Scalable Distributed Process Group Control and Inspection via the File System

Michael J. Brim and Barton P. Miller
{mjbrim,bart}@cs.wisc.edu
Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, U.S.A.

1 Research Overview

The size of distributed systems is rapidly expanding to meet the computational demands of the world. The largest current distributed systems for corporate intranets, high-performance computing (HPC) systems, and cloud computers contain tens of thousands of hosts. HPC systems on the horizon are expected to have counts in the millions. Developers of tools and middleware for distributed systems face the daunting task of enhancing or redesigning their software to operate at ever-increasing scale. Unfortunately, the group operation requirements of tools and middleware are often ignored when distributed systems are initially designed and deployed. As a result, each tool is forced to support the required group operations, leading to replication of effort and limiting the generality of these techniques and adoption by others.

As discussed in Section 2, many classes of tools and middleware share a requirement for scalable group operations on distributed files. My Ph.D. thesis, entitled "Scalable Distributed Process Group Control and Inspection via the File System", addresses the need for a common, scalable infrastructure for operating on large groups of distributed files using a new idiom, *group file operations*, that provides a simple, intuitive interface. I aim to provide solutions for the largest of current distributed systems, and techniques that will continue to scale for use with future systems.

The cornerstone of the group file operation idiom is a new `gopen` operation that provides a group file handle for use with existing file system operations (e.g., `read` and `write`). The key benefit of the group file abstraction is eliminating explicit iteration, which leads to serial behavior when applying the same operation to a group of files. A file-based idiom also promotes conciseness and portability for group operations. File operations are well-understood and intuitive, and support in programming languages and operating systems is ubiquitous. Also, file operations are data format agnostic and work on files containing binary data, text, or both. Despite the benefits, introducing group semantics for existing file operations presents several unique challenges. In particular, I address the interface and scalability issues associated with group status and data operands. My approach is to define intuitive group semantics for existing file operation interfaces and make extensions only when necessary.

Still, the group file operation idiom alone does not provide scalability when operating on large groups of distributed files. The mechanisms underlying group file operations also must be scalable. To this end, I have designed TBON-FS, a new distributed file system that provides scalable group file operations by leveraging tree-based overlay networks (TBONs) for scalable multicast distribution of group file operation requests and aggregation of group status and data results. Aggregation is a powerful and flexible technique for scalable analysis of the vast amount of data produced by tools and middleware at large scales. Figure 1 shows the proposed architecture of TBON-FS.

Throughout my research, I employ a practical evaluation methodology that involves prototyping and real-world integration of my ideas, as well as quantitative measurement of scalability and performance on large scale distributed systems. I also use qualitative observations gleaned from integrating my ideas into new and existing tools as inspiration for improving the usability and applicability of the group file operation idiom.

2 Research Significance

The overarching goal of my research is to advance the state of the art in the development of tools and middleware for large-scale distributed systems. Scalable tools and middleware are crucial for efficient use of HPC resources, as they provide the means for running user applications and problem diagnosis. Only by addressing the scarcity of tools and middleware that can effectively function on the largest of current and upcoming systems can we hope to maximize the efficient use of HPC systems. By providing an intuitive, general idiom and infrastructure for scalable group operations on distributed files, I hope to encourage the rapid development of new scalable tools, as well as allow existing tools and middleware to easily adopt a scalable solution, thereby ending unnecessary duplication of effort.

Several classes of tools and middleware share a requirement for operating on groups of distributed files. Distributed system management tools update or view software or configuration files across large groups, and middleware such as distributed monitoring uses process and host information found in files on each monitored system. Distributed computing middleware may need to distribute applications or data as files to groups of hosts, and collect groups of result files for analysis. On systems

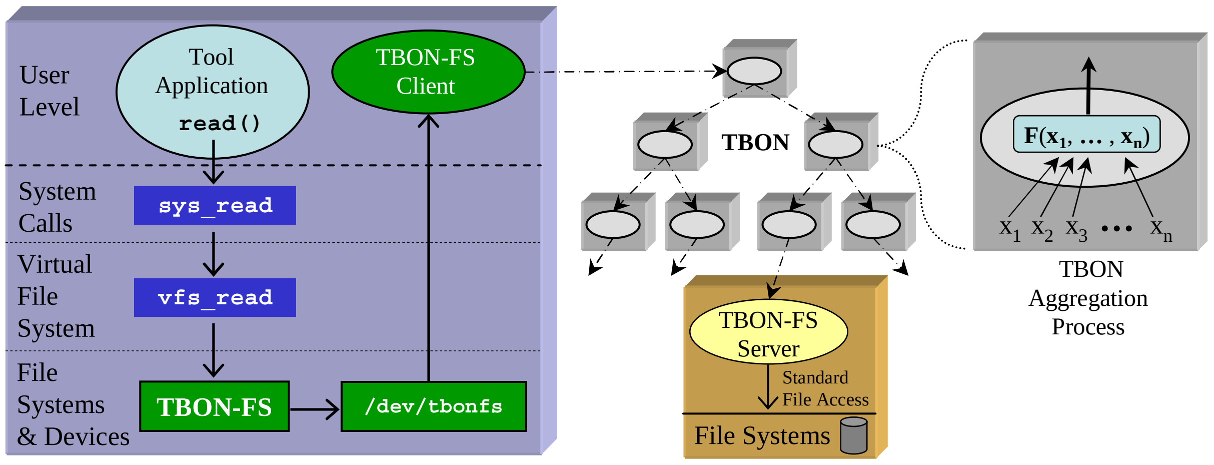

Figure 1 TBON-FS Architecture

An application `read` is a system call to the Virtual File System (VFS) layer, which maps the request to the TBON-FS file system. Requests are passed to a TBON-FS client process through a character device (`/dev/tbonfs`). The client forwards requests to servers via the TBON. The servers provide a simple file operation proxy service by transforming requests into local file system operations. Operation results are aggregated via the TBON and then returned to the application.

using a file abstraction for processes (e.g., the `/proc` file system on Plan 9, various UNIX systems, and Linux), process control and inspection involves operating on files. Application run-time environments [1][4][10] control distributed process groups, resource and performance monitoring tools [12][13][14][18][25] perform group process and host inspection, and distributed debuggers [9][26][28] and computational steering systems [15] require both group control and inspection.

3 Related Work

Group file operations on distributed files combine properties of both distributed and parallel file systems. Similar to distributed file systems, a client accesses files located on independent servers. Akin to parallel file systems, a single operation potentially involves multiple servers. Therefore, group file operations require concurrent access to a large set of independent file servers from a single client. The single-client, multiple-server model is distinctly different from existing parallel and distributed file systems. Parallel file systems [6][22][23] enable many cooperating clients to access large shared datasets that span few to many servers. Distributed file systems [11][21][24] provide many independent clients access to shared files exported from a small set of servers. Current distributed and parallel file systems focus on scaling the number of simultaneous clients that can be served, but the focus for group file operations is on scaling the number of independent servers that can be accessed concurrently from a single client. Due to the differences in client-server model, no existing distributed or parallel file system makes sense as a starting point for TBON-FS.

Like TBON-FS, CFS [7] and PAST [20] use overlay networks for achieving scalable file storage and retrieval. However, they only support read-only files and do not provide in-network aggregation of data from groups of independent files located across many servers. San Fermín [5] provides large-scale, fault-tolerant distributed data aggregation using a binomial swap forest established over a peer-to-peer overlay network. To achieve fault tolerance, each distributed data source exchanges aggregated data with peers to compute the final aggregate. Unfortunately, the overhead of this duplicate computation and communication is too high for many distributed systems, most notably HPC environments.

Group file operations are related to the MapReduce system [8], as both the map and reduce operations are distributed aggregation of file data. In MapReduce, a large data set is partitioned into many small fixed-size chunks stored at hundreds or thousands of servers. Considering chunks as files permits a MapReduce operation to be cast as a group file operation where both map and reduce are implemented as a single aggregation. Google's MapReduce system is tightly bound to the Google File System, as group file operations are currently tightly bound to TBON-FS. Group file operations are similar to the Sawzall [17] and Pig Latin [16] programming languages in the desire to expose data parallelism in a simple interface that hides the underlying parallel processing system. Unlike these languages, group file operations use familiar file system interfaces.

4 Progress and Future Work

After formal admittance to the Ph.D. program at the University of Wisconsin-Madison in December 2006, I investigated intuitive semantics for synchronous group file operations and started development on a user-level prototype of TBON-FS. In April 2007, I introduced the new group file operation idiom and its utility for tools and middleware at the annual Paradyn/Condor Week in a talk entitled “Group File Operations: A New Idiom for Scalable Tools”.

4.1 Case Study 1: Ganglia

Next, I began an evaluation of the group file operation idiom by integrating the prototype TBON-FS into the widely-used Ganglia Distributed Monitoring System [12]. With no prior knowledge of the Ganglia software, I finished this integration in a few weeks, effectively demonstrating the applicability of the idiom and its ease of adoption. The performance of the original Ganglia versus the TBON-FS version was compared on two cluster systems, the 150-node GLOW cluster in our CS department and the 1024-node (4096 processors) Thunder cluster at Lawrence Livermore National Laboratory (LLNL). Results showed that the TBON-FS version was able to significantly reduce the computational and network load on monitored hosts by eliminating the use of IP multicast among cluster nodes. IP multicast has poor scaling behavior that results in linear increases in resource utilization at each host as the size of the cluster grows.

4.2 Case Study 2: Parallel UNIX Tools

Having demonstrated the benefit for an existing middleware system, I proceeded to investigate the use of group file operations for rapid development of new scalable tools. Motivated by past research on parallel versions of common UNIX utilities and my prior work on cluster administration tools [2], I developed scalable parallel versions of five simple, yet powerful command-line tools: `cp`, `rsync`, `grep`, `tail`, and `top`.

File distribution is a common task when administering large clusters and distributed systems where individual hosts are similarly configured. There are two basic approaches to managing these homogenous configurations, file replication or file sharing. With replication, configuration files are kept on disks local to each host and must be updated when the global configuration changes. To improve the scalability of file replication, I have developed parallel versions of both `cp` and `rsync`. `pcp` uses group `write` operations to multicast an entire source file to all servers. `psync` uses a group `read` aggregation based on the `rsync` block checksum comparison algorithm [27] to identify differences between the servers’ copies of a file and the client source file, and only multicasts the data that has been changed or added.

System administrators use `grep` for various tasks including searching configuration files, scanning system and application logs for interesting or alarming events, and gathering system or process information. Leveraging `grep` in parallel on distributed files provides the ability to spot configuration differences, correlate distributed events, and monitor system resource use. The new `pgrep` tool supports simple textual search strings, as opposed to regular expressions. For scalable printing of results, a line-based equivalence aggregation is used that prepends the constituent files to each matched line.

The `tail` utility with the "-f" option is often used to follow system log activity in real-time, and a parallel version provides a simple form of distributed event correlation. Correlation of events across distributed hosts has many useful applications, including identifying misconfigured network services (e.g., multiple clients of the service notice a problem and generate an error) and security audits (e.g., distributed intrusion or denial-of-service attacks). `ptail` uses a version of the line equivalence aggregation used for `pgrep` extended with an option to strip host-specific information (e.g., hostname and process ids) from lines using the standard `syslog` message format.

The `top` utility is a simple yet powerful method for displaying real-time resource utilization by processes on a single host. A parallel `top` provides a powerful method for monitoring distributed resource utilization. To enhance the abilities of `ptop`, two new grouping facilities were included that enable display of summary information for processes with the same command name, both for a specific user and across all users. When grouping processes, the user has the option of viewing total, average, or maximum resource utilization. As shown in Figure 2, `ptop` is able to aggregate resource utilization for file groups consisting of hundreds of thousands of distributed processes (several hundred processes per host) using the same default delay interval as `top`.

Initial results on the performance and scalability of these new parallel tools were presented as part of my talk entitled “TBON-FS: A New Infrastructure for Scalable Tools and Runtimes” at Paradyn/Condor Week in April 2008. Full experimental results were collected using two LLNL clusters, Thunder and the 1024-node (8192 processors) Atlas. These results, as well as an interesting application of group file operations for distributed debugging, are reported in a paper [3] under submission.

4.3 Future Work

I plan to further my investigation of group file operations and TBON-FS along several dimensions. First, I will continue evaluating group file operations and TBON-FS within new and existing tools. A noteworthy target for evaluation is the TotalView Debugger [9], the most widely-used debugger for parallel applications. As part of a Department of Energy funded FAST-OS research proposal, and with direct technical guidance from

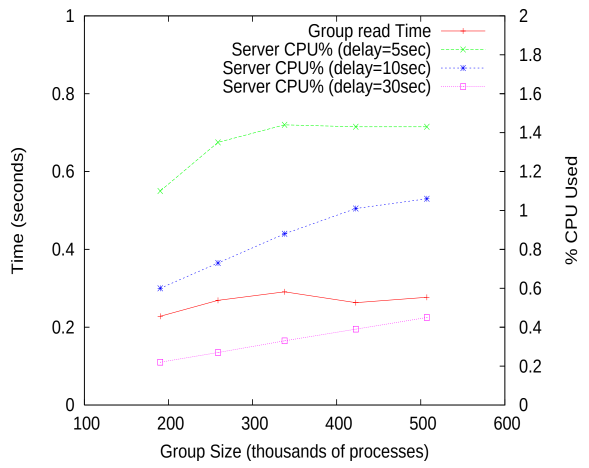

Figure 2 `ptop` Scalability
Server CPU Utilization and group `read` Time vs. number of monitored processes.

TotalView developers, I aim to use group file operations to improve the scalability of TotalView for use on the largest of current HPC systems and applications. The target system for this evaluation is Jaguar, a Cray XT system at Oak Ridge National Laboratory. Second, I will continue to identify the proper abstractions and semantics for the group file operation idiom. Specifically, I seek to define scalable abstractions for definition of file groups and study the impact of group file operations on higher level file access methodologies such as file streams. I also plan to evaluate the semantics of group file operations in the context of asynchronous I/O and other operations that create or use file descriptors (e.g., `select`, `socket`, and `mmap`). Finally, I wish to explore group file operations and TBON-FS within the context of a real client file system for Linux. Using a real file system will allow for validating my design for integrating group file operations within the Linux Virtual File System, measuring the impact to existing file systems, and observing performance.

5 References

[1] R. Brightwell and L. Fisk, “Scalable Parallel Application Launch on Cplant”, *SC 2001*, Denver, Colorado, November 2001.

[2] M. Brim, R. Flanery, A. Geist, B. Luethke, and S.L. Scott, “Cluster command & control (c3) tool suite”, *Parallel and Distributed Computing Practices* **4**, 4, December 2001, Nova Science Publishers, pp. 381-399.

[3] M. Brim and B.P. Miller, "Group File Operations for Scalable Tools and Middleware", University of Wisconsin Computer Sciences Technical Report TR1638, under submission, October 2008.

[4] R. Butler, W. Gropp, and E. Lusk, “A Scalable Process Management Environment for Parallel Programs”, Recent Advances in Parallel Virtual Machine and Message Passing Interface - *LNCS* **1908**, Springer, September 2000, pp. 168-175.

[5] J. Cappos and J. Hartman, "San Fermín: Aggregating Large Data Sets using a Binomial Swap Forest", *5th USENIX Symposium on Networked Systems Design and Implementation*, April 2008.

[6] P. Carns, W. Ligon III, R. Ross, and R. Thakur, "PVFS: A Parallel File System for Linux Clusters", *Fourth Annual Linux Showcase and Conference*, pp. 317-327, October 2000.

[7] F. Dabek, F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS", *SIGOPS Oper. Sys. Rev.* **35**, 5, December 2001, pp. 202-215.

[8] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", *6th Symposium on Operating Systems Design and Implmentation*, December 2004.

[9] “Debugger for Multi-core, Multi-Threaded, Multi-Processor Applications”, TotalView Technologies, LLC, http://www.totalviewtech.com/productsTV.htm, 2007.

[10] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, **PVM: A User’s Guide and Tutorial for Networked Parallel Computing**, MIT Press, 1994.

[11] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West, “Scale and Performance in a Distributed File System”, *ACM Transactions on Computer Systems* **6**, 1, February 1988.

[12] M. Massie, B. Chun, and D. Culler, “The Ganglia Distributed Monitoring System: Design, Implementation, and Experience”, *Parallel Computing* **30**, Elsevier B.V., 2004.

[13] B. Miller, M. Callaghan, J. Cargille, J. Hollingsworth, R. Irvin, K. Karavanic, K. Kunchithapadam and T. Newhall, “The Paradyn Parallel Performance Measurement Tool”, *IEEE Computer* **28**, 11, November 1995, pp. 37-46.

[14] R. Mooney, K. Schmidt, R. Studham, and J. Nieplocha, “NWPerf: A System Wide Performance Monitoring Tool for Large Linux Clusters”, *2004 IEEE International Conference on Cluster Computing*, pp. 379-389, September 2004.

[15] A. Morajko, O. Morajko, T. Margalef, and E. Luque, “MATE: Dynamic Performance Tuning Environment”, *10th International Euro-Par Conference*, pp. 98-107, August 2004.

[16] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig Latin: A Not-So-Foreign Language for Data Processing", *SIGMOD ’08*, Vancouver, British Columbia, 2008.

[17] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, "Interpreting the Data: Parallel Analysis with Sawzall", *Scientific Programming*, **13**, 4, October 2005, pp. 277-298.

[18] “Open|SpeedShop”, http://www.openspeedshop.org/.

[19] P. Roth, D. Arnold, and B. Miller, "MRNet: A Software-Based Multicast/Reduction Network for Scalable Tools", *SC 2003*, Phoenix, Arizona, November 2003.

[20] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility", *SIGOPS Oper. Sys. Rev.* **35**, 5, December 2001, pp. 188-201.

[21] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, “Design and Implementation of the Sun Network Filesystem”, *USENIX Technical Conference*, pp. 119-130, Portland, Oregon, June 1985.

[22] F. Schmuck and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters", *File and Storage Technologies*, pp 231-244, January 2002.

[23] P. Schwan, "Lustre: Building a File System for 1000-node Clusters", *2003 Linux Symposium*, July 2003.

[24] S. Soltis, T. Ruwart, and M. O'Keefe, "The Global File System", *5th NASA Goddard Conference on Mass Storage Systems and Technologies*, pp. 319-342, September 1996.

[25] M. Sottile and R. Minnich, “Supermon: a high-speed cluster monitoring system”, *IEEE Conference on Cluster Computing*, pp. 39, Chicago, Illinois, September 2002.

[26] “The Distributed Debugging Tool (DDT)”, Allinea Software, http://www.allinea.com/DDT.pdf.

[27] A. Tridgell and P. Mackerras, "The rsync algorithm", Australian National University Technical Report TR-CS-96-05, June 1996.

[28] G. Watson and C.E. Rasmussen, “A Strategy for Addressing the Needs of Advanced Scientific Computing Using Eclipse as a Parallel Tools Platform”, Los Alamos National Laboratory Technical Report LA-UR-05-9114, December 2005.