Enabling High Performance Application I/O

Final Report

Northwestern University
Alok Choudhary (PI) and Wei-keng Liao (Co-PI)
Part of the Scientific Data Management Center

Summary

This work provides software that enables scientific applications to more efficiently access available storage resources at different levels of interfaces. We developed scalable techniques and optimizations for PVFS parallel file systems, MPI I/O, and parallel netCDF I/O library. These implementations were evaluated using production application I/O kernels as well as popular I/O benchmarks and demonstrated promising results. The software developed under this work has been made available to the public via MCS, ANL web sites.

Introduction

Today's scientific applications access datasets ranging from gigabytes (GB) to terabytes (TB), checkpoint frequently, and create large volumes of visualization data. Demanding high-performance I/O systems, such applications are hamstrung by bottlenecks anywhere in the I/O path, including the storage hardware, file system, low-level I/O middleware, application level interface, and in some cases the mechanism used for Grid I/O access. The goal of this project is to provide significant improvements in the parallel I/O subsystems used on today's machines. This work addresses inefficiencies in all the software layers by carefully balancing the needs of scientists with implementations that allow the expression and exploitation of parallelism in access patterns.

Sitting in between the storage devices and user applications, file systems play an important role on how to interpret application's I/O requests and to deliver the data efficiently. We developed several techniques for the Parallel Virtual File System (PVFS), including data types packing for multiple non-contiguous I/O requests, and native PVFS data type I/O. These methods reduce communication cost between application client processes and the I/O servers and hence significantly improve the performance.

MPI-IO has become the standard utility for parallel applications to obtain high-performance I/O. It is important for MPI-IO implementation to closely interpret high-level access information and utilize proper system functionalities to achieve the best data bandwidth. At the file system level, we have integrated the PVFS data types into ROMIO to enable applications to implicitly invoke the data type functionality. We further addressed scalable implementation issues for MPI file consistency and atomicity. We also designed and developed a client-side caching sub-system for MPI-IO. By placing a thin caching layer in MPI-IO library, we can capture high-level applications' I/O patterns, such as the information across multiple I/O calls, and perform file caching more effectively while maintaining data coherency.

Scientific applications desire more structured file formats that map closely to the data structures used, such as multidimensional datasets and their associated attributes. One of the standard file formats commonly used in scientific community is netCDF. NetCDF defines a file format that is portable across different platforms and provides a set of programming interfaces to access data stored in such format. Since only interfaces for sequential access are provided in netCDF, we constructed a new set of parallel I/O interfaces that were built on top of MPI-IO. The implementation ensures the file format conformation and backward compatibility to the legacy codes. Parallel netCDF library (pnetCDF) has been adopted by a number of application groups in the climate community, such as WRF at NCAR.

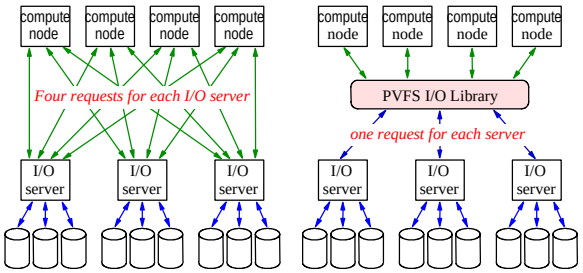


Figure 1 Comparison of data flow between the traditional I/O design and the PVFS list-IO design. The number of I/O requests to the I/O servers is reduced to one for each server through PVFS library.

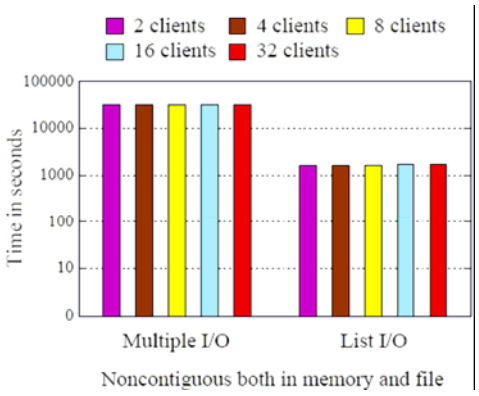


Figure 2 FLASH I/O results that compare between traditional multiple I/O and list I/O strategies. The performance of list I/O is approximately two orders of magnitude improved from the multiple I/O.

PVFS Data Types for Non-contiguous Access

Many scientific applications require non-contiguous access of small regions of file data. These access patterns can be described by MPI derived data types, which will be used in the MPI-IO operations later. However, they do not map well to the traditional block access interfaces of most file systems. Traditionally, parallel file systems perform each of the contiguous-region access as an individual I/O request to satisfy these types of operations, resulting in a large request-processing overhead. We implemented the PVFS's list-IO operation that internally combined multiple requests into single ones. This work demonstrated beneficial to many non-contiguous I/O patterns. The follow-on PVFS datatype I/O implementation used PVFS derived data types to describe non-contiguous I/O in a concise representation that can be handled by the file system natively and more efficiently. We completed this work for both client-side (MPI-IO library and client-side file system) and the server system side. Underneath, the communication between PVFS clients and servers for non-contiguous I/O requests, as depicted in Figure 1, is made through a derived data type, instead of a list of offset-length pairs as the traditional approach.

We benchmarked this work using three different test suites: an artificial benchmark, an I/O of the FLASH astrophysics application, and an I/O simulation of a tiled visualization application. Our results show that in most cases, list I/O outperforms traditional noncontiguous methods by up to two orders of magnitude. By providing file system support for more structured accesses, and supporting this mechanism in ROMIO, our performance improves by two or more orders of magnitude for many common multidimensional array accesses. An example performance results is given in Figure 2. This work is applicable to most applications running on clusters today and is available in current PVFS and ROMIO releases. The publications resulted from this work include [1,2,3,4,5,6].

Scalable Implementations for MPI File Atomicity and Data Consistency

MPI atomicity is defined as: in concurrent overlapping MPI-IO operations, the results of the overlapped regions shall contain data from only one of the MPI processes that participates in the I/O operations. Since each MPI-IO operation can cross multiple non-contiguous file regions, the overlaps can also contain non-contiguous regions. Traditional approach uses file locking to ensure the exclusive access to the conflicted accesses, which can easily serialize the I/O parallelism. We have designed scalable approaches to avoid the serialization by having processes negotiate with each other for overlapped access. The performance results demonstrated higher scalability obtained on multiple high-performance computers.

File locking is also used on the platforms where client-side file caching is performed to fulfill the requirement by the MPI file consistency

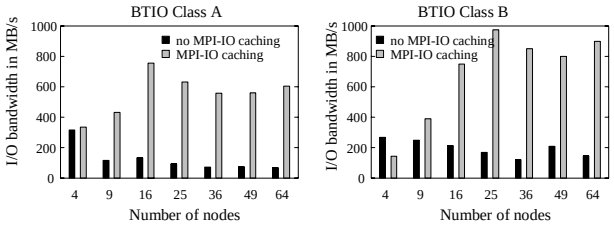


Figure 3. BTIO benchmark performance results for MPI-IO caching.

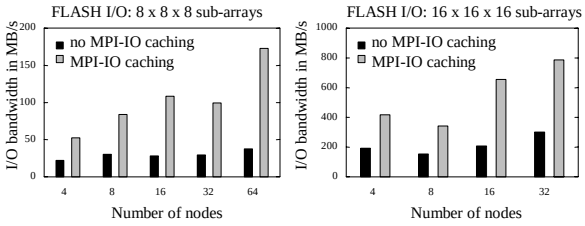


Figure 4. FLASH I/O benchmark performance results for MPI-IO caching.

semantics. Incoherent cache occurs when multiple copies of the same data are stored at different clients and a change to one copy does not propagate to others in time, leaving the cached data in an incoherent state. Since MPI semantics requires a write from one process to be immediately visible to all the processes that open the file, we propose a scalable approach, called *Persistent File Domain (PFD)*, which reuses the file access information from the preceding MPI-IO operations to guide the subsequent I/O to the processes that hold the most up-to-date cache. Its implementation has been embedded into ROMIO and scalable performance results were obtained for several I/O benchmarks. The publications resulted from this work include [7,8].

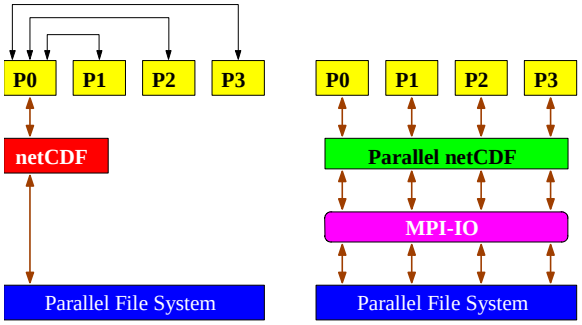
Client-side File Caching in MPI-IO

We developed a file caching sub-system at MPI library level in order to gain more control on data caching. The motivation came from the inadequate adoption by the parallel file systems from the traditional strategies that treat each client independently when performing client-side caching. Our strategy is to cooperate application processes to perform caching and coherence control without involving I/O servers. Leaving the I/O servers from coherence control would dramatically reduce the communication overhead and I/O burden at the servers. The techniques involved in this task include the management of distributed cache metadata, locks, local and global coherence control, and caching policies within the group of clients. We designed two approaches. The first is to create a POSIX thread at each I/O aggregator and lets the threads handle caching and all I/O operations at the background. The second approach is to utilize MPI remote-memory-access functions for coherence control. Since not all existing parallel machines support multi-threading and MPI RMA, we exercise the cooperated caching through these two methods and obtained a certain degree of performance improvement. Figures 3 and 4 give the results for BTIO benchmarking and FLASH I/O. Comparing to the runs relying on traditional file caching system, we obtained a great performance enhancement. The publications resulted from this work include [9,10,11,12,13,14,15].

Parallel NetCDF

NetCDF is a high level API widely used in the climate and fusion areas. The netCDF API is designed for serial codes to perform file I/O through a single processor as shown in Figure 5(a). Its interface does not define parallel access semantics and the implementation underneath does not support parallel I/O. We designed a new set of parallel APIs for accessing netCDF data sets and implement them using optimal I/O strategies such that I/O can be concurrently performed. Our new parallel API closely mimics the original API, but is designed with scalability in mind and is implemented on top of MPI-IO. The design infrastructure of this work is illustrated in Figure 5(b).

In our implementation, the access pattern of an I/O request is represented in an MPI file view constructed from the API arguments. Similar to MPI-IO, both collective and independent I/O calls are provided to users such that proper calls can be determined for application specific I/O requests. File access hint object is also added to the new API argument list and, in our parallel netCDF implementation, it is passed down to MPI-IO in



(a) Sequential file access (b) Concurrent file access

Figure 5. When using netCDF APIs, parallel I/O requires the root process perform the access to the file system for other processes. On the other hand, the parallel netCDF APIs that is built on top of MPI-IO can perform the writes concurrently and portably across different platforms.

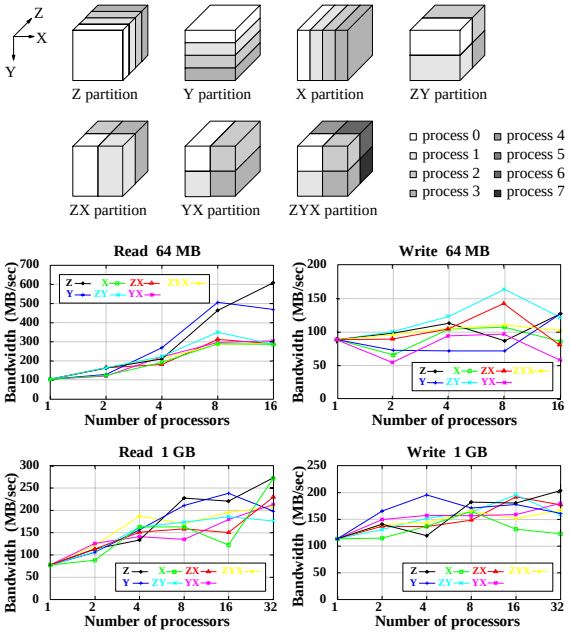


Figure 6. Performance results of parallel netCDF using a test suite that performs various block pattern accesses to a 3D array on the IBM SP at SDSC.

case certain I/O optimization and characteristics are available for a specific file system. Figure 6 gives the results for I/O that uses regular array partitioning patterns and pnetCDF clearly can provide a scalable performance. Since the parallel netCDF is built on top of MPI-IO and only MPI standard feature are used in its implementation, it is clear that the new APIs are portable. The publication on this work includes [11] and the software package of version 1.0.1 has been released to the public.

We have completed the following implementations. MPI derived data type support for flexible API that allows applications to describe flexible buffer memory layout. Mapped strided sub-array access functions are incorporated into the library. Fortran binding has been built and tested successfully. Test package includes flexible parallel APIs, collective var/vara/vars/varm functions and subarray/darray/vector MPI derived data types. PnetCDF is successfully installed in many TeraGrid parallel computers. It is supported across multiple MPI implementations, include MPICH, IBM MPI, MAVPICH and others. Figure 7 shows the performance of the I/O kernel of the astrophysics FLASH application. We compare the pnetCDF and HDF5 methods.

Future Plans

Distributed File Locking for PVFS

Many parallel scientific applications use high-level I/O APIs that offer atomic I/O capabilities. Atomic I/O in current parallel file systems is often slow when multiple processes simultaneously access interleaved, shared files. Current atomic I/O solutions are not optimized for handling noncontiguous access patterns because current locking systems have a fixed file system block-based granularity and do not leverage high-level access pattern information. We plan to develop a new lock protocol that takes advantage of new list and datatype byte-range lock description techniques to enable high performance atomic I/O operations for these challenging access patterns.

MPI-IO Caching

We plan to extend the MPI-IO caching work on the machines with intermediate I/O nodes between clients and I/O servers. We would like to

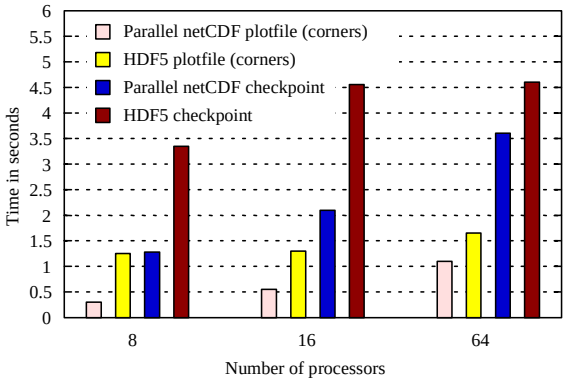


Figure 7. The I/O performance results of the FLASH astrophysics application using HDF5 and parallel netCDF. Two types of I/O are shown in FLASH: plotfile and checkpoint.

investigate the idea of cooperated caching on dedicate compute resource that run concurrently with I/O clients and servers.

Parallel NetCDF

We plan to support large file size and large array size. Currently netCDF file format and implementation uses 32-bit integers that limits the size of a file or array to 4GB. The large-size support will include an extended file format, adopting 64-bit integers throughout the implementation, and changes of using MPI-IO functions that support 64-bit integers. This software package continues to mature as application groups work with the software. We have checked the software into a revision control system and created a web page (via MCS at ANL) in order to provide the community with easy access to the software and to aid in debugging.

Personnel for Northwestern University

- Professor Alok Choudhary (PI)
- Professor Wei-keng Liao (Co-PI)
- Ph.D. students
    - Jianwei Li, graduated in 2006
    - Avery Ching, graduated in 2007
    - Kenin Coloma, graduated in 2007

Software

i. **Parallel Virtual File System**

http://www.pvfs.org/

ii. ROMIO MPI-IO Implementation

http://www.mcs.anl.gov/romio

iii. **Parallel NetCDF**

http://www.mcs.anl.gov/parallel-netcdf

Publications

1. A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp, “Noncontiguous I/O through PVFS,” in the Proceedings of *Cluster 2002*, September 2002.
2. A. Ching, A. Choudhary, K. Coloma, W. Liao, R. Ross, and W. Gropp, “Noncontiguous I/O Accesses Through MPI-IO,” in the Proceedings of *the CCGrid2003*, May 2003.
3. A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp, "Efficient Structured Access in Parallel File Systems," in the Proceedings of *the 2003 IEEE International Conference on Cluster Computing*, December 2003.
4. A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp. “Efficient Structured Data Access in Parallel File Systems, in the *International Journal of High Performance Computing and Networking*, issue 3, 2004.
5. A. Ching, W. Feng, H. Lin, X. Ma, and A. Choudhary. Exploring I/O strategies for parallel sequence database search tools with S3aSim. In Proceedings of the *International Symposium on High Performance Distributed Computing*, 2006.
6. A. Ching, A. Choudhary, W. Liao, L. Ward, and N. Pundit. Evaluating I/O characteristics and methods for storing structured scientific data. In Proceedings of the *International Parallel and Distributed Processing Symposium*, 2006.
7. W. Liao, A. Choudhary, K. Coloma, G. Thiruvathukal, L. Ward, E. Russell, and N. Pundit. Scalable Implementations of MPI Atomicity for Concurrent Overlapping I/O. In the Proceedings of the *International Conference on Parallel Processing*, 2003.
8. W. Liao, K. Coloma, A. Choudhary, L. Ward, E. Russell, and N. Pundit. Scalable Design and Implementations for MPI Parallel Overlapping I/O.

In the *IEEE Transactions on Parallel and Distributed System*, Volume 17, Number 11, pp. 1264-1276, November 2006.

9. W. Liao, K. Coloma, A. Choudhary, and L. Ward. Cooperative Write-Behind Data Buffering for MPI I/O. In Proceedings of the 12th European Parallel Virtual Machine and Message Passing Interface Conference, September 2005.

10. W. Liao, K. Coloma, A. Choudhary, L. Ward, E. Russel, and S. Tideman. Collective Caching: Application-Aware Client-Side File Caching. In Proceedings of the 14th International Symposium on High Performance Distributed Computing, July 2005.

11. K. Coloma, A. Choudhary, W Liao, L. Ward, and S. Tideman. DACheː Direct Access Cache System for Parallel I/O. In Proceedings of the 20th International Supercomputer Conference (ISC), June 2005.

12. W. Liao, A. Ching, K. Coloma, A. Choudhary, and L. Ward. Implementation and evaluation of client-side file caching for MPI-IO. In Proceedings of *the International Parallel and Distributed Processing Symposium*, March 2007.

13. W. Liao, A. Ching, K. Coloma, A. Choudhary, and M. Kandemir. Improving MPI Independent Write Performance Using A Two-Stage Write-Behind Buffering Method. In the Proceedings of *the Next Generation Software (NGS) Workshop, held in conjunction with the 21st International Parallel and Distributed Processing Symposium*, March 2007.

14. Avery Ching, R. Ross, W. Liao, L. Ward, and A. Choudhary. Noncontiguous locking techniques for parallel file systems. In Proceedings of *Supercomputing*, November 2007.

15. W. Liao, A. Ching, K. Coloma, A. Choudhary, J, Chen, R. Sankaran,, and S. Klasky. Using MPI file caching to improve parallel write performance for large-scale scientific applications. In Proceedings of *Supercomputing*, November 2007.

16. J. Li, W. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale. "Parallel netCDF: A Scientific High-Performance I/O Interface," in the Proceedings of *Supercomputing Conference*, November 2003.

17. Y. Liu, J. Pisharath, W. Liao, G. Memik, A. Choudhary, and P. Dubey. Performance Evaluation and Characterization of Scalable Data Mining Algorithms. In Proceedings of the *16th International Conference on Parallel and Distributed Computing and Systems*, November 2004.

18. K. Coloma, A. Choudhary, A. Ching, W. Liao, S. Son, M. Kandemir, and L. Ward. Power and Performance in I/O for Scientific Applications. In Proceedings of *the Next Generation Software Workshop (NGS), held in conjunction with the 21st International Parallel and Distributed Processing Symposium*, April 2005.

19. Y. Liu, W. Liao, and A. Choudhary. A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets. In Proceedings of the *9th Pacific-Asia Conference on Knowledge Discovery and Data Minin*,, May 2005.

20. J. Pisharath, W. Liao, and A. Choudhary. Design and Evaluation of Database Layouts for MEMS-Based Storage Systems. In Proceedings of *the International Database Engineering and Applications Symposium*, July 2005.

21. Y. Liu, W. Liao, and A. Choudhary. A Fast High Utility Itemsets Mining Algorithm. In the Proceedings of *the Workshop on Utility-Based Data Mining, held in conjunction with the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2005..

22. J. Li, A. Choudhary, N. Jiang, and W. Liao. Mining Frequent Patterns by Differential Refinement of Clustered Bitmaps. In Proceedings of *the SIAM International Conference on Data Mining*, April 2006.