DAFS Storage for High Performance Computing using MPI-I/O: Design and Experience

Vijay Velusamy, Anthony Skjellum
MPI Software Technology, Inc.
Email: {vijay, tony}@mpi-softtech.com

Arkady Kanevsky*†, Peter Corbett
Network Appliance
Phone: (781) 768-5395
Fax: (781) 895-1195
Email: {arkady, pcorbett}@netapp.com

A key goal of this effort is to demonstrate and develop heterogeneous and distributed computing technologies that are applicable to DoD and scientific communities, while maintaining and benefiting from industry standards that could be applied to high performance computing.

High performance computing systems based on clusters of compute nodes, connected via a high speed interconnect, are becoming popular for large-scale parallel applications, forming a highly scalable infrastructure. Most large-scale scientific applications are highly I/O-centric and have a tremendous need for a similarly scalable file I/O subsystem. DAFS is a well-defined high-performance protocol for file access across a network as well as set of APIs, uDAFS, for user level OS-bypassing application programming, designed to take advantage of RDMA-based transports, such as Virtual Interface Architecture (VIA), InfiniBand Architecture[1], and iWARP.

Although DAFS was not originally designed for parallel I/O, this effort aims to demonstrate that DAFS can easily used directly or extended for the creation of a parallel file system. Most parallel file systems are designed to use metadata for parallel files, stored as objects (often themselves files) separate from the data of those files, and the data of the individual files is striped across a number of different server nodes. It is envisioned that this effort would demonstrate the adoption of DAFS and its parallel features to high performance computing environments, strengthening the technology base, and providing an opportunity for the adoption of widely used technology to the largest high performance computing programs such as ASCI.

This effort utilizes ChaMPIon/Pro, an efficient commercial implementation of the MPI-1.2 standard for message passing interfaces [3], to demonstrate the applicability of DAFS for scalable I/O. The MPI I/O implementation in ChaMPIon/Pro is designed to support both parallelism, and portability [2]. Parallelism is achieved in the MPI I/O layer, by implementing the MPI I/O APIs, while the BAFS layer provides portability (Figure 1). BAFS consists of a set of APIs that provide the necessary functionality for parallel I/O. BAFS provides a thin abstract I/O interface between MPI I/O and DAFS. It provides a non-collective I/O interface that function independently for each MPI process created. MPI I/O operates on the communicator level, involving communication between multiple processes. The MPI I/O layer manages collective I/O and shared file pointers. This layer maintains data consistency across different processes and delivers file atomicity semantics.

The BAFS implementation benefits from early binding and persistency for repeated file access patterns that use the same data structure. In this case, the data structure is allocated and initialized only once. The user can then call the non-blocking data access routines to fetch the data repeatedly making use of the same data structure.

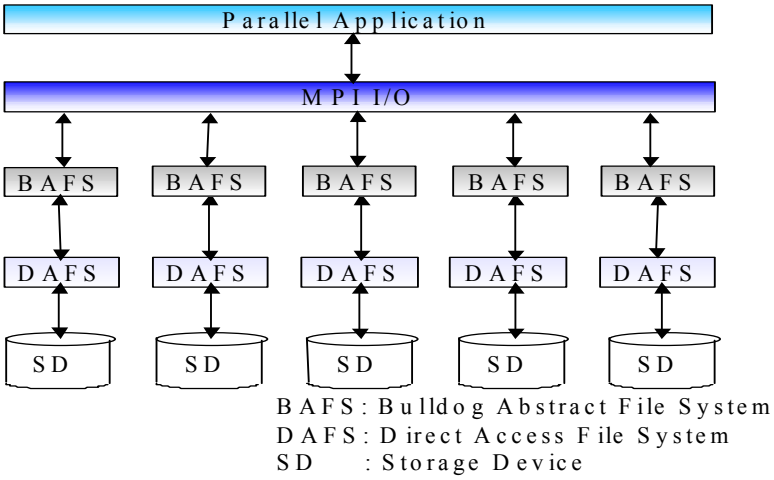
---

* Corresponding Author
† Presenting Author

In parallel file systems, files are generally striped across the server nodes, with each server node managing its own pool of disk space. File data is distributed by dividing the file into subfiles, referred to as cells [4]. Cells are distributed among the server nodes, so that each cell resides entirely on one server node, and zero, one or more cells of a file may reside on each of the server nodes in the parallel file server. The cell usually has an inode, including a blocklist, and the server node allocates space to each cell that it owns from its own pool of disk. In this effort for adapting DAFS for parallel I/O, a cell specific decomposed view of the parallel file is presented, so that the I/O library can perform the mapping of data to cells directly. Also file metadata servers are not separated from file data servers. In other words, the external metadata of a file, that is visible to the clients, including access control lists, cells locations, file attributes, are stored in a single metadata object, called metanode of the file. The details of the metadata need not be known to any application that needs to access the file.

The design of MPI I/O for DAFS allows the cache coherence and token/lock based read and write access control that are generally present in cluster file systems to be eliminated. There is no contention among the nodes of a parallel application for temporary access rights to data once the parallel program gains access rights to the entire file. This avoids any bottlenecks to restrict data flow, enhances the scalability of the system. Since DAFS is designed to benefit from high bandwidth low-latency RDMA access to file data, and because it can offload the client almost completely from performing file system and I/O tasks, the client CPU utilization for I/O is reduced tremendously, allowing the overlap of computation and I/O and better utilization of these extra CPU cycles.

It is expected that performance numbers for this effort would be available in September.



References:

[1] DAFS Collaborative, "Direct Access File System API Specification v1.0," http://www.dafscollaborative.org

[2] Kumaran Rajaram, Anthony Skjellum, Rossen P. Dimitrov, Purushotham V. Bangalore, Vijay Velusamy, and David Leimbach, "Design, Implementation, and Evaluation of a High Performance Portable Implementation of the MPI-2 I/O Standard API," submitted to Parallel Computing, November 2002.

[3] MPI Forum, "**MPI** - The Message Passing Interface Standard,"

http://www-unix.mcs.anl.gov/**mpi**/

[4] Peter Corbett, "DAFS Extensions for Parallelism," Network Appliance, August 2002.