Building on Lessons Learned From Over a Decade of MRNet Research and Development

Barton P. Miller*, Dorian C. Arnold†, Michael J. Brim‡, Philip C. Roth‡,
Evan Samanas*, Benjamin Welton*, Bill Williams*

*University of Wisconsin-Madison, Madison, WI 53706
†University of New Mexico, Albuquerque, NM 87131
‡Oak Ridge National Laboratory, Oak Ridge, TN 37831

Tools for debugging, performance analysis, and system administration must scale at least as well as the software or hardware they are meant to monitor and control. For instance, a debugger that provides traditional “single-step” thread control operations and the ability to examine the memory contents of any process in a parallel application must be scalable to the size of an extreme scale system if it is to be useful to diagnose problems that only occur when the application is run on the whole system. In the traditional tool organization, tool *back-ends* that monitor and control application processes communicate directly with a tool *front-end* that interacts with the user (or some other data consumer such as an automated fault diagnosis component). This organization fails to provide the desired data transfer bandwidth and tool responsiveness as system sizes moved beyond a few handfuls of nodes. Although these performance problems are especially problematic for *on-line* tools that collect and analyze application data as the application runs (e.g., for computational steering or efficient performance diagnosis), the problems impact most types of parallel tools.

Over a decade ago, we explored the concept of tree-based overlay networks (TBONs) to help address these problems and built MRNet [1], the initial TBON implementation that specifically targeted scalable tools. In a TBON, tool infrastructure processes connected in a hierarchical organization cooperate to provide a scalable substrate for data transfer amongst a tool’s back-ends and its front-end. Filters running in the TBON processes can manipulate and synchronize data as it passes through the network. MRNet was initially developed to improve the scalability of the automated performance diagnosis component within the Paradyn parallel performance tool [2], [3]. Subsequently, it has been adopted by several other scalable parallel tools, like the stack trace analysis tool [4] (STAT) and Open|SpeedShop [5], and even some applications [6], [7]. Other tools and distributed infrastructures have incorporated alternative TBON implementations or share many TBON characteristics.

Over the last decade of MRNet development and use, we accumulated substantial experience with tool scalability by recognizing and overcoming the challenges that limited a tool’s ability to scale. These challenges and the lessons we have learnt include:

- Any approach that requires linear data growth proportional to the number of application processes (or tool back-ends) will eventually limit tool scalability. Even growth that requires only one bit per back-end will eventually limit tool scalability, either due to memory capacity limitations, front-end processing capability, or network bandwidth limitations. It was this recognition that prompted the development of a hierarchical tool infrastructure with the ability to manipulate data as it was transferred across the infrastructure.
- The tool must balance computation and communication. Just like a parallel scientific application, load imbalance can greatly limit the scalability of a tool. It can be extremely beneficial to trade off time for space and computation for communication by spending more time processing data at a node internal to the TBON process tree to reduce the volume of data transferred up the tree, as opposed to transferring a larger amount of data with less delay through the TBON’s internal nodes.
- It is critical to identify the right location for a given operation. Operations implemented in the back-ends (at the leaves of the tree) are replicated at the scale of the system, so can have the greatest computation benefit. Operations implemented in the tree internal nodes can scalably distribute a function that globally operates over all the back end data, but must (as we said above) maintain balance and constant size data outputs. Operations implemented at the front-end (the root of the tree) must be constant in cost to prevent linear (or worse) growth.
- The tree organization provides intrinsic fault tolerance support. Intuitively, since the various levels of the tree are added for performance and not functionality, this means they often render inherent information redundancies that can enable fail-forward error recovery for several types of common operations that do not require explicit checkpointing, replication or logging [8].
- There are a surprising number of unforeseen operating system/systems software limitations that impact tool scalability. For instance, the operating system imposes a limit on the number of open file descriptors and the number of file descriptors that may be passed to system calls that check those descriptors for activity. Although these limits at first may appear to be so large as to be effectively infinite, users of tools that treat them as such will eventually be surprised (in an extremely negative way) when the software fails to work for larger applications.

In our presentation, we will discuss these and other tool scalability challenges and the lessons we learned in overcoming them.

We continue to explore and refine TBON concepts, search for new ways to apply TBONs and work to improve the MRNet implementation. For example, we recently applied TBON concepts to address the scalability problems of accessing files (and other data that the OS exposes using a file system interface) using the concept of a *group file* [9], [10]. We demonstrated the efficacy of group files within the context of tools for distributed system administration and monitoring and parallel application debugging. Also, because many of today’s high performance computing (HPC) platforms include hardware accelerators such as graphical processing units (GPUs), we are investigating the benefits and costs of using these accelerators within TBON filters to manipulate data as it is transferred across a TBON process network [7]. Such accelerators can provide a tremendous

The submitted abstract has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

benefit with respect to processing data as compared to a traditional CPU, but the accelerator's connection to the host system has relatively low bandwidth. Thus, using accelerators to implement TBON filters must be done with great care. In fact, the problem of making efficient use of accelerator hardware is but one instance of an interesting, more general problem: how best to map TBON components onto the target HPC platforms as those systems' software and hardware increases in complexity. In addition to presenting several lessons learned over the past decade, we will also provide more detail about our current research and development activities and discuss how this work may serve as the foundation for research and development on scalable tools infrastructure for the next ten years.

References

[1] P. C. Roth, D. C. Arnold, and B. P. Miller, "MRNet: A software-based multicast/reduction network for scalable tools," in *Proceedings of the 2003 International Conference for High Performance computing, Networking, Storage and Analysis*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 21–36.

[2] B. P. Miller, M. D. Callaghan, J. M. Cargille, J. K. Hollingsworth, R. B. Irvin, K. L. Karavanic, K. Kunchithapadam, and T. Newhall, "The Paradyn parallel performance measurement tool," *Computer*, vol. 28, no. 11, pp. 37–46, Nov. 1995. [Online]. Available: http://dx.doi.org/10.1109/2.471178

[3] P. C. Roth and B. P. Miller, "On-line automated performance diagnosis on thousands of processes," in *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*, ser. PPoPP '06. New York, NY, USA: ACM, 2006, pp. 69–80. [Online]. Available: http://doi.acm.org/10.1145/1122971.1122984

[4] D. C. Arnold, D. H. Ahn, B. R. de Supinski, G. L. Lee, B. P. Miller, and M. Schulz, "Stack trace analysis for large scale debugging," March 2007.

[5] M. Schulz, J. Galarowicz, D. Maghrak, W. Hachfeld, D. Montoya, and S. Cranford, "Open|SpeedShop: An open source infrastructure for parallel performance analysis," *Sci. Program.*, vol. 16, no. 2-3, pp. 105–121, Apr. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1402716.1402727

[6] D. C. Arnold, G. D. Pack, and B. P. Miller, "Tree-based overlay networks for scalable applications," in *Proceedings of the 20th international conference on Parallel and distributed processing*, ser. IPDPS'06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 223–223. [Online]. Available: http://dl.acm.org/citation.cfm?id=1898699.1898719

[7] B. Welton, E. Samanas, and B. P. Miller, "Mr. Scan: extreme scale density-based clustering using a tree-based network of GPGPU nodes," April 2013, under submission.

[8] D. C. Arnold and B. P. Miller, "Scalable failure recovery for high-performance data aggregation," in *Proceedings of the 2010 International Parallel Distributed Processing Symposium (IPDPS 2010)*, 2010, pp. 1–11.

[9] M. J. Brim and B. P. Miller, "Group file operations for scalable tools and middleware," in *Proceedings of the 2009 IEEE International Conference on High Performance Computing*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2009, pp. 69–78. [Online]. Available: http://dx.doi.org/10.1109/HIPC.2009.5433223

[10] M. J. Brim, B. P. Miller, and V. Zandy, "FINAL: Flexible and scalable composition of file system name spaces," in *Proceedings of the 1st International Workshop on Runtime and Operating Systems for Supercomputers*. New York, NY, USA: ACM, 2011, pp. 25–32. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1988796.1988801