# Job-Oriented Monitoring of Clusters

Vijayalaxmi Cigala
Computer Engineering Department,
MIT College of Engineering, Pune

Dhirajkumar Mahale
Computer Engineering Department,
MIT College of Engineering, Pune

Monil Shah
Computer Engineering Department,
MIT College of Engineering, Pune

Sukhada Bhingarkar
Computer Engineering Department,
MIT College of Engineering, Pune

***Abstract*—There has been a lot of development in the field of clusters and grids. Recently, the use of clusters has been on rise in every possible field. This paper proposes a system that monitors jobs on large computational clusters. Monitoring jobs is essential to understand how jobs are being executed. This helps us in understanding the complete life cycle of the jobs being executed on large clusters. Also, this paper describes how the information obtained by monitoring the jobs would help in increasing the overall throughput of clusters. Heuristics help in efficient job distribution among the computational nodes, thereby accomplishing fair job distribution policy. The proposed system would be capable of load balancing among the computational nodes, detecting failures, taking corrective actions after failure detection, job monitoring, system resource monitoring, etc.**

***Index Terms—Cluster computing, Job monitoring, HPC clusters.***

## I. INTRODUCTION

Cluster computing is a technology by which a huge problem is solved by splitting it into smaller parts and distributing it among several parallel computing nodes. Cluster, basically follows Divide-n-Conquer policy to solve huge problems. HPC clusters are the clusters that are required to deliver high performance and low response time along with accurate results. Their intentional role is to provide high computational power against high availability. Clusters are subset of Grids. A grid comprises of several geographically distributed homogenous or heterogeneous clusters.

Grids become difficult to manage with large number of clusters in it. But, if the clusters were managed at local-level, it would be very helpful in monitoring huge grids. Several job processing systems do exist today, but they are either standalone or do not have the detailed monitoring recovery mechanism. Some of the obvious questions are:

1. Does the system perform job monitoring?
2. Does it provide a reporting capability?
3. Does it detect failures?
4. Does it have capability to recover from failures?
5. Does it notify failure to the administrator?

The aim of the proposed system is to monitor jobs being executed at every node in a cluster. This in turn helps in better resource utilization by the jobs and better throughput. This information further can be used for fair job distribution on the nodes and load balancing.

## II. RELATED WORK

Large numbers of tools have been developed in the field of clusters. But the previously developed tools follow system centric approach. A tool named Ganglia[1] is used widely to monitor system resources available on the computational nodes. Ganglia provides web portal based system resource monitoring tool which helps to find out overall system resources available on every computational node in a graphical way. The ganglia system

comprises two unique daemons, a PHP-based web front-end, and a few other small utility programs namely Ganglia Monitoring Daemon (gmond) and Ganglia Meta Daemon (gmetad). Ganglia has a PHP-based frontend.

Similarly LVS Cluster Management[2] helps in monitoring system resources. The major work of cluster monitoring in LVS is to monitor the availability of real servers and load balancers, and reconfigure the system if any partial failure happens, so that the whole cluster system can still serve requests. To monitor the availability of real servers, there are two approaches, one is to run service-monitoring daemons at the load balancer to check server health periodically, the other is to run monitoring agents at real servers to collect information and report to the load balancer. Cerebro is another tool that monitors system resources by removing the overhead of XML as against Ganglia that uses XML. Another tool named MonALISA[3] is a globally scalable framework of services to monitor and help manage and optimize the operational performance of Grids, networks and running applications in real-time. A tool named Ka-admin[4] was developed for data aggregation over a large cluster. But, as said before, the tools developed have been system-centric.

Reference [5] proposes a job monitoring system for clusters with different system architecture. Reference [5] proposes a scheme wherein administrator administrates distantly over internet. Reference [6] proposes a job monitoring system with different architecture. Also in reference [6], heuristics are not considered for load-balanced job distribution among the nodes. Thus, the proposed system achieves job monitoring of clusters with an altogether different architecture and policies.

## III. PROPOSED ARCHITECTURE

System-centric approach stresses more on the system resources available on cluster and on its computational nodes rather than on jobs. Our paper describes a new approach that is job-centric. It captures every phase of the lifecycle of a job. This helps in better understanding about the job execution. Also, failures if any can be detected and corrected quickly with this approach. This approach not only looks into job monitoring, but also system resource monitoring.

The sequences of operations that are performed are:

1. User is authenticated at the master node.
2. User registers the job to be executed at the master node.
3. Network scan is performed to determine the available nodes by sending heartbeat messages to all the nodes.
4. Master node then allows user to make a choice of nodes for job execution.
5. Master node then learns about the efficiency of the selected nodes and then dispatches them according to their capability.
6. The sub tasks dispatched to the nodes are executed concurrently at the respective nodes.
7. Reporting is done during concurrent execution at the master node to gather status of the jobs being executed.
8. Once the job completes, result is made available at the master node by aggregating results from the nodes.

Figure 1 shows proposed system architecture which consists of 3 key components:

1. Job Dispatcher
2. Job Monitor
3. Job Logs

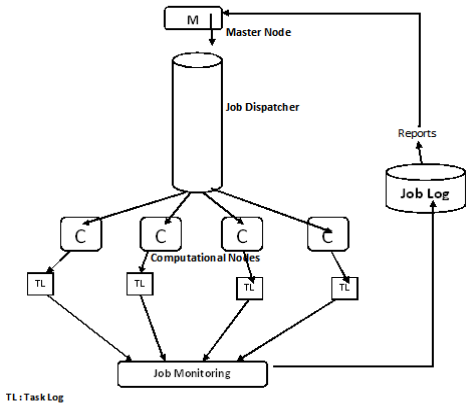


Figure 1: Architectural Diagram

*Job Dispatcher* – A job once accepted by the master node, the job is left to the Job Dispatcher to dispatch the sub-tasks to the respective computational nodes. The Job Dispatcher dispatches the jobs to the nodes based on the policies fixed for the system for their further execution.

*Job Monitor* – This component monitors all the activity and processing performed by computational nodes. It records status of each process and maintains its log. Also it integrates status of the sub-tasks executing at different processors and maintains overall job log.

*Job Logs* – It is used for storing information of job (both static as well as run-time). This information is then used to prepare report and for analysis purpose. Report data is stored in log files as well as some internal files.

Once a job is registered at the master node, the efficiency, current load and the previous performances of the nodes are taken into consideration for load balancing among the nodes. This policy helps in providing better throughput of the system. The jobs get distributed according to the above said policy on the respective nodes.

The sub tasks assigned to the nodes start being executed on the respective nodes. During execution, the nodes log the job status details as well as report them to the master node periodically. A daemon process specially dedicated to a specific job is responsible to collect all the status information of the jobs. This daemon process then reports the status of the job at the master node. All the jobs are monitored in the similar way and are reported at the master node. During sub task execution, if any sub task fails or node fails due to some unknown reason, the daemon reports this at the master node. Notification is also sent to the administrator. Master node then determines the node that is best available at that point for the re-execution of the failed sub-task thereby assuring failure handling.

Reports are generated in both, overall as well as at individual sub-task level using graphs as well as text. This helps the administrator in viewing the availability of the resources in the cluster. Also, administrator can make decisions of distributing the jobs among nodes making assumptions of whether the job can be executed or will it fail.

## IV. DESIGN AND IMPLEMENTATION

The technology used to implement the system is Java. The cluster being used is Fedora Cluster. Fedora is a Linux- based operating system. Fedora has been used because of it being an open-source operating system along with being easily programmable. For data persistence, XML files have been used for storing information instead of a database. Technologies such as RMI and FTP have been used. FTP servers are made available on every node in the cluster. The master node is the RMI client and computational nodes are the RMI servers logically. In order to implement the proposed system, following is gist of operations that have been done:

- Authentication of the administrator happens using the details maintained in the XML files at the master node.
- Registration details of the jobs are maintained in XML files at the master node for future use.
- Input files are collected at the master node.
- Based on the efficiency analysis, distribution of the workload is calculated for the nodes.
- Based on the selected computational nodes, files are transferred to them from the master node using FTP.
- Master node (RMI client) then invokes method on the remote object located in the computational nodes for the job to be executed.
- Thus execution of the sub-tasks of on the selected nodes takes place.
- Provisions for Job cancellation, node shutdown, node restart etc. is made available to the administrator at the master node.
- During execution of sub-tasks, the computational nodes log the reporting details in text files.
- A daemon process dedicated to individual job reports to the master node the details of the job periodically. It reports an overall job as well as individual sub-task report.
- Reporting details are displayed graphically as well as textually to the administrator.
- Once the job is completed, all the processed files are transferred to the master node using FTP again so as to provide a unified result at the master node.
- Master node then merges and sorts all the processed files and presents the output.

Supporting applications with the system are:

- Compiling
- De-Compiling
- Sorting
- Web- Crawling

## V. EXPERIMENTAL RESULT

The process execution view shows selected node list on which a particular job is running. Each node is indexed using its IP address. For each node, the view provides information as number of input files given for execution, number of output files on which operation is completed and completion percentage. Figure 2 shows status of a job in execution on 2 computational nodes in a tabular fashion. Figure 3 shows status of the above said job using bar chart. In addition, administrator can view the status using a pie chart for better interpretation. The idea behind presenting information in various forms is to enable the administrator to visualize the situation precisely.

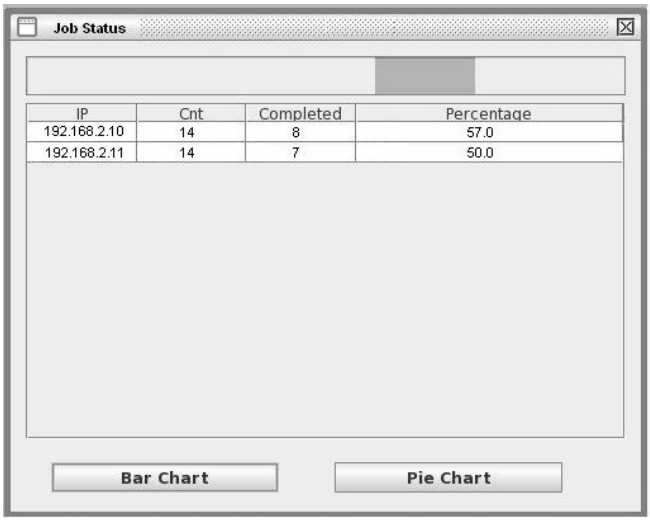


Figure 2: User process view

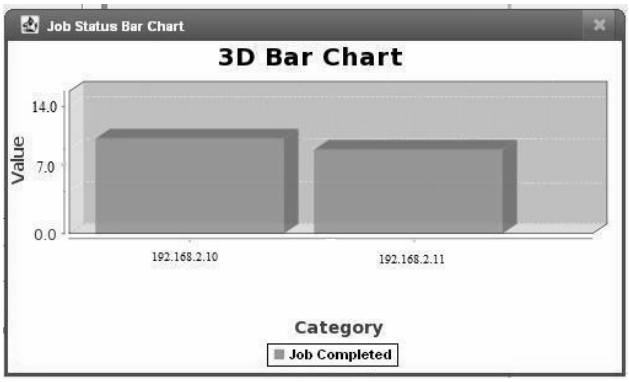


Figure 3: User process view (Graphical)

## VI. CONCLUSION AND FUTURE WORK

The system monitors the jobs being executed and the resources consumed by them. Also monitoring of the system resources available on each node like RAM, disk space, CPU, etc. is also achieved. Idle resources can be used effectively and thus the performance of the coupled computer cluster system increases. Time-critical jobs are monitored and tracked easily. Job failure can be detected and accordingly job re-distribution takes place. Load balancing is achieved according to the efficiency analysis of the computing nodes. The system is made fault-tolerant by applying job re-distribution policies.

Proposed architecture with further enhancement can be used to monitor multiple clusters simultaneously. External job schedulers like Maui scheduler can be used for better scheduling policies.

## VII. REFERENCES

[1] Ganglia http://www.ibm.com/developerworks/wikis/display/WikiPty pe/ganglia
[2] LVS Cluster Management http://kb.linuxvirtualserver.org/wiki/LVS_Cluster_Ma nagement
[3] MonALISA for cluster monitoring http://monalisa.cern.ch/blog/2010/11/24/monalisa-for-cluster-monitoring/,

[4] Augerat, P. Martin, C. Stein, B. Inst. Nat. Polytech. de Grenoble "Scalable monitoring and configuration tools for grids and clusters", Parallel, Distributed and Network-based Processing, 2002.
[5] Min Li, Yi- Sheng Zhang, "Job Monitoring Analysis Tool for HPC Clusters", International Conference on Information Science and Engineering 2009.
[6] P. Srinivas Rao, Ambika Prasad Mohanty, Dr. A. Govardhan, Dr. P. C. Rao, A Distributed Monitoring System for Jobs Processing, International Conference on Computer Design and Application 2010, May 29, 2010.
[7] Torque Documentation http://www.clusterresources.com/products/torque/docs/

## AUTHORS' PROFILES

Vijayalaxmi Cigala is a student in Department of Computer Engineering in MIT College of Engineering, Pune, India. She is pursuing her Bachelor's degree in Computer Science and will complete her engineering in May, 2011.

Dhirajkumar Mahale is a student of MIT College of Engineering, Pune, India. Presently he is final year student in a Department of Computer Engineering and going to complete his Bachelor's degree in May 2011.

Monil Shah is presently pursuing his Bachelor's degree in Computer Science in MIT College of Engineering, Pune, India. He is going to complete his engineering in May, 2011 and hopeful of career in Software Industry.