Final Progress Report:
A Path to Operating System and Runtime Support for Extreme Scale Tools

August 2008 – August 2012

DE-PS02-08ER25843

Barton P. Miller
bart@cs.wisc.edu

Computer Sciences Department
University of Wisconsin
Madison, WI 53706-1685

Philip Roth
rothpc@ornl.gov

Future Technologies Group
Computer Science & Math Division
Oak Ridge National Laboratory
One Bethel Valley Road
P.O. Box 2008, MS-6173
Oak Ridge, TN 37831-6173

John DelSignore
jdelsign@totalviewtech.com

Rogue Wave Software
Suite 106
24 Prime Park Way
Natick, MA 01760

1 Technical Areas of Progress

This report describes progress made over the course of funding for this project, including work at the University of Wisconsin, Oak Ridge National Laboratory and Rogue Wave Software.

1.1 Work at the University of Wisconsin

Tools and middleware are crucial to the effective use of large distributed systems. Middleware enables efficient utilization of resources, and tools help to diagnose and fix problems in distributed programs. A common requirement among tools and middleware is operating on groups of distributed processes and files, but prior work has failed to provide a solution that both addresses the key scalability barriers and is easy to use within new and existing software. Each group operation involves a single process communicating with distributed hosts to apply the operation to each member, and collecting status results or data produced by the operations. Often, group operation status or data results are further processed to derive information that summarizes group behavior or to guide further operations on the group. In many tools and middleware, both the distributed operations and data analysis need to be completed in a time span suitable for providing interactive functionality. For large distributed groups, however, the distributed communication and data processing required represent critical scalability barriers.

In this project, we cast distributed resource access as operations on files in a global name space and developed a common, scalable solution for group operations on distributed processes and files. The resulting solution enables tool and middleware developers to quickly create new scalable software or easily improve the scalability of existing software.

The cornerstone of the project was the design of a new programming idiom called group file operations that eliminates iterative behavior when a single process must apply the same set of file operations to a group of related files [3]. The keys to the idiom are explicit identification of file groups using directories as the grouping mechanism, the ability to name a file group as the target for POSIX I/O operations such as read and write, and explicit semantics for aggregation of group data and status results. Group file operations provide an interface that eliminates forced iteration, thus enabling scalable implementations on distributed files. Given underlying mechanisms for distributed data aggregation, group data and status results can be processed in a distributed fashion that eliminates or reduces the need for centralized analysis or large data storage.

Using group file operations, the actions of tools and middleware during distributed group operations are mapped to a common set of tasks: defining a file group; reading or writing the group's data; and optionally processing data read from the group. Due to the idiom's flexibility, many tool activities can be mapped to group file operations. For example, group read operations can be used to gather system performance and configuration data, examine system or application logs, and query host or process status information. In many group read scenarios, group data aggregation can be used to further classify or transform raw, per-member data into summarized information for all hosts or processes. Group write operations are useful for writing messages to synthetic file systems providing operating system or process control (e.g., sysfs and procfs on Linux), as used in distributed system management and parallel debugging tools. Group writes can also be used for centralized management of shared-nothing clusters where software and configuration updates must be applied to the local file systems of each host.

Forming file groups efficiently is an important but less obvious problem, as it is a precursor to using group file operations. To avoid iteration and provide scalable definition of file groups spanning thousands of file servers, we devised a two-step process that relies on file system name space composition [6]. First, tools provide a specification that is used at each independent server to generate a custom view of the server's local name space; the custom views are structured to naturally support efficient construction of a global view. Second, the custom server views are merged into a global name space using a composition that automatically groups related files (i.e., files providing similar data or functionality). Name space specifications are written using a new language called FINAL, the FIle Name space Aggregation Language, that models hierarchical file system name space composition as operations on trees. FINAL provides flexible composition semantics based on common tree operations such as copying, pruning, and grafting. To support efficient composition of many trees without explicit iteration, FINAL introduces a merge operation over set of trees, and supports customizable resolution for the name conflicts that can occur during a merge. FINAL's merge operation provides the key semantics required for composing file group directories containing files from independent name spaces.

To demonstrate our novel and scalable ideas for group file operations and global name space composition, we developed a group file system called TBON-FS that leverages a tree-based overlay network (TBON), specifically MRNet [4,5], for logarithmic communication and distributed data aggregation. Similar to NFS, TBON-FS does not provide any data storage, and simply acts as a proxy at each file server to access the locally visible file systems. MRNet is used for scalable multicast of group file operation requests to thousands of independent servers, scalable aggregation of group data and status results from group file operations, and scalable composition of the TBON-FS global name space. TBON-FS also supports synthetic file systems within servers as a means for providing file-based interfaces to arbitrary tool functionality. Because TBON-FS is implemented entirely at user-level, it is easy to deploy on a wide variety of distributed systems.

A particular focus of the project has been the application of our techniques to control and inspection of distributed process and thread groups. In this regard, we also developed proc++, a new synthetic file system co-designed for use in scalable group file operations. The design of proc++ is based on the proc file system found in Solaris, with extensions to provide abstractions that naturally encapsulate interaction-intensive operations, such as breakpoint management and gathering stack traces, as a means to reducing the number of interactions between the tool and the process control layer. The primary challenges in designing proc++ were to identify the tool behaviors that were hindered by interactions with existing process control interfaces, and to develop appropriate abstractions that enable group file operations to take advantage of newly provided tool-level capabilities. The difficulty of the latter challenge concerns the placement of the abstractions within the layers of the distributed tool hierarchy, which is key to enabling the parallelism necessary to scale, and choosing the proper amount of tool functionality captured by the abstractions requires careful thought to avoid limiting their utility.

Over the course of the project, we evaluated the utility and performance of group file operations, global name space composition, TBON-FS, and proc++ in three case studies. The full details of each study are available in [8].

The first study, initially discussed in [3], focused on the ease in using group file operations and TBON-FS to quickly develop several new scalable tools for distributed system administration and monitoring. We developed parallel versions of five common UNIX utilities (cp, rsync, grep, top, and tail) and showed good performance and scalability on two Linux clusters with over one thousand hosts.

The second study evaluated the integration of group file operation and TBON-FS within the Ganglia Distributed Monitoring System to improve its scalability for clusters. This study was also initially described in [3], and we demonstrated significant reductions in CPU and network utilization for our modified version.

The final study involved the integration of group file operations, TBON-FS, and proc++ within TotalView, the widely-used parallel debugger. Using a large Cray XT system, we demonstrated our improved version of TotalView debugging over 150,000 processes, while the original version failed to debug more than 12,000 processes. Further, we showed that using TBON-FS and proc++, group file operations on groups with over 200,000 distributed processes (or files) can be completed in a quarter of one second, which is suitable for interactive tasks. When using scalable data aggregations such as equivalence classes or simple statistical summaries, our results project that group file operations on groups with over one million members should complete in just one second.

1.2 Work at Oak Ridge

For this project, the work of the Oak Ridge National Laboratory (ORNL) team occurred primarily in two directions: bringing the MRNet tree-based overlay network (TBON) implementation to the Cray XT platform, and investigating techniques for predicting the performance of MRNet topologies on such systems.

The ORNL team was instrumental in bringing the MRNet to the Cray XT and subsequent X-series platforms, with early emphasis on the Cray XT systems deployed at ORNL. Prior to this effort, the MRNet implementation supported traditional Linux clusters and the IBM Blue Gene platform but several technical barriers presented by the Cray software environment prevented it from working on the Cray platform. Cray's transition from using the Catamount lightweight com-

pute node kernel to Linux removed some barriers such as the restriction on the use of TCP/IP sockets between compute nodes. With information and support libraries from Cray engineers, we developed and evaluated techniques for overcoming problems in starting and connecting MRNet overlay network processes within the Cray batch runtime environment. In response to requests by project personnel from Rogue Wave and from other potential MRNet users on the Cray platform, we developed and implemented a technique that allows MRNet processes to be placed entirely on the same compute node resources as a parallel application's processes so as not to require users of MRNet-based tools to request extra compute nodes for tool processes.

As part of this project, we also investigated approaches for predicting the performance of MRNet-based tools and applications. Within the TBON concept, there is a great deal of flexibility for the topology used for process organization, with topologies varying by degree of connection fanout, tree depth, and balance. Although this flexibility is beneficial in that it allows people to deploy a TBON tailored to their particular requirements, it often leads to questions of identifying the "best" process topology and process placement. To partially address this problem, we investigated an approach for predicting the performance of TBON-based tools and applications under various topology and process placement scenarios. Our approach, presented at Dagstuhl and CScADS workshops, uses discrete event simulation for performance prediction and event traces as simulation input.

Early in the project, an unexpected opportunity arose to collaborate with faculty and students at North Carolina State University to explore the use of the TBON concept within the context of their efforts to trace a program's I/O activity when running on a large-scale system. The most recent I/O tracing approach[7] uses a TBON for inter-node I/O event trace compression leaving the final compressed trace at the tree's root node. We also began to explore the use of compressed traces similar to these compressed I/O event traces as input to the TBON performance prediction framework, but that work has not yet resulted in publications or presentations.

1.3 Work at Rogue Wave Software (RWS)

Rogue Wave Software (RWS), formerly TotalView Technologies Inc., worked with the University of Wisconsin (UW) team on the design and prototyping of a scalable version of the TotalView debugger. Throughout the project, RWS managed bi-weekly conference calls where design and implementation strategies were discussed. Access to the TotalView sources and RWS TotalView machine environment was provided to UW, which allowed UW to edit, build, test, and run TotalView. UW was granted access to the TotalView CVS source repository to allow them to create a source branch to contain their modifications to TotalView. RWS provided guidance to UW in understanding the overall structure, organization, and layering of the TotalView sources. RWS granted UW access to key TotalView documents and information contained with the TotalView Toolworks (TW) document catalog, which collectively describe important aspects of TotalView's internal operation. In addition to providing guidance with regard to existing documentation, RWS also wrote new documentation, including tw236 and tw237, and added them to the TW document catalog. These new documents addressed several questions raised by UW regarding the internal operation of TotalView.

RWS assisted UW with their "proc++" design effort. The design was the main topic of the bi-weekly conference calls, but it was also discussed in email. In addition to time spent on the conference calls, TotalView spent substantial additional time considering issues which came up during the calls. RWS reviewed design documents produced by UW. During the project UW provided

RWS with several documents proposing aspects of the design, and RWS reviewed and responded to these documents, providing feedback by email and feedback during the bi-weekly discussions. RWS critiqued the ongoing design work, made several recommendations for solutions to problems contained within the designs, and collaborated on refinements to the design. Specific areas in which RWS was involved included debugger target event management, event reporting, group formation, location addressing issues, provisional addresses, and indexed addresses.

RWS assisted UW with strategy for integrating proc++ into TotalView. Beyond the basic proc++ design issues, several additional considerations arose in the area of the interface between proc++ and the TotalView client. RWS was involved in working out the details of the issues surrounding this interface, including layering issues for where to attach proc++ to TotalView, whether to use existing parts of TotalView's low-level debugger infrastructure, design of interface between TotalView and proc++, and integration of proc++ with the TotalView event tracer.

Finally, RWS reviewed and commented on Mike Brim's Ph.D. thesis from UW, concentrating on Chapter 9 which focused on his TotalView work.

2 Publications

1. Philip C. Roth and Jeffrey S. Vetter, "Scalable Tool Infrastructure for the Cray XT Using Tree-Based Overlay Networks," *Cray User Group 2009 Meeting (CUG 2009),* Atlanta, May 2009.
2. Karthik Vijayakumar, Frank Mueller, Xiaosong Ma, and Philip C. Roth, "Scalable I/O Tracing and Analysis," *2009 Petascale Data Storage Workshop, Portland,* Oregon, November 2009.
3. Michael J. Brim and Barton P. Miller, "Group File Operations for Scalable Tools and Middleware", *IEEE 16th International Conference on High Performance Computing (HiPC 2009),* Kochi, India, December 2009. **Best Paper Award.**
4. Michael J. Brim, Luiz DeRose, Barton P. Miller, Ramya Olichandran, and Philip C. Roth, "MRNet: A Scalable Infrastructure for the Development of Parallel Tools and Applications," *Cray User Group 2010 Meeting,* Edinburgh, Scotland, May 2010.
5. Emily R. Jacobson, Michael J. Brim, and Barton P. Miller, "A Lightweight Library for Building Scalable Tools", *Para 2010: State of the Art in Scientific and Parallel Computing,* Reykjavik, Iceland, June 2010.
6. Michael J. Brim, Barton P. Miller, and Victor Zandy, "FINAL: Flexible and Scalable Composition of File System Name Spaces", International Workshop on Runtime and Operating Systems for Supercomputers 2011 (ROSS'11), Tucson, Arizona, May 2011.
7. Xing Wu, Karthik Vijayakumar, Frank Mueller, Xiaosong Ma, and Philip C. Roth, "Probabilistic Communication and I/O Tracing With Deterministic Replay at Scale," *2011 International Conference on Parallel Processing (ICPP 2011),* Taipei, Taiwan, September 2011.
8. Michael Joseph Brim, "Control and Inspection of Distributed Process Groups at Extreme Scale via Group File Semantics", *Ph.D. Dissertation in Computer Sciences,* University of Wisconsin-Madison, May 2012. (Available online at `ftp://ftp.cs.wisc.edu/paradyn/papers/Brim12Dissertation.pdf`)

3 Students Supported and Student Progress

Michael Brim (Wisconsin, Ph.D. in Computer Science)
Avrilia Floratou (Wisconsin)

James Jolly (Wisconsin, M.S. in Computer Science)
Aishwara Kumar (Wisconsin, M.S. in Computer Science)
Kevin Roundy (Wisconsin, Ph.D. in Computer Science)

A total of five graduate Research Assistants were supported during the reporting period. No post doctoral nor undergraduate student were supported during this period. During this time, Wisconsin also supported a full-time research staff member, Madhavi Krishnan, from the University of Texas.

4 Outreach and Transitions

While still early in the project, we have made a good start on technology transitions:

- Cray Research: Cray is working to adopt MRNet and other related tools as the foundation for their new suite of scalable "Abnormal Process Termination" tools.
- The ORNL team presented MRNet for the Cray XT to the team from the Coordinated Infrastructure for Fault Tolerant Systems (CiFTS) project as a potential "fault tolerant backplane" for their infrastructure.
- Barton Miller was appointed to the *Petascale Software Advisory Committee,* NSF/"Blue Waters".
- Cray developed at least one tool using the result of our work in bringing the MRNet infrastructure to the Cray XT platform. MRNet is part of the software stack delivered on current Cray systems.

5 External Presentations

In addition to conference and workshop papers listed in Section 2 above, the following presentations were given:

- Barton Miller, *Invited talk*, Hebrew University of Jerusalem, "Tree-based Overlay Networks for Scalable Middleware and Systems", December, 2008.
- Barton Miller, *Invited Talk*, "Random Testing with 'Fuzz': 18 Years of Finding Bugs", Forschungszentrum Jülich, Germany, October 2008.
- Philip C. Roth and Jeffrey S. Vetter, "Scalable Tool Infrastructure for the Cray XT Using Tree-Based Overlay Networks," (presentation) 2009 CScADS Workshop on Performance Tools for Petascale Computing, Tahoe City, California, July 2009.
- Barton P. Miller, *Keynote Address,* ""How much Security is Enough? And at What Cost?", Open Grid Forum 28, Munich Germany, March 2010.
- Philip C. Roth, "Toward Performance Prediction of Tree-Based Overlay Networks on the Cray XT," (presentation) Dagstuhl Seminar on Program Development for Extreme-Scale Computing, Wadern, Germany, May 2010.

6 Awards and Honors

- Barton Miller: University of Wisconsin **Vilas Associate** Award.
- MRNet, as part of the Stack Trace Debugging Tool (STAT) won an **R&D 100 Award** in 2011.