# TOWARD UNDERSTANDING SOFT FAULTS IN HIGH PERFORMANCE CLUSTER NETWORKS

Jeffrey J. Evans,[1] Seongbok Baik,[1] Cynthia S. Hood,[1] William Gropp[2]

[1] *Department of Computer Science*
*Illinois Institute of Technology*
*10 West* $31^{st}$ *St.*
*Chicago, Illinois 60616*
*Email: {evanjef, sbbaik, hood}@iit.edu*

[2] *Mathematics and Computer Science Division*
*Argonne National Laboratory*
*9700 South Cass Avenue*
*Argonne, Illinois 60439*
*Email: gropp@mcs.anl.gov*

**Abstract:** Fault management in high performance cluster networks has been focused on the notion of *hard faults* (i.e., link or node failures). Network degradations that negatively impact performance but do not result in failures often go unnoticed. In this paper, we classify such degradations as *soft faults.* In addition, we identify consistent performance as an important requirement in cluster networks. Using this service requirement, we describe a comprehensive strategy for cluster fault management.

**Keywords:** Cluster, fault management, interconnection networks, soft faults

## 1. Introduction and Motivation

Cluster computing systems have been rapidly evolving over the past decade. A variety of system architectures exist ranging from tightly coupled proprietary systems to loosely coupled commodity-based systems. The relatively low cost of commodity-based systems along with the availability of public domain software makes them an attractive option. In research environments, clusters are replacing supercomputers. The processing power of multiple networked PCs or workstations is utilized through parallel computing software. Computational clusters are used to tackle complex problems that require large amounts of computing resources.

Our research focuses on computational clusters. Paramount here is the concept of a "coordinated team" of nodes and their communication environment working together as a single entity. One of the key elements of a computational cluster system is the interconnection mechanism. Since the nodes do not physically share memory, they rely on message passing through the network. In order to achieve good performance in message-passing multicomputer systems, consistently low latency and high bandwidth are required. Given these stringent requirements on the network, effective fault management is critical.

The negative impact of performance degradation, which we term *soft faults*, is often greater than that of hard faults. Soft faults are *performance degrada-*

*tions requiring corrective action.* The requirement of corrective action is then a function of the system performance *expected* or *assumed* by an application.

For example, when a user wishes to execute an application, a *request for service* is submitted to a centralized scheduler. The user requests a subset of compute nodes for a finite period of time, thus requiring the user to know approximately how long the job will take to run. If the application execution happens to run longer than the user expected (or guessed), a timeout results. The application is terminated by the scheduler and the user may or may not receive useful data - a waste of time and resources. Conversely, if the user reserves compute nodes conservatively and the application executes in a shorter period of time (i.e., no faults or optimum performance), resources are again wasted (unused), since the compute nodes were conservatively reserved.

Inconsistent performance can cause scheduling problems at many different levels. Soft faults may cause synchronization problems when the impact of a network fault accumulates through the run. Many variables affect application performance in computational cluster networks, and further investigation is required to fully understand the impact of network faults. The area of soft fault (or degraded service) management in cluster environments will grow in importance as the area of computational Grids and Grid computing evolves.

To better understand performance degradation in the context of clusters, one must consider performance relationships in both the horizontal and vertical planes. The horizontal plane includes the causal relationships that occur between "peers" at any layer (physical, link, kernel, application, etc.). The vertical plane includes causal relationships between layers, adjacent or not, all the way up through the operating system and into the applications themselves. We are studying performance degredation on the Chiba City cluster [6] at Argonne National Laboratory.

## 2. Related Work

Considerable effort both in commercial products and in the research community has been devoted to traditional "hard" fault management issues in cluster environments. There have also been efforts exploring performance issues in parallel program execution. These efforts include evaluation of network effects [2, 7], performance analysis using application and kernel code instrumentation [4, 9, 11], performance prediction [8, 10], and program steering [3, 13]. Additionally, adaptive techniques have been explored for predictive signaling and control in cluster environments for performance management [12] and in highly distributed networks for use in fault management [5]. Our focus, however, is on soft faults. Specifically, we wish to understand the mechanisms behind network contributions to soft faults and to identify ways to signal or ultimately control such faults.

## 3. The Problem of Cluster Fault Management

As high-performance systems, clusters require strong performance from each component of the system, including the application, the operating system, and the communication network itself. Additionally, when determining how to distribute processing across the nodes of a cluster, parallel computing software assumes consistent network performance. Therefore, the type of service required from cluster interconnection networks is different from that for traditional best effort or telephone networks. In best effort networks, applications tolerate variations in network performance, and real-time fault management primarily focuses on hard faults. In telephone networks, faults are tightly coupled to voice service. In clusters, however, good system performance is required to execute a large-scale application in a timely fashion. Hence, network performance degradations (soft faults) need to be addressed along with link and node failures (hard faults).

This distinction is necessary because of the time scale of action. Once detected, hard faults are corrected. Soft faults, however, are generally tolerated in the short term and may be monitored for longer-term trends. The goal of cluster fault management is to address both hard and soft faults to maintain consistent network performance. The execution of a parallel application is complex by definition. Performance tuning to achieve the optimum balance between computation and communication for a given data set can be both time consuming and unproductive. Tuning models depend on the computation and communication speeds of the hardware and software as well as the specifics of the application data set. Another major factor that is more difficult to incorporate into the models is run-time environment.

Cluster fault management can be used to maintain good system performance in two different ways. First, fault management techniques can detect and correct soft faults, thereby maintaining consistent network performance. In addition, when correction is not possible, feedback can be provided to the parallel computing software, allowing a more accurate description of current network conditions to be reflected in the modeling.

## 4. Summary and Ongoing Work

This paper has described cluster fault management in terms of both hard and soft faults. We defined a soft fault as a degradation resulting in inconsistent performance. Network behaviors impacting performance include localized hot spots, dropped packets, retransmissions or unordered messages, routing effects, and delayed transmissions because of flow control. Ongoing research is in two directions, (1) understanding the propagation or impact of soft faults and (2) developing mechanisms to detect and correct soft faults.

To better understand the impact of soft faults on cluster applications and other system components, we are currently running experiments on the Chiba City cluster at Argonne National Laboratory. In area (1), we are exposing horizontal and vertical performance relationships that can be cast into classes of soft faults. Once cast, these relationships can be further explored to better understand their causes, propogation and impact on the overall cluster system. In area (2), we are developing new adaptive routing techniques for Myrinet interconnection networks [1]. Myrinet uses source routing, with minimal intelligence and monitoring within the network, so existing techniques cannot be used.

Our next step is to extend low level network monitoring capabilities to better understand component issues. These tools will be used in conjunction with system and application monitoring tools to enhance our understanding and develop strategies for adaptive cluster management.

## Acknowledgments

This work was supported in part by the U.S. Department of Energy, under Contract W-31-109-Eng-38 and NSF 9984811.

## References

[1] S. Baik, C. Hood, and W. Gropp. Prototype of am3: Active mapper and monitoring module for the Myrinet environment. In *Proceedings of the HSLN Workshop*, Nov. 2002.

[2] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. Eicken. Logp: Towards a realistic model of parallel computation. In *Proceedings of the Fourth ACM SIGPLAN Symposium on Princples and Practices of Parallel Programming*, May 1993.

[3] W. Gu, G. Eisenhauer, and K. Schwan. Falcon: On-line moniroting and steering of parallel programs. In *Ninth International Conference on Parallel and Distributed Computing and Systems (PDCS'97)*, Oct. 1997.

[4] J. Hollingsworth and B. Miller. Dynamic control of performance monitoring on large scale parallel systems. In *International Conference on Supercomputing*, July 1993.

[5] C. S. Hood and C. Ji. Proactive network-fault detection. *IEEE Transactions on Reliability*, 46(3):333–341, September 1997.

[6] Argonne National Laboratory. Chiba City, the Argonne scalable cluster, 1999. http://www-unix.mcs.anl.gov/chiba/.

[7] R. P. Martin, A. M. Vahdat, D. E. Culler, and T. E. Anderson. Effects of communication latency, overhead, and bandwidth in a cluster architecture. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 85–97, June 1997.

[8] C. Mendes and D. Reed. Performance stability and prediction. In *IEEE International Workshop on High Performance Computing (WHPC'94)*, March 1994.

[9] D. M. Ogle, K. Schwan, and R. Snodgrass. Application-dependent dynamic monitoring of distributed and parallel systems. *IEEE Transactions on Parallel and Distributed Systems*, 4(7):762–778, July 1993.

[10] J. M. Orduna, F. Silla, and J. Duato. A new task mapping technique for communication-aware scheduling strategies. In *International Conference on Parallel Processing Workshops*, pages 349–354, 2001.

[11] D. A. Reed, R. A. Aydt, R. J. Noe, P. C. Roth, K. A. Shields, B. W. Schwartz, and L. F. Tavera. Scalable performance analysis: The pablo performance analysis environment. In *Proceedings of the IEEE Computer Society Scalable Parallel Libraries Conference*, October 1993.

[12] J. Vetter and D. Reed. Managing performance analysis with dynamic projection pursuit. In *Proceedings of SC'99*, November 1999.

[13] J. Vetter and K. Schwan. Progress: A toolkit for interactive program steering. In *Proceedings of the International Conference on Parallel Processing*, August 1995.