very useful capability. For example, it can support cooperating applications, where one application writes its data to disk in the format required by another application. In fact, the visualization application discussed in this section is a good example of such cooperation between applications. The investigation of cooperating applications began during the course of this project and will continue into the future.

**High Performance ADIO Driver for the Lustre File System**

It is important to note that while the concepts of object-based caching and object-based files are related to object-based file systems (such as Lustre and Panasas) that store files as objects they are not the same. In particular, the objects we create and store are derived from the I/O access patterns as described by the MPI file views. Such objects are created, managed, and stored transparently to the user. The objects stored in file systems such as Lustre are created by the user, and generally consist of a single stripe of a file. They do not contain any information about file access patterns, and, in general, do not reside in the file in a way that matches its use by the application. However, the object-based cache is well suited to such file systems, and we have shown that our approach can significantly increase the performance of MPI in such environments.

This is a particularly important issue with respect to MPI executing in Lustre file systems. Lustre is becoming an increasingly important file system for large-scale computing clusters, but the performance of MPI-IO in this environment is quite poor. While this poor performance has been well documented, the reasons for such performance have heretofore not been well understood. We have been studying this problem, with the goal of aligning our object-based caching system with the Lustre object-based file model.

We have performed extensive experimentation showing that the primary performance issues have to do with the assumptions underpinning most of the parallel I/O optimizations implemented in MPI-IO, which do not appear to hold in a Lustre environment. Perhaps the most important assumption is that optimal performance is obtained by performing large, contiguous I/O operations. Our research suggests that this is often the worst approach to take in a Lustre file system. In fact, we found that the best performance is often achieved when each process performs a series of smaller, non-contiguous I/O operations. We believe the reasons for these non-intuitive results have to do with the communication patterns between MPI-IO and Lustre's Object Storage Targets (OST). In particular, writing large, contiguous blocks often results in what we term an all-to-all communication pattern where each aggregator process is communicating with all OSTs. This results in significant contention at all levels of the system (network, locking, protocol processing), which apparently overwhelms the benefits of performing fewer, larger, I/O operations.

We have reached these conclusions based on extensive experimentation on four large-scale Lustre file systems on the TeraGrid: Tungston (NCSA), Lonestar (TACC), BigRed (Indiana University), and Ranger (TACC). In these studies, we implemented several data aggregation patterns, where each pattern varied the number of OSTs with which each aggregator process communicated. In every system studied, the experiments showed that the best I/O performance was achieved when there was a one-to-one or one-to-two communication pattern (i.e., each aggregator process communicated with exactly one or two OSTs respectively). It was also observed that the worst performance was obtained when there was a many-to-many communication pattern and when we used the native implementations of MPI-IO.

Over the course of this research, we have developed a user-level library (termed Y-Lib) that redistributes application data into the one-to-one OST pattern and then performs the requested I/O operation. Extensive experimentation on the Ranger cluster showed a speedup between 300% and 1000% when compared with un-modified MPI-IO. We also obtained significant speedups on BigRed (using the Data Capacitor), although not on the order of those obtained on Ranger (maximum of 36%). We are continuing our work in this area.

**Poster Presentations**

- Dickens, P. and Logan, J. Y-Lib: A User Level Library to Improve the Performance of MPI-IO in a Lustre File System Environment. Poster presentation: *HEC FSIO Research and Development Conference,* Arlington, Virginia, August 2009.
- Dickens, P. and Logan, J. Toward a High Performance Implementation of MPI-IO for the Lustre File System. Poster presentation: *HEC FSIO Research and Development Conference,* Arlington, Virginia, August 2008.
- Logan, J. and P. Dickens. Towards an Understanding of the Performance of MPI-IO on the Lustre File System. Poster presentation: *Cluster 2008*, September 29 to October 1, 2008, Tsukuba, Japan.

**Research Presentations:**

- Dickens, Phillip and Jeremy Logan. Y-Lib: A User Level Library to Increase the Performance of MPI-IO in a Lustre File System Environment. Presented at: The *International Symposium on High Performance Distributed Computing (HPDC 2009).* Munich, Germany, June 11th – 13th, 2009.
- Logan, J. and Dickens, P. Improving I/O Performance through the Dynamic Remapping of Object Sets. Presented at: The 5th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. August, 2009.
- Dickens, P. and Logan, J. Towards a High Performance Implementation of MPI-IO on the Lustre File System. Presented at: GADA'08: Grid computing, high-performance and Distributed Applications. Monterrey, Mexico, November 13 - 14, 2008.
- Logan, J., and Dickens, P. Using Object Based Files for High Performance Parallel I/O. Presented at: The *IEEE 4th International Workshop on Intelligent Data acquisition and Advanced Computing Systems: Technology and Applications*. Dortmund, Germany, September 6-8 2007.

## Major Findings

The research thus far has led to several important findings:

- Object-based caching can provide up to a nine-fold increase in the I/O performance of MPI applications executing in Atomic mode.
- Object-based caching can provide up to a four-fold increase in the I/O performance of MPI applications when using an overlapping, strided access pattern.
- The translator mechanism can provide the dynamic remapping of object sets, which can support, for example, what we term *cooperating applications*. In this paradigm, one application dynamically remaps its object set into the set required by another application.
- Object-based caching can significantly improve the performance of MPI-IO when periodically writing large checkpoint files. In fact, our research has demonstrated an increase in performance in the range of 400% to 3000% in the case of the FLASH I/O benchmark.
- The long-held assumptions about how to maximize parallel I/O performance are at the root of the poor performance of MPI-IO in Lustre file systems. In particular, the assumption that I/O performance is maximized when performed in large, contiguous blocks does not appear to hold in Lustre. The problem is that this leads to a many-to-many communication pattern, which results in significant contention within a Lustre file system.
- Our user level library (Y-Lib) provides high-performance, scalable I/O to MPI applications executing in a Lustre environment. It accomplishes this by managing the communication patterns. Y-Lib has been shown to increase the performance of MPI-IO by up to a factor of thirty by enforcing a one-to-one or one-to-two communication pattern.