

# Building a Resources Monitoring System for SMILE Beowulf Cluster

Putchong Uthayopas, Surachai Phaisithbenchapol, Krisana Chongbarirux  
Parallel Research Group  
Computer and Network System Research Laboratory  
Department of Computer Engineering  
Faculty of Engineering, Kasetsart University  
Bangkok, Thailand  
Email: pu@nontri.ku.ac.th

## Abstract

*As the PC cluster becomes a popular low cost high-performance computing platform, it is hard to manage this system due to the lack of powerful resource management and monitoring tool. This paper presents our effort to mitigate this problem by developing a resource monitoring system. This system also provides a set of API in C, Java and TCL/TK that can be used to develop management applications on top of it. This resource monitoring system has currently been used in our SMILE Beowulf cluster system. We found that this resource management system is a powerful tool for real-time performance monitoring and can be extended later to support dynamic load balancing and task scheduling.*

## 1. Introduction and Related Work

In the past, researchers who need extremely high computing power have no choice but to use a supercomputing facilities located around the globe. In most cases, it is difficult to have an access to the facilities due to distant, poor communication link, or budget constrains. Sometime, due to the high load of the system, researchers have to wait for a very long time before getting a result from supercomputer. This problem lower the productivity and creativity of the scientists and engineers dramatically.

Some early approach to alleviate this problem is to use the idle time on network of workstations as a source of computing power. Many software systems has been developed such as condor, DQS [1,2]. These systems usually focus on resource allocation issues such as process distribution among group of workstations, how to locate idle workstation, how to resume and migrate processes, and batch scheduling of tasks. The tasks scheduled are usually a batch of sequential tasks only. To exploit the power of heterogeneous workstation cluster for parallel processing, software that present a virtual parallel machine view to programmer has been developed. These software such as PVM [3], MPI[4] help present a

standardize and uniform programming model based on message passing paradigm over a workstation network.

Most of the non-dedicate clusters still use conventional 10 Mbps ethernet as an interconnection link. Although some researcher showed that using these software, it is possible to gain an acceptable performance for many class of important parallel algorithms on workstation network with only 10 Mbit ethernet[7,8,9]. The communication link speed are usually a cause for low performance for most compute intensive and communication intensive application. One way to alleviate the problem is to use a dedicated workstation cluster and high speed network technology such as fast ethernet, ATM or Myrinet. Berkeley NOW project [10] is an example of this effort.

Recent literatures [11,12] have reported a successful attempt to exploit PC clusters as low cost high-performance computing platform. PC cluster is an excellent solution to HPC need in Asian developing country due to the low startup cost, ease of maintenance and use, and scalability of the system[13,14]one of the major difficulty encounter in the exploitation of PC Cluster is the lack of management tools that enable users and systems administration to use and control cluster efficiently.

To mitigate this problem on our system, we have developed a resource management system on our cluster as a solution for this problem. Our system gathers useful information from every node into a single places in real-time. Management application can retrieve this information using the API we provided. Currently, the API is available in C, Java, TCL/TK. We have also developed some system monitoring tools based on this API. The focus of our resource management system is on a small to medium dedicated cluster (between 4 to 64 nodes). This paper presents the detail and results of our development.

This paper is organized as follows. Section 2 presents the systems configuration of our cluster. Section 3 discuss the structure of our monitoring and API needed to contact monitoring system. Section 4 show some application that use this API. Finally, we provide the conclusion and discuss some future work in section 5.

## 2. SMILE Beowulf Cluster

SMILE (Scalable Multicomputer Implementation using Low-cost Equipment) Beowulf Cluster (<http://smile.cpe.ku.ac.th>) has been operating for more than a year at Computer and Network Systems Research Laboratory, Department of Computer Engineering, Kasetsart University[15,16]. This system is currently used as a platform for parallel processing research, computational science application development. The picture of system is shown in Figure 1.



Figure 1. SMILE Beowulf Cluster System, Kasetsart University

### 2.1 Hardware

Current hardware configuration of SMILE system consists of 8 compute nodes and 1 front-end node connected by one 100 MBPS Fast Ethernet Hub. Four of the compute nodes contain Pentium II 233MHz, 256 KB L2 Cache, 64 MB of memory, 2 GB EIDE disk, and 3COM Ethernet card 3C905TX. The rest of the compute nodes are Pentium 133MHz, 256 KB L2 Cache, 64 MB of memory, and 3COM Ethernet card 3C905TX. The front-end nodes are Pentium 133MHz with 256 KB L2 Cache, 64 MB of memory. The difference is that the front-end node has 2 Ethernet card, one is Fast Ethernet card and the other is 10 Mbps Ethernet. Front-end node also have a CD-ROM drive and 17 inches monitor. Front-end node is also used as a gateway to separate the rest of cluster from outside network. This setup helps as minimize the use of the Internet address and separate heavy network traffic of computation tasks from external network. In our system, 100 Mbps Fast Ethernet Hub has been used to link all nodes together. The reason is that fast ethernet can deliver up to 100 Mbps at low cost. We plan to upgrade the connection to fast ethernet switch in the near future.

### 2.2 Software

The most popular OS for clustered computing and the one that we choose to use are Linux Red Hat 5.0 since Linux is free and contains source code to the systems that make it suitable for research and development. Moreover, there are plenty of good public domain software are available.

To use cluster for parallel processing, a parallel programming system must be installed to take care of parallel task creation/deletion, passing data among tasks, and synchronize tasks execution. Currently, available parallel programming system on our system are PVM, MPI-CH[5], BSP [17] and Scalapack Parallel Math Library.

## 3. SMILE Resource Monitoring System

### 3.1 Basic Requirements

Managing a cluster is different from managing a single system since resources are distributed among multiple nodes. Good management system must help cluster administrator to perform some management task such as:

- Shutting down or booting up any nodes.
- Remote rlogin and remote execute command to any node.
- Submit parallel command that execute cluster-wide and get the result back at management point.
- Browse system configuration of any node from a single point.
- Query important statistic such as CPU, memory, I/O, and network usage from any node or the whole cluster.

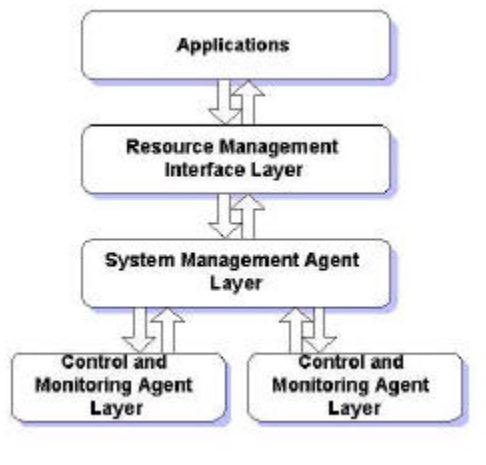
As we implement these requirements, we found non real-time management task can be accomplished using script language (such as perl, TCL/TK). In contrast, real-time task is difficult since it need a special resource monitoring system. Consequently, we implemented a monitoring system as presented in the following section.

### 3.2 System Architecture

The architecture of resource monitoring system is shown in Figure 1. Each compute node has a daemon named **CMA (Control and Monitoring Agent)** that collect node statistic continuously. This statistic is reported to a central resource management server that we named **SMA (Systems Management Agent)**. This server keeps track of system information for later retrieval. Management information is presented to user by a set of specific management applications.

The reason that we use centralize server is simplicity and efficiency. This is true for small to

medium size cluster. For large cluster, we plan to use hierarchical structure by dividing management responsibility into zones or domains and use multiple communicating servers to handle the information.



**Figure 2 Cluster Monitoring System Architecture**

A set of API called **RMI (Resources Management Interface)** has been provided for the developer of system monitoring and logging applications. This interfaces are available in C , TCL/TK , and Java Class library. Using these interface help speed up the development dramatically. Example of C language API is listed in Table 1. Also, Java Class enable use to integrate world wide web technology into monitoring system by having monitoring software coded as an applet and store them on www server. Users can access monitoring system by browsing through a pre-defined web pages and loaded the applet to their browser. More information about this approach can be found in [18].The advantages of our approach are:

- Simplified the design of system management agent.
- Reduce concern about an information inconsistency.
- Provide the uniform view of resources through API.

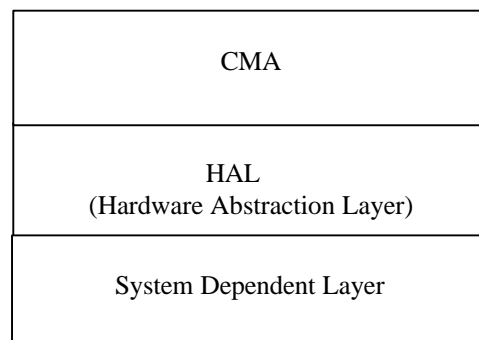
**Table 1 List of RMI C interface**

| Function       | Parameter           | Description   |
|----------------|---------------------|---|
| RMI_init       | None                | Perform some initialization tasks.                        |
| RMI_getnode    | Node id, Attributed | Get the value of specified attribute from particular node |
| RMI_getcluster | Attribute           | Get the value of specified attribute from all nodes       |
| RMI_close      | None                | Perform some clean up action                              |

However, this design has some drawbacks such as:

- This design leaves SMA as a single point of failure. If SMA goes down, all management system will cease to function properly. This problem can be solved by having a secondary SMA which handle all task after primary SMA stop functioning. In addition, we can nominate one of the CMA nodes to do this task.
- Single server implementation does not scale well large cluster. However, since we target our system to a cluster smaller than 64 nodes, current implementation is more than enough to handle the task. For large cluster, we plan to partition it into multiple partition and use hierarchical system of SMA to handle it.

Figure 3 shows the structure of CMA. To make CMA portable, we provide a layer called HAL (Hardware Abstraction Layer) that provides a uniform and machine independent interface to node statistical information. This design allows a very flexible and portable implementation. Porting CMA to other platform can be done easily by changing System Dependent Layer to match target hardware. Currently we support only LINUX operating system. In the future, we plan to extend to SOLARIS and Windows based systems also.



**Figure 3 Structure of CMA**

For SMA, the function of this subsystem is to gather the information from CMA on every node and store in an internal table. In addition, this subsystem response to user command to query system status and also forward some command to CMA on behalf of the user.

At present, all subsystem communicating using UDP protocol in order to reduce traffic and avoid using many TCP connection. We have added a simple protocol on top of UDP packet. The format of this protocol are as shown in Figure 4. The usage of each filed are:

| Protocol Type | Option | Data Length | Data |
|---------------|--------|-------------|------|
|---------------|--------|-------------|------|

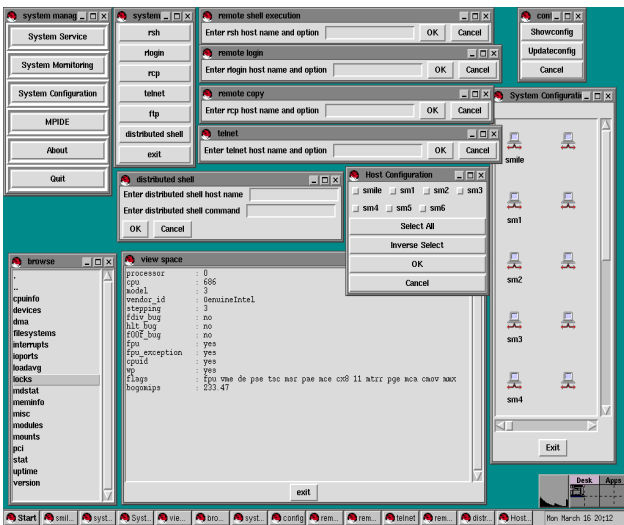
**Figure 4 Protocol Used to Communicate among Subsystem**

- Protocol Type: Specify the type of protocol used to communicate between each sub-system. This is useful for classifying the type of source and destination subsystem. Current types are CMA-SMA, SMA-RMI.
- Option: Specify message function that drive the handler routine in each subsystem.
- Data length : Specify the length of the data field.
- Data : contain information that need to be exchanged.

**4. Resource Monitoring Applications**

Employing the API as we have mention earlier, we develop a real-time monitoring of processor workload and memory usage of all nodes. This monitoring system is part of cluster management system that we are developing (see Figure 5). Current load monitoring program that we developed using the API provided is presented in Figure 6. This application enable us to easily observe problems or overload condition takes place on any node.

**Figure 5 SMILE Cluster Management Systems**

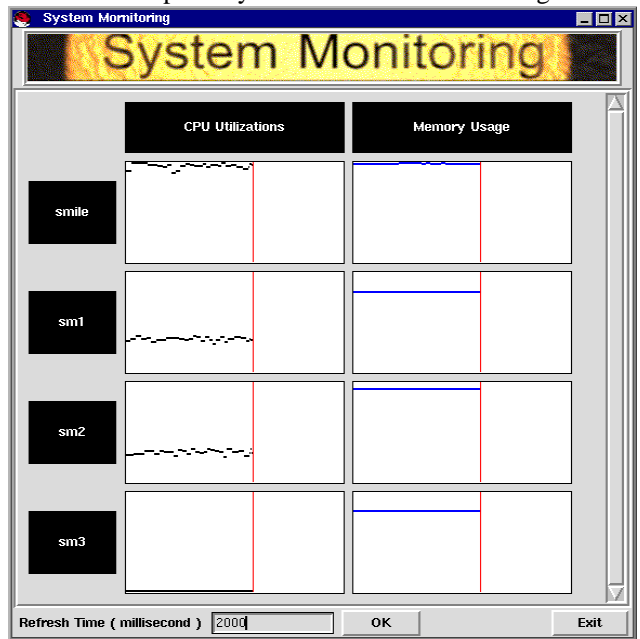


**5. Conclusion and Future Work**

In this paper, we describe our design and development of resources monitoring system on SMILE Beowulf Cluster. The information obtained enable us to gain deeper understanding about behavior of our cluster. Nevertheless, this work is far from complete since our goal is to provide a total solution for cluster management system. There are many issues that we plan address such as:

1. Evaluate users requirement and extend RMI interface to better suite the need. For example, we plan to add the API that allow user to start a process on a particular host.
2. Adding the support for efficient and scalable parallel UNIX command. We are currently working on script based paralle unix command. However, this approach suffers form high start up time or remote shell. Better and more efficient approach is to build the support and API for efficient scalable Unix command in CMA and SMA to shorten the process startup overhead.
3. Making all subsystem fault tolerance especially SMA.
4. Interface to Task scheduling and parallel programming systems to make all these tools make intelligent decision based on real time systems load.
5. Develop more monitoring application such as usage statistic for further analysis.

Finally, we hope that our contribution will lead to a powerful and publicly available cluster management



system that help engineer and scientist to harness the power of PC cluster for their tasks.

**Figure 6 Real-Time Load Monitoring**

**6. References**

1. M. Litzkow, M. Livny, and M.W. Mutka, "Condor- A Hunter of Idle Workstation," Proceedings of the 8<sup>th</sup> International Conference on Distributed Computing Systems, pp. 104-111, June,1988.

2. Green, Thomas P. , DQS User Interface Preliminary Design Document, Supercomputer Computations Research Institute, Florida State University, Tallahassee, Florida.
3. Sunderam, "PVM: A Framework for Parallel Distributed Computing", *Concurrency: Practice and Experience*, December 1990, pp. 315-339.
4. MPI Forum,"MPI: A Message Passing Interface", Supercomputing 93,1993
5. MPICH-A Portable Implementation of MPI, "<http://www-c.mcs.anl.gov/mpi/mpich>"
6. LAM/MPI Parallel Computing, <http://www.osc.edu/lam.html>
7. Venkatesh Krishnamoorthy, Putchong Uthayopas, and Kemal Efe, "Partitioning Strategies for Multiplying Dense Matrices on Workstation Networks", Proceeding of First International Workshop on High-Speed Network Computing HiNet'95 , April 25,1995 ,Santa Barbara, CA. pp. 74-82.
8. K.Efe, P. Uthayopas, V. Krishnamoorthy and T. Dechsakulthorn, "Parallel Algorithms for Workstation Clusters", Proceeding of 1995 ICPP Workshop on Challenges for Parallel Processing, 1995.(pp.140-147).
9. Putchong Uthayopas, Boriboon Patwiwat, Parallel Image Velocity Estimation using Workstation Clusters, Proceeding of 5<sup>th</sup> Annual Conference on Technologies Transfer between Thai Researchers in North America and Thailand, Edmonton, Canada , 1996.
10. David E. Culler, Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau, Brent Chun, Steven Lumetta, Alan Mainwaring, Richard Martin, Chad Yoshikawa, Federick Wong, "Parallel Computing on Berkeley NOW", Computer Science Division, University of California, Berkley,1997.
11. Thomas Sterling, Donald J. Becker, Daniel Savarese, John E. Dorband Udaya A. Ranawake, Charles V. Packer, "BEOWULF: A Parallel Workstation for Scientific Computation", Proceedings of ICPP95, 1995.
12. Martin Ewing, "Beowulf at SC97",<http://www.yale.edu/secf/etc/sc97/beowulf.htm>
13. Yuen Poovarawan, Putchong Uthayopas, "High Performance Computing: Needs and Application fo Thailand", EECON 20<sup>th</sup>, Bangkok, Thailand, November 1997.
14. Putchong Uthayopas, "Beowulf Class Cluster: Opportunities and Approach in Thailand", Presented in First NASA Workshop on Beowulf Class Computer Systems, NASA Jet Propulsion Laboratory, Pasadena, California, October 1997.
15. Putchong Uthayopas, Supaket Katchamart, Jittat Fakcharoenpol, Panit Hemunnopjit, SMILE: Toward the Affordable High-Performance Computing Platform using Network-Based Multicomputer, First National Computational Science and Engineering Symposium, Bangkok, Thailand, March 1997.
16. Putchong Uthayopas, Thara Angskun, Jullawadee Maneesilp, "Building a Parallel from Cheap PCs: SMILE Cluster Experiences", Proceedings of the 2<sup>nd</sup> Annual National Symposium on Computational Science and Engineering, pp. 255-264, Bangkok, Thailand, March 1998.
17. Jonathan M.D. Hill, D.B. Skillicorn," Lessons Learned From Implementing BSP", Technical Report PRG-TR-21-96, Oxford University Computing Laboratory, Oxford
18. Putchong Uthayopas, Chaiyaporn Jaikaew, Thitiwan Srinark, Interactive Management of Workstation Clusters using World-Wide-Web, Proceedings of Cluster Computing Conference 97, Atlanta, Georgia, March 1997.