NAS Requirements Checklist for Job Queuing/Scheduling Software

James Patton Jones[1]

NAS Technical Report NAS-96-003 April 96

jjones@nas.nasa.gov
NAS High Performance Processing Group
NASA Ames Research Center
Mail Stop 258-6
Moffett Field, CA 94035-1000

Abstract

The increasing reliability of parallel systems and clusters of computers has resulted in these systems becoming more attractive for true production workloads. Today, the primary obstacle to production use of clusters of computers is the lack of a functional and robust Job Management System for parallel applications. This document provides a checklist of NAS requirements for job queuing and scheduling in order to make most efficient use of parallel systems and clusters for parallel applications. Future requirements are also identified to assist software vendors with design planning.

1.0 Introduction

The Numerical Aerodynamic Simulation (NAS) supercomputer facility, located at NASA Ames Research Center, has, for the last few years, been working to bring parallel systems and clusters of workstations into a true production

1. MRJ, Inc., NASA Contract NAS 2-14303, Moffett Field, CA 94035-1000

environment. One of the primary difficulties has been identifying a robust Job Management System (JMS) capable of handling parallel jobs. For a complete discussion of the role and need of a JMS, see [Sap95a].

The purpose of this document is to supplement the above referenced JMS paper, by providing a checklist of the current requirements for the job queuing, scheduling, and resource management software components of a JMS in order to efficiently run parallel jobs on parallel computers and clusters of workstations. This document will be used as the basis for an evaluation of potential job management systems for NAS and the AHPC project, NASA Cooperative Agreement NCC3-413. (For a complete description of the cooperative agreement see [CAN95].)

This paper is divided into two sections: (1) initial definitions and (2) actual requirements. The list of requirements is divided into three main categories: absolute requirements, recommended capabilities, and future requirements. Each is rated as high, medium, or low priority. The priorities indicated are those assigned by the AHPC project team members.

2.0 Definitions

A *job management system* (JMS) generally fills three roles, often as separate modules: Queuing, Scheduling, and Resource Management. (For reasons for this trichotomy and a detailed analysis of Job Management needs in general, see [Sap95a].)

The *Queuing* role has traditionally been filled with batch systems, such as NQS [Kin86]. The *Scheduling* role is the process of selecting which jobs to run, according to a predetermined policy. *Resource management* refers to the monitoring, tracking and reservation of system resources; and enforcement of usage policy.

For the purposes of this discussion, a *job* is a sequence of operations requested by a user, of which a parallel application is the main part. The terms “job” and “application” may be used interchangeably. A job may consist of processes on one or more processors. There are several terms describing jobs that need to be clarified for use in this document:

> A *serial job* can be thought of as a special case of the above job definition where there is only one process, instead of multiple, parallel processes.
>
> A *batch job* is submitted through a JMS and run when the requested resources are available.
>
> An *interactive job* is generally the opposite of a batch job in that a user is

implicitly permitted to log directly into a computer and run jobs without going through a JMS. For example, a researcher who runs a job on the workstation on her desk.

An *interactive-batch job* is a special type of batch job in which the job is submitted to a JMS, but when the job runs, the user is given access to and control of the job's input, output, and error file streams (stdin, stdout, and stderr, respectively), allowing the user to interact with the job.

A *foreign job* is an interactive job running on a compute resource to which it has not been granted access. For example, when a user runs a job on a dedicated compute resource without going through a required JMS.

Also for the purposes of this paper, the term *node* refers to a single computer that contains specific resources, such as memory, network interfaces, one or more CPUs, etc. For definitions of other terms relating to Queuing and Job Management Systems in general, see [Hen95].

3.0 Requirements

This paper focuses on the requirements for Queuing and Scheduling. However, since these two elements are tightly integrated with the Resource Management, requirements are given for the Resource Manager as well.

3.1 Job Management System

The complete JMS has several overall requirements, including:

High Priority

3.1.1 Must operate in a heterogeneous multi-computer environment. The JMS must be capable of managing any number of machines, in any number of combinations: including shared-memory multi-processor computers, parallel machines, and both tightly and loosely coupled clusters of workstations.

3.1.2 Must be capable of integrating with frequently used distributed file systems, to include NFS, AFS, and DFS. Specifically, the JMS must be able to execute in the DCE/DFS environment utilizing the DCE/DFS ACL features for file permissions.

3.1.3 Must possess a command line interface to all modules of the JMS.

3.1.4 Must include a published API to every component of the JMS (i.e. resource manager, scheduler, and queuing subsystems) that allows

local tools and utilities to be written to interface with the JMS components.

3.1.5 Must be able to enforce resource allocations and limits on:

- Number of CPUs per job
- Number of nodes per job
- Type of nodes per job
- Number of jobs executing per user
- Number of jobs executing per group
- Wall clock time
- CPU time (per node and per application)
- System time
- Memory utilization
- Disk usage
- Swap space
- Dedicated access
- Shared access
- Network adapter access

3.1.6 Software must permit multiple instances and versions to exist and run simultaneously on the same systems. This is needed, in part, for testing new releases before production use.

3.1.7 Source code must be available for complete JMS. This is primarily needed for bug-fixes.

3.1.8 Must supply the ability to define more than one user id as the manager of the software. These ids, ideally, would not need overall root privileges on the machine running the JMS software. It is also desirable to be able to define JMS operator ids as well. The operator ids would have a subset of the manager's privileges which would include, but not be limited to starting and stopping queues, suspending, moving, restarting, and killing jobs.

Medium Priority

3.1.9 Must provide a means of user identification outside the password file. This would provide a much more convenient way of maintaining authorized users for cluster configurations.

3.1.10 Must be scalable. Specifically the JMS must be capable of:

- Managing very large clusters (> 500 nodes)
- Allowing very large parallel jobs (> 200 nodes)

3.1.11 Must meet all requirements of appropriate standards, including:

- POSIX 1003.2d “Batch Queuing Extensions for Portable Operating Systems”.

3.2 Resource Manager Requirements

High Priority

3.2.1 Must be “parallel aware,” i.e. understand the concept of a parallel job and maintain complete control over that job. This capability requires:

- Tracking all processes (and sub-processes) of the job.
- Being able to kill any job completely, including sub-processes, without leaving orphaned processes. This implies that the JMS must be aware of distributed processes and capable of forwarding signals.
- Being able to “clean up” after jobs, i.e. provide node condition equivalent to the state before the given job existed.
- Collecting complete job accounting information for all processes of a job, which must be combined to provide an aggregate job accounting record, in addition to per-node totals. The job accounting record must indicate total usage of all resources allocated to the job, and which limits, if any, were exceeded. See also [Sap95b].
- Providing a mechanism (i.e. a programming interface) which allows a parallel program to communicate with the JMS to coordinate resource usage and to start processes.

3.2.2 Must be able to support and interact (i.e. coordinate resource allocations) with the following:

- MPI
- PVM
- HPF

3.2.3 Must provide file “stage-in” and “stage-out” capabilities that allows the user to identify files that should be transferred to and from appropriate locations on the computer system on which his/her job will be run. Stage-in needs to occur before the actual job starts, but after disk resources have been allocated for that job. Stage-out should follow termination of the job, but before the disk resources are released for re-allocation.

3.2.4 User-level checkpointing/restart (AHPC project completion date: March 31st1996):

- Allowing application to periodically checkpoint its state without system support.
- JMS should have a well-defined interface to facilitate checkpoint and restart.
- JMS default should be able to checkpoint (if possible) when stopping a job or if the JMS goes down; and to restart the job (from the checkpoint if available, otherwise from the beginning of job). The user should be able to override this default if job restart is not wanted, in the event that the job is stopped.

Medium Priority

3.2.5 Must provide a history log of all jobs, to include:

- Time job entered batch system
- Time job entered (each) queue
- Time job started execution
- Time job suspended execution
- Time job restarted execution
- Time job terminated
- Exit status of job
- Total usage and identification of each resource allocated to job (as specified in 3.1.5).

3.2.6 Asynchronous communication between application and Job Manager via a published API:

- To request specific resource (number of nodes, amount of time, etc., as specified in 3.1.5)
- Acquire resources through non-blocking request with asynchronous notification of resource availability
- Specify if and when an application can release resources
- Provide for Job Manager preemptive and cooperative resource re-acquisition for reallocation

3.2.7 Must be integrated with authentication/security system. This includes providing:

- Well documented interface with security/authentication system
- Site configurable authentication mechanism
- Necessary hooks for site to interface JMS with local environment
- Out-of-the-box support for common and standard authentication systems, including DCE.

3.2.8 Interactive-batch jobs must run with standard input, output, and error file streams connected to a terminal.

3.3 Scheduler Requirements

High Priority

3.3.1 Must be highly configurable, supporting:

- Complex scheduling, allowing different scheduling policies at different times of the day, and distinction between prime and non-prime time
- Dynamic and preemptive resource allocation (reshuffling queue, tiling, etc.)
- Awareness/distinction between batch, interactive, interactive-batch, and foreign jobs

3.3.2 Must provide simple, out-of-the-box scheduling policies, including:

- First in, first out (FIFO)
- Shortest job first
- Favor large memory (or CPU) jobs, or small jobs
- Favor long running jobs, or short jobs
- Load balancing (time-shared systems)
- User or group priority
- Fair sharing (past usage consumption)

3.3.3 Must schedule multiple resources simultaneously, including at least the following:

- Number of nodes
- Type of nodes (compute, I/O, big memory, multiprocessor)
- Number of processors per node
- Memory per node
- Network connections (Ethernet, HiPPi, FDDI, ATM, etc.)
- Disk (local, system, scratch, temporary, fast, etc.)
- System specific resources (e.g. switch adapter mode on SP2)
- Operating system version

3.3.4 Must be able to change the priority, privileges, run order, and resource limits of all jobs, regardless of the job state.

3.3.5 Coordinated scheduling is absolutely critical for almost all message passing jobs as there is severe performance degradation when resources are simultaneously used by different jobs. Space-sharing (or

tiling, allocating nodes or Processing Elements as dedicated resources to support non-overlapping jobs) is the only effective way to accomplish coordinated scheduling in the absence of gang-scheduling (synchronized time-sharing, see also 5.1.1).

Medium Priority

3.3.6 Must provide mechanism to implement any arbitrary policy. Policy expressed by a simple set of rules is generally not sufficient, as it does not allow for complete flexibility within a given site. This requires:

- Scheduler must be separable from JMS. A site needs the ability to both modify and replace the scheduler.
- A published API must be available to the system administrator to implement his/her specific site scheduling policy. A parsed configuration file alone is not sufficient.

3.3.7 Must support unsynchronized timesharing of jobs. Unsynchronized time-sharing (time-sharing with no guarantee of synchronization across nodes) can be used on interactive nodes where the performance degradation from the unsynchronized time-sharing across nodes is not as important, and for general interactive debugging.

3.3.8 Sites need to be able to define which, if any, nodes are to be time-shared as well as the number of processes and users per time-shared node. There may be limitations in the number of applications that can simultaneously use the network adapter, or a given node may have a small amount of memory or swap space.

3.4 Queuing System Requirements

High Priority

3.4.1 Must handle two job types with a common set of commands:

- Interactive (stdin, stdout, and stderr connected to the terminal session)
- Batch (stdout and stderr directed to files)

3.4.2 User Interface must provide information on at least the following (AHPC project completion date is March 31st 1997):

- Unique identifier for each job
- User id job executing under
- Group id job executing under

- Job state (including running, queued, suspended, held, exiting)
- Job priority
- Why a given job is not running
- Information about the consumed and remaining resources available to a job
- List of allocated or requested resources for each job (as specified in 3.1.7)
- Status of all system resources (idle, reserved, available, down, allocated)

3.4.3 Must provide for restricting access to the batch system in a variety of site-configurable methods, to include:

- specific user restrictions
- specific group restrictions
- restrictions based on past resource consumption
- restrictions based on per user or group current resources in use
- origin of job

3.4.4 Must be able to sustain hardware or system failure, i.e. no jobs get lost; restart, rerun, or checkpoint interrupted jobs.

3.4.5 Must be able to configure and manage one or more queues.

3.4.6 Administrator must be able to create, delete, and modify resources and resource types.

3.4.7 Administrator must be able to change a job's state (queued, running, suspended, held, etc.)

3.4.8 Must allow dynamic system reconfiguration by administrator with minimal impact on running jobs. Administrator needs to be able to selectively remove or add nodes to the cluster without impacting overall access to JMS functions or to the remaining nodes. Any jobs which were running on the removed nodes must, at least, be suspended and started up again once those nodes are turned back over for general use. Ideally, the jobs which were running on the affected nodes would be checkpointed, moved, and continue running on available nodes not affected by the reconfiguration.

3.4.9 Must provide centralized administration. Log files and administration commands must be centrally located.

3.4.10 Users must be able to reliably kill their own job. See 3.2.1.

Medium Priority

3.4.11 Must provide administrator configurable scripts/programs to be run by JMS before and after a job, respectively. This may be used for initialization or node clean-up.

3.4.12 Must include user specifiable job inter-dependency based on:

- Job state (see 3.4.2)
- Job return status (success, failure)
- Job submission time (e.g. "run my jobs in the order I submitted them")
- Job start time (e.g. "don't run my job before noon on Tuesday")
- Status of other computer systems (e.g. mass storage)

3.4.13 Must allow jobs to be submitted from one cluster and run on another. (AHPC project completion date is June 30th 1996).

3.4.14 Must provide a site-configurable mechanism (at both the user and group levels) to permit users to have access to information about jobs from other submitters.

4.0 Requested Capabilities

The following capabilities would be extremely useful but are not absolutely critical. The timeframe in which these capabilities should be provided is six to eighteen months.

High Priority

4.1.1 Job scheduler should support dynamic policy changes (from any computable scheduling policy to any other) without restarting the batch system.

4.1.2 Possess a Graphical User Interface (GUI) to all modules of the JMS.

4.1.3 Provide a graphical representation of the configuration and usage of the resources under the JMS. There should also be an option to view other clusters within the same graphical display, instead of opening up multiple displays from each defined cluster.

Medium Priority

4.1.4 The time-sharing configuration information should be available to the

job scheduler for optimizing job scheduling (i.e. which nodes and jobs are time-shared, if any, and for how long resources have been committed to time-sharing).

4.1.5 Provide a graphical monitoring tool with the following capabilities:

- View history of host load
- Be able to adjust the sample time
- Be able to store data to separate output file
- Be able to capture a snapshot of the graphical representation in postscript, TIFF, and GIF formats.

4.1.6 Should be able to support both hard and soft limits when appropriate.

- Each supported resource should have a corresponding hard limit. Jobs exceeding a hard limit should typically be killed, suspended, held, or rejected, but this should be a function of the job manager, and site configurable.
- Each supported resource should also have a corresponding soft limit. Jobs exceeding a soft limit should be notified and allowed to continue until the hard limit is reached.

4.1.7 Should be readily available. The marketplace must be able to support the continued development and support of the product. This can be a defacto standard public domain package with a “marketplace” that supports it or a commercially supported product with the appropriate target market.

4.1.8 Should supply some kind of a proxy account optional setup. If it is deemed necessary that certain machines be available for “open” use, configuration would be made much easier if the JMS had a few ids “owned” by the software that are available for use to any user id defined by a group, subnet, etc. This would make the JMS much more accessible without the overhead of unnecessary user ids across multiple systems.

4.1.9 Should provide at least the following accounting capabilities:

- Recorded in flat ascii files to make UNIX command processing of the data easier
- GUI interface to control data collection

and at least the following datapoints, per node and per cluster:

- Usage of each resource defined to JMS (as specified in 3.1.5)
- Fraction of time JMS was available

- Total and percent of available CPU time used
- Number of logins and users
- Load average of nodes
- Number of batch jobs

Low Priority

4.1.10 Must allow a site to choose to run separate resource managers for each system (or cluster), as well as a single resource manager for all systems. A single resource manager for an entire site would allow a single entry point to which any job could be submitted, and then routed to the correct system (or cluster) at the site.

4.1.11 Interactive jobs allow user to "detach" from the job, requiring that output be logged to a file as well. See [DJM93].

4.1.12 Provide a mechanism to allow reservations of any resource (for example, a capability similar to Session Reservable File System (SRFS)).

4.1.13 Should provide at least the following attributes for jobs:

- Set of resource consumption counters
- Set of resource limits

4.1.14 Should be able to define and modify a separate access control list for each supported resource (as specified in 3.1.5).

4.1.15 Should provide wide area support allowing clusters separated by large distances with relatively slow (> 56Kb/sec) network connections to share resources. (AHPC project completion date is June 30th 1997)

4.1.16 Should allow an interactive user on a workstation console to instruct the JMS to suspend or migrate a job to a different workstation.

4.1.17 Should provide both client and server capabilities for Windows NT.

5.0 Future Requirements

The following capabilities are recognized as being difficult to implement, and will be required in the future. They are listed here to assist vendors in design and feature planning.

High Priority

5.1.1 Gang-scheduling (AHPC project completion date of June 30th 1997): fully synchronized time-sharing of parallel processes across distributed nodes. This feature is critical to statically balanced application and tightly coupled parallel applications, where resonance in communication delays from time-sharing delays may significantly degrade performance. Gang-scheduling or co-scheduling should be implemented on:

- Fixed size partitions
- Variable sized or dynamic partitions

5.1.2 Dynamic load balancing (AHPC project completion date of December 31st 1996), including the ability to:

- Change resource allocation dynamically
- Migrate a running application to other nodes
- Reduce/increase nodes as availability/priority changes
- Fault tolerance

5.1.3 Job migration: Ability to suspend a job or part of a job, and move its full computing environment (binary, local files, etc.) to a different node, or set of nodes, of the same architecture. Information about the best migration point could be given by the user to simplify the migration process. A published API should be available for communication between a job and the JMS to provide the system with necessary information on how and/or when to best suspend or restart the job. (AHPC project completion date of December 31st 1996)

Medium Priority

5.1.4 OS level checkpointing, providing the ability for the JMS to restart a job from where it left off and not simply from the beginning.

- Needed for true fault tolerance
- Needed for true dynamic resource allocation

6.0 Acknowledgments

The requirements within this paper are the results of iterations of discussions with the NAS Parallel Systems group and project members of NASA Cooperative Agreement NCC3-413, including representatives from NASA Ames, NASA Langley, NASA Lewis, United Technologies Pratt & Whitney group, CFD Research Center, MacNeal Schwendler, Corp., Massachusetts Institute of Tech-

nology, and the State University of New York [CAN95]. Comments were also received from the Platform Computing, NAS PBS group, Cray Research, IBM, and SGI regarding requirements of their customers for JMS software.

7.0 References

[CAN95] NASA Cooperative Agreement NCC-413.
URL: http://www.lerc.nasa.gov/Other_Groups/NPSS/html/can95.html

[DJM93] "Distributed Job Manager Administration Guide," AHPCRC, Minnesota Supercomputer Center, 1993.

[Hen95] "Portable Batch System: Requirements Specification," Robert Henderson and Dave Tweten, Numerical Aerodynamic Simulation Facility, NASA Ames Research Center, April 1995.

[Kin86] "The Network Queuing System," B.A. Kinsbury, *Cosmic Software*, NASA Ames Research Center, 1986.

[Sap95a] "Job Management Requirements for NAS Parallel Systems and Clusters", William Saphir, Leigh Ann Tanner, and Bernard Traversat, NAS Technical Report NAS-95-006, Numerical Aerodynamic Simulation Facility, NASA Ames Research Center, February 1995.

[Sap95b] "JSD: Parallel Job Accounting on the IBM SP2," William Saphir and James Patton Jones, NAS Technical Report NAS-95-016, Numerical Aerodynamic Simulation Facility, NASA Ames Research Center, July 1995.

<table>
<tr><td></td><td>NAS TECHNICAL REPORT
Title: NAS Requirements Checklist for Job Queuing/Scheduling Software</td></tr>
<tr><td></td><td>Author(s): James Patton Jones</td></tr>
<tr><td></td><td>Reviewers:
“I have carefully and thoroughly reviewed this technical report. I have worked with the author(s) to ensure clarity of presentation and technical accuracy. I take personal responsibility for the quality of this document.”</td></tr>
<tr><td>Two reviewers must sign.</td><td>Signed: _____
Name: _____
Signed: _____
Name: _____</td></tr>
<tr><td>After approval, assign NAS Report number.</td><td>Branch Chief:
Approved: _____</td></tr>
<tr><td>Date:</td><td>NAS ReportNumber:</td></tr>
</table>