Application Oriented Management Services in Time -Shared Clusters

José Manuel Pereira

Departamento de Informática, Universidade do Minho
4710 - 057 Braga, Portugal
josemanuelapereira@hotmail.com

Abstract. The need for cluster management software is increasing as cluster computing becomes more common. This communication presents and briefly explains some services that applications usually require, and therefore should be provided in cluster software management packages; it also looks into some cluster software management packages that provide those services. An application-oriented method to select a cluster software management package is presented, and its drawbacks are analysed.

1 Introduction

Analysing cluster management services for applications implies that some terms are introduced and briefly explained.

Pfister [1] defines a cluster as "a parallel or distributed system that consists of a collection of interconnected whole computers, that is used as a single unified resource". These "whole computers" are usually called nodes, which are complete computers with CPU, RAM and disk, such as a standard PC. A homogeneous cluster could be defined as a cluster that uses compatible nodes, e.g. compatible hardware and software components in each node.

A time shared cluster could be defined as a cluster running multiple applications (jobs) over time, without any of them knowing of each other and each application assuming the cluster is available for itself. The cluster performance varies accordingly to jobs workload.

A cluster is usually intended to achieve three goals:

- *Single system image* the ability to use the cluster as a single, unified computer. The goal is to share the overall workload among the cluster nodes automatically without applications or users knowing about it.
- *Fault tolerance* is the ability of the cluster to recover from the failure of one of its nodes. Ideally the node failure should be unknown to applications, the node workload migrated and resumed in another cluster node.
- *Scalability* is the ability to increase/decrease the cluster performance by simply adding/removing nodes from the cluster. Linear scalability is achieved if the cluster computing power grows/decrease in linear proportion to the number of nodes added to/removed from the cluster.

Cluster management software tries to implement these goals through software services usually available as add-on packages on common mass-market operating systems. Each package is looked into in search of these services and their implications to applications. This communication addresses cluster management services available for two recent operating systems families: Linux OS packages and Windows NT cluster management software.

These services can generally be defined as:

- Cluster management services: generally each node runs its own copy of the operating system. For some operations, applications want to view the cluster as a single com-

puter, e.g., single system image to execute system wide operations like cluster wide command execution, file distribution and gathering, remote shutdown and restart, process control and system images updates.

- Workload management: when applications are started jobs are created and must be run on the cluster. A typical Workload management system tries to ensure that every process gets their fair share of cluster execution and also tries to use resources efficiently. Workload management has three basic components:
- Resource management, Job management and Job scheduler: once a job is queued, workload manager is responsible for monitoring the state of the cluster , handling jobs that start and finish and handling data input/output on nodes. The job scheduler is responsible for scaling a job to run on a cluster node. The node resources are managed by the node local operating system.
- Programming environments: clusters are usually built to run software written by their users and usually some kind of parallel programming is needed. There are several ways to write software for clusters but message passing approach is the most common message passing interface (MPI) [2] and parallel Virtual Machine (PVM) [3] are the more common message passing software libraries and therefore are searched for.
- Nodes interconnections: which hardware devices and software protocols are supported and its implications in overall cluster performance are analysed.
- Single point of failure: true if the cluster has a centralised component, which, in case of failure, halts the cluster. Also compromises scalability.
- Web interface: the ability to use the cluster via Web to enable remote administration and job submission.

2 Cluster Management Services

The packages there were considered for analysis based on available www information and were divided into 2 groups: Linux based clusters and Windows NT clusters.

In the Linux group were considered: Oscar, Scyld, MSC-Linux, Npaci Rocks, SCoreD and MOSIX

	Cluster Management Software	**Workload Management**	**Nº FrontEnd Nodes**	**Supported Nodes Interconnections**	**Node Operating System**
Oscar	C3 – 2.6	PBS + PBS standard scheduler. Maui scheduler soon.	1	- Gigaethernet - Myrinet	Linux Red-Hat 6.2
Scyld	C3	PBS – PRO + PBS scheduler	1	- Gigaethernet - Myrinet	Linux RedHat with Kernel 2.2
Msc-Linux	C3	PBS	1	- Gigaethernet - Myrinet	Linux RedHat
NPaci – Rocks	C3 2.7	PBS + Maui Scheduler	1	- Gigaethernet - Myrinet	Linux RedHat 7.0 + Updates

As seem in this table some packages use the same service components and so the services provided to applications will greatly rely on the behaviour of these service components. We will try to show how they work and what implications they have to serviceability on applications. About parallel programming environments, all support MPI and PVM.

Monitoring & Management of Multiple Clusters (M3C). M3C is a project for developing user interfaces for assistance in the management of PC clusters. M3C tool is a web based interface application that enables remote monitoring of several clusters at the same

time. It also provides a web-based interface for users remote execution of tasks. M3C operation interfaces with C3 tools for issuing user commands to the cluster.[1]

Cluster Command and Control (C3). C3 is a tool to provide single system image to the execution of commands. One possible implementation would be, for example for file operations, to mount a file system on the server, which would include all cluster nodes. The centralised approach simplifies the cluster operation but derives the least scalable cluster due to server-centralised workload of operations as well as network traffic congestion due to nodes to server communication.

C3 takes the approach to decentralise the execution of commands. Each cluster runs its own copy of the operating system and C3 tools move data in/out of cluster nodes transparently.

1. Send command via rsh/ssh.
2. Execute command on node.
3. Output return via rsh/ssh

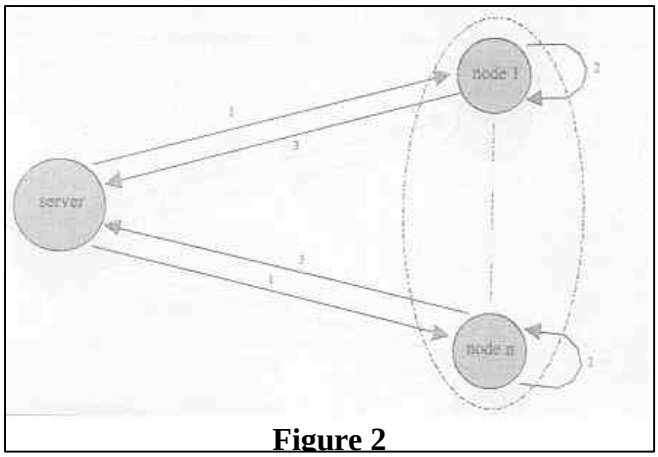

Figure 2

The command is sent to all cluster nodes, executed in parallel and its results returned to the server. Although this algorithm grants parallel execution of commands in cluster nodes it also has limitations due to the network traffic increase that is proportionality to the number of nodes. Server overhead grows in management of cluster nodes since server creates a process for each cluster node and within each process opens a network connection to a cluster node. It is predictable that congestion hits both the server and the network as scalability grows. According to Brim [4], C3 performs well on 64 cluster nodes but starts to reveal excessive congestion over 256 nodes.

Future research is been made on various algorithms for distribution of the server load among all cluster nodes, as well as trying to reduce network traffic.

Job Scheduling in PBS. PBS is a batch queue and workload management. It operates networked, multi platform UNIX environments including homogeneous and heterogeneous workstations, supercomputers and massively parallel computers. Key features are compatibility with Unix standards, configurable job scheduler, programmable scheduling policies, dynamic workload distribution and support for batch and interactive jobs [5]. Figure 3 describes the scheduling algorithm of PBS [6]:

1- Application sends a job to a server called an event.
2- Server contacts scheduler and sends reason for contact. Scheduler becomes a privileged client process of the server.
3- Scheduler requests cluster state info to cluster nodes; in figure 4 for each job each cluster node is contacted to gather information. This can compromise scalability in a larger cluster. The cluster monitor is a programmable module, which can change this behaviour.

[1] M3C project is available at http://www.csm.ornl.gov//torc/M3C

4- Cluster monitor returns cluster nodes (MOM) info.
5- Scheduler requests job info from server.
6- Server returns job info to scheduler. Scheduler runs *policy* algorithm to calculate the node to run the job. Pbs has a number of included *policy* algorithms such as generic purpose; specific purposes and it can be programmed with other scheduling policies.[2]
7- Scheduler sends id of cluster node that will run job to server.
8- Server sends job directly to node.

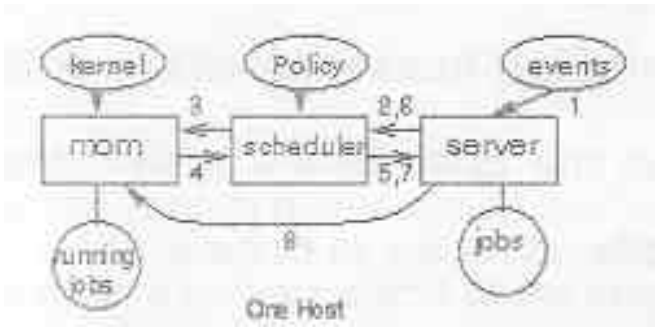

Figure 3 [6]

Although PBS grants flexibility in configuring the scheduling policy it also implies that for the application point of view this means that the dynamic workload distribution could rely on application programmers diverging them from the original task of application building. The *Maui* scheduler promises to have a scientific solution to workload distribution.[3]

The Maui Scheduler. Fig.5 shows Maui scheduler components that run in it own process, iterating repetitively 3 tasks.

First the scheduler pools the resource manager, second it attempts to schedule jobs in the queue and finally manages existing reservations. The scheduling routine takes a queue of prioritised jobs and tries to make an immediate reservation for the highest priority job and start it, repeating until running out resources. Then starting at the first job not scheduled the process of backfilling starts. Backfilling refers to the process of making reservations for N (configurable) jobs to run immediately or at some time later.

The scheduling algorithm preserves priority of jobs and also optimises the use of cluster resources. However, this scheduling algorithm is inadequate for jobs of unpredictable execution times due to recursion and different amount of processing.

The ScoreD System. Score-D is a user level parallel operating system providing single system image for a cluster. The scheduling method used in ScoreD is called "time and space sharing scheduling" e.g. multiple users jobs are executed in a dynamic workload distribution fashion sharing nodes among multiple jobs. This strategy scheduling means that users cannot specify where to run their jobs. ScoreD implements *Gang Scheduling* [8] that could be defined as "all of a program's threads of execution being grouped together into a gang and concurrently scheduled on distinct processors of a parallel computer. Time slicing is supported by concurrent pre-emption and later rescheduling of the gang"[4]

Figure 8 shows the philosophy behind a ScoreD cluster. Connected to ScoreD there are only a few servers running broadcast programs that clients use to get cluster monitor information instead of directly to ScoreD. This cascading reduces workload on ScoreD and network traffic as well as promotes fault tolerance since the failure of a server only affects its clients.

---

[2] *Policy* algorithms can be programmed in TCL and C languages. NOTE: For building a policy scheduler all that is needed is to build a function called sched_main() (and all functions supporting it) in a module called *pbs_sched.c* [6]

[3] Maui Scheduler is being used in some of major clusters around the world and is being ported to smaller clusters.[7]

[4] Gang Scheduling timesharing on Parallel computers, Lawrence Livermore National Laboratory http://www.llnl.gov/sccd/lc/gang

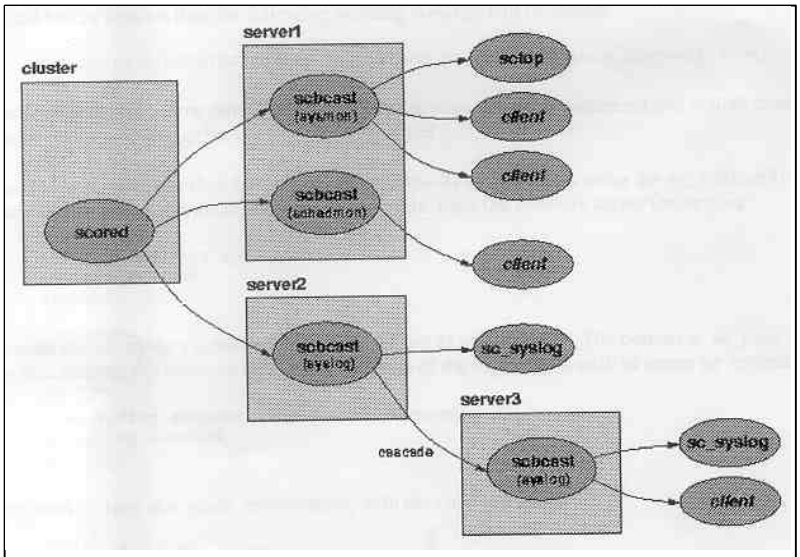

Figure 8-Organization of a ScoreD Cluster

This previous systems assume some sort of calculation of jobs workload or priority and centralised scheduling of jobs based on that workload or priority which has to be done by the application, operating system or an administrator. The next system will avoid this application ´s extra work.

The Mosix System. Mosix is a cluster computing system made of a collection of nodes working co-operatively as a single system. Each node is both a single system for running local created processes and a cluster node for running remote processes that migrated from other nodes. The core of the Mosix system are adaptive load balancing and file I/O optimisation algorithms that respond to uneven load distribution or excessive disk swapping of one of the nodes due to lack of memory. Migration policy is particularly useful in CPU bound processes but it poses some problems when dealing with I/O or file intensive jobs.

The mosix distribution scheduler policy takes into account I/O operations basically trying to migrate jobs to nodes where I/O takes place. [9]

Mosix process management main characteristics are [10]:

- Probabilistic information dissemination algorithms
- Pre-emptive process migration
- Dynamic load
- Memory ushering
- Efficient Kernel communication

Mosix also implements a direct file system access (DFSA) to optimise file access. MFS is a file system implementation on Mosix. [10]

Mosix promises a large-scale distributed cluster since there is no centralised management of the cluster and load balancing is automatic. Furthermore there is no need for job workload calculation. MPI and PVM pose reference problems on MOSIX due to static reference to nodes in these languages.

Windows NT Clusters Systems. Windows NT clusters are based in a three part clustering architecture as shown is figure 11 with:

- Network Load Balancing: provides load balancing and fail over support for IP based applications and services.

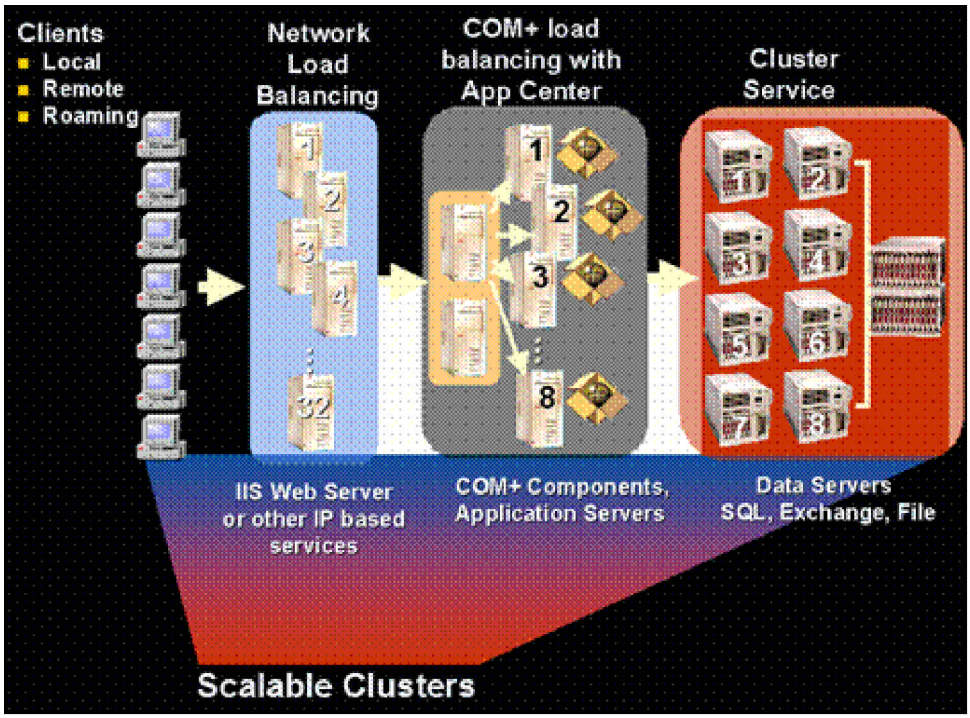

Figure 11 [11]

- Component load balancing: provides dynamic load balancing for middle-tier applications components using a proprietary Microsoft technology called COM+.[5]
- Server Clusters: provides fail over support for applications and services.[6]

Network Load Balancing Algorithm in Windows NT. Network Load Balancing (NLB) uses a virtual IP address to which clients requests are directed. When a load balanced resource fails remaining servers take over the workload of the failed server. When server comes back on-line can automatically rejoin the group (usually it takes less then 10 sec).

The NLB Algorithm [12] runs on parallel on all cluster nodes. An arriving request (job) is broadcasted to all cluster nodes, which run the load-balancing algorithm (based on a randomisation function) that calculates a host priority to run the job. Elected host maintains the request and the other cluster nodes discard it. [12] The algorithm does not respond to changes in each cluster node workload such as CPU and memory usage only the raw performance of the node is calculated upon joining the node to the cluster. This architecture may be well suited for small, independent from each other and numerous jobs like a mail server, an internet site server or a central database but could be not so suited to uneven workload or interdependent jobs. Since each cluster node runs the balancing algorithm no network traffic in cluster state monitoring is generated.

Due to the scheduling distribution, scalability is also promoted although the cluster limit is set, for now, to 32 nodes. Fault tolerance is another issue since the scheduling algorithm is run independently on each node assigning a job to a node that as failed is not known by the remaining nodes. This situation is dealt by a periodic message detection mechanism called *heartbeats* to detect non-responding systems. [13]

---

[5] Proprietary technology for background services load balancing . Available at windows2000 site http://www.microsoft.com/windows2000/server/

[6] Cluster Server & Load Balancing at http://www.microsoft.com/windows2000/technologies/clustering

Component Load Balancing. Application Workload Distribution in Windows NT Clusters uses a proprietary technology (COM®+) with which several copies of an application are started in several, limited for now up to 8, systems (or nodes). A routing list contains a list of applications servers and response times. The response times are tracked in a memory table updated in round robin fashion by the time tracker. The router read the table to route incoming requests to the application server with fastest response time enabling workload distribution by increasing workload in faster application servers.

Cluster Architecture in Windows NT. In Windows NT the cluster service is a collection of software on each node that manages all cluster related activity. A resource is the simplest item managed by the cluster service, which sees the other resources as opaque objects with a callable public object interface. A Resource can be hardware devices like disk drives and network cards or logical devices like logical network disks, applications or databases. A *group* is a collection of resources managed as a single resource and usually contains the resources needed for running a specific application and enable client connections. Services in a Windows NT cluster are exposed as virtual servers. Client applications believe they are connecting to a physical system, but in fact, are connecting to a service that which may be provided by one of several systems. Clients create a TCP/IP session with a service using a known IP address. In case of failure the service will transparently be moved to another *system* and the client will detect a session failure and will reconnect again in exactly the same manner as the previous connection.

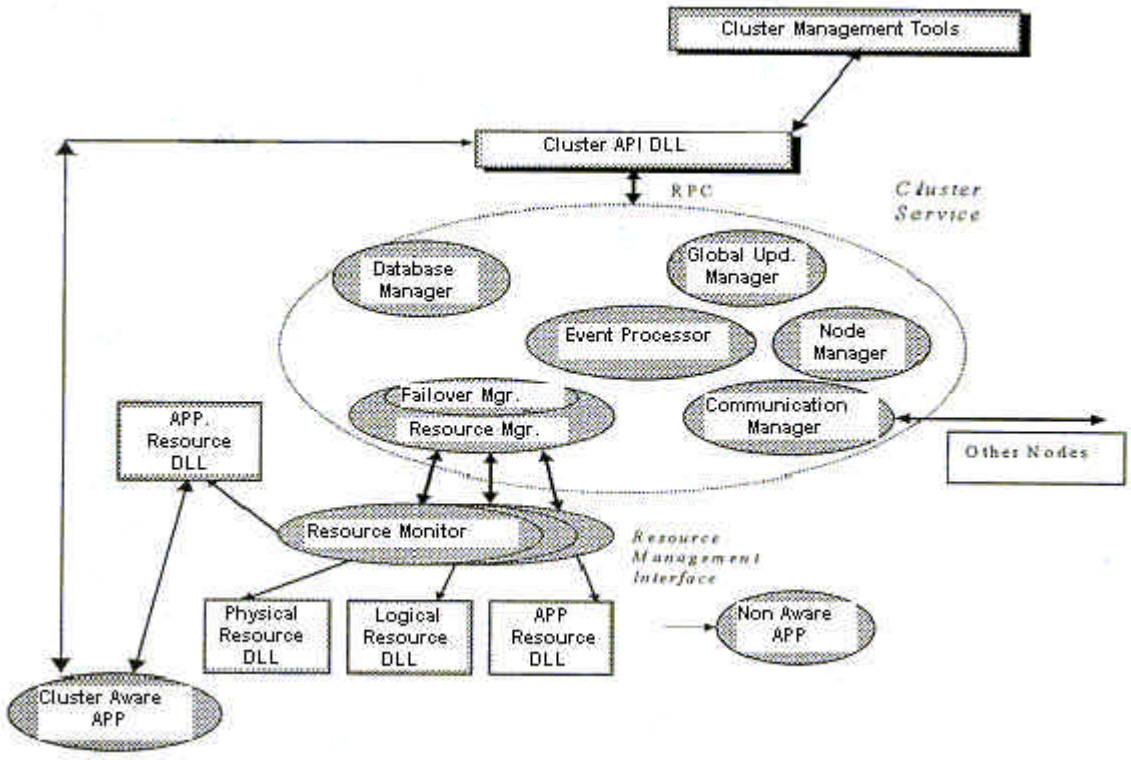

Figure 14 [13]

Figure 14 shows an overview of the components and their relationships in a single system of a Windows NT cluster.

3 Conclusions

First it must be said that none of these packages have been installed, used or tested. Instead was gathered information about their basic components architecture and was tried to present their main features for applications serviceability in a manner that someone with a specific application and knowing main features of each package could choose the most appropriate one. From an application ´s point of view and considering the components described in previous sections these packages could be shortly classified in 3 groups shown in figure 14.

	Windows NT	**Mosix**	**Linux**
Fault Tolerance	Distributed, based in group communication.		Some packages have checkpointing.
Scalability	For now, limited to 8 systems.	64 CPU configurations are in operation. Most Benchmarking results are based in 16 to 32 CPU range configurations.	128 CPU configurations are in operation. Several hundred CPU configurations for the near future.
Workload distribution	Based in node average response time.	Automated probabilistic load balancing algorithms. Also based on several optimisations made to operating system services.	Default, configurable, programmable and application driven job scheduling policies area available.

Figure 14

From figure 14 we can deduce that Windows NT is more oriented to applications that need fault tolerance and are not CPU intensive but rather I/O directed such as database, mail or Web servers. Mosix like the Linux-based group do not have robust fault tolerance mechanisms as windows NT but possess more scalability and workload distribution capabilities enabling them for CPU demanding applications such as physics or chemistry simulations, image analysis, computer movie animation, etc. Mosix has a “fork and forget” parallel kind of programming philosophy with automated job management and applications can not override these algorithms. Each node can be used by a local user and at the same time executing background jobs for the cluster. The Linux-based term is used to refer to the packages shown in figure 1.As seen these packages use identical public available components and therefore we believe they perform identically under applications. We believe they can be considered as component-integrated packages much in the same way as the mass-market Linux distributions available today. However ScoreD implements some more advanced features like *Gang Scheduling* and *Cascading Servers* not yet available on the other packages of this group. The main difference in workload distribution is that in Mosix it is automatic and transparent to applications while in the Linux group it can be configured or application driven.

A method for choosing a cluster management package depends on the application’s requirements, usually demanding. First, requirements should be estimated in terms of fault tolerance, scalability and workload distribution. Second knowing the application requirements and the packages features we can choose a package that better meets the application requirements. For instance if an application workload can be determined then a Maui Scheduling policy will most certainly perform better and therefore a package supporting it would be recommended. An application that manages workload distribution among nodes could perform badly with Mosix due to transparent job migration algorithms. If we want to run a standard single system unix application Mosix would be a good choice providing extra

scalability without changing the application. MOSIX changes to operating system services could reveal not fully compatible with standard specifications causing incompatibilities to applications.

Finally, most of these packages, except Windows NT, have downloadable versions that can be used for testing. A choice is relatively simple between windows NT and others (MOSIX and the linuxes group) since NT is more oriented to fault tolerant file I/O applications then MOSIX and the linuxes group. The drawback is that, comparing cluster packages oriented to CPU applications, such as MOSIX and the group of linuxes, we can not, just by knowing the packages features, determine which is more suited for a specific application. This variable performance due to probabilistic behaviour of the workload distribution algorithms derives that performance of these algorithms is difficult to quantify therefore measurements to the applications would be necessary to determine which package performs better.[7] CPU intensive applications that vary greatly in CPU and RAM usage and job interaction could pose difficulties to automatic dynamic load balancing algorithms. In these cases configurable or application driven workload distribution could be useful. For a mass-market cluster operating system, automatic load-balancing algorithms with average acceptable performance with the most common type of applications could be a good compromise between efficient usage of cluster resources and overall throughput.

References

[1] G. Pfister, In Search of Clusters (Prentice-Hall, Inc., 1995) page 72

[2] MPI, http://www-unix.mcs.anl.gov/mpi

[3] PVM, http://www.epm.ornl.gov/pvm

[4] *Cluster Command & Control Suite (C3)* Michael Brim, Ray Flanery, Al geist, Brian Luethe and Stephen Scott, Oak Ridge National Laboratory http://www.csm.ornl.gov/torc/c3

[5] PBS Product features, http://www.openpbs.org/about.htm, http://www.openpbs.org/features.htm

[6] The Job Scheduler, related information in http://openpbs.org/scheduling.htm

[7] The Maui scheduler, http://sourceforge.net/projects/mauischeduler and http://havi.supercluster.org/documentation/maiu

[8] For more information on ScoreD scheduling see foundation for real world computing Japan, http://pdswww.rwcp.or.jp/dist/score/html/sys/scored/scheduling.html.

[9] Amar, Lior and Barak, Amnon Hebrew University, Israel , The MOSIX system, http://www.cs.huji.ac.il/mosix , for a description of file related migration policies in Mosix see Section 3.1 Bringing the process to the file in The MOSIX Scalable Cluster File Systems for LINUX (www.cs.huji.ac.il/mosic/publications.)

[10] Barak, Amnon and La' adan, Oren Hebrew University, Israel , The Mosix Multicomputer Operating System for High Performance Cluster Computing.

[11] Windows Clustering Technologies –An Overview, a Microsoft windows 2000 Server Technical Article www.microsoft.com/windows2000/techinfo/planning/clustering.asp

[7] As an example of applications performance testing for parallel applications over MOSIX see section 4 [10]

[12] Network Load balancing overview white paper Microsoft www.microsoft.com/windows2000

[13] Short,Rob Gamache,Rod Vert,john and Massa, Mike Windows NT Clusters for Availability and Scalability white paper , Microsoft Corporation and Windows 2000 clustering technologies: cluster service architecture white paper Microsoft Corporation.