PARMON: A Comprehensive Cluster Monitoring System

RAJKUMAR
School of Computing Science
Queensland University of Technology
825a, PLAS Lab, S-Block
Gardens Point Campus, Brisbane, Australia
rajkumar@fit.qut.edu.au

KRISHNA MOHAN and **BINDU GOPAL**
Operating Systems Group
Centre for Development of Advanced Computing
2/1, Ramanashree Plaza
Brunton Road. Bangalore, India
{krishmo, bindug}@cdacb.ernet.in

Abstract - Workstation clusters have off late become a cost-effective solution for high performance computing. C-DAC's PARAM OpenFrame is a large cluster of high performance workstations interconnected through low-latency, high bandwidth communication networks. Monitoring such huge systems is a tedious and challenging task since typical workstations are designed to work as a standalone system, rather than a part of workstation clusters. System administrators require tools to effectively monitor such huge systems. PARMON provides the solution to this challenging problem.

PARMON is a portable, flexible, interactive, scalable, location-transparent, and comprehensive environment for monitoring of large clusters. It follows client-server methodology and provides transparent access to all nodes to be monitored from a monitoring machine. PARMON allows to monitor critical system resources activities and their utilization at three different levels: entire system, node, and component level. It allows monitoring multiple instances of the same component such as CPU in SMP node.

The two major components of PARMON are ***parmon-server***—system resource activities and utilization information provider and ***parmon-client***—GUI based client responsible for interacting with parmon-server and users for data gathering in real-time and presenting information graphically for visualization. The PARMON Client is designed, developed, and implemented using the state-of-the-art object-oriented, client-server, and Java computing technologies. The PARMON-server is developed as a multithreaded server using POSIX/Solaris threads and C as Java does not support interfaces to access system internals. PARMON is being successfully used in monitoring PARAM OpenFrame Supercomputer, which is a cluster of 48 Ultra-4 workstations running SUN-Solaris operating system.

Keywords: Clusters, Single System Image, Monitoring, System Activities, and Resource Utilization.

1. Motivations and Goals

During 1980s, computer scientists believed that the performance of the computer can be improved by creating faster, more efficient processors. But this idea is being challenged by the concept of clustering which, essentially, means linking two or more computers to perform functions. The goal is to develop infrastructure (both hardware and software) so that end users need not know on which computer they are actually working. This enables multiple computers to work together as one system. Rapid changes in both areas of computing and communication (availability of commodity high-performance microprocessors and high-speed networks) facilitate this transition. These two enabling technologies are making Networks of Workstations (NOW) or Clusters of Workstations (COW) (in short, Network of Computers—PCs/Workstations) an appealing vehicle for parallel processing [1] leading to low-cost *commodity supercomputing*.

Computer scientists for years have tackled the problem of adding more performance while maintaining the integrity of systems. A major advancement that addresses both concerns is called *clustering* or *cluster computer* in which multiple computers are linked together to work as one system.

Monitoring of a cluster computer is a tedious and challenging task since typical workstations are designed to work as a standalone system, rather than a part of workstation clusters. This can be eased by software systems, which allows monitoring the entire system at different levels through an integrated GUI display. PARMON is one such system, which allows to monitor entire cluster, a single node, a component of node, or activities of a component. A node can have multiple instances of the same component types and PARMON allows to monitor such multiple instances.

As stated above, high performance computing on commodity hardware is gaining wide acceptance. For this to be practicable, it is important that systems provide a single system image at any one (or more) of the following levels:

- Hardware Level
- Operating System Level
- Message Passing Interfaces Level
- Language/Compiler Level
- Tools/Application Level

The transparent access to cluster computer is possible on a system exhibiting a single system image. The above approaches for achieving unified access to system resources have been discussed elsewhere [2]. In this paper, we discuss a cluster monitoring system, PARMON, which provides a unified means to monitor cluster computer at tools/application level.

A large network interconnects many computers from different vendors having different machine architectures and running different operating systems. In such a network of heterogeneous systems, the monitoring system must be portable. This is achieved by developing PARMON using Java programming language.

The motivations for using Java are its features as highlighted by its definition [3], "Java: A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi-threaded, and dynamic language". Among these features, Java makes network programming easier by encapsulating connection functionality in socket classes [4]. Java is mostly used in writing distributed computing programs or GUI based programs because of its portability feature and Java goes a lot further than most languages to obtain not mainly just portability but identical program behavior on different platforms. Java is mainly intended for the development of object-oriented network based software for Internet applications. Networking, multithreading and GUI-components features of Java have been used extensively for developing PARMON client.

2. Overview of C-DAC HPCC Software Architecture

C-DAC HPCC (High Performance Computing and Communication) software is an open and flexible parallel-and-distributed processing environment for a cluster of UNIX workstations [5]. The components of C-DAC software architecture are shown in Figure 1. The software architecture allows the collection of workstations to be viewed as independent workstations, cluster of workstations, or a MPP (Massively Parallel Processing) system connected through a scalable low latency and high bandwidth interconnection network.

The HPCC software allows users to develop and execute sequential, message passing and data parallel programs. Efficient MPI (Message Passing Interface) implementation provides the required support for developing message passing applications, while support for data parallel programs is provided through HPF (High Performance Fortran). High performance communication protocols (Light-Weight Protocols) and a rich set of program development, system management and software engineering tools are aimed at offering high performance services and usability.

C-DAC HPCC software is a complete solution for enterprises that need to create and execute parallel programs on UNIX clusters. Performance for both parallel and distributed applications is guaranteed through efficient implementation of lightweight communication protocols. In addition, the environment supports effective monitoring and administration of the multiple resources of large UNIX clusters.

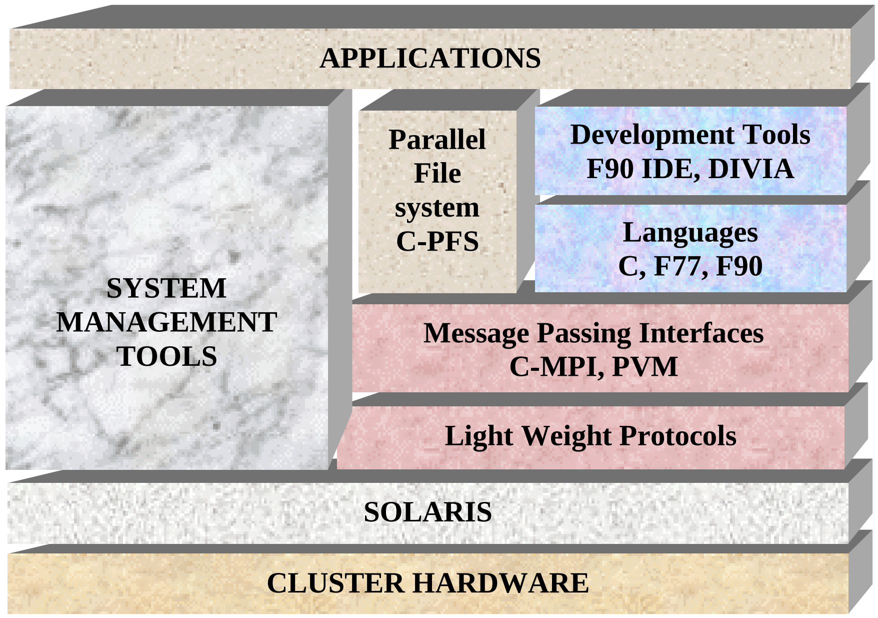


Figure 1: C-DAC HPCC Software Architecture

The following section discusses the components of C-DAC HPCC system software in brief:

PULP: Light weight communication software. Layered over ParamNet and Myrinet for UNIX Clusters.

C-MPI: Optimized implementation of MPI for Cluster of Multi Processors (CLUMPS). Both point-to-point and collective calls have been optimized. Effectively uses both shared and distributed memory of CLUMPS.

C-PFS: Parallel File System. Provides MPI-IO file system interface to parallel applications.

CAF90: Fortran 90 compiler. Jointly developed by C-DAC and APOGEE, USA. Integrated development environment with debugging, profiling, browsing and project management support.

C-F77to90: A tool for converting FORTRAN-77 to Fortran 90 programs. An essential product for programmers who have large FORTRAN 77 code to maintain.

DIViA: Parallel program correctness and performance debugger. Detects communication bottlenecks and supports message debugging. Provides architecture and communication layer neutrality.

PARMON: Cluster monitoring tool. Monitors the cluster as a unified resource. Helps in detecting performance bottlenecks and also acts as an interface to diagnostic tools.

PARCOM: Parallel Unix Commands. Provides parallel extensions to traditional UNIX commands.

MetricAdvisor: Software engineering tool for metrics. Evaluates Halstead, McCabe, Complexity Density, and Fan-in and Fan-out metrices.

The detailed discussion of the above components except PARMON is beyond the scope of this paper and the reader is advised to refer to the C-DAC website, http://cdacb.ernet.in, or the paper [5] for details. The remaining part of this paper discusses PARMON system model and architecture, monitoring system resources and activities with PARMON, implementation of PARMON, related works, and future directions.

3. PARMON Overview

PARMON allows the user to monitor system activities and resource utilization of various components of workstation clusters. It monitors the machine at various levels: component, node and the entire system level exhibiting a single system image. PARMON allows to monitor the following:

- Aggregate of System Resources Utilization
- Process Activities
- System Log Activities
- Kernel Activities
- Multiple instances of the same resource

PARMON allows to define events and its automatic triggering whenever event condition is met during runtime. It also provides physical and logical views of the components of the system.

The important features of PARMON include the following:

- Exploits developments of latest technologies, hardware and software features for communication and imaging.
- Allows the user to build system database (nodes and groups) comprising node name, communication interfaces, and specific area of a disk to be monitored.
- Supports listing of system information and machine configuration of all nodes of a cluster.
- Allows instrumentation of system resources such as CPU, Disk, Memory, and Network and their parameters both at macro and micro level.
- Supports monitoring of cluster at node level, group level, or entire system level and thus exhibits a single system image.
- Allows the parallel execution of selected operations on a single or group of workstations, real-time and interactive resource monitoring (e.g., processor, memory, disk, and network utilization), and normal maintenance tasks such as node or cluster shutdown.
- PARMON Client is portable to all platforms supporting Java Runtime System, JVM (Java Virtual Machine) and PARMON Server is portable across all machines running Solaris.

4. PARMON System Model: Architecture

PARMON consists of the parmon-client and the parmon-server (parmond). The system model is shown in Figure 2 and it follows client-server paradigm with nodes to be monitored acting as servers and monitoring system acting as a client. The cluster nodes can be monitored from any workstation, PC, or a node of the cluster itself.

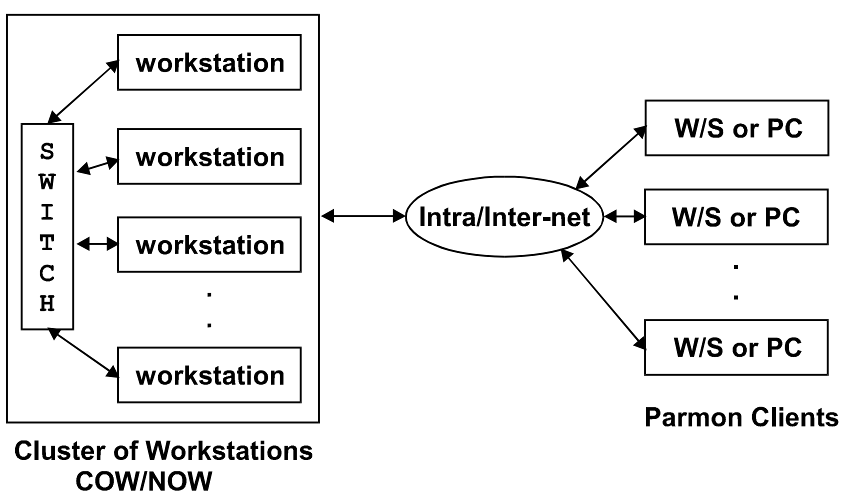


Figure 2: PARMON Cluster Monitoring System Model

The PARMON server is loaded on all the nodes that need to be monitored. A client requests for parameters through message passing and gets server response in the form of messages. The client interprets the messages and converts them into appropriate format for graphical visualization.

A client can either monitor all the nodes or selectively monitor a few nodes of the cluster. For effective monitoring, the concept of group is supported. A set of nodes forms a group and nodes are selected based on the allocation of resources to various user groups. Such grouping mechanism helps in monitoring and gathering usability statistics, with which the system administrator can change the resource allocation strategy.

5. Monitoring System Activities and Resource Utilization

The following section discusses monitoring of system activities and resource utilization using PARMON. This knowledge helps in understanding techniques used in the design, development and implementation of PARMON. The user interacts with PARMON through PARMON-Launcher shown in Figure 3 to monitor utilization of system resources such as Memory and Disk and system activities such as execution of program [6]. It also allows software instrumentation of resource activity parameters through kernel-data-catalog.

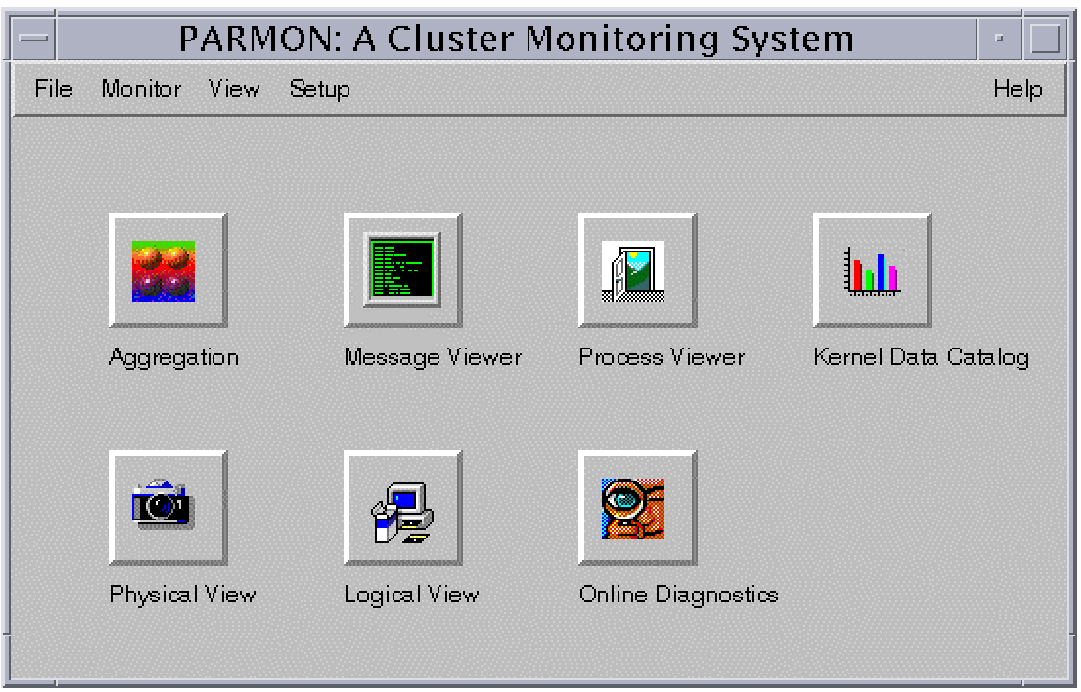
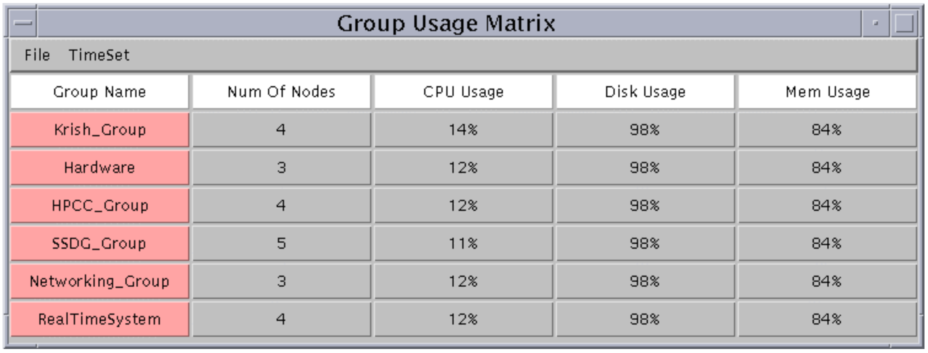


Figure 3: PARMON Launcher

5.1 Aggregation in Visualization of Resource Utilization

The use of aggregation in visualization allows to scale the visualization to whole cluster. The administrator can create user-groups containing a set of nodes based on the allocation of resources policy and monitor them using PARMON as shown in the Figure 4. The same statistics for different nodes are combined to get a single statistic and this technique is called group/machine utilization. The results shown in Figure 4 can also be presented using pie charts or bar graphs in PARMON.


Group Usage Matrix

File TimeSet

Group Name	Num Of Nodes	CPU Usage	Disk Usage	Mem Usage
Krish_Group	4	14%	98%	84%
Hardware	3	12%	98%	84%
HPCC_Group	4	12%	98%	84%
SSDG_Group	5	11%	98%	84%
Networking_Group	3	12%	98%	84%
RealTimeSystem	4	12%	98%	84%

Figure 4: Resource Utilization by Groups in a Cluster

5.2 Process Activities

The utilization of CPU resource can be measured by monitoring process activities, which helps in identifying CPU/Memory-intensive processes. The parameters that can be monitored using PARMON are: pid, command, uname, uid, nice, status, user (%), sys (%), total (%), total CPU (%), and start up time. PARMON continuously updates the process and system data at a user specified sampling interval. The snapshot of process activities is shown in Figure 5 and this information can be order based on selected parameters such as process-id, user name, etc. The information displayed by this appears similar to `top` utility.

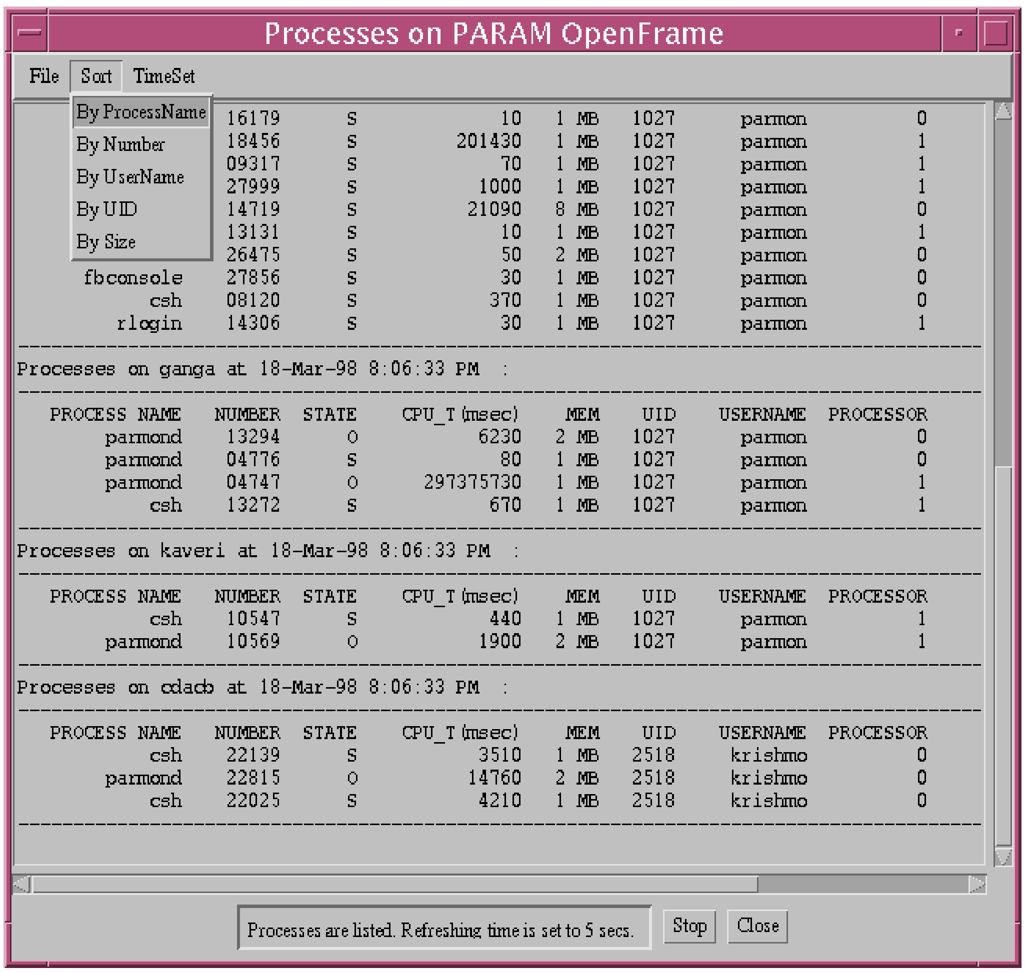


Figure 5: Information on Resources Consumed by Processes

5.3 System Logs

PARMON helps in effective monitoring of system logs maintained by the operating system. It allows to process system messages and syslog files for entries that occur at a specific time, or for entries that contain a specific keyword or word pattern.

5.4 Kernel Activities

PARMON supports software instrumentation of system resources such as CPU, Memory, Disk, and Network and their activities. When a particular resource has more than one instance, PARMON allows monitoring of each resource instance individually. Invocation of kernel-data-catalog option allows instrumentation of kernel activities related to resources such as CPU, memory, disk, and network as discussed below.

CPU Parameters

The monitoring of CPU parameters helps in understanding how CPU is being utilized. PARMON allows to monitor the number of mutex, interrupts, context switches, system calls (Figure 6), forks, execs, page in, page out, swap-in, and swap-out operations performed per second and displays these activities using graphs. It also allows monitoring of process run queue, I/O queue, and swap queue.

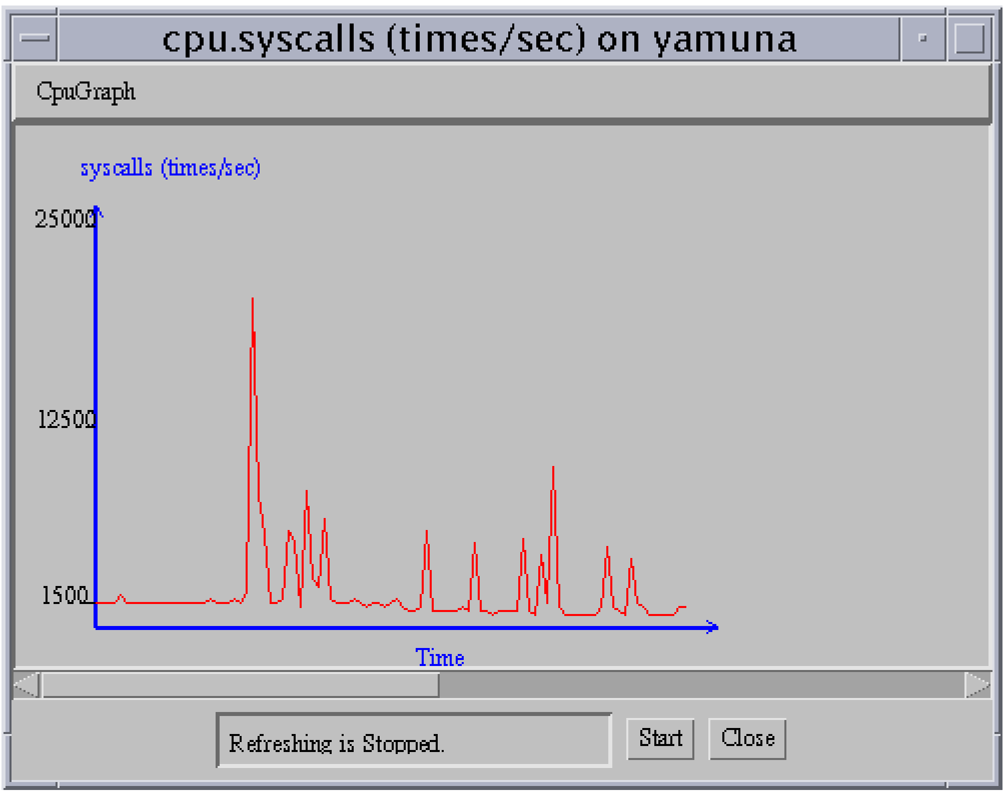


Figure 6: Systems Calls generated by Processes Running on the node in a Cluster

Memory Parameters

PARMON allows continuous instrumentation of memory availability, memory in-use, free memory, percentage of memory in-use, reserved swap space, allocated swap space, and available swap space.

Disk Parameters

PARMON allows to monitor disk operations such as reads, writes, number of jobs waiting in the queue for disk service, and disk's run-time and wait-time.

Network Parameters

The software instrumentation of network parameters such as input packets, output packets, errors in packet transmission, helps in detecting network bottlenecks. PARMON also allows displaying the percentage of incoming and outgoing data packets containing packet format errors.

5.5 Component View: Physical and Logical

PARMON allows displaying system picture and that of a few important components, which helps the user in quick understanding of machine's physical look-and-feel. It also displays different views of the machine.

The Logical View allows to display system components in a hierarchical manner. System components include processing elements, file system, and network components, which are probed dynamically. Each item in the hierarchy diagram can be further probed for more detailed information. For instance, when file system (fs) is probed, it shows all logical partitions of the disk and other details such as partition name, allocated disk space, and disk usage both in terms of bytes and percentage.

5.6 Device Control

PARMON also supports control of multiple instances of the same resource, like the multiple CPUs in a SMP node where it allows to set a CPU in on-line or off-line mode.

5.7 Events Generation

PARMON allows to define events such as sending e-mail when the user crosses resource utilization limits. This helps the administrator to have effective control over system resource utilization and for changing resource allocation policies.

5.8 Diagnostics

Under Solaris, PARMON integrates SunVTS to do validation, testing, and stress testing on external devices such as disk and tape drives and network connectivity. SunVTS helps in locating problems under Solaris with components that have managed to pass the configuration test (POST), as well as problems with external devices that POST knows nothing about.

5.9 Data Representation

PARMON uses Pie chart, Bar chart and Line graphs for representing resource usage such as disk, memory and graphs of representing various kernel activity parameters of CPU, disk, network, etc.

5.10 On-line Help

PARMON offers comprehensive online help. User can choose the functionality in the main help menu and then navigate to specific details of how each service can be accessed and interpreted. The online help also has a glossary of technical terms and parameters that are often used in PARMON or among cluster computing community.

5.11 Miscellaneous Features

The additional features supported by PARMON includes the following:

- Allows to issue a specific command on selected nodes or all nodes in a cluster.
- Allows to list users working on selected nodes or entire cluster.
- Allows to broadcast message to selected or all nodes in a cluster.
- Allows to retrieve system information and configuration.
- Allows to retrieve information related to packages installed on nodes.

6. PARMON Implementation

The two major components of PARMON are ***parmon-server***—system resource activities and utilization information provider and ***parmon-client***—GUI based client responsible for interacting with parmon-server and users for data gathering in real-time and presenting information graphically for visualization. The PARMON Client is designed, developed, and implemented using the state-of-the-art object-oriented, client-server, and Java Computing technologies. The PARMON-server is being developed as a multithreaded server using POSIX/Solaris threads and C as Java does not support interfaces to access system internals. The PARMON-client also maintains information related cluster setup, groups, node location details. This information needs to be created by PARMON user so that it knows about all those nodes, which forms a cluster or a group.

6.1 Server Implementation

Multithreaded implementation of servers allows them to serve requests of multiple clients simultaneously with improved response time and throughput. The server creates a new thread for every request it receives, and assigns responsibility of serving the request to this new thread so that the main server-process can handle other requests waiting in queue or wait for new requests.

PARMON server is a multitheaded server and it communicates with clients over parmon-port (one of the free sockets reserved for parmon) and provides defined services as discussed below:

6.1.1 PARMON Server Socket

PARMON server opens a connection endpoint by invoking `socket()` and binds it to a specified port or parmon-port through `bind()`. It blocks and waits for a client request for a connection by invoking `listen()`. When the connect request arrives at the server machine, the server calls `accept()` to accept the connection over which server and clients communicate. Once the connection is established, the server creates a service-thread, which is responsible for serving the client request and then it goes ahead with serving other requests.

6.1.2 Multithreading PARMON Server

PARMON server is multithreaded using POSIX and Solaris Threads Interface to enhance its portability. However, only one of them is used for creating executable file through conditional compilation. PARMON server and clients communicate with each other by exchanging messages. The client requests for parameters through messages, the server creates a thread, called service-thread, for each request by calling `thr_create()` for Solaris threads and `pthread_create()` for POSIX threads. The service-thread interprets the meaning of requests, processes them on local node and communicates the results to the client. Both service-thread and client may communicate with each other more than once in order to complete the request. The service-thread dies after providing requested service by releasing resources (such as buffers, sockets) occupied by it during its lifetime.

6.1.3 PARMON Services and their Implementation

The services provided by server can be mainly classified into two categories: the first, kernel related services and the second, system status and their configuration related services. The kernel related services focus on the activities on CPU, Disk, Memory, Network during process execution. The second type of services offered by PARMON includes the information about process activities, system logs, users, physical and logical view, system information, and system configuration. The CPU parameters include busy(%), idle(%), sys(%), user(%), wait(%), context_switches, interrupts, system calls, forks, execs, pgin, pgout, run_queue_length, io_queue_length, etc. The Disk parameters include operations such as reads, writes, disk wait percentage, disk active percentage, disk service percentage, etc. The memory parameters consist of available memory, free memory, used memory, swap avail, swap free, swap reserved, etc. Input packets, output packets, input error, output error, collisions, error percentage, defer percentage are some of the network parameters.

Kernel Parameters

In Solaris, the kstat (kernel statistics) facility is a general purpose mechanism for providing kernel statistics to the users. The kernel maintains a linked list of kstats. Each kstat has a common header section and a type-specific data section and it can be characterized as belonging to some broad class of statistics, for instance, disk, net, vm, etc. This field can be used as a filter to extract related kstats. Solaris offers a set of interface functions to access kernel statistics. The kstat_open() initializes the kstat control structure, which provides access to the kernel statistics library. Call to kstat_read() gets data from the kernel for the kstat, and kstat_close() frees all the resources that were associated with the pointer returned by kstat_open().

Physical memory parameters data can be obtained through sysconf(). Swap memory information is obtained through calls to Kernel VM routines: The kvm_open() initializes a set of file descriptors to be used in subsequent calls to these routines. The kvm_read()transfers data from the kernel image specified by pointer returned by the kvm_open(). The kvm_close() closes all the file descriptors associated with the pointer returned by kvm_open().

System Status and Configuration

PARMON supports access to parameters related to system configuration, users, and processes; and they are implemented by using relevant system calls [7]. The getutxent() allows to access information related to system users. The readdir_r() allows to access information related to proccesses. The sysconf() allows to access system configuration such as maximum number of files that can be opened simultaneously, maximum size of a core memory page. The sysinfo() allows to access system information such as host name, release number, system architecture, vendor details.

6.2 PARMON Client Implementation in Java

PARMON client is a GUI based tool, which interacts with users and server. The communication between PARMON server and client takes place by sending or receiving messages over TCP/IP sockets. The following interaction takes place between PARMON user, client, and server repeatedly for every action of the user:

1. The user interacts with PARMON client for monitoring and selects appropriate option.
2. The PARMON client interprets user requests and invokes appropriate event handler or closes the current window for exit or close window request.
3. The event-handler performs appropriate function. In most cases, it creates an object of user-defined service class extending the Frame class and having the capability to interact with server and the user.
4. The user selects a set of nodes to be monitored.
5. PARMON client maps node name to I/P address by accessing node-database and establishes a connection with the server by creating an object of the Socket class by supplying IP address and parmon-socket number (default or user supplied). It also creates input and output streams for this socket object.
6. PARMON client converts user request into messages and communicates the same to server.
7. PARMON server processes client requests and communicates the results to the client.
8. PARMON client interprets messages and converts the same into appropriate format—presents it in the form of graph, pie chart, bar graph, table, or in a text box.
9. PARMON client then closes all connections made to servers.

The user class ParmonSocket is one of the most commonly used classes. It is responsible for mapping node names to their I/P-address by accessing the node database, establishing the socket connection to the server, and creating input and output streams to the socket. This class has methods for reading from the server and writing commands to the server through socket. The constructor of the class `ParmonSocket` has the following syntax:

```
public ParmonSocket( String nodes[], int port_id)
```

The `ParmonSocket` constructor takes two arguments, `nodes` and `port_id`. The argument `nodes` is an array of the class `String`, which contains the names of the servers to which the client may request for connection through socket. The argument `port_id` is an integer variable representing socket/port number over which clients requests for connection to the server.

The PARMON client need to interpret server response for its request as follows: The server response to client request contains the following two items.

| **Request_Status** | **Data/Error** |

The *Request_Status* can be SUCC, STOP, or CONT and the client needs to interpret it as follows:

SUCC: The server honors the client request and next item, *Data*, contains the result.

STOP: The server could not serve the request from the client, so stop further requests through the current connection. The second item, *Error*, contains the error message and the reason for this error.

CONT: The current request cannot be served and remaining requests from client can continue. The second item, *Error*, contains the error message and the reason for this error.

The processing is simplified by the class `ParmonSocket` by supporting methods `read()` and `response()`. After sending request, invoke `read(int i)` which reads `node[i]` messages and returns *Request_Status*. It also separates status condition with the response portion, which can be accessed through the method `response()`.

Kernel Data Catalog window in Java

The Kernel Data Catalog window shown in Figure 7 allows to select node and type of parameters to be instrumented. This class extends the class `java.awt.Frame` and its layout is set to null by `setLayout(null)`, so that components position can be controlled by the programmer. This window's resizable property is set to false by `setResizable(false)`.

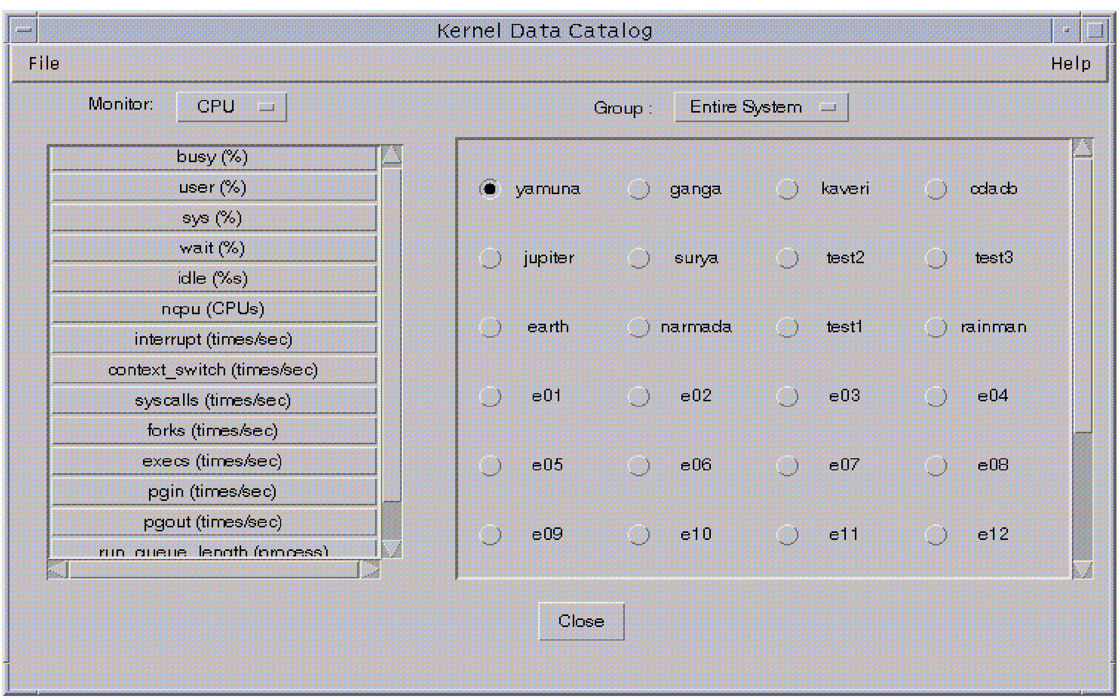


Figure 7: Kernel Data Catalog

Two panels are created and added to this window. The left panel shows parameters related to the resources, which the user selects for viewing the activities. This panel has a choice component whose items are CPU, disk, network and Memory and another panel contains buttons indicating parameters related to the resource selected in the choice component.

The right panel shows the group lists and respective nodes in them. Any one of these nodes needs to be selected at a time for monitoring resource activities. The required resource activity will be chosen from left panel. This panel has choice component and another panel with radio-buttons. The choice component has group names as its items. The group information will be obtained from group database. The panel having radio-buttons represent nodes and therefore only one node can be selected for monitoring. When the group is selected from the choice component, the nodes belonging to that group are displayed in the radio-buttons panel.

When the user clicks on any parameter, the event handler will be invoked by supplying event details. The event handler will determine type of parameter to be monitored and node name and invokes/creates appropriate method/object, which performs instrumentation of the selected parameter on the selected node.

6.3 Node and Group Database

The PARMON client maintains nodes and group information in `Nodes.db` and `Groups.db` files. The file `Nodes.db` maintains the information about nodes in the system for monitoring. The user interface for creating node database is shown in Figure 8. It takes the name of the server and its IP address. On the righthand side, there are a list of nodes. If the node has to be removed from this list, select respective node name and click on Remove Node button and click on Ok button to incorporate these changes into the `Nodes.db` file.

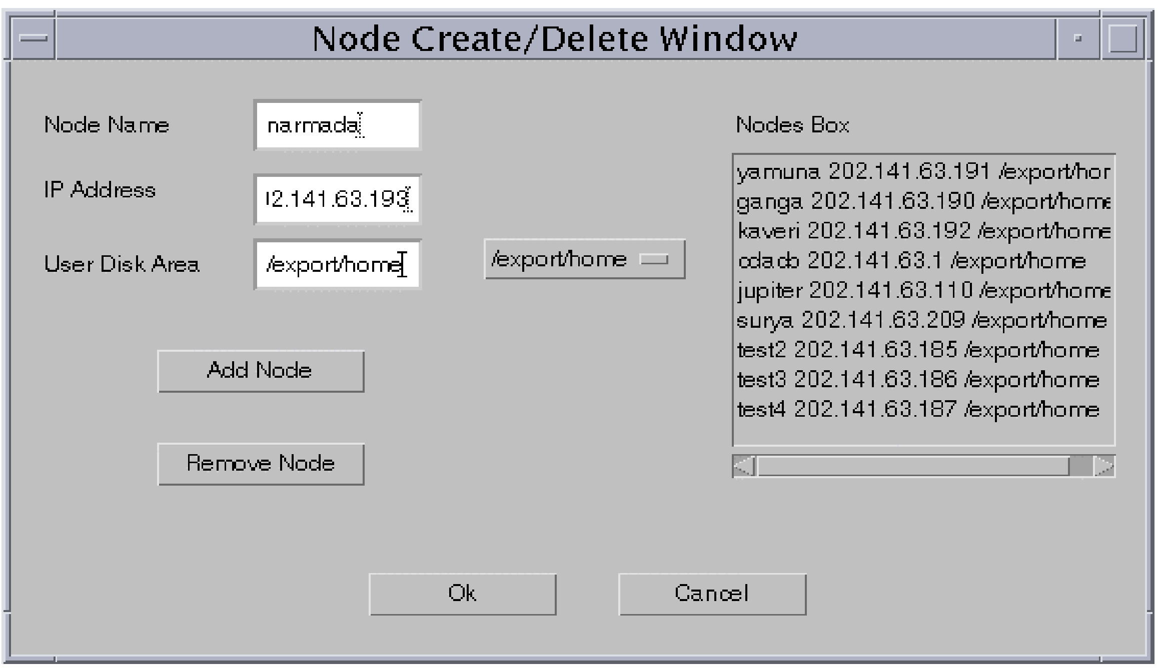


Figure 8: Node Setup window

PARMON allows creation of user groups containing a set of nodes based on resource allocation and its monitoring. The Group Create window allows the user to select a set of nodes that forms the user group. The group details (group name and its nodes) are maintained in the `Groups.db` file. The Group Modify window offers the flexibility to change the configuration of the group by selecting new nodes or deselecting nodes from the list of nodes.

7. Related Works

There are many projects investigating system administration of workstations and clusters supporting parallel computing across networks of workstations, including NOW (Network of Workstations) at the UC Berkeley and SMILE (Scalable Multicomputer Implementation using Low-cost Equipments) at Kasetsart University, Bangkok, and Solstice™ SyMon™ at Sun Microsystems. NOW system administration tool gathers and stores data in the relational database. It uses a Java applet as primary interface allowing users anywhere in the world to monitor the system from their browser [8]. SMILE administration tool is called K-CAP and its environment consists of *compute nodes*—executes the compute-intensive tasks, *management node*—file server and cluster manager, and *management console*—controls and monitors the cluster [9]. The K-CAP user interface uses WWW and Java Applets for connecting to K-CAP management node through predefined URL address in the cluster. SyMon allows to monitor a standalone workstation, but uses client-server technology for separating the node to be monitored and the monitoring station [10].

The NSR (Node Status Reporter) provides a standard mechanism for measurement and access to status information in clusters of workstations [11]. Parallel applications/tools can access NSR through its Tool/NSR Interface.

Unlike NOW and SMILE cluster monitoring systems, PARMON user interface is developed as a Java application instead of Applet. PARMON also allows the user to monitor the cluster system from anywhere in the world if nodes of the cluster are accessible through Internet. Global parallel clusters built using Internet servers can easily be monitored using PARMON.

8. Conclusions and Future Directions

PARMON is being successfully used in monitoring PARAM OpenFrame Supercomputer, which is a cluster of 48 Ultra-4 workstations running SUN-Solaris operating system. It is scalable across a cluster of hundreds of workstations including Internet-based global parallel clusters.

PARMON introduces some overhead on network as its clients and server communicates with each other by exchanging messages. The message size and the amount of time required to process a request depends on the request-type. There is no way to avoid network overhead introduced by PARMON as it uses client-server paradigm.

Our experience in building PARMON shows that Java offers a flexible environment for building scalable and extensible applications. PARMON can be easily extended to support new features as it has been designed using latest object-oriented and internet-based Java technology. PARMON client is fully portable as it has been developed using Java, and the server needs a few changes to support new platforms.

PARMON can easily be extended to support web-based user interface for monitoring. This can be achieved by converting PARMON-client (which is a Java application) into Applet; and building an intermediate server running on web-sever that acts as link between PARMON server running on cluster nodes and PARMON web-based user interface containing Applet. Security is another important issue that PARMON needs to address when web-based monitoring support is provided.

Good cluster management systems are crucial in exploiting the computer clusters as a high performance computing platform. The future (high performance) computing systems are expected to be cluster based and heterogeneous in nature. To continue with this trend, we are porting PARMON to Linux. We are also planning to port PARMON to Windows NT, as NT-based clusters are becoming popular. The availability of PARMON for Unix, Linux, and NT-based clusters, makes it a truly heterogeneous cluster monitoring system.

Acknowledgments

The authors are grateful to all the members of C-DAC, Bangalore. In particular, we thank Mohan

Ram, Arun Babu, Nitin, Raghavendra, Vikas, and Soma Sundaram for their support, suggestions, comments, and advises during the design and development.of PARMON. We thank Smrithi R; and Mudlapur R and Binu Thomas of Queensland University of Technology, for proof reading drafts of this paper.

References

1. T. Anderson, D. Culler, D. Patterson, and the NOW team. *A Case for NOW (Networks of Workstations)*. IEEE Micro, pages 54-64, February 1995.
2. Rajkumar. *Single System Image: Need, Approaches, and Supporting HPC Systems*. Proceedings of the Fourth International Conference on Parallel and Distributed Processing, Techniques and Applications (PDPTA'97), CSREA Publishers, Las Vegas, USA, 1997.
3. Sun Microsystems. *Java Language - A White Paper*. Sun Microsystems Computer Company, 1996.
4. Patrick Naughton and Herbert Schildt. *JAVA: The Complete Reference*. Mc-Graw Hill Inc., 1997.
5. Mohan Ram N, Sasi Kumar S, Vijay P Bhatkar, and Arora R K. *HPCC Software: A Scalable Parallel Programming Environment for UNIX Clusters*. 1998.
6. HPCC System Software PARMON Group. *PARMON User Manual*. © C-DAC, 1998.
7. Berney Goodheart and James Cox. *The Magic Garden Explained: The internals of UNIX System V Release 4*. Prentice Hall, 1993.
8. Eric Anderson and Dave Patterson. *Extensible, Scalable Monitoring for Clusters of Computers*. Proceedings of the 11th Systems Administration Conference (LISA '97), October 26-31, 1997, San Diego, California, USA.
9. Putchong Uthayopas, Chaiyaporn Jaikaew, and Thitiwan Srinak. *Interactive Management of Workstation Clusters Using World Wide Web*. Cluster Computing Conference-CCC '97, Web Proceedings, http://www.mathcs.emory.edu/~ccc97/, (http://www.eng.ku.ac.th/~pu/smile/), 1997.
10. Sun Microsystems. *Solstice SyMON 1.1 User's Guide*, Palo Alto, CA 1996.
11. C. Roder, T. Ludwig, and A. Bode. *Flexible Status Measurement in Heterogeneous Environment*. Proceedings of the Fifth International Conference on Parallel and Distributed Processing, Techniques and Applications (PDPTA'98), CSREA Publishers, Las Vegas, USA, 1998.