Job Management Systems Analysis

Emir Imamagić Branimir Radić Dobriša Dobrenić

Department of Computer Systems,
University Computing Centre, Croatia
{eimamagi, bradic, dobrisa}@srce.hr

Abstract

Job Management System (JMS) is a system responsible for management of user's jobs on computer cluster. In this paper, we describe common JMS architecture and functionalities that JMS has to implement. Furthermore, we describe in details features and our own experiences with following systems: Condor, Torque and SGE.

1. Introduction

The main goal of computer cluster is to create an illusion of large multiprocessor system. On such environment, users can execute parallel jobs or serial jobs that consist of large number of independent tasks. In either case, management of users' jobs is necessary. Thus, cluster provides the system that distributes tasks over cluster nodes and enables creation of cluster usage policies. This system is called Job Management System.

Job Management System (JMS) is a cluster component responsible for users' jobs control, their dispatching and scheduling. Main goals of JMS are to create interface for users' job requests, to achieve efficient resource utilization and to allow cluster owners to define cluster usage policies. Furthermore, JMS should gather usage statistics information, which may be later used to charge users for consumed time or analyze cluster utilization. JMS is also known as Resource Management System, Workload Manager, Batching System, Local and Distributed Resource Manager.

In this paper, we emphasize functionalities of JMS that are important for Grid systems. There is a difference between using cluster as a part of Grid and as individual system. When using cluster as a part of Grid, some functionalities are necessary. From the Grid perspective, some of the most important functionalities are advance reservation, standard API (e.g. DRMAA) and integration with existing Grid middleware (e.g. Globus Toolkit or UNICORE).

The remaining of the paper is organized as follows. In following section, we describe common architecture of JMS. In third section, we described set of functionalities that JMS has to implement and that we used as evaluation criteria. Fourth section contains evaluation results of three systems: Torque & Maui, SGE and Condor. In fifth section, we state our conclusions.

2. Job Management System Architecture

JMS usually consists of three components (Figure 1):

- Queue Manager
- Scheduler
- Resource Manager.

Queue Manager is responsible for queue management and interaction with users. Users submit, monitor and control their jobs through Queue Manager. Furthermore, Queue Manager records resource consumption and jobs' execution history.

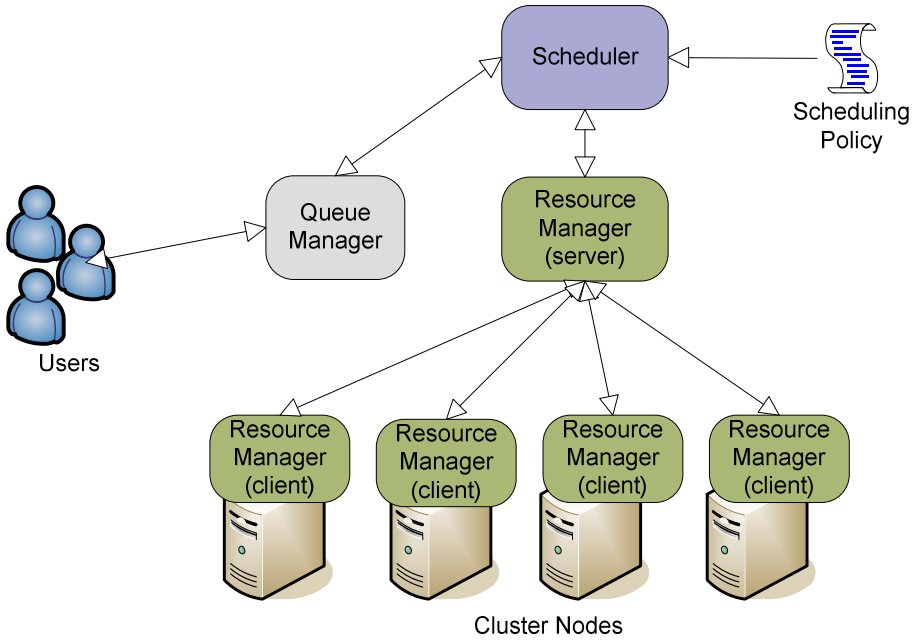

Figure 1 Job Management System Architecture

Scheduler selects and assigns nodes to jobs. Nodes selection is based on users' resource requests, nodes status and scheduling and usage policies. Administrator defines usage policies. For example, administrator can reserve nodes for group of users or limit resource (e.g. memory, CPU, disk space) consumption.

Resource Manager is responsible for resource monitoring and job execution. Resource Manager consists of server and client component. Client component is placed on every cluster node and it is responsible for execution of jobs and monitoring of jobs and nodes. Server component collects monitoring information from client components and dispatches users' jobs to client components. Furthermore, server provides Queue Manager with jobs' status information and Scheduler with nodes' status information.

3. Evaluation Criteria

In order to evaluate existing job management systems, we have defined set of criteria. Criteria are based on these similar studies: [1], [2], [3], [4], [5], [6] and [7]. Criteria are divided in 6 groups: general features, job support, queuing, scheduling, resource management and security. Within every group, we will emphasize features that are important for Grid. List of all criteria is presented in table 1. Functionalities that are important for Grid are written with bold letters.

Table 1 Job Management System evaluation criteria

Criteria group	Criteria
General	Platforms supported
	User interface • **API** (e.g. **DRMAA**) • GUI • CLI • Web Portal
	Open source vs. commercial
	Cluster distribution
	Installation method (install script, package)
	Support
	Documentation
Job support	Multiple job types: • batch • parallel • interactive • workflow • array of jobs
	Job description
	Parallel library integration (PVM, MPI)
Queue Management	Multiple queues
	Job control
	Prologue & epilogue scripts
	Job history & statistics
Scheduling	Standard scheduling algorithms
	User defined algorithm
	External Scheduler module
	Advance reservation
	Job Preemption
	Fair share
	Backfilling
Resource Management	Node configuration
	Process migration
	Checkpointing
	Dynamic load balancing
	Fault tolerance
	CPU harvesting
	File stage in/out
Security	Authentication

	Authorization: • job types • resources • queues
	Accounting
	Secure communication

3.1. General features

First set of criteria are general characteristics of JMS software. Most important characteristic is supported platforms. It is important that JMS can be deployed on as many platforms as possible. At least, support for various UNIX flavor OSs and commodity architectures (e.g. x86, AMD) is needed. Furthermore, JMS has to support clusters with heterogeneous nodes.

JMS should provide at least one of following user interfaces: graphical user interface (GUI), command line interface (CLI), web portal or application programming interface (API). From aspect of Grid systems, it is necessary that JMS provide some form of API. Preferred API is DRMAA (Distributed Resource Management Application API). DRMAA is API for submitting and controlling jobs developed by a working group in the Global Grid Forum (GGF).

JMS should provide some sort of installation method (such as installation script or program) and detailed administration and user's manual. Furthermore, at least, basic user support (such as mailing lists) is needed.

Finally, we will prefer open-source to commercial solutions. One of advantages of open-source system that we find important is possibility of making various changes to JMS services.

3.2. Job Support

JMS should support at least standard types of user's jobs: serial, interactive and parallel jobs. Support for more complex jobs is advantage. The most widely used complex job types are array of jobs or workflow.

JMS has to provide job description language for users to describe resources needed for their jobs (e.g. memory, cputime). Detailed jobs' descriptions are essential for efficient job scheduling.

To enable execution of parallel jobs, it has t be possible to integrate at least two widely used parallel execution libraries: MPI (Message Passing Interface) and PVM (Parallel Virtual Machine). Some JMSs provide interface for enabling integration of custom parallel execution library.

3.3. Queue Management

This set of criteria is related with Queue Manager. The key Queue Manager functionality is interaction with user. Thus, Queue Manager has to provide mechanisms for user to submit jobs, monitor job's execution and control jobs (e.g. hold or stop job's execution).

JMS should enable creation of multiple queues, where every queue is intended for specific type of jobs or user groups.

Queue Manager has to provide a mechanism for storing job's execution history and cluster usage statistics.

Last functionality from this group is the ability to define prologue and epilogue scripts. Prologue script is set of commands that are executed before every job is executed. Epilogue is script executed after the job. Administrators use these scripts to perform set of actions specific for their cluster (e.g. removing temporary files).

3.4. Scheduling

Scheduler component of JMS has to provide following functionalities: standard set of scheduling algorithms, usage of user defined scheduling algorithm, usage of external Scheduler module, job preemption and advance reservation.

Examples of standard scheduling algorithms are: First In First Out (FIFO), algorithms that favor jobs based on resource request (e.g. large jobs first, parallel jobs first), fair-share, backfilling. Fair-share is scheduling algorithm that tracks users' resource consumption history and limits the consumption in order to ensure fair cluster usage among users. Backfilling is scheduling algorithm that distributes short jobs on resources that are reserved for usage in future. Backfilling is very important for Schedulers that enable advanced reservation.

From the Grid perspective, the most important functionalities are advanced reservation and job preemption. Scheduler has to provide mechanism for making reservations of one or set of resources over extended period of time. By using advanced reservation, Grid system can reserve resources for Grid jobs on one or more clusters. Job preemption is process of suspending one job in favor of other.

3.5. Resource Management

Resource management is related to management of cluster resources and active jobs. JMS should enable the administrator to define policies for node usage. By using such policies, administrator should be able to define in which cases, which resources of node can be used. For example, administrator can allow JMS to exploit only limited amount of resource (e.g. memory or cputime) or in case when nodes are not dedicated administrator can define that node should not be used when keyboard is used.

Node usage configuration is necessary for enabling CPU harvesting. CPU harvesting is process of exploiting non-dedicated computers (e.g. desktop computers) when they are not used.

Related to active jobs, Resource Manager has to enable checkpointing, process migration, fault tolerance and dynamic load balancing.

Checkpointing is procedure of storing the state of active process on hard drive. Stored state of process is used to restart the process from that point.

Process migration is movement of jobs or processes from one node to another. Jobs are then restarted from the last available checkpoint or from the beginning.

Dynamic load balancing is process of balancing the load on nodes. Process migration is necessary for dynamic load balancing.

Resource Manager has to make possible job recovery in case of node failure. Fault recovery is usually done by process migration. In that case, checkpointed state of process has to be stored on special nodes.

The last functionality from this group is file stage in/out. File stage in/out is process of copying defined set of files on the node before the job is executed and copying set of files after the job is executed. This

capability is important for JMS that is used to control cluster without shared file system.

3.6. Security

Security requirements are authentication, authorization, accounting, and protection of communication between JMS services. JMS should provide its own authentication mechanisms and support existing solutions (e.g. Kerberos, GSI). Authorization should be performed for queues, resources and job types. Secure communication between JMS services is necessary in case when the nodes are placed in public network.

4. Job Management Systems Overview

We used criteria described above to evaluate numerous existing JMS solutions. Evaluated systems are Condor, CSS, LSF, Loadleveler, OpenPBS, Torque, PBSPro and SGE. Based on preliminary research we decided to implement and practically test following three: Condor, Torque with Maui and SGE.

In following part of this section, we describe features of JMS's that we have installed and our own experiences with those systems. All JMSs provide most of the features so we will outline only those that are not provided or those that are significantly good.

4.1. Torque & Maui

Torque [11] is based on OpenPBS [14] with improved scalability and node fault tolerance. Torque was installed together with Maui scheduling system.

Torque is open source JMS with support for wide set of systems (except Windows OS). Clusters with heterogeneous nodes are supported. It provides well manuals and mailing list. Torque does not provide install scripts but it is available as part of OSCAR and Rocks cluster distributions. User interfaces are CLI, GUI and C API. It is possible to find contributed API's for other programming languages (e.g. Python, Perl). GUI allows only job control functionalities and not administrator functions. We find that the GUI is not very functional, especially compared to SGE's GUI.

Torque supports batch, parallel and interactive jobs. It does not support submission array of jobs or workflow, although is possible to make basic synchronization between jobs. For example, it is possible to define that job has to wait one or set of existing jobs to finish before it starts with execution. Our experience shows that interactive jobs do not work smoothly. Instead of getting output and input of job, user gets direct access to node via rsh or ssh and then has to execute the application.

Good feature of Torque is that it can use external scheduling module. We used Torque with Maui scheduler. Maui scheduler provides advanced reservation, complex scheduling policies, fair share scheduling and various tools for managing and diagnosing cluster resources.

Main disadvantage of Torque are Resource Management features. Torque does not provide CPU harvesting, advanced node configuration (e.g. definition of custom sensors), dynamic load balancing and process migration. File stage in/out is checkpointing with external libraries is supported.

Torque provides authentication and authorization mechanisms. Integration with other security systems is not provided. Maui provides accounting mechanism.

4.2. SGE

SGE [10] is a product of Sun Microsystems Company. SGE is open source JMS that supports same set of platforms as Torque.

User interfaces and manuals are much better then Torque's or Condor's. Especially well developed is GUI that enables complete management of cluster. In addition, SGE provides scripts for automatic installation and it is available as part of Rocks cluster distribution.

SGE supports standard job types and array of jobs. Workflows are not supported. It provides modules for integration with MPI and PVM parallel libraries, but our experience shows that those do not work smoothly. Module for integration with MPI was unable to clean all processes in case of failure.

Queue Management and scheduling capabilities are one of SGE's weaknesses. Creation of multiple queues is not explicitly possible. Job statistics feature is SGE does not provide. SGE does not support external Scheduler module (although support for Maui is being implemented). Scheduler component provides set of standard scheduling algorithms (FIFO, fair-share) but custom algorithms, advance reservation, backfilling and job preemption are not supported.

Resource management functionalities are one of the SGE's advantages. SGE provides support for job migration, load balancing and fault tolerance. Furthermore, SGE enables advance node configuration and definition of custom sensors. By using custom sensors (e.g. sensors for keyboard usage) SGE can be used for CPU harvesting. Resource Manager only lacks support for file stage in/out.

SGE provide mechanisms for authentication and authorization. Accounting is implemented as commercial module that has to be purchased independently. SGE does not enable integration with other accounting mechanisms.

We analyzed here SGE version 5.3. It is important to emphasize that next version (6) will add most of the functionalities that are missing. Some of the most important added features are: job preemption, advance reservations, creation of multiple queues, file stage in/out and DRMAA interface.

4.3. Condor

Condor [8] is an open source project of University of Wisconsin. Condor is profoundly different from two other JMS's. Condor is designed specifically for High Throughput Computing and CPU harvesting.

Condor provides installation scripts, and CLI and API interface. For monitoring of nodes and jobs, Hawkeye can be used. Hawkeye provides web portal interface. Next versions of Condor will provide DRMAA interface.

Condor is used mainly for serial jobs but it provides limited support for parallel PVM and MPI jobs. Language ClassAds is provided for complex description of jobs. Same language is used for description of nodes. Condor also supports array of jobs and complex workflows. There is a special system – Condor DAG for workflow jobs.

All queue management functionalities are provided except multiple queues. For detailed jobs' statistics is necessary to install additional software – Condor View.

Condor uses matchmaking mechanism for job scheduling. Nodes contact central server and ask for jobs. Matchmaking

service compares job requests with nodes' requests and assigns jobs to nodes. Explicit definition of custom scheduling algorithm is not possible. However, it is possible to configure nodes to favor specific type of jobs. Furthermore, usage of external Scheduler module and backfilling and fair share algorithms are not enabled. Job preemption and advanced scheduling functionalities are provided.

Resource Manager enables advanced node configuration, checkpointing, process migration, fault tolerance and file stage in/out. Checkpointing is possible for serial jobs only and application needs to be re-linked with Condor libraries. Condor allows creation of special server for storing checkpoints. In case of node failure checkpoint is fetched from checkpoint server and job continues execution on available node. Additionally, Condor enables sending jobs from one Condor cluster to another.

Condor provides authentication and authorization mechanisms. It is possible to integrate Condor with Kerberos and GSI security systems. Condor also enables secure communication between services.

5. Conclusion

In this paper, we outline main functionalities that Job Management System has to provide. We divided functionalities in 6 groups and used them to evaluate three widely used JMSs: Torque with Maui, SGE and Condor. The main motivation of this review is to find JMS that is most appropriate for use in Grid systems. Therefore, we emphasize the JMS's functionalities that are important to Grid systems.

Analysis shows that all three systems satisfy most of the criteria. However, every system has some advantages and disadvantages. Condor is ideal for serial jobs, especially because of checkpointing, process migration and fault tolerance. Torque with Maui works well with parallel jobs and enables advanced scheduling options. Current SGE version has strong support for node configuration and well GUI but weak integration with parallel libraries and lack of multiple queues are disadvantage.

In conclusion, we find that Torque with Maui is currently the best for Grid integration. Main basis for this conclusion is advance reservation capability and integration with most existing Grid systems (e.g. Globus Toolkit, UNICORE). We also recommend SGE version 6, especially because of DRMAA support and advance reservation capability. Condor is recommended in case when Grid is used for HTC applications (e.g. parameter sweep) or workflows. Furthermore, Condor enables mechanisms for Grid scheduling: Condor/G and glide-in.

6. References

[1] J. A. Kaplan, M. L. Nelson: **«A Comparision of Queueing, Cluster and Distributed Computing Systems»**, June 1994.

[2] M. Baker, G. Fox, H.W. Yau: **«Cluster Computing Review»**, NPAC Technical Report SCCS-748, Northeast Parallel Architectures Center, Syracuse University, November 1995.

[3] J. P. Jones: **«NAS Requirements Checklist for Job Queuing/Scheduling Software»**, NAS Technical Report NAS-96-003, April 1996.

[4] C. Byun, C. Duncan, S. Burks: **«A Comparision of Job Management Systems in Supporting HPC**

ClusterTools», Proc. SUPerG, Vancouver, Fall 2000.

[5] O. Hassaine: **«Issues in Selecting a Job Management Systems (JMS)»**, Proc SUPerG, Tokyo, April 2001.

[6] T. El-Ghazawi, K. Gaj, N. Alexandridis, F. Vroman, N. Nguyen, J. R. Radzikowski, P. Samipagdi, S. A. Suboh: **«A Performance Study of Job Managements Systems»**

[7] T. El-Ghazawi, K. Gaj, N. Alexandridis, B. Schott, A. V. Staicu, J. R. Radzikowski, N. Nguyen, S. A. Suboh: **«Conceptual Comparative Study of Job Management Systems»**, Technical Report, February 2001

[8] Condor, http://www.cs.wisc.edu/condor/

[9] LSF, http://www.platform.com

[10] SGE, http://gridengine.sunsource.net/

[11] Torque, http://supercluster.org/torque

[12] IBM Loadleveler, http://publib.boulder.ibm.com/clresctr/

[13] PBSPro, http://www.pbspro.com/

[14] OpenPBS, http://www.openpbs.org