Fast and Scalable Real-time Monitoring System for Beowulf Clusters

Putchong Uthayopas, Sugree Phatanapherom

Parallel Research Group, CONSYL
Department of Computer Engineering
Faculty of Engineering, Kasetsart University,
Bangkok, Thailand 10400
Email: {pu, b40sup}@ku.ac.th

Abstract. Fast real-time monitoring of system information is important to the understanding of parallel system especially for a large cluster system that appeared recently. Making the system fast and scalable at the same time is still a challenging task. This paper presents the design and implementation of a fast and real time monitoring system called SCMS/RMS. This system is a part of more comprehensive cluster management tool called SCMS. SCMS/RMS is designed to be flexible, highly scalable, and efficient. Many techniques that are used to increase the monitoring speed and to achieve high scalability have been described in this paper. The experiment has been conducted on a 72 nodes Beowulf Cluster and the results show that SCMS/RMS is very fast and highly scalable.

1 Introduction and Related Works

System performance monitoring is important since it allows system administrator to understand system behavior or, in many cases, predict the malfunction earlier. Moreover, performance information can help many important subsystems such as batch scheduler to make a better decision about system resource allocation. Nevertheless, the timelines of the information is crucial for its usefulness. The implementation of fast and efficient real time monitoring is a challenging task especially for the recent large-scale clusters with thousands of node.

Many works related to system monitoring appeared in the literatures. Many tools such as CIS [1], ClusterProbe [2], GARDMON [3], PARMON [4], Co-Pilot [5], are built specifically for system monitoring. Moreover, many monitoring systems appear as a part of system management tools such as SCMS [6] and VACM [7]. The design of each tools is rather different due to the goal and complexity of the monitoring subsystem itself. For example, many tools rely on usual system interface such as /proc in Linux to access operating system performance data. But some tools such as CIS developes their own kernel probe to increase the speed of access and to reduce the level of intrusiveness. Scalability is also a major issue being addressed by many works. Many monitoring subsystems [1][3][4][8][5] are still based on centralized daemon or applications that collect the information from the distributed agents

running on every node in the system. This obviously limits the scalability of the system. Hierarchical monitoring is employed to enhance the scalability in ClusterProbe. A performance study of this is presented in [9]. ClusterProbe uses advanced techniques such as data filtering and merging to further reduce the monitoring traffic. The tool called CIS introduces the technique of adjusting the monitoring frequency to reduce the impact. This technique and more are also supported by our implementation. Most monitoring is based on their own protocol over TCP/UDP link except in [10] which builds a large scale monitoring based on SNMP protocol. Extension to the large system such as grid is discussed in [11]. The use of information obtained to predict the behavior of large wide area system is presented by the work on Network Weather Service (NWS)[12].

This paper presents the work on real-time monitoring subsystem called SCMS/RMS. This subsystem is a part of a more comprehensive cluster management tool called SCMS. The goals of SCMS/RMS are to provide a monitoring subsystem for Beowulf cluster that is fast, efficient, low intrusive, and scalable to a very large cluster. Many design techniques and experiences are presented.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of SCMS/RMS system architecture and implementation. Section 3 gives the detail of performance improvement techniques used. The experiments performed to evaluate the performance and results obtained are described in Section 4. Finally, the conclusion and future work are discussed in Section 5.

2 SCMS/RMS System Architecture

SCMS/RMS consists of two kinds of agent processes, namely, CMA (Control and Monitoring Agent) and SMA (System Management Agent). CMA is an agent that runs on every node in the system. CMA main function is to collect the information requested by SMA.

SMA has several functions in the system. Firstly, SMA is a contact point for application. Applications under SCMS/RMS contact SMA through a set of API called RMI (Resource Monitoring Interface). API bindings are currently supported C, C++, Java, and Python. Secondly, SMA also acts as a message router so that the system can be configured to form tree structure as described in the following section.

The organization of SCMS/RMS monitoring subsystem is as illustrated in **Fig. 1**. To make system scalable and efficient, SCMS/RMS monitoring system is organized into *Partition*. Partition is a set of one SMA and multiple CMA, or a set of SMA. In **Fig. 1**, three partitions are shown. Partition 2 and 3 forms a basic structure of monitoring system. They consist of one SMA and a number of CMAs. In this partition, SMA knows about the underlining CMA. SMA and SMA can be connected together to form a partition as illustrated by partition 1. This connection allows us to connect the basic partition into a tree structure. This technique allows SCMS/RMS to easily scale to a very large number of nodes. Using this design also allows user to freely configure the monitoring subsystem into many forms. One example is the partitioning of monitoring system to match the underlining physical sub networks and localize the communication traffic.

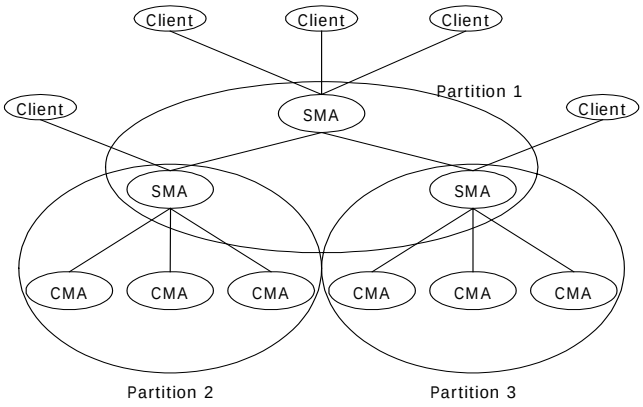

Fig. 1. SCMS/RMS System Structure

Clients are free to connect to any of the SMA. But the scope of information is limited to the information gather from that SMA and the partition below it. Hence, only clients that connect to top-level SMA can access all the information provided by the cluster. In practice, the system can be designed such that client that connectd to any SMA can request any information. But doing so will increase the number of control message substantially.

In SCMS/RMS, one decision is to clearly separate the information retrieval mechanism from data interpretation. SCMS/RMS only provides mechanism to name any object in cluster system and gather the information about named objected efficiently. The interpretation of data gathered is left to application that uses API. The purpose of this design is to allow SCMS/RMS to be employed in a much broader scope rather than limited to resource monitoring task only. For instance, one can write a probe module that attaches to MPI parallel program. Then gather the usage information of MPI via the MPI profiling interface. Then, a real-time visualization application can be developed easily to gather such information using SCMS/RMS as a delivering mechanism. This is, in fact, being target as one of our future work.

To allow extreme flexibility and extensibility, SCMS/ RMS allows users to build a new plug-in module and load it dynamically into CMA. This feature is based on shared library feature supported by many operating system including Linux. Each plug-in module is identified by 32-bit plugging id. In each plug-in, user can define many interface functions. This set of function is an agreement between end application and plug-in. SCMS/RMS only delivers the information between these two endpoints. Currently, there are two standard plug-in: Hardware Plug-in and Process Plug-in. These two plug-ins provide the interface to local operating system and get system information for SCMS. Using plug-in makes SCMS/RMS very modular and very portable.

In SCMS/RMS, applications can easily access the monitoring services using a set of API called Resource Management Interface (RMI). These API allows users to get node id and name of all live nodes in the cluster, get any information from any object in the system known to SCMS/RMS, and manage plug-in module attached to CMA.

3 Performance Improvement Techniques

The key concept in achieving high scalability and performance is to reduce amount of information passed on the network and the overlapping of processing and communicating activities. The following techniques are employed in SCMS/RMS to reduce network traffic and optimize the performance of monitoring subsystem.

3.1 Dual Mode Operation

SCMS/RMS separates its operation into two modes: Asynchronous and Synchronous mode. By default, SCMS/RMS will operate in Asynchronous mode. In this mode, the information will be sent back along the tree by each CMA and collected centrally in SMA. The frequency of information gathering can be adjusted by users to a suitable one depending on the level of intrusiveness required. Application that requests the information will get the one that are cached. So, the information is only the estimation of the system state at that point but not a precise one. The purpose of this mode is to support the system monitoring tools.

In contrast, synchronous mode is designed for the application that requires more precise and more updated system state. When application requests the information in synchronous mode, all the cached data is bypassed for that session and the request is passed through the system to destination CMA. The information obtained is delivered to requested application through SMA agent that application attachs to. This mode is intended to support systems such as batch scheduling where the correctness of information has more impact.

3.2 Compact Name Space

One important issue is the naming of resources in cluster systems. One approach is to use SNMP style of hierarchical object named space. Although very flexible, this scheme results in a very complex and slow implementation.

In SCMS/RMS, each resource is given a unique integer id. So, the resource can be addressed with a tuple of *<node id, resource id>*. Multiple resources information can be accessed simultaneously by forming a *Resources vector*. Resource vector is a list of integer values that represent a set of resources or objects. Since most API requires users to supply node id, multiple resource addresses on one node can be derived from the resources vector. By using resource vector, the naming can be done very efficiently.

3.3 Message Pipelining

Since SCMS/RMS is structured as a tree of process, it is natural choice to exploit this to speed up the system. All requests are pipelined along the tree structure of the system. The effect is the overlapping between processing time and communicating

time. Moreover, the tree structure guarantees that the time complexity of the information gathering scale as O (log n) where n is number of nodes in the system.

3.4 Filtering and Merging

In SCMS/RMS, the requests from several client applications will be merged together and sent down the tree to destination CMA. Once CMA gets the required data, it will be sent back to destination SMA. Intermediate SMA will merge the data along the way up to the top level SMA. Then the data will be splitted and distributed to the client application that requests them. This method reduces the number of message exchanged substantially.

4 Performance Evaluation

SCMS/RMS has been evaluated on PIRUN Beowulf Cluster system, the main computing facility at Kasetsart University. The experimental system has 72 diskless nodes of Pentium III 500 MHz with 128 Mbytes RAM, 3 Xeon 500 MHz fileserver with 1 Gigabytes of memory. The interconnection used is 3COM Superstack II Fast Ethernet switch. The tests had been carefully conducted when the system had no task to interfere with the communication and computation time. Due to node availability, we were able to test the configuration up to 64 nodes only. Each experiment has been conducted 3-5 times and the average results are used.

The first experiment simulates the task of finding least loaded nodes in the cluster. This is done by getting load information from all nodes and comparing them to find the minimum one. This operation is usually performed by a batch scheduler to locate a set of best node in the cluster. The elapse time from the start of the request to the end of request is measured. The results are as shown in Table 1.

Table 1. Time used to determine the least loaded node

Nodes	Time Used (msec)
1	86
2	46
4	38
8	33
16	45
32	68
64	116

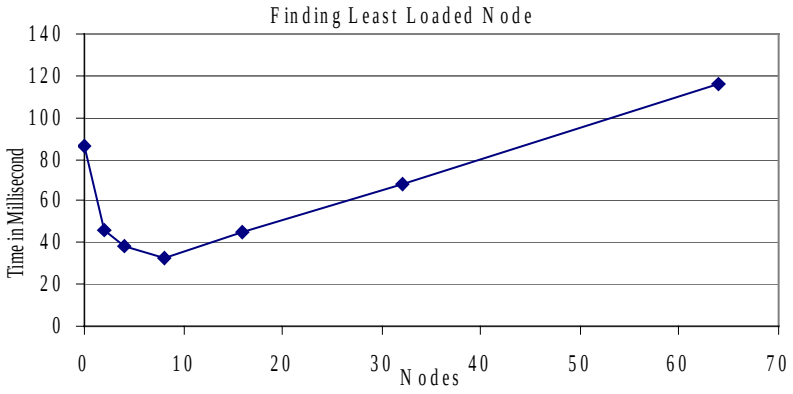

Fig. 2. Timed used to find least loaded node as number of nodes increases

Table 1 shows that SCMS/RMS monitoring system is very fast. The time used to find this information is less that 116 millisecond for 64 nodes. One fact that can be observed from the graph in Fig. 2 is that the time used is quite high for a small number of nodes and then decreases to a minimum at about 10 nodes, then increases again. This is caused by the overhead of monitoring system such as the hierarchical structure organization. So for a small number of nodes, the system will not benefit from the optimization techniques used. Only medium to large cluster can take the advantages of the technique.

In the second experiment, the test is conducted by measuring the average response time when requesting large number of performance data from multiple nodes. This test simulates the case when system administrator wants to monitor the whole system in very detail. The size of information from each node is about 11 Kbytes. Number of nodes used in the experiment is 8, 16, 32, and 64. The comparison has also been made between SCMS/RMS and an open source monitoring software from SGI Incorporate called performance co-pilot (PCP). This software is available on Internet at http://oss.sgi.com/projects/pcp/. The performance co-pilot version 2.2.0 has been used in this experiment. The time used is measured as presented in **Table 2** and the graph are as illustrated in Fig. 3.

Table 2. Time used to request large amount of information from a set of nodes

Nodes	SCMS/RMS (millisecond)	PCP (millisecond)
8	18.53	17.20
16	36.35	49.39
32	45.69	147.71
64	65.76	340.57

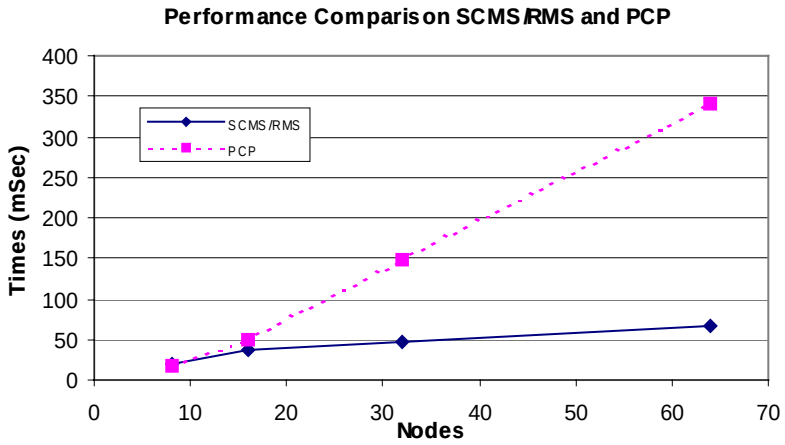

Fig. 3. Comparison of the Monitoring time for SCMS/RMS and Performance Co-pilot

As can be seen from Fig. 3, the time used increases as number of node increases. In all cases, SCMS/RMS is much faster than performance co-pilot. As the number of nodes increase, the monitoring time of SCMS/RMS increases much slower than performance co-pilot. This means that SCMS/RMS can achieve a better scalability.

5 Conclusion

Good real-time performance system is an important component in cluster software tools. Apart from using them to obtain more insight about large system behavior, this subsystem can be applied in a much wider context. For example, real-time monitoring can be used to feed the resource information to batch scheduler. This allows us to build a smart scheduling policy that better utilizes system resources. Another application is the monitoring of running parallel program for debugging or optimizing purposes. This is the area that will be explored as a future work for SCMS/RMS.

Acknowledgement

This work was partially funded by Kasetsart University Research and Development Institute under SRU Grant and Faculty of Engineering Research Grant. Equipments used for the development is supported in part by AMD Far East Inc.

References

1. J. Astalos, "CIS - Cluster Information Service". http://ups.savba.sk/parcom/cluster/
2. Z. Liang, Y. Sun, and C. Wang. "ClusterProbe: An Open, Flexible and Scalable Cluster Monitoring Tool", Proceedings of the First International Workshop on Cluster Computing, pp. 261-268, Dec. 2-3, 1999
3. R. Buyya, B. Koshy, and R. Mudlapur, "Gardmon: A javabased monitoring tool for gardens non-dedicated cluster computing", In Proceedings of Workshop on Cluster Computing Technologies, Environments, and Applications, PDPTA 99, Monte Carlo Resort, Las Vegas, Nevada, USA, 1999
4. R. Buyya, "PARMON: A Portable and Scalable Monitoring System for Clusters", International Journal on Software: Practice & Experience (SPE), John Wiley & Sons, Inc, USA, pp. 723-739, June 2000
5. Silicon Graphics, Inc, "Performance Co-Pilot". http://oss.sgi.com/projects/pcp/
6. Putchong Uthayopas, Jullawadee Maneesilp, Paricha Ingongnam,"SCMS: An Integrated Cluster Management Tool for Beowulf Cluster System", Proceedings of the International Conference on Parallel and Distributed Proceeding Techniques and Applications 2000 (PDPTA'2000) , Las Vegas, Nevada , USA , 26-28 June 2000
7. VA Linux Systems, "VACM Cluster Management". http://www.valinux.com/projects/vacm/
8. R. Flanery, A. Geist, B. Luethke, and S. Scott, "Cluster Command & Control (C3) Tools Suite", http://www.epm.ornl.gov/~sscott/
9. K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldzmidt, and Y. Yemini, "Decentralizing Control and Intelligence in Network Management", Proceedings of 4th International Symposium on Integrated Network Management, Integrated Network Management IV, Ed. Sethi et.al., pp. 4-15, ISBN 0412715708, Chapman &Hall, 1995
10. Rajesh Subramanyan, Jose Miguel-Alonso, and Jose A.B Fortes, "A scalable SNMP-Based distributed monitoring system for heterogenous network computing", Proceedings of SC2000, Dallas, Texas, 2000
11. B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, M. Swany, and the Grid Performance Working Group, "White Paper: A Grid Monitoring Service Architecture (DRAFT)", Global Grid Forum, February 2001. http://www-didc.lbl.gov/papers/Grid.Monitoring.wp.pdf
12. R. Wolski, N. T. Spring, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing", Proceedings of 6th High-Performance Distributed Computing, 1998