1995

Cluster Computing Review

Mark Baker
Syracuse University

Geoffrey C. Fox
Syracuse University, Northeast Parallel Architectures Center

Hon W. Yau
Syracuse University, Northeast Parallel Architectures Center

Recommended Citation

Baker, Mark; Fox, Geoffrey C.; and Yau, Hon W., "Cluster Computing Review" (1995). *Northeast Parallel Architecture Center.* Paper 33.
http://surface.syr.edu/npac/33

Cluster Computing Review

Mark A. Baker, Geoffrey C. Fox and Hon W. Yau

Northeast Parallel Architectures Center

111 College Place Syracuse University New York 13244−4100 USA

(mab@npac.syr.edu)

16 November 1995

Version 1.1

Preface

In the past decade there has been a dramatic shift from mainframe or ‘host−centric’ computing to a distributed ‘client−server’ approach. In the next few years this trend is likely to continue with further shifts towards ‘network−centric’ computing becoming apparent. All these trends were set in motion by the invention of the mass−reproducible microprocessor by Ted Hoff of Intel some twenty−odd years ago.

The present generation of RISC microprocessors are now more than a match for mainframes in terms of cost and performance. The long−foreseen day when collections of RISC microprocessors assembled together as a parallel computer could out perform the vector supercomputers has finally arrived.

Such high−performance parallel computers incorporate proprietary interconnection networks allowing low−latency, high bandwidth inter−processor communications. However, for certain types of applications such interconnect optimisation is unnecessary and conventional LAN technology is sufficient. This has led to the realisation that clusters of high−performance workstations can be realistically used for a variety of applications either to replace mainframes, vector supercomputers and parallel computers or to better manage already installed collections of workstations. Whilst it is clear that ‘cluster computers’ have limitations, many institutions and companies are exploring this option.

Software to manage such clusters is at an early stage of development and this report reviews the current state−of−the−art. Cluster computing is a rapidly maturing technology that seems certain to play an important part in the ‘network−centric’ computing future.

Tony Hey The University of Southampton

Chapter 1

1. Cluster Computing Review

1.1 Summary of Conclusions

· The use of clusters of workstations to increase the throughput of user applications is becoming increasingly commonplace throughout the US and Europe.

· There now exists a significant number of Cluster Management Software (CMS) packages and more than twenty are mentioned in this review. These packages almost all originate from research projects, but many have now been taken−up or adopted by commercial vendors.

· The importance of cluster software can be seen by both the commercial take−up of these products and also by the widespread installation of this software at most of the major computing facilities around the world.

· It is not clear that CMS is being used to take advantage of spare CPU cycles, but it is evident that much effort is being expended to increase throughput on networks of workstations by load balancing the work that needs to be done.

· Six current CMS packages, two public domain and four commercial, out of the nineteen reviewed, have been identified as being worth serious investigation. The advertised functionality of these six packages is very similar and an informed choice would need installation and detailed testing.

· If finances permit, it is probably wise to choose one of the commercial packages. This will minimise the efforts of the on−site staff and leave the onus on vendors to ensure that their software is installed, used and supported properly.

· A number of packages were identified that were either newly established CMS projects at an early stage of development or which could be considered as complete Cluster Computing Environments (CCE). In a CCE the cluster is used in a fashion similar to a distributed shared memory environment.

· Nearly all the CMS packages are designed to run on Unix workstations and MPP systems. Some of the public domain packages support Linux, which runs on PCs. One commercial package, Codine, supports Linux. JP1[1] from Hitachi ltd. is designed to run on Windows NT platforms. No CMS package supports Windows−95.

· The software being developed for CCE is potentially the most viable means of utilising clusters of workstations efficiently to run user applications in the future.

· `WWW` software and `HTTP` protocols could clearly be used as part of an integrated CMS package. Little software of this type has so far been developed at present but several of the packages reviewed used a `WWW` browser as an alternative GUI.

1.2 Introduction

This review was commissioned by the Joint Information Systems Committee (JISC) New Technology Sub Committee (NTSC) and follows two other reports: a historic review of Cluster Computing produced by the Manchester Computing Centre [1], and a critical review of Cluster Computing projects funded by the NTSC [2] and produced by the University of Southampton.

The overall aim of this review is to guide the reader through the *mine−field* that surrounds distributed and cluster computing software. In particular it aims to provide sufficient information for staff in a typical university Computing Service to:

− Identify the potential benefits and costs associated with Cluster Computing.

− Understand the important features of CMS.

− Provide an overview of the CMS packages presently available.

− Act as a guide through the maze of CMS packages.

− Be used as an aid for making decisions about which CMS package to use.

− Provide references, technical details and contact points.

In addition the review provides information about CCE. These are predominantly new projects which should provide indicators to show the future direction of CMS.

The material contained in this report has been put together using information gathered from numerous World Wide Web (WWW) sites, product descriptions supplied by vendors or included in software releases, and from various cluster review documents. Due to the limitations of time, the software discussed (nearly thirty packages) in this document has only been reviewed by addressing the issue of functionality (a paper exercise) rather than practically assessing each product, by installing it and testing it for quality and other factors, such as ease of use by administrators and users.

1.3 Organisation of this Review Document

This report is organised as follows:

Chapter 1 − introduces and then briefly discusses the scope and range of the review.

Chapter 2 − describes and discusses the criteria which will be used to judge and evaluate the functionality of the various Cluster Management Software (CMS) packages. This entails judging what the user of such a system wants against what a particular packages provides.

Chapter 3 − describes briefly each CMS package and its functionality.

Chapter 4 − in this chapter the CMS packages mentioned in the previous chapter are evaluated against the criteria discussed in chapter 2.

Chapter 5 − in this chapter each of the Computing Cluster Environments (CCE) is described.

Chapter 6 − includes comments about CMS, a step−by−step guide for choosing a CMS package, a short discussion about CMS and some views about the future of CMS.

Chapter 7− a glossary of the terms used throughout this review is included in this chapter.

Chapter 8 − includes a list of references and a table of contents.

1.4 Comments about this Review

Whilst gathering the information to produce this review it became evident that software for cluster computing software fell into one of two camps:

1. Cluster Management Software (CMS) − this software is designed to administer and manage application jobs submitted to workstation clusters. It encompasses the traditional batch and queuing systems.

2. Cluster Computing Environments (CCE) − with this software the cluster is typically used as an applications environment, similar in many ways to distributed shared memory systems. This chapter also includes descriptions of environments which are at an early development stages.

This review is primarily aimed at CMS, which is reviewed in Chapter 4. The CCE packages and ones in early stages of development or implementation are briefly described in Chapter 5.

The authors wish, in particular, to acknowledge the work of Kaplan & Nelson [3 & 4], and their evaluation criteria (Chapter 2, page 6) used in this review is based on their work. The authors acknowledge the help and assistance of vendors and numerous staff

tasked with supporting the various packages. The authors would like to thank the JISC−NTI, and in particular Tom Franklin, for sponsoring this review. Finally, we wish to thank Tony Hey for his comments on this review.

1.5 Cluster Software and Its Interaction With the Operating System

In order to understand cluster software it is necessary to know how it interacts with the operating systems of a particular platform.

The CMS described in this review works completely outside the kernel and on top of a machines existing operating system. This means that its installation does not require modification of the kernel, and so basically the CMS package is installed like any other software package on the machine.

The situation is very different with the CCE. Here, typically, a micro−based kernel with customised services needs to be installed instead of the existing kernel to support the desired environment. The reason for this more radical solution with CCE is the need to support functionality, such as virtual shared memory. This could not be achieved efficiently outside the kernel.

The BSP and PVM packages can exist in two forms. The public domain versions of these packages are installed on top of an existing operating system. Whereas the vendor versions of the software are often integrated into the kernel to optimise their performance.

1.6 Some Words About Cluster Computing

So called cluster computing systems and the means of managing and scheduling applications to run on these systems are becoming increasingly common place. CMS packages have come about for a number of reasons; including load balancing, utilising spare CPU cycles, providing fault tolerant systems, managed access to powerful systems, and so on. But overall the main reason for there existence is their ability to provide an increased, and reliable, throughput of user applications on the systems they manage.

The systems reviewed in this report are those that are more commonly known, not all, and for that matter not necessarily the best. Approximately twenty CMS systems are briefly reviewed, at least two more packages are known to exist, but have been excluded because the authors had great difficulty getting further information on these products.

This review attempts to gauge and assess the features and functionality of each CMS package against a set of criteria deemed to be highly desirable or useful additions. The criteria that has been adopted for assessing the CMS packages is based heavily upon that devised by Kaplan & Nelson [3 & 4] at NASA. However, the criteria in this review has been broadened and modified to reflect the needs of the JISC−NTI and the knowledge and experience of the authors [5, 6, 7, 8, 9 & 10].

1.7 The Workings of Typical Cluster Management Software

To run a job on a CMS batch system it is usual to produce some type of resource description file. This file is generally an ASCII text file (produced using a normal text editor or with the aid of a GUI) which contains a set of keywords to be interpreted by the CMS. The nature and number of keywords available depends on the CMS package, but will at least include the job name, the maximum runtime and the desired platform.

Once completed, the job description file is sent by the client software resident on the user's workstation, to a master scheduler. The master scheduler, as its name implies, is the part of the CMS that has an overall view of the cluster resources available: the queues that have been configured and the computational load on the workstations that it is managing. On each of the cluster workstations daemons are present that communicate their state at regular intervals to the master scheduler. One of the tasks of the master scheduler is to evenly balance the load on the resources that it is managing. So, when a new job is submitted it not only has to match the requested resources with those that are available, but also needs to ensure that the resources being used are load balanced.

Typically a batch system will have multiple queues, each being appropriate for a different type of jobs. For example, a queue may be set up for a homogeneous cluster which is primarily used to service parallel jobs, alternatively queues may be set up on a powerful server for CPU intensive jobs, or there may be a queue for jobs that need a rapid turnaround. The number of possible queue configurations is large and will depend on the typical throughput of jobs on the system being used.

The master scheduler is also tasked with the responsibility of ensuring that jobs complete successfully. It does this by monitoring jobs until they successfully finish. However, if a job fails, due to problems other than an application runtime error, it will reschedule the job to run again.

1.8 Clusters of Workstations: The Ownership Hurdle.

Generally a workstation will be "owned" by, for example, an individual, a groups, a department, or an organisation. They are dedicated to the exclusive use by the "owners". This ownership often brings problems when attempting to form a cluster of workstations. Typically, there are three types of "owner":

· Ones who use their workstations for sending and receiving mail or preparing papers, such as administrative staff, librarian, theoreticians, etc.

· Ones involved in software development, where the usage of the workstation revolves around the edit, compile, debug and test cycle.

· Ones involved with running large numbers of simulations often requiring powerful computational resources.

It is the latter type of "owner" that needs additional compute resources and it is possible to fulfill their needs by fully utilising spare CPU cycles from former two "owners". However, this may be easier said than done and often requires delicate negotiation to become reality.

Chapter 2

2. Evaluation Criteria

2.1 Introduction

In this chapter the functionality and features (criteria) that a potential system administrator or user of CMS are defined. The criteria used are based on those originally set out by Kaplan and Nelson [3 & 4], have been modified to reflect the views and experiences of the authors. Apart from when explicitly mentioned, the criteria used are all deemed to be *highly desirable* features of a CMS package.

2.2 Computing Environments Supported

2.2.1 COMMERCIAL/RESEARCH

Is it a commercial or a research/academic product? This factor will determine the cost and level of support that can be expected − as most users of commercial and public domain software will understand.

Comments

A commercial site is likely to require a higher degree of software stability, robustness and more comprehensive software support services than an academic site. The software is likely to be managing workstations which are not only crucial to internal productivity but may be critical for services that are being provided to a commercial entity. In the case of research sites there is often more leeway and flexibility about the levels of service that are provided or are expected. It is also possible that a research site does not have the funds available to purchase expensive "turn−key" software with high levels of user support.

2.2.2 HETEROGENEOUS

Does the software support homogeneous or heterogeneous clusters?

2.2.3 PLATFORMS

What are the hardware platforms supported?

2.2.4 OPERATING SYSTEMS

Vendor operating systems supported.

2.2.5 ADDITIONAL HARDWARE/SOFTWARE

Is there any need for additional hardware or software to be able to run the CMS package? For example, additional diskspace or software such as AFS or DCE (see glossary for explanations of terms).

Comments

It is important that the CMS chosen by a site fully supports the platforms that it is intended for. For example, problems are bound to occur if you have Sparc platforms running Solaris 2.4 and the Cluster Management Software only supports SunOS 4.1.3.

Note − It is assumed that software such as NIS and NFS is available on the clusters being used.

2.3 Application support

2.3.1 BATCH JOBS

Are batch submissions of jobs supported?

2.3.2 INTERACTIVE SUPPORT

Are jobs that would normally be run interactively supported? For example, a debugging session or a job that requires user command−line input .

2.3.3 PARALLEL SUPPORT

Is there support for running parallel programs on the cluster, for example PVM, MPI or HPF.

Comments

The provision of support for parallel computing is probably more relevant to research sites, at the moment, than commercial sites. However, support for parallel computing will often mean that the CMS is more flexible in how it can be configured, than one that only supports sequential jobs.

Note − should also be taken of the parallel software packages supported by the Cluster Management Software. As a minimum acceptable criteria it should support PVM 3.3.x and, at least, have plans to support the common industry interface MPI.

2.3.4 QUEUE TYPE

Are multiple, configurable, queues supported? This feature is necessary for managing large multi−vendor clusters where jobs ranging from short interactive sessions to compute intensive parallel applications need to run.

Comments

The configuration of queues within a managed cluster is a matter that should considered carefully. The number and configuration will determine how effectively and efficiently a cluster can be utilised.

2.4 Job Scheduling and Allocation Policy

2.4.1 DISPATCHING POLICY

Is there a configurable dispatching policy, allowing for factors such as system load, resources available (CPU type, computational load, memory, disk−space), resources required, etc.

2.4.2 IMPACT ON WORKSTATION OWNER

What is the impact of the CMS on the owner of the workstation? Is it possible to minimise the impact on a workstation? It should be possible to configure the CMS to, for example, suspend jobs when an owner is using his/her workstation or set jobs to have a low priority (`nice`) value.

Comments

Ownership of a workstation and its resources can be a rather problematic matter. To utilise CPU cycles of workstations physically distributed around a site which are owned by individuals, groups and departments can become a point of aggravation and heated debate. It is vital that the CMS is able to minimise and manage the impact of running jobs on remote workstations.

Alternatively, CMS software to manage a "headless" workstation cluster does not need to be aware of the owner. Then it is just a matter of allocating resources to jobs and ensuring that throughput is carried out efficiently and effectively.

2.4.3 IMPACT ON THE WORKSTATION

What is the impact of running the CMS package on a workstation? There will be an obvious impact when a job is running, but there also may be an undesirable impact when a job is suspended, checkpointed or migrated to another workstation. For example process migration requires that a job saves its state and then is physically moved over the local network to another workstation. This will impact on the workstation (CPU/memory and diskspace) while the state is saved and then on the network bandwidth when tens of Mbytes of data is transferred across the network.

2.4.4 LOAD BALANCING

The CMS should load balances the resources that it is managing. It is useful if the system administrator can customise the default configuration to suit the local conditions in the light of experience of running the CMS.

2.4.5 CHECK POINTING

This is a means of saving a job’s state at regular intervals during its execution. If the workstation fails then the job can be restarted at its last checkpointed position.

Comments

This is a useful means of saving a jobs state in case of failure while a job is running. Its usefulness needs to be weighed carefully against the costs in additional resources required to support it. For small jobs (ones that do not take long to run), ones that are not time critical checkpointing is unnecessary. If the jobs are time critical, for example at a commercial site where results equate directly to income, then checkpointing would be absolutely necessary.

The main cost of checkpointing is in the need for hardware to support the activity. Saving a jobs state at regular intervals, for even relatively small jobs, may require tens of Mbytes of diskspace per workstation to achieve. This may mean that:

· Additional diskspace per workstation is needed.

· Home filestore my be remotely mounted, this will have an impact on NFS performance and the network bandwidth.

· Many existing clusters will have not have the physical resources (local diskspace) to support checkpointing.

2.4.6 PROCESS MIGRATION

This is a means of migrating an executing jobs from one workstation to another. Its is often used when owner takes back control of his/her workstation. Here the job running on the workstation will, be suspended first, and then migrated onto another workstation after a certain time interval. Another use for job migration is to move jobs around to load balance the cluster.

Comments

Like checkpointing, process migration can be a very useful feature of a CMS package. Typically it is used to minimise the impact on a owner workstation (a job will be suspended and eventually migrated on to a different resource when the workstation is used by the owner) and also as a means of load balancing a cluster (migrating processes of heavily load workstation and running them on lightly loaded ones). The impact of using process migration is similar to checkpointing, but has the additional disadvantage that large state files will be moved around the network connecting the cluster. This can have a serious impact on users of the network.

2.4.7 JOB MONITORING AND RESCHEDULING

The CMS should monitor that jobs running and in the event of a job failure should reschedule it to run again.

2.4.8 SUSPENSION/RESUMPTION OF JOBS

The ability to suspend and then resume jobs is highly desirable. This feature is particularly useful to minimise the impact of a jobs on the owner of a workstation, but may also be useful in the event of a system or network wide problem.

2.5 Configurability

2.5.1 RESOURCE ADMINISTRATION

The cluster administrator should have control over the resources available. The administrator should be able to, for example, control who has access to what resources and also what resources are used (CPU load, diskspace, memory).

2.5.2 JOB RUNTIME LIMITS

The CMS should enforce job runtime limits, otherwise it will be difficult to fairly allocate resources amongst users.

2.5.3 FORKED CHILD MANAGEMENT

It is common for a job to fork child processes. The CMS should be capable of managing and accounting for these processes.

Comments

Control over forked child processes is not common under most Unix operating systems. A parent processes can spawn child process which are not managed, cannot be accounted/charged for, and can potentially have a serious impact on load balancing ability of CMS.

2.5.4 PROCESS MANAGEMENT

The CMS should be able configure the available resources to be either shared or be exclusive to a given job.

Comments

Efficient use of resources may require close control over the number of processes running on a workstation, it may even be desirable to allow exclusive access to workstations by a particular job. It should also be possible to control the priority of jobs running on a workstation to help load balancing (`nice`)and minimise the impact of jobs on the owner of the workstation.

2.5.5 JOB SCHEDULING CONTROL

The user and/or administrator should be able to schedule when a job will be run.

2.5.6 GUI/COMMAND−LINE

What user interface do users and administrators of the CMS have.

Comments

The interface, for both a user or administrator, of a software package will often determine the popularity of a package. In general, a GUI based on Motif is the standard. However, there has been a dramatic increase in usage and popularity of the HTTP protocol and the WWW, so a GUI based on this technology seems likely to be a common standard in the future.

2.5.7 EASE OF USE

How easy and/or intuitive it is for users and administrators to use the CMS.

2.5.8 USER ALLOCATION OF JOBS

Can a user specify the resources that they require. For example, the machine type, job length and diskspace.

2.5.9 USER JOB STATUS QUERY

Can a user query the status of their job. For example, to find out if it is pending/running or perhaps how long before it completes.

2.5.10 JOB STATISTICS

Are statistics provided to the user and administrator about the jobs that have run?

2.6 Dynamics of Resources

2.6.1 RUNTIME CONFIGURATION

Can the resources available, queues and other configurable features of the CMS be reconfigured during runtime? i.e. it is not necessary to restart the CMS.

2.6.2 DYNAMIC RESOURCE POOL

Is it possible to add and withdraw resources (workstations) dynamically during runtime?

2.6.3 SINGLE POINT OF FAILURE (SPF)

Is there a particular part of the CMS that will act as a SPF. For example, if the master scheduler fails does the CMS need restarting from scratch?

Comments

If the CMS has an SPF then there is no guarantee that a job submitted will complete. A typical SPF is only being able to run one master scheduler, so if it then fails the whole CMS fails. Ideally, there should be a backup scheduler which takes over if the original master scheduler fails. An SPF will also mean that the operators of the cluster will need to monitor closely the master scheduler in case of failure.

2.6.4 FAULT TOLERANCE

Is there fault tolerance built in to the CMS package? For example, does it check that resources are available before submitting jobs to them, and will it try to rerun a job after a workstation has crashed.

Comments

The CMS should be able to guarantee that a job will complete. So, if while a job is running the machine that it is running on fails then the CMS should not only notice that the machine/s are unavailable, but it should also reschedule the job that did not complete as soon as it is feasible.

Also, if a machine running a queue or CMS scheduler fails, the CMS should be able to recover and continue to run.

The real need for fault tolerance is determined by the level of service that is being provided by the cluster. However, fault tolerance is a useful feature in any system.

2.6.5 SECURITY ISSUES

What security features are provided.

Comments

The CMS should provide at least normal Unix security features. In addition it is desirable that it takes advantage of NIS and other industry standard packages.

Chapter 3

3. Cluster Management Software

3.1.1 INTRODUCTION

The aim of this chapter is to provide a brief description of each of the CMS packages, listed in the table above. The descriptions consist of information coalesced from vendor publicity, user guides and on−line (WWW) documents.

At least two other CMS packages; Balens (VXM Technologies Inc. USA) and J1 (Hitachi Inc.) − formerly known as the NC Toolset. Some difficulty has been found in trying to get further information about these packages, but when found it will be added to this review.

3.2 Commercial Packages

3.2.1 CODIINE

`URL − http://www.genias.de/genias/english/codine/`

Codine [11] is a software package targeted at utilising heterogeneous networked environments, in particular large workstation clusters with integrated compute servers, like vector and parallel computers. Codine provides a batch queuing framework for a large variety of architectures via a GUI−based administration tool. Codine also provides dynamic and static load balancing, checkpointing and supports batch, interactive and parallel jobs.

Main Features of Codine

· Support for batch, interactive and parallel (Express, p4 and PVM) jobs.

· Support for multiple queues.

· Support for checkpointing.

· Static load balancing.

· Dynamic load balancing by checkpointing and job migration.

· NQS interface − used for integration with existing NQS−based systems.

· Accounting and utilisation statistics.

· X11 Motif GUI, command−line and script interfaces for administrators and users.

· POSIX compliance.

· Support for DCE technology.

3.2.2 CONNECT:QUEUE

URL http://www.sterling.com/

This package is a commercial variation of NQS (until recently know as Sterling NQS/Exec) that is commercially marketed and supported by Sterling Software Inc. Its feature and functionality are very similar to GNQS.

The Package provides a Unix batch and device queuing facility capable of supporting wide range of Unix−based platforms. It provides three queue types:

· Batch queue, providing a percentage mechanism for scheduling and running jobs;

· Device queue, which provides batch access to physical devices;

· Pipe queue, which transports requests to other batch, device or pipe queues at remote locations.

An intelligent batch job scheduling system provides job load balancing across the workstation clusters being managed. Load balancing is based on a scheduling algorithm which uses three statistics :

· Percentage of physical memory used.

· CPU utilisation

· Queue job limit.

3.2.3 LOAD BALANCER

URL
http://www.unison.com/main−menu/products/operations/loadbalanc

Load Balancer attempts to optimise the use of computer resources by distributing workloads to available UNIX systems across the network. Load Balancer determines availability based on the:

· Capacity and current workload of each computer.

· Resources required by the submitted job.

· User defined and other constraints

Load Balancer tries to increase the overall job throughput by making use of idle computer resources. At the same time, it attempts to prevent systems becoming overloaded by distributing the load evenly across the available computers.

Major Features

· No modification to applications or kernel.

· Central control and administration.

· Policy and constraint options.

· Job/application distribution criteria include:

− CPU Speed.

− Current load average.

− Time of day.

− RAM and Swap space.

− CPU type (HP, IBM, etc.).

− Presence and absence of interactive user(s).

− Other user−defined criteria.

· Robust and scalable.

· Works with both batch jobs and interactive applications.

· Security features are built in to restrict unauthorised access.

3.2.4 LOADLEVELER

`URL http://lscftp.kng.ibm.com/pps/products/loadlev.html`

Load Leveler is a job scheduler that distributes jobs to a cluster of workstations and/or to nodes of a multi−processor machine [12]. LoadLeveler decides when and how a batch job is run based on preferences set up by the user and system administrator. Users communicate with LoadLeveler using a few simple LoadLeveler commands, or by using the LoadLeveler GUI. Jobs are submitted to LoadLeveler via a command file, which is much like a UNIX script file. Jobs are held in the queue until LoadLeveler can allocate the required resources to run the job. Once the job has completed, LoadLeveler will (optionally) notify the user. The user does not have to specify what machines to run on, as LoadLeveler chooses the appropriate machines.

Features

· Distributed job scheduler.

· Serial/parallel batch and interactive workload.

· Workload balancing.

· Heterogeneous workstation support.

· Compatibility with NQS.

· Central point of control.

· Scalability (queues and master schedulers).

When a job is scheduled, its requirements are compared to all the resources available to LoadLeveler. Job requirements might be a combination of memory, disk space, architecture operating system and application programs. LoadLeveler's central manager collects resource information and dispatches the job as soon as it locates the suitable resources. Load Leveler accepts shell scripts written for NQS so that jobs can be run under LoadLeveler or under NQS−based systems. LoadLeveler also provides a user or system−initiated checkpoint/restart capability for certain types of Fortran or C jobs linked to the LoadLeveler libraries.

Interactive session support – LoadLeveler's interactive session support feature allows remote TCP/IP applications to connect to the least loaded cluster resource.

Individual control – Users can specify to LoadLeveler when their workstation resources are available and how they are to be used.

Central control – From a system management perspective, LoadLeveler allows a system administrator to control all the jobs running on a cluster. Job and machine status are always available, providing administrators with the information needed to make adjustments to job classes and changes to LoadLeveler controlled resources.

Scalability – As workstations are added, LoadLeveler automatically scales upward so the additional resources are transparent to the user.

Load Leveler has a command line interface and a Motif−based GUI. Users can:

· Submit and cancel jobs.

· Monitor job status.

· Set and change job priorities.

3.2.5 LOAD SHARING FACILITY (LSF)

URL http://www.platform.com/products/overview.html

LSF is a distributed load sharing and batch queuing software package for heterogeneous UNIX environments. LSF manages job processing by providing a transparent, single view of all hardware and software resources, regardless of systems in the cluster. LSF supports batch, interactive and parallel jobs and manages these jobs on the cluster making use of idle workstations and servers.

Specifically, LSF provides the following:

· Batch job queuing and scheduling.

· Interactive job distribution across the network.

· Load balancing across hosts.

· Transparent access to heterogeneous resources across network.

· Site resource sharing policy enforcement

· Network−wide load monitoring.

Other LSF Features

· Comprehensive load and resource information.

· Transparent remote execution. The same execution environment (user ID, umask, resource limits, environment variables, working directory, task exit status, signals, terminal parameters, etc.) are maintained when jobs are sent to execute remotely.

· Batch and interactive processing.

· Sequential and parallel processing.

· Heterogeneous operation.

· Application Programming Interface (API) − allowing software be developed for new distributed applications.

· Fault tolerance − LSF is available as long as one host is up and no job is lost even if all hosts are down.

· Site policy options − provides configuration options for sites to customise their own resource sharing policies.

· Job accounting data and analysis tools.

3.2.6 NETWORK QUEUING ENVIRONMENT (NQE)

URL http://www.cray.com/PUBLIC/product−info/craysoft/WLM/NQE/NQE.h

NQE [13] provides a job management environment for the most popular workstation by distributing jobs to the most appropriate UNIX systems available in a heterogeneous network, allowing users to share resources. NQE is compatible with Network Queuing System (NQS) software, but has a functionality that exceeds the basic NQS capability. NQE has the following features:

· Automatic load balancing across an entire network.

· Parallel Virtual Machine (PVM) support.

· Job−dependency capability that allows users to specify the criteria that must be met before a specific job may be initiated.

· Automatic job routing to systems that match user−specified attributes.

· Unattended and secure file transfer to/from any client using the File Transfer Agent (FTA).

· Batch job submission from remote locations, using the NQE client software.

· Compatibility with other UNIX systems running public domain versions of NQS.

· Security features and multiple authentication mechanisms.

· Motif GUI for network system and NQE job, installation, and on−line documentation.

· Automatic placement of jobs in wait queues for systems not available.

· Multiple job and administration. logs.

· Application programming interfaces (APIs) for customising applications to take advantage of NQE’s functionality.

· World−wide web interface to submit NQE jobs, view job status, and receive back output.

3.2.7 TASK BROKER

URL http://www.hp.com/

Task Broker is a software tool that attempts to distribute computational tasks among heterogeneous UNIX−system−based computer systems. Task Broker performs its computational distribution without requiring any changes to the application. Task Broker will relocate a job and its data according to rules set up at initialisation. The other capabilities provided by Task Broker include:

· The ability to load balancing computational load among a group of computer systems.

· Intelligent targeting of specialised jobs so that they run on the most appropriate server for the specific task. For example, a graphics application may be most efficiently run on a machine with a graphics accelerator or − the user needs have no machine−specific knowledge.

· DCE inter−operability.

Task Broker automates the distribution of computations by:

− Gathering machine−specific knowledge from the user.

− Analysing machine−specific information and selecting the most available server.

− Connecting to a selected server via `telnet`, `rsh` (remote shell), or `crp` (create remote process).

− Copying program and data files to a selected server via `ftp` or NFS.

− Invoking applications over the network

− Copying the resulting data files back from the server via `ftp` or NFS.

Each of the above steps is done automatically by Task Broker without the user needing to be aware of, or having to deal with, the details of server selection and data movement.

Task Broker Features:

· Runs without requiring application code to be recompiled.

· Adds servers and clients to a network simply by hooking them to a LAN, updating the central configuration file to include the new client server, and starting the Task Broker daemon.

· Provides an accounting of services used on a given server.

· Allows limiting of the number of applications running on a computer at any one time. This prevents degradation of server performance.

· Permits control of the use of computers as servers. For example, users can specify that their workstations be accessed only at night.

· GUI provides a visual interface to most of the Task Broker command set and configuration information. In addition, task status monitoring and control are provided for the user.

· An on−line, context−sensitive help system.

3.3 Research Packages

3.3.1 BATCH

`URL`
`http://cornelius.ucsf.edu/~srp/batch/ucsf/batchusr_toc.html`

This software [14] is designed to manage multiple batch job queues under the control of a daemon process. This daemon controls the batch jobs through the use of the BSD job control signals, while the client programs, `batch`, `baq`, and `barm` provide for submission, examination and removal of jobs, respectively.

The capabilities include:

· Start and suspend running jobs at specific times and/or days.

· Start and suspend running jobs at specific machine loads.

· Start and suspend jobs based on whether someone is using the console.

· Logging to the system log at varying levels. Statistics and database information are also available via signals sent to the daemon process.

· Accounting of jobs for each queue.

· Remote queues. Queues may be localised to a specific machine.

· Load balanced queues. Given a set of hosts supporting the same queue, the machines (batch daemons) decide amongst themselves which host will run the current job, based on load averages, currently running jobs and other details.

· Job runtime limitations.

· Run jobs with specific `nice` values.

· Limited console user control over the spooled jobs to minimise the effect of batch jobs on interactive response during work hours.

· Ancillary programs for job and queue inspection, removal or manipulation.

· A GUI (using the forms library) is also available to the job status and removal tools.

3.3.2 COMPUTING CENTRE SOFTWARE (CCS)

URL http://www.uni−paderborn.de/pcpc/work/ccs/

The Computing Center Software [15 & 16] is itself a distributed software package running on the front−end of an MPP system. The Mastershell (MS), which runs on a front−end workstation, is the only user interface to CCS. It offers a limited environment for creating Virtual Hardware Environments (VHE) and running applications in interactive or batch mode.

The Port−Manager (PM) is a daemon that connects the MS, Queue−Manager (QM) and the Machine−Managers (MM) together. The MS can be started manually by the user or automatically by the surrounding Unix system as the user's default login shell. In either case a connection to the PM is established and data identifying the is transferred. The PM uses this information to initiate a first authorisation, on failure the user session is aborted immediately. Otherwise, the user has the whole command language of the MS at his/her disposal.

If a user requests a VHE consisting of a number of processors in a certain configuration and wants exclusive usage for one hour. The number of VHEs a user can handle simultaneously is only restricted by the limitations of the metacomputer and the restrictions set up by the administrator or given by the operating system. When a VHE is ordered from the MS side the PM checks the user's limitations first, i.e. the maximum number and kind of resources allowed for the requesting user or project. If the request validation is successful, the VHE is sent to the QM.

The QM administers several queues. Depending on priority, time, resource requirements, and the application mode (batch or interactive), a queue for the request is chosen. If the scheduler of the QM decides that a certain VHE should be created, it is sent to the PM. The PM configures the request in co−operation with appropriate MMs and supervises the time limits. In addition, the PM generates operating and accounting data.

The user is allowed to start arbitrary applications within his VHE. The upper level of the three level optimisation policy used by CCS, corresponds to an efficient hardware request scheduling (QM). The mid−level maps requests onto the metacomputer (PM) and the third level handles the system dependent configuration software, optimising request placements onto the system architectures (MMs). This three level policy leads to a high level load balancing within the metacomputer.

3.3.3 CONDOR/

URL http://www.cs.wisc.edu/condor/

Condor [17 & 18] is a software package for executing batch type jobs on workstations which would otherwise be idle. Major features of Condor are automatic location and allocation of idle machines, checkpointing and the migration of processes. All of these features are achieved without any modification to the underlying UNIX kernel. It is not necessary for a user to change their source code to run with Condor, although programs must be specially linked with Condor libraries.

The Condor software monitors the activity on all the participating workstations in the local network. Those machines which are determined to be idle are placed into a resource pool. Machines are then allocated from the pool for the execution of jobs. The pool is a dynamic entity − workstations enter when they become idle, and leave again when they get busy.

Design Features

· The user only needs to relink their code to run under Condor.

· The local execution environment is preserved for remotely executing processes.

· The Condor software is responsible for locating and allocating idle workstations.

· "Owners" of workstations have complete priority over their own machines.

· Jobs are guaranteed to complete eventually.

· Local diskspace is not used by remotely executing Condor job.

· Condor works outside the kernel, and is compatible with BSD and other Unix derivatives.

3.3.4 DISTRIBUTED JOB MANAGER (DJM)

`URL http://www.msc.edu/`

DJM is a job scheduling system designed to allow the use of massively parallel processor (MPP) systems more efficiently. DJM provides a comprehensive set of management tools to help administrators utilise MPP systems effectively and efficiently.

Main features:

· Pack jobs into the MPP in order to minimise the number of idle processing elements (PEs). DJM uses a packing algorithm that attempts to select the best physical location and order that the jobs should be run on the MPP.

· Ensure that jobs complete. DJM forecasts future job activity to prevent jobs from being killed because of scheduled events, such as a maintenance shutdown.

· Select jobs to run based on local site policy. DJM can be configured to determine the order in which jobs are run, including the time of day, the class of user, and the size of the job.

· Reserve the MPP for real−time applications, heterogeneous applications, or live demonstrations.

· Manage access to the MPP by both interactive and batch users.

· Runs transparently underneath NQS (Network Queuing System).

3.3.5 DISTRIBUTED QUEUING SYSTEM (DQS 3.X)

URL http://www.scri.fsu.edu/~pasko/dqs.html

DQS 3.1 [19] is an experimental UNIX based queuing system being developed at the Supercomputer Computations Research Institute (SCRI) at The Florida State University. DQS development is sponsored by the United States Department Of Energy. DQS is designed as a management tool to aid in computational resource distribution across a network. DQS provides architecture transparency for both users and administrators across a heterogeneous environment.

NOTE: Dynamic Network Queuing System (DNQS) has now been superseded by DQS 3.x. DNQS was based on an earlier version of DQS produced by Tom Green at Florida State University in 1990. DNQS was basically re−written by people at McGill University in Toronto to increase the functionality and make the software easier to use and install. DNQS is still available, information about can be found at the following URLs:

URL http://www.physics.mcgill.ca/cmpdocs/dnqs.html

URL http://docserver.bnl.gov/com/www/dnqs/DNQS.html

Some features of DQS

· GUI−based interface.

· GUI−based accounting.

· Parallel Make Utility.

· Parallel package support − PVM, TCG/MSG, P4/P5 and MPI (planned).

· Interactive support.

· Scheduler based on Queue−Complexes.

· No single point of failure, redundant queue masters are supported.

Qmon – `Qmon` is a GUI to DQS based on `X/Xt`. The top module has menus for executing DQS commands and other utility functions. An icon window displays the current states of the queue and, finally a text output window to record the response of DQS commands launched from `qmon`.

Qusage – This is a `Xt`−based accounting package provided by DQS. Accounting information in a variety of forms can be retrieved via `Qusage`, it features on−line help and Postscript output. All accounting information is stored in one place, making retrieval of accounting information quick and easy.

Dmake – Distributed Make is a generic parallel make utility designed to speed up the process of compiling large packages. `Dmake` is designed for use with DQS, but can also be easily used as a standalone parallel make utility (separate from DQS). `Dmake` was developed with simplicity in mind, no daemons or other modifications to network configurations are required.

3.3.6 EXTENSIBLE ARGONNE SCHEDULER SYSTEM (EASY)

`URL`
`http://info.mcs.anl.gov/Projects/sp/scheduler/scheduler.html`

The goals of EASY [20], Argonne National Laboratory’s job scheduler, are fairness, simplicity, and efficient use of the available resources. These goals are in conflict, but the scheduler is designed to be a compromise. Users will be able to request a set of nodes for any type of use. In order to maintain the quality of machine access, the scheduler provides a single point of access, `submit`. This program allows users to queue both interactive and batch access jobs. When resources are available, the user is notified by the scheduler and at that time has exclusive access to the number of nodes requested. Having exclusive access to the nodes allows the user to have optimum cache performance and use of all available memory and `/tmp` disk space. This type of access allows users to run benchmarks at any time and also to predict how long it will take for their job to complete. Having exclusive access is essential so that users can predict wall−clock run time for their jobs when they submit them to the scheduler.

While there are currently no limits to the number or size of jobs that can be submitted, the scheduler uses a public algorithm to determine when batch or interactive time is actually provided. Any modifications to this algorithm will be made public. Argonne has also implemented an allocation policy as a separate part of the scheduler. The intent of the policy is to ensure all users some set amount of resource time and to prevent people from using more than their share of resources.

3.3.7 FAR − A TOOL FOR EXPLOITING SPARE WORKSTATION CAPACITY

URL http://www.liv.ac.uk/HPC/farHomepage.html

This project [21] is being carried out by the Computing Services Department of the University of Liverpool and is funded by the JISC New Technologies Initiative. The **far** project has developed a software tool to facilitate the exploitation of the spare processing capacity of UNIX workstations. The initial aims of the project are to develop a system which would:

· Detect lightly loaded workstations.

· Provide automatic load balancing.

· Allow a flexible workstation access policy.

· Allow users to request dedicated clusters of free workstations.

· Support a general PVM service.

far provides an environment in which the user could `rlogin` to, or issue a command via the Unix commands `at` or `rsh` which would be run automatically on the most lightly−loaded workstation in the network. The implementation of the system is based on a managed database of current workstation usage which could be inspected to find a suitable workstation. **far** also supports the exploitation of a network for running message passing parallel programs, i.e. as a loosely−coupled distributed−memory parallel computer. Functionality, such as checkpointing and process migration, has been deliberately omitted from **far**.

far release 1.0 has the following features:

· A managed database of workstation loads at any given instant.

· Ability to run remote commands on the most lightly loaded workstations.

· Support for PVM and other message passing libraries.

· System administrator control of remote access to workstations.

· Facilities for the dynamic configuration of workstation clusters

· A set of system administrator controls.

3.3.8 GENERIC NETWORK QUEUING SYSTEM (GNQS)

URL
http://www.shef.ac.uk/uni/projects/nqs/Product/Generic−NQS/v3.

The Networked, UNIX based queuing system, NQS [22 & 23], was developed under a

US government contract by the National Aeronautics and Space Administration (NASA). NQS was designed and written with the following goals in mind:

· Provide for the full support of both batch and device requests. Here a batch request is defined as a shell script containing commands not requiring the direct services of some physical device. In contrast, a device request is defined as a set of independent instructions requiring the direct services of a specific device for execution (e.g. a line printer request).

· Support all of the resource quotas enforceable by the underlying UNIX kernel implementation that are relevant to any particular batch request, and its corresponding batch queue.

· Support the remote queuing and routing of batch and device requests throughout the network of machines running NQS.

· Modularise all of the request scheduling algorithms so that the NQS request schedulers can be easily modified on an installation by installation basis, if necessary.

· Support queue access restrictions whereby the right to submit a batch or device request to a particular queue can be controlled, in the form of a user and group access list for any queue.

· Support networked output return, whereby the `stdout` and `stderr` files of any batch request can be returned to a possibly remote machine.

· Allow for the mapping of accounts across machine boundaries.

· Provide a friendly mechanism whereby the NQS configuration on any particular machine can be modified without having to resort to the editing of obscure configuration files.

· Support status operations across the network so that a user on one machine can obtain information relevant to the NQS on another machine, without requiring the user to log in on the target remote machine.

· Provide a design for the future implementation of file staging, whereby several files or file hierarchies can be staged in or out of the machine that eventually executes a particular batch request.

NQS (modified by Monsanto) has been superseded by Generic NQS 3.4, which in turn is being further developed and supported by the University of Sheffield − URL http://www.shef.ac.uk/uni/projects/nqs

3.3.9 MULTIPLE DEVICE QUEUING SYSTEM (MDQS)

URL ftp://ftp.arl.mil/arch/

The Multiple Device Queuing System (MDQS) [24] is designed to provide UNIX with a fully functional, modular, and consistent queuing system. The MDQS system was designed with portability, expandability, robustness, and data integrity as key goals.

MDQS is designed around a central queue which is managed by a single privileged daemon. Requests, delayed or immediate, are queued by non−privileged programs. Once queued, requests can be listed, modified or deleted. When the requested device or job stream becomes available, the daemon executes an appropriate server process to handle the request. Once activated, the request can still be canceled or restarted if needed.

MDQS can serve as a delayed− execution/batch. MDQS provides the system manager with a number of tools for managing the queuing system. Queues can be created, modified, or deleted without the loss of requests. MDQS recognises and supports both multiple devices per queue and multiple queues per device by mapping input for a logical device to an appropriate physical output device. Anticipating the inevitable, MDQS also provides for crash recovery.

The MDQS system has been developed at the U.S. Army, Ballistics Research Laboratory to support the work of the laboratory and is available to other UNIX sites upon request.

3.3.10 PORTABLE BATCH SYSTEM (PBS)

URL http://www.nas.nasa.gov/NAS/Projects/pbs/

The Portable Batch System (PBS) project [25] was initiated to create a flexible, extensible batch processing system to meet the unique demands of heterogeneous computing networks. The purpose of PBS is to provide additional controls over initiating or scheduling execution of batch jobs, and to allow routing of those jobs between different hosts.

PBS’s independent scheduling module allows the system administrator to define what types of resources, and how much of each resource, can be used by each job. The scheduling module has full knowledge of the available queued jobs, running jobs, and system resource usage. Using one of several procedural languages, the scheduling policies can easily be modified to suit the computing requirements and goals of any site. The batch system allows a site to define and implement policy as to what types of resources and how much of each resource can be used by different jobs. PBS also provides a mechanism which allows users to specify unique resources required for a job to complete.

A Forerunner of PBS was Cosmic NQS, which was also developed by the NAS program. It became the early standard for batch system under Unix. However Cosmic NQS had several limitations and it was difficult to maintain and enhance.

PBS Supported Provides:

· User commands − for manipulating jobs, i.e. submit, deletion, status, etc.

· Operator commands − for setting up, starting and stopping PBS.

· Administration commands − for managing PBS (setting up, start/stop queues).

· Applications Programming Interface − design applications to run with PBS.

· Resource Management − for manipulating PBS to best use the resources available.

· Job routing − the process of moving jobs from one destination to another.

· Job initiation − the process of selecting and running certain jobs.

· File staging − process of managing disk space and moving files to and from the system executing the job.

· Job tracking − monitoring the status of jobs submitted remotely.

· History File − log maintained by PBS of job resource usage statistics.

· Error detection and recovery.

· Client software can be installed on all cluster workstations.

3.3.11 THE PROSPERO RESOURCE MANAGER (PRM)

`URL http://nii−server.isi.edu/gost−group/products/prm/`

The Prospero Resource Manager (PRM) [26] supports the allocation of processing resources in large distributed systems, enabling users to run sequential and parallel applications on processors connected by local or wide−area networks. PRM has been developed as part of the Distributed Virtual Systems Project at the Information Sciences Institute of the University of Southern California.

PRM enables users to run sequential or parallel jobs on a network of workstations. Sequential jobs may be off loaded to lightly loaded workstations while parallel jobs can make use of a collection of workstations. PRM supports the message passing libraries CMMD and a PVM interface (V3.3.5). PRM also supports terminal and file I/O activity by its tasks, such as keyboard input, printing to a terminal or access to files that may be on a filesystem not mounted by the host on which the task is running. Further more, the components of an application may span multiple administrative domains and hardware platforms, without imposing the responsibility of mapping individual components to nodes on the user.

PRM selects the processors on which the jobs will run, starts the job, supports communication between the tasks that make up the job, and directs input and output to and from the terminal and files on the user's workstation. At the job level, location transparency is achieved through a dynamic address translation mechanisms that translate task identifiers to physical workstation addresses.

PRM's resource allocation functions are distributed across three entities: the system manager, the job manager, and the node manager. The system manager controls access to a collection of processing resources and allocates them to jobs as requested by job managers. Large systems may employ multiple system managers, each managing a subset of resources. The job manager is the principal entity through which a job acquires processing resources to execute its tasks. The job manager acquires resources from one or more system managers and initiates tasks on these workstations through the node manager. A node manager runs on each workstation in the PRM environment. It initiates and monitors tasks on the workstation on which it is running.

3.3.12 QBATCH

URL
http://gatekeeper.dec.com/pub/usenet/comp.sources.misc/volume2

QBATCH is a queued batch processing system for UNIX. Each queue consists of a file containing information about the queue itself, and about all jobs currently present in the queue. When the program `qp` is run for a given queue, it will fork a child process for each job in the queue in turn, and wait for it to complete. If there are no jobs present in the queue, `qp` will wait for a signal from one of the support programs, which will 'tell' it that another job has joined the queue, or that it should terminate.

Features:

There can be as many queues as the system can support. Queues are named, and run asynchronously. The processing of jobs running in one queue are totally independent of those in any other (subject of course to the independence of potentially shared resources such as data files and devices).

· Queues can be started and stopped independently.

· The priority of all jobs running are defined when the queue is created.

· Jobs running in a queue consist of shell scripts. The jobs run with the `uid` and `gid` of the submittor, and run in the environment ruling when the job was submitted.

· `stdout` and `stderr` of running jobs is redirected to a 'monitor'. This will normally be the queue monitor, but may be specified for individual jobs.

· The status and contents of queues can be listed at any time.

· If `qbatch` is configured in the system `init` script, all queues which were running when the system went down will be restarted. Any jobs present in the queues will also be restarted.

Facilities:

− Inform user when job is complete.

− Cancel jobs in queues, or kill running jobs.

− Change the running order of jobs in queues.

− Suspend and resume processing of jobs.

− Start and stop the queue process engine at any time.

− Sizable submissions to a queue, and re−enable them.

− Stop a queue, and wait until the current job has finished.

− List the status and contents of a queue.

− Repeat (or kill and repeat) the currently running job in a queue.

− Examine or edit jobs in a queue, and examine the monitors.

− Monitor (and perhaps account) the times taken by jobs in the queue.

− Prevent CPU hogging jobs from swamping the system.

− Separate queue for large jobs that is started and stopped by `cron` out of working hours.

− Protect sensitive applications and data with file and directory permissions.

Chapter 4

4. Assessment of Evaluation Criteria

In this chapter we compare the evaluation criteria laid out in chapter 2 with each CMS package.

4.1 Commercial Packages

Num.	Package	Vendor
	1. Codine − Computing in Distrib. Network Envn.	GENIAS GmbH, Germany
	2. Connect:Queue	Sterling Corp., USA
	3. Load Balancer	Unison Software, USA
	4. Load Leveler	IBM Corp., USA
	5. LSF − Load Sharing Facility	Platform Computing, Canada
	6. NQE	Craysoft Corp., USA
	7. Task Broker	Hewlett−Packard Corp.

4.1.1 CODINE

Functionality	Supported	Comments
Commercial/Research	Commercial	Fully supported commercial Genias GmbH
Cost	Yes	There are academic and comme
User Support	Yes	tel. & Email − seems to be ve
Heterogeneous	Yes	
Platforms	Most Major	Convex, Cray, DEC, HP, IBM,
Operating Systems	Most Major	ConvexOS, UniCOS, OSF/1, Ul AIX, Irix, SunOS and S
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	
Parallel Support	Yes	PVM and Express
Queue Types	Yes	
Dispatching policy	Yes	Configurable.
Impact on Cluster Users	Yes	Can be set up to minimis
Impact on Cluster w/s	Yes	If process Migration and chec
Load Balancing	Yes	No details.
Check Pointing	Yes	Optional − Relink code wi libraries
Process Migration	Yes	Optional
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	BSD Signals suppor
Resource Administration	Yes	Yes, via a master sch
Job Runtime Limits	Yes	
Forked Child Management	No	No migration or checkpointi processes.
Job Scheduling Priority	Yes	
Process Management	Yes	Runtime limits and exclusiv
Job Scheduling Control	Yes	
GUI/Command−line	Yes	
Ease of Use	Unknown	
User Allocation of Job	Yes	
User Job Status Query	Yes	

Job Statistics	Yes	
Runtime Configuration	Yes	Dynamic
Dynamic Resource Pool	Yes	
Single Point of Failure	Yes	Being worked upon
Fault Tolerance	Some	Master scheduler will resch
Security Issues	Yes	Normal Unix − but seems to thought out.

Contact : Andreas Schwierskott

GENIAS Software GmbH

Erzgebirgstr. 2

D−93073 Neutraubling , Germany

Tel: +49 9401 9200−33

Email: andy@genias.de

Systems Supported:

DEC Alpha − OSF/1 v1.3, v2.x, v3.0, DEC SMM − OSF/1 v3.0

HP − HP−UX v9.0

IBM − AIX 3.2.5, IBM SP1, SP2 − AIX 3.2.5

INTEL − LINUX

SGI − IRIX v4.0.5, v5.x, SGI Power Challenge − IRIX v6.0

SUN Sparc − SunOS 4.1, Solaris 2.1, SUN SMM − Solaris 2.1

Upon request, the software can be ported to other platforms.

4.1.2 CONNECT:QUEUE

Functionality	Supported	Comments
Commercial/Research	Commercial	Sterling Software − Dall
Cost	?	
User Support	Yes	By Sterling
Heterogeneous	Yes	
Platforms	Most Major	Data General, Digital, IB Amdahl, Meiko, and
Operating Systems	Most Major	DG/UX, OSF/Ultrix, Aix/ECS, U SunOS/Solaris
Additional	No	

Hardware/Software		
Batch Jobs	Yes	
Interactive Support	No	Planned for the future but w interactive i/o
Parallel Support	Yes	PVM and Linda
Queue Types	Yes	Multiple queue typ
Dispatching policy	Yes	CPU/Memory/Queue ba
Impact on w/s Owner	Yes	Owner has no control over w
Impact on Cluster w/s	Yes	CPU/Memory/Swap−sp
Load Balancing	Yes	CPU/Memory/Queue ba
Check Pointing	No	Planned for the fut
Process Migration	No	Planned for the fut
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	No	
Resource Administration	Yes	
Job Runtime Limits	Yes	
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	Need to configure whole
Job Scheduling Control	Yes	
GUI/Command−line	Both	GUI for user and admini
Ease of Use	Unknown	
User Allocation of Job	Limited	User can only specify wh
User Job Status Query	Yes	
Job Statistics	Yes	
Runtime Configuration	Yes	Administrator can modify que policies on "fly
Dynamic Resource Pool	Yes	Probably yes
Single Point of Failure	Minimised	Multiple master scheduler
Fault Tolerance	Yes	Scheduler will attempt to r after a failure
Security Issues	Yes	Uses normal Unix sec

Contact: Peter Johnson

Sterling Software

Communications Software Division

5215 North O’Conner Boulevard, Suite 1500

Irving, TX 75039

Tel: +1 (800) 700 5599

Email: connect@sterling.com

Systems Supported:

Data General − DG/UX

DEC − OSF/1 & Ultrix

IBM − Aix/ESA

HP − HP−UX

SGI − Irix

Amdahl − UTS

Sun − Solaris & SunOS

4.1.3 LOAD BALANCER

Functionality	Supported	Comments
Commercial/Research	Commercial	Unison Software Solu
Cost	POA	
User Support	Yes	Email/Tel
Heterogeneous	Yes	
Platforms	Most Major	Sun, HP, IBM, DEC,
Operating Systems	Most Major	Ultrix, HP−UX, Aix, Irix, Su
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	
Parallel Support	No	
Queue Types	Multiple	
Dispatching policy	Multiple	Numerous factors can be taken
Impact on w/s Owner	Configurable	User have rights over t
Impact on Cluster w/s	Yes	CPU/Memory/Swap
Load Balancing	Yes	Tries to distribute load
Check Pointing	Yes	
Process Migration	Yes	
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	
Resource Administration	Yes	Numerous factors can be taken
Job Runtime Limits	Yes	
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	Exclusive CPU access av
Job Scheduling Control	Yes	
GUI/Command−line	Both	
Ease of Use	Unknown	
User Allocation of Job	Yes	Submit to one of many
User Job Status Query	Yes	
Job Statistics	Yes	
Runtime Configuration	Yes	

Dynamic Resource Pool	Yes	Master scheduler check res regularly
Single Point of Failure	Yes	New release will overcome multiple master sched
Fault Tolerance	Partially	If machine crashes, scheduler re−run Job.
Security Issues	Yes	Unix security

Contact:

Unison Software

9−11 Vaughn Road, Harpenden

Hertfordshire AL5 4HU

United Kingdom

Tel: +44 1582 462424

Email: −

URL
http://www.unison.com/main−menu/contacts/mailsupport.html

Systems Supported:

IBM − Aix

Sun − SunOS & Solaris

DEC − Ultrix

HP − HP−UX

SGI − Irix

4.1.4 LOADLEVELER

Functionality	Supported	Comments
Commercial/Research	Commercial	IBM Kingston v 1.2
Cost	POA	
User Support	Yes	IBM type − depends on leve
Heterogeneous	Yes	
Platforms	Most Major	SP, RS/6000, Sun SPARC, SGI HP 9000
Operating Systems	Most Major	AIX 4.1.3, SunOS 4.1.2, Sola 5.2, HP−UX 9.01

Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	Only on Spx, planned for c future release
Parallel Support	Yes	Only on Spx, planned for c future release − P
Queue Types	Multiple	
Dispatching policy	Configurable	
Impact on w/s Owner	Configurable	User determines when their available
Impact on Cluster w/s	Yes	CPU/RAM/Swap
Load Balancing	Yes	based on fair and optimal use
Check Pointing	Yes	Need to relink with LL l
Process Migration	Yes	Need to relink with LL l
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	
Resource Administration	Yes	Administrator can config
Job Runtime Limits	Yes	
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	Can have exclusive CPU
Job Scheduling Control	Yes	
GUI/Command−line	Both	NQS compatible
Ease of Use	Unknown	
User Allocation of Job	Yes	
User Job Status Query	Yes	
Job Statistics	Yes	
Runtime Configuration	Yes	
Dynamic Resource Pool	Yes	
Single Point of Failure	Limited	Master schedulers − but checkpointing
Fault Tolerance	Limited	Schedulers will try and re
Security Issues	Yes	Normal Unix Securi

Contact: Joe Banas

IBM UK Ltd

1 New Square, Bedfont Lakes

Feltham

Middlesex TW14 8HB, UK

Tel: +44 181 818 4000

Email: banas@vnet.ibm.com

Systems Supported:

IBM RS/6000 & SP − AIX 4.1.3

Sun Sparc − SunOS 4.1.2 & Solaris 2.3

SGI − IRIX 5.2

HP Apollo 9000 Series 700 − HP−UX 9.01

4.1.5 LSF

Functionality	Supported	Comments
Commercial/Research	Commercial	Platform Computing C
Cost	POA	
User Support	Yes	
Heterogeneous	Yes	
Platforms	Most Major	Convex, Digital, HP, IBM,
Operating Systems	Most Major	ConvexOS, Ultrix, HP−−UX, Ai SunOS and Solari
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	
Parallel Support	Yes	PVM, P4, TCCMSG, DSM an
Queue Types	Multiple	
Dispatching policy	Configurable	Satisfy job resource an requirements
Impact on w/s Owner	Yes	LSF will migrate jobs on a
Impact on Cluster w/s	Yes	CPU/RAM/Swap
Load Balancing	Yes	Tries to distributed work across cluster
Check Pointing	No	Planned in future rel
Process Migration	Yes	
Job Monitoring and Rescheduling	Yes	Migration of job
Suspension/Resumption of Jobs	Yes	
Resource Administration	Yes	
Job Runtime Limits	Yes	
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	Exclusive CPU usage po
Job Scheduling Control	Yes	
GUI/Command−line	Both	
Ease of Use	Unknown	
User Allocation of Job	Yes	Limited
User Job Status Query	Yes	
Job Statistics	Yes	
Runtime Configuration	Yes	
Dynamic Resource Pool	Yes?	Not sure but seems li

Single Point of Failure	No	Master scheduler re−electe schedulers
Fault Tolerance	Yes	jobs restarted
Security Issues	Yes	Normal Unix, trusted hosts support

Contact:

Platform Computing Corporation

5001 Yonge St., #1401

North York, ON, M2N 6P6

Canada

Tel: +1 (416) 512−9587

Email: info@platform.com

Systems Supported:

CONVEX − C−Series systems running ConvexOS 10 or 11

Digital − Ultrix 4.2 − 4.4, ALPHA/AXP − Digital UNIX 2.x & 3.x

HP 9000/300 − HP−UX 8.x, 9.x 9000/700 systems & 9000/800 systems 9.x & 10.x

IBM RS/6000 − AIX 3.2.x & 4.1

SGI − IRIX 4.x, 5.2, 5.3 & 6.01

SUN − SunOS 4.1.x, Solaris 2.3 & 2.4

4.1.6 NQE

Functionality	Supported	Comments
Commercial/Research	Commercial	CraySoft Corp.
Cost	Yes	$2875 per 10−user network −
User Support	Yes	email, tel, etc
Heterogeneous	Yes	
Platforms	Most Major	SPARC, IBM RS/6000, SGI, Digi and Cray Research
Operating Systems	Most Major	Solaris, SunOS, Aix, Irix, HP−UX & Unicos (lat
Additional Hardware/Software	No	
Batch Jobs	Yes	

Interactive Support	Yes	
Parallel Support	Yes	PVM
Queue Types	Multiple	Batch and pipe que
Dispatching policy	Yes	Configurable by adminis
Impact on w/s Owner	Configurable	w/s owner can configure
Impact on Cluster w/s	Yes	CPU/RAM/Swap
Load Balancing	Yes	
Check Pointing	No	Possible Cray if NQS av
Process Migration	No	Possible Cray if NQS av
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	
Resource Administration	Yes	
Job Runtime Limits	Yes	
Forked Child Management	No	Only under UniCO
Job Scheduling Priority	Yes	
Process Management	Yes	Workstation can be configur exclusive CPU
Job Scheduling Control	Yes	
GUI/Command−line	Both	WWW interface
Ease of Use	Unknown	
User Allocation of Job	Yes	To one of a number of
User Job Status Query	Yes	
Job Statistics	Yes	
Runtime Configuration	Yes	
Dynamic Resource Pool	Yes	
Single Point of Failure	No	Client forwards jobs to mast multiple schedule
Fault Tolerance	Some	Jobs not sent to queues not
Security Issues	High	Has US DoD B1 security ratin Multi−level securi

Contact:

Cray Research, Inc.

655 Lone Oak Drive

Eagan, Minnesota 55121

Tel: +1 (612)452−6650

Email: craysoft@cray.com

Systems Supported:

Sun − Solaris & SunOS

SGI − IRIX

IBM - AIX

HP - HP-UX

DEC - OSF/1

Cray - UNICOS

4.1.7 TASK BROKER

Functionality	Supported	Comments
Commercial/Research	Commercial	
Cost	Yes	Doc & Media = $315, 10 w/s L
User Support	Yes	tel/email - depends on how
Heterogeneous	No	Version supplied by HP only platforms.
Platforms	One	HP - 3rd Party versions a
Operating Systems	One	HP-UX
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Limited	X supported for I/O, stdin/ special w/s configur
Parallel Support	No	
Queue Types	Many	Configurable Queu
Dispatching policy	Yes	
Impact on w/s Owner	Yes	Can be achieved, but need to configuration fil
Impact on Cluster w/s	Yes	CPU/RAM/Disk
Load Balancing	Yes	Scheme where each w/s "bid
Check Pointing	No	
Process Migration	No	
Job Monitoring and Rescheduling	Yes & No	No automatic reschedulin
Suspension/Resumption of Jobs	Yes	Supports all BSD sig
Resource Administration	Yes	Local and global configura
Job Runtime Limits	No	No built in
Forked Child Management	No	See above...
Job Scheduling Priority	Yes	
Process Management	Yes	w/s can be configured to prov or shared access to
Job Scheduling Control	Yes	
GUI/Command-line	Both	Motif based GUI
Ease of Use	Unknown	
User Allocation of Job	Yes	
User Job Status Query	Yes	
Job Statistics	Yes	Admin and user stati
Runtime Configuration	Yes	Need to "edit" configuration w/s - static
Dynamic Resource Pool	Yes	Need to "edit" local configur

		add/withdrawn w/s
Single Point of Failure	No	Schedulers distributed − j guaranteed to compl
Fault Tolerance	Yes	Jobs are not guaranteed to
Security Issues	Yes	Normal Unix

Contact: Terry Graff

Hewlett−Packard Corp.

300 Apollo Drive

Chelmsford

Ma 01824 − USA

Tel: +1 (508) 436 5931

Email:

Systems Supported:

HP Task Broker runs on HP 9000 Series 300, 400, 600, 700, and 800 computers running the HP−UX operating system, and the HP Apollo workstations DN2500, DN3500, DN4500,

DN5500, and DN10000 running Domain/OS. In addition, Scientific Applications International Corporation (SAIC) has ported Task Broker to the Sun3, Sun4, and SPARCstation platforms.

4.2 Research Packages

Num.	Package	Vendor	V
	1. Batch	UCSF, USA	
	2. CCS − Computing Centre Software	Paderborn, Germany	
	3. Condor	Wisconsin State University, USA	
	4. DJM − Distributed Job Manager	Minnesota Supercomputing Center	
	5. DQS 3.x	Florida State University, USA	
	6. EASY	Argonne National Lab, USA	
	7. far	University of Liverpool, UK	
	8. MDQS	ARL, USA	
	9. Generic NQS	University of Sheffield, UK	
	10. Portable Batch System	NASA Amass & LLNL, USA	
	11. PRM − Prospero Resource	University of Southern	

Manager	California
12. QBATCH	Vita Services Ltd., USA

4.2.1 BATCH

Functionality	Supported	Comments
Commercial/Research	Research	Unclear, IPR seems to belo
Cost	PD	GNU Types licens
User Support	Yes	email Author
Heterogeneous	Yes	
Platforms	Many	IBM RS/6000, Sparc, MIP
Operating Systems	Many	AIX 3.x, IRIX 3.x, 4.x, & 5. and Ultrix 4.1
Additional Hardware/Software	No	NFS
Batch Jobs	Yes	
Interactive Support	Yes	
Parallel Support	No	
Queue Types	Yes	Multiple Queues
Dispatching policy	Configurable	
Impact on w/s Owner	Controllable	
Impact on Cluster w/s	Some	Uses w/s CPU, memory and
Load Balancing	Some	Load balanced queu
Check Pointing	No	
Process Migration	No	
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	Start/suspend Job
Resource Administration	Some	
Job Runtime Limits	Yes	
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Some	Unix "nice"
Job Scheduling Control	Yes	
GUI/Command−line	Yes	Based on Forms library, Job removal.
Ease of Use	Unknown	
User Allocation of Job	Some	Control over Jobs on own w
User Job Status Query	Yes	
Job Statistics	Yes	
Runtime Configuration	Some	
Dynamic Resource Pool	Unknown	
Single Point of Failure	Unknown	Multiple queues, but proba administrative dae
Fault Tolerance	No	Not mentioned in docume
Security Issues	Some	Seems to be normal

Contact: Scott Presnell

Zymo Genetics Inc., 1201 Eastlake Ave. E, Seattle, WA, USA

Tel: +1 (206) 442 6751

Email: srp@zgi.com

Systems Supported:

IBM − AIX 3.x,

SGI − IRIX 3.x, 4.x, and 5.x,

Sun − SunOS 4.x

DEC − Ultrix 4.1.

4.2.2 COMPUTING CENTER SOFTWARE

Functionality	Supported	Comments
Commercial/Research	Research	Paderborn Center for Parall
Cost	Unknown	Enquire
User Support	Yes	tel./email from Paderborn Parallel Computing (Ge
Heterogeneous	No	Based around Parsytec
Platforms	Parsytec	Sparc, Parsytec G
Operating Systems	Parix	SunOS, Parix
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	
Parallel Support	Yes	
Queue Types	Yes	
Dispatching policy	Yes	Static configuration, dy allocated by user
Impact on w/s Owner	N/A	
Impact on Cluster w/s	Yes	Resources, once allocated to not be released2
Load Balancing	No	
Check Pointing	No	
Process Migration	No	
Job Monitoring and Rescheduling	No	Dependent upon the
Suspension/Resumption of Jobs	Yes	
Resource Administration	Yes	
Job Runtime Limits	Yes	
Forked Child Management	N/A	
Job Scheduling Priority	Yes	Administrator
Process Management	No	
Job Scheduling Control	Yes	Administrator configures

GUI/Command−line	Command−line	
Ease of Use	Moderate	
User Allocation of Job	Yes	
User Job Status Query	Yes	
Job Statistics	Some	Not extensive
Runtime Configuration	Yes	Dynamically configur
Dynamic Resource Pool	Yes	Resources dynamically co
Single Point of Failure	Yes	Port and queues mana
Fault Tolerance	No	Jobs need to be re−sub
Security Issues	Yes	Normal Unix securi

Contact: Friedhelm Ramme

Paderborn Center for Parallel Computing

University of Paderborn

Fuerstenallee 11

33102 Paderborn Germany

Tel: +49 5251 60 6322

Email: ram@uni−paderborn.de

Systems Supported:

Frontend : Sun − Solaris & SunOS

Backend: Parsytec − GCel

4.2.3 CONDOR

Functionality	Supported	Comments
Commercial/Research	Academic	Wisconsin State Unive
Cost	PD	
User Support	Some	tel/email + email maili
Heterogeneous	Yes	
Platforms	Most Major	DEC, HP, IBM, Sequent, SG
Operating Systems	Most Major	OSF/1, Ultrix, HP−UX, AIX, SunOS and Solari
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	?	
Parallel Support	No	PVM support plann
Queue Types Dispatching policy	Yes	
Impact on w/s Owner	Yes	Controllable

Impact on Cluster w/s Load Balancing	Yes	Checkpoints to local
Check Pointing	Yes	Code must be relinked with Co
Process Migration	Yes	Code must be relinked with Co
Job Monitoring and Rescheduling		
Suspension/Resumption of Jobs		
Resource Administration	Yes	
Job Runtime Limits	Yes	
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	
Job Scheduling Control	Yes	
GUI/Command−line	Command−line	
Ease of Use	unknown	
User Allocation of Job	Yes	
User Job Status Query	Yes	
Job Statistics	Yes	
Runtime Configuration	Yes	
Dynamic Resource Pool	Yes	But only when machines fall added to the Condor
Single Point of Failure	Yes	At the master schedule
Fault Tolerance	Yes	Restart job at last che
Security Issues	Yes	Tries to maintain normal Un

Contact: Miron Livny

University of Wisconsin

1210 W. Dayton St.

Madison, WI 53706−1685

Tel: +1 (608) 262−1204

Email: miron@cs.wisc.edu

Supported Systems

SUN SparcStation − SunOS 4.1.3

DEC Alpha − OSF/1 1.3

DEC DecStation − ULTRIX 4.3

HP HP700 (snake) − HPUX 9

IBM R6000 − AIX 3.2

INTEL − LINUX.

4.2.4 DJM

Functionality	Supported	Comments
Commercial/Research	Research	Minnesota Supercomputin
Cost	PD	
User Support	Limited	
Heterogeneous	No	
Platforms	Limited	Front−end − Sun’s and SGI’s, and CM−5
Operating Systems	Limited	
Additional Hardware/Software	CM−X	
Batch Jobs	Yes	
Interactive Support	Yes	I/O redirected if running in batch
Parallel Support	Yes	
Queue Types	Multiple	Queues Jobs to run on CM
Dispatching policy	Yes	No. of factors, including partition loading, time in
Impact on w/s Owner	No	
Impact on Cluster w/s	No	
Load Balancing	Yes	
Check Pointing	No	
Process Migration	No	
Job Monitoring and Rescheduling	Yes	Restart job
Suspension/Resumption of Jobs	Yes	Restart job
Resource Administration	Yes	Partition manage
Job Runtime Limits	Yes	To groups and users per
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	
Job Scheduling Control	Yes	
GUI/Command−line	Command−line	NQS type interfac
Ease of Use	Unknown	
User Allocation of Job	Limited	Direct job to a qu
User Job Status Query	Yes	Command−line
Job Statistics	Yes	Limited
Runtime Configuration	Yes	Partitions and queues chang
Dynamic Resource Pool	Yes	partitions can be recovered
Single Point of Failure	Yes	Queue manager runs on w/s dependent on partitions
Fault Tolerance	Yes	Will restart job − as lon manager remains act
Security Issues	Yes	Normal Unix securi

Contact:

Minnesota Supercomputer Center, Inc.

1200 Washington Avenue South

Minneapolis, MN 55415

Tel: +1 (612) 337−0200 (Main)

Email: salesinfo@msc.edu (E−mail)

Systems Supported:

CRAY T3D − UNICOS MAX 1.x

Thinking Machines Corporation CM−5 − CMOST 7.x

4.2.5 DQS 3.X − SUPERSEDES DQS 2.X AND DNQS

Functionality	Supported	Comments
Commercial/Research	Research	Supercomputer Computation Institute (SCRI) at
Cost	PD	
User Support	Yes	email and tel. support from S lists
Heterogeneous	Yes	
Platforms	Most Major	Digital, Intel, HP, SGI, I
Operating Systems	Most Major	OSF/Ultrix, Linux, UX, Irix and Solaris
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	Via BSD sockets
Parallel Support	Yes	PVM, P4 and P5
Queue Types	Multiple	Queue complexes defined by a
Dispatching policy	Yes	By queue and weighted
Impact on w/s Owner	Minimal	Can be configured not to aff all
Impact on Cluster w/s	Yes	Especially if checkpointin
Load Balancing	Yes	Configurable − CPU memory, j
Check Pointing	Yes	Must be relined with DQS
Process Migration	No	
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	Configurable
Resource Administration	Yes	
Job Runtime Limits	Yes	
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	Will allow exclusive CP
Job Scheduling Control	Yes	

GUI/Command−line	Both	
Ease of Use	Unknown	
User Allocation of Job	Yes	Based on queues us
User Job Status Query	Yes	GUI
Job Statistics	Yes	
Runtime Configuration	Yes	By root or designated DQS ma
Dynamic Resource Pool	Yes	On the "fly" resource
Single Point of Failure	Minimal	Multiple instances of Maste
Fault Tolerance	Yes	Tries to complete job af
Security Issues	Yes	Normal Unix + AFS/Ker

Contact: Joe Pasko

400 Dirac Science Center, FSU Tallahassee,

FL 32306, USA

Tel:

Email: pasko@scri.fsu.edu

Systems Supported:

DEC Alpha − OSF/1, Ultrix 4.2

Intel I486 − Linux

HP − HP−UX

SGI − Irix4.0.5, 5.1.1

Next − NeXT3.2

IBM RS6000 − Aix3.2

Sun Sparc − SunOs 4.1.3 & Solaris 2.3

4.2.6 EASY

Functionality	Supported	Comments
Commercial/Research	Research	Argonne National Labo
Cost	PD	
User Support	Limited	Email support by author, permits. + mailer g
Heterogeneous	Limited	Written in Perl, should be e
Platforms	Limited	IBM (SP1 & SP2) and DE
Operating Systems	Limited	Aix and OSF/1
Additional	No	Will work with AF

Hardware/Software		
Batch Jobs	Yes	
Interactive Support	Yes	I/O from batch jobs is deliv at end of job − actual inter
Parallel Support	Yes	MPL, PVM, P4 and MPI − wil support any interf
Queue Types	Single	Configurable
Dispatching policy	Yes	Configurable
Impact on w/s Owner	Yes	Only normal Unix (n
Impact on Cluster w/s	Yes	CPU/Memory/Swap
Load Balancing	No	
Check Pointing	No	
Process Migration	No	
Job Monitoring and Rescheduling	Yes/No	Jobs that fail are not re
Suspension/Resumption of Jobs	No	
Resource Administration	Yes	By administrator − conf
Job Runtime Limits	Yes	Runtime limits are put on a to the resources allo
Forked Child Management	No	Not applicable as user has access to node.
Job Scheduling Priority	Yes	
Process Management	No	Each user has exclusive ac nodes allocated to
Job Scheduling Control	Yes	
GUI/Command−line	Command−line	GUI planned
Ease of Use	Unknown	
User Allocation of Job	Limited	Direct job to a qu
User Job Status Query	Yes	Command−line
Job Statistics	Yes	Limited − extensions p
Runtime Configuration	Yes	Queue needs to be turned off, and turned back o
Dynamic Resource Pool	Yes	
Single Point of Failure	Yes	Scheduler need to write to
Fault Tolerance	No	Jobs need to be reschedul failure.
Security Issues	Yes	Normal Unix securi

Contact: David Lifka

Cornell Theory Center

Frank H.T. Rhodes Hall

Hoy Road, Cornell University

Ithaca, New York 14853−3801

Tel: +1 (607) 254−8686

Email: lifka@tc.cornell.edu

Systems Supported:

IBM SP1 and SP2

DEC Alpha Farm

4.2.7 FAR

Functionality	Supported	Comments
Commercial/Research	Research	University of Liverpo
Cost	PD	
User Support	Yes	email/tel. − UK JISC fund
Heterogeneous	Some	Only implemented on Sun’s
Platforms	Several	Sun − SGI and HP (b
Operating Systems	Several	SunOS and Solaris Irix and
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	BSD Sockets
Parallel Support	Yes	NAS−HPF & PVM
Queue Types	No	
Dispatching policy	Yes	Automatic or manu
Impact on w/s Owner	Configurable	Console user has priority − jobs killed by defa
Impact on Cluster w/s	Yes	CPU/RAM/diskspac
Load Balancing	Limited	Based on w/s loads − can be ridden.
Check Pointing	No	Deliberately omitt
Process Migration	No	Deliberately omitt
Job Monitoring and Rescheduling	No	User initiated (man
Suspension/Resumption of Jobs	No	User initiated (man
Resource Administration	Some	Static database
Job Runtime Limits	No	Normal Unix can be invoke
Forked Child Management	No	
Job Scheduling Priority	No	
Process Management	No	
Job Scheduling Control	No	
GUI/Command−line	Command−line	
Ease of Use	Unknown	
User Allocation of Job	Yes	
User Job Status Query	Yes	User initiated (man
Job Statistics	No	Administrator can add Unix st not standard
Runtime Configuration	Limited	Editing far database
Dynamic Resource Pool	Limited	Required editing master
Single Point of Failure	Yes	Master daemon
Fault Tolerance	No	When Master deamon dies, far
Security Issues	Yes	Normal Unix

Contact: Steve Morgan

Computing Services

University of Liverpool

PO Box 147

Abercromby Square

LIVERPOOL

L69 3BX, UK

Tel: +44 151 794 3746

Email: j.s.morgen@liv.ac.uk

Systems Supported:

Sun − SunOS & Solaris

HP − HP−UX (beta)

SGI − Irix (beta)

4.2.8 GENERIC NQS

Functionality	Supported	Comments
Commercial/Research	Research	University of Sheff
Cost	PD	GNU License
User Support	Yes	Supported by the University (JISC−NTI) until Jul
Heterogeneous	Yes	
Platforms	Most Major	IBM, Fujitsu, HP, SGI, Inte DEC & Cray
Operating Systems	Most Major	AIX, UXP/M, HP−UX, IRIX, Lin SunOS, Ultrix, OSF/1 &
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	NQS nets to be configure stdin/err to file
Parallel Support	No	Not yet.
Queue Types	Yes	One on each server, mini configuration.
Dispatching policy	Static	Each queue knows about its performance

Impact on w/s Owner	Yes	Queues can be day/night sens can "nice" jobs
Impact on Cluster w/s	Yes	CPU/RAM//Diskspac
Load Balancing	Static	Master scheduling (option), perf. and load at each
Check Pointing	No	
Process Migration	No	
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	Supports normal Unix s
Resource Administration	Yes	Normal Unix
Job Runtime Limits	Yes	Normal Unix
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	Manage the number of jobs run
Job Scheduling Control	Yes	
GUI/Command−line	Command−line	WWW based interface pl
Ease of Use	Unknown	
User Allocation of Job	Yes	
User Job Status Query	Yes	
Job Statistics	Yes	Amount depends on the platfor
Runtime Configuration	Yes	Dynamic
Dynamic Resource Pool	Yes	Dynamic
Single Point of Failure	Yes/No	Yes if master scheduler, configured with "peer"
Fault Tolerance	Yes	Queue will try run and compl crash
Security Issues	Yes	Normal Unix

Contact: Stuart Herbert

Academic Computing Services,

University of Sheffield, UK

Tel: +44 − 114 282 4254

Email: S.Herbert@sheffield.ac.uk

Systems Supported:

IBM − AIX 3 & 4

Fujitsu − UXP/M 10

HP − HP−UX 8, 9 & 10

SGI − IRIX 4, 5 & 6

Intel − Linux/ELF

NCR − UNIX

Sun − Solaris 2.x & SunOS 4.x

DEC − ULTRIX & OSF/1

Cray − UNICOS v8

4.2.9 MDQS

Functionality	Supported	Comments
Commercial/Research	Research	Ballistics Research Lab
Cost	PD	
User Support	Some	
Heterogeneous	Yes	
Platforms	Some	Sun Sparc +??
Operating Systems	Some	SunOS 4.1.x, BSD 4.2, SY
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	Redirected I/O and soc
Parallel Support	No	
Queue Types	Multiple	
Dispatching policy	Unknown	
Impact on w/s Owner	Unknown	
Impact on Cluster w/s	Normal	
Load Balancing	Probably	Master scheduler probably do
Check Pointing	No	
Process Migration	No	
Job Monitoring and Rescheduling	unknown/Yes	On failure job will be re
Suspension/Resumption of Jobs	Yes	BSD signals
Resource Administration	Yes	Via multiple queu
Job Runtime Limits	Probably	No known if enforc
Forked Child Management	No	
Job Scheduling Priority	Probably	
Process Management	Unknown	
Job Scheduling Control	Probably	
GUI/Command−line	Command−line	
Ease of Use	Unknown	
User Allocation of Job	Yes	
User Job Status Query	Probably	
Job Statistics	Probably	
Runtime Configuration	Unknown	
Dynamic Resource Pool	Unknown	
Single Point of Failure	Yes	
Fault Tolerance	Yes	On reboot, system will attem jib.
Security Issues	Yes	Normal Unix

NOTE: Unable to find, or get hold of, further information about MDQS from BRL.

Contact: Mike Muuss

US Army Ballistics Research Laboratory

Aberdeen Proving Ground,

Maryland, 21005−5067,

USA

Tel:

Email: mike@arl.mil

Systems Supported:

Sun − SunOS 4.1.1

Various − BSD 4.2

Various − Sys III & V

4.2.10 PORTABLE BATCH SYSTEM (PBS)

Functionality	Supported	Comments
Commercial/Research	Research	NASA Ames & LLNL
Cost	PD	
User Support	Yes	NASA Ames − email/tel/mai
Heterogeneous	Yes	
Platforms	Multiple Platforms	Sun, SGI, IBM, Intel, Thinki Cray
Operating Systems	Multiple OS/s	SunOS, Solaris, Aix, Intel−O Unicos
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	stdin/stderr via Xt
Parallel Support	Yes	On Parallel machines − i
Queue Types	Multiple	Definable by administ
Dispatching policy	Configurable	
Impact on w/s Owner	Yes	Configurable − mouse/keyboa available
Impact on Cluster w/s	Yes	CPU/RAM/Swap
Load Balancing	Yes	
Check Pointing	No	Vendor specific − reliant on
Process Migration	No	

Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	
Resource Administration	Yes	
Job Runtime Limits	Yes	
Forked Child Management	No	
Job Scheduling Priority	Yes	
Process Management	Yes	Configure for exclusive
Job Scheduling Control	Yes	
GUI/Command−line	Command−line	Tcl/Tk planned for next
Ease of Use	Unknown	
User Allocation of Job	Yes	To specific queu
User Job Status Query	Yes	
Job Statistics	Yes	
Runtime Configuration	Yes	
Dynamic Resource Pool	Yes	
Single Point of Failure	Unsure	Unsure about master sche
Fault Tolerance	Minimised	Restart jobs after failure about crashes
Security Issues	Yes	Normal Unix trusted clients type authorisatio

Contact : Dave Tweten

NASA Ames Research Center

Tel : +1 415−604−4416

Email: tweten@nas.nasa.gov

Systems supported:

− In production at NAS − IBM SP−2, SunOS 4, IRIX 4, 5, 6 and AIX 3

− Testing at NAS − UNICOS 7 and 8

− No Longer Available − Thinking Machines CM−5

− Being Ported − Cray T3D, HP−UX SunOS 5 and FreeBSD 2

4.2.11 PRM

Functionality	Supported	Comments
Commercial/Research	Research	Information Sciences Institut of S. California
Cost	PD	Free to non−commercial sit Agreement)

User Support	Yes	email/tel/WWW
Heterogeneous Platforms	Yes but ! Sun & HP	Only supports two pla Sun and HP
Operating Systems	Yes	SunOS and HP−UX
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	I/O redirected to users
Parallel Support	Yes	CMMD, PVM, and Express − M
Queue Types	Yes	
Dispatching policy	No	
Impact on w/s Owner	No	Configurable + migrate proce
Impact on Cluster w/s	Yes	Diskspace and Netw
Load Balancing	No	
Check Pointing	Yes	Taken from Condo
Process Migration	Yes	Taken from Condo
Job Monitoring and Rescheduling	Yes/No	No automatic reschedu
Suspension/Resumption of Jobs	Some	Processes can be suspended i
Resource Administration	Yes	Through "system manager"
Job Runtime Limits	No	None imposed
Forked Child Management	No	
Job Scheduling Priority	No	
Process Management	Yes/No	Yes, but when w/s owner retu be suspended and mig
Job Scheduling Control	No	
GUI/Command−line	command−line	
Ease of Use	Unknown	
User Allocation of Job	Yes	Using job configuratio
User Job Status Query	Yes	
Job Statistics	Some	
Runtime Configuration	Yes	
Dynamic Resource Pool	Yes	
Single Point of Failure	Yes/No	Failed resources will be acqu system managers
Fault Tolerance	No	
Security Issues	Yes	Normal Unix, plans for Ker authentication

Contact: Santosh Roa

Distributed Virtual Systems Project,

Information Sciences Institute,

University of Southern California.

4676 Admiralty Way

Marina Del Rey

CA, 90292, USA

Tel: +1 (310) 822−1511

Email: info−prospero@isi.edu

Systems supported :

Sun − SunOS 4.1.x

HP 9000/7xx − HP−UX version 9

4.2.12 QBATCH

Functionality	Supported	Comments
Commercial/Research	Research	Alan Saunders
Cost	PD	
User Support	None	
Heterogeneous	Yes	Limited multi−platform
Platforms	Several	Sun, DEC and IBM
Operating Systems	Several	SunOS, Ultrix and
Additional Hardware/Software	No	
Batch Jobs	Yes	
Interactive Support	Yes	Send to queue monitor, b reconfigured.
Parallel Support	No	
Queue Types	Multiple	One or more queues per
Dispatching policy	Static	Configured when queue is
Impact on w/s Owner	Yes	Queue can be configured with
Impact on Cluster w/s	Yes	CPU/Memory/Diskspa
Load Balancing	Static	Seems necessary send jobs queues
Check Pointing	No	
Process Migration	No	
Job Monitoring and Rescheduling	Yes	
Suspension/Resumption of Jobs	Yes	Normal BSD signals sup
Resource Administration	Static	Configured at startup
Job Runtime Limits	Yes	Unix limits suppor
Forked Child Management	No	
Job Scheduling Priority	Yes	Prioritise queue
Process Management	Probably	
Job Scheduling Control	Yes	
GUI/Command−line	command−line	
Ease of Use	unknown	
User Allocation of Job	Yes	
User Job Status Query	Yes	
Job Statistics	Yes	Amount unknown, but probably

		accounting.
Runtime Configuration	Static	
Dynamic Resource Pool	Static	
Single Point of Failure	No	Multiple queues − each kno about the others
Fault Tolerance	Yes	Machines crashes jobs in q
Security Issues	Yes	Normal Unix

Contact: Alan Saunders

Vita Services Ltd.

Source `ftp://gatekeeper.dec.com/pub/usenet/comp.sources.misc/volume25`

Systems supported:

Sun − SunOS 4.1.x

DEC − Ultrix 4.0

IBM − AIX 3.1.5

4.3 A Discussion About the CMS Packages Reviewed

4.3.1 INTRODUCTION

The aim of this section is to narrow down the list of possible packages by eliminating those that do no meet the required criteria set out in Chapter 2. It should be noted that not all the packages have been installed and extensively tested,

4.3.2 COMMERCIAL PACKAGES

Two packages can be eliminated from the list immediately. If the support of parallel jobs is seen to be an highly desirable feature of a CMS package, neither Load Balancer or Task Broker support this feature. In addition, Task Broker does not provide heterogeneous system support

The remaining five systems: Codine, Connect:Queue, LoadLeveler, LSF and NQE, are functionally very similar. These five packages all support parallel jobs, but, at present LoadLeveler only supports parallel jobs on IBM SP2 systems rather than workstation clusters, thus eliminating it from our shortlist. The other packages all support PVM but none mention future support for MPI and HPF.

If checkpointing is deemed to be important then Connect:Queue, LSF and NQE are eliminated from the shortlist. The authors of this report take the view that checkpointing is a useful additional feature rather than being highly desirable.

If security is a major issue then only LSF and NQE have security features over and above normal Unix features. Both packages support Kerboros type user authentication, but NQE additionally has a US DoD security rating.

Of the remaining packages, Connect:Queue, LSF and NQE all use multiple master schedulers to minimise problem associated with Single Points of Failure − and a similar feature is planned for the next release of Codine. All the packages claim to maximise their resilience and fault tolerance.

Conclusion

Four packages remain on our shortlist (Codine, Connect:Queue, LSF and NQE) out of the original seven. It is difficult to reduce the list further without installing and detailed testing of the functionality, robustness and stability of each package. However, an additional consideration is the cost of a site license and software maintenance support for each package.

4.3.3 RESEARCH PACKAGES

Two packages, CCS and DJM, do not support heterogeneous systems. These packages are primarily targetted at MPP systems (Parsytec, Cray T3D and Thinking Machines CM−5), and have not been ported or tested on heterogeneous workstation clusters. Four packages (EASY, **`far`**, MDQS and PRM) presently only support a limited number of vendor platforms. Of these four, EASY is written in Perl and would be relatively simple to install and run on other platforms.

Five packages (Batch, Condor, GNQS, MDQS and Qbatch) do not support parallel jobs which these authors deem to be highly desirable. Support for parallel jobs is planned in both Condor and GNQS.

Three packages, EASY, **`far`** and PRM have limited functionality under the Job Scheduling and Allocation Policy section (see section 2.4). EASY, at present, has no concept of load balancing, however, it is planned in a future release. **`far`** is able to accomplish some load balancing, but it appears to be rather crude (based on `rup`) and its usefulness on a large diverse cluster would be limited. PRM has no dispatching or load balancing capability, even though it is capable of migrating processes. All three packages are incapable of suspending, resuming or rescheduling jobs without intervention at the processes level by a systems administrator.

Conclusion

Two packages remain on the selection list (DQS and PBS) out of the original twelve. It is difficult to reduce the list further without actually installing and practically testing the functionality, robustness and stability of each package. A key factor in choosing between these two remaining packages would be the maturity of the software, how widespread is its usage and how much user support could be expected from the authors or supporting site.

If sequential jobs, rather than parallel ones, are the overriding concern then Condor and GNQS should be the packages of choice. GNQS in particular is a mature, robust and widely used package. Support for it can be found at a number of sites, including CERN and the University of Sheffield.

4.3.4 OVERALL CHOICE

In this section we have reduced choice of CMS package down from the original nineteen to six − four commercial and two public domain. If finances permit, it would be wise to choose one of the commercial packages. The main reason for this decision is that it should minimise the amount of time and effort that is expended by staff on a site to understand, set up, install and run the package. The onus will be on the vendors to ensure your site has a smooth passage with the software. The actual choice of which commercial package to buy will probably be based on price performance, as each vendor is trying sell software that provides the same cluster services.

Chapter 5

5. Cluster Computing Environments

Num.	Package	Vendor
	1. Amoeba	Vrije Universiteit, Amsterdam
	2. Beowulf	NASA, USA
	3. BSP	Oxford Parallel, UK
	4. DCE − Distribute Computing Environment	OSF, USA
	5. DOME − Distributed Object Migration Envn.	Carnegie Mellon, USA
	6. GLU	Stanford Research Lab. USA
	7. LAM	Ohio Supercomputer Center, USA
	8. Networks Of Workstations (NOW)	Berkeley, USA
	9. Shrimp	Princeton, USA
	10. Thesis	MCC, UK
	11. WANE	SCRI, FSU, USA

Introduction

The aim of this chapter is to provide a brief description of each of the packages, listed in the table above. The description consists of information has been coalesced from user guides and on−line (WWW) documents. The software packages described in this chapter consist of ones that can be described as Cluster Computing Environments (CCE), it also contains descriptions of two CMS packages which are at the early stages of development (NOW and WANE) as well as the Parallel Virtual Machine (PVM) environment.

5.1 Amoeba

URL http://www.cs.vu.nl/vakgroepen/cs/amoeba.html

Amoeba is a general purpose distributed operating system developed and supported by Vrije Universiteit in Amsterdam [28]. Amoeba is designed to take a collection of machines and make them act together as a single integrated system. In general users are unaware of the number and location of the processors that run their commands, nor of the number and location of the file servers that store their files.

Amoeba is an on−going research project and should be regarded as a platform for doing research and development in distributed and parallel systems, languages, protocols and applications. Amoeba provides some Unix emulation and has a Unix−like feel, but it is not a Unix replacement.

Amoeba was designed around a micro−kernel architecture − here each machine runs a kernel that supports the basic process, communications and objective primitives. It also handles raw I/O and memory management. Everything else is built on top of these fundamentals.

A typical Amoeba system will consist of three fundamental classes of machines. First, each user has a workstation for running the user interface, based on the X Windows system. Second, there exists a pool of processors that are dynamically allocated to users as required. Third, there are specialised servers, such as file servers ands directory servers that run all the time. All these components are connected via a fast LAN − at present only Ethernet is supported.

5.2 Beowulf

URL
http://cesdis.gsfc.nasa.gov/pub/people/becker/beowulf.html

Beowulf [29] is a project to produce the software to control workstations based on commodity PC−class hardware and the Linux operating system. Beowulf couples the low cost, moderate performance of commodity PC subsystems with the emergence of *de facto* standards in message passing hardware and software, utilising local file storage capacity and bandwidth. This experimental system is driven by the requirements of NASA Earth and Space science applications including data assimilation, data set browsing and visualisation, and simulation of natural physical systems.

The Beowulf parallel workstation architecture comprises PC processor subsystems, each with a disk, controller and 16 Mbytes of DRAM. The processing elements are interconnected by two parallel Ethernet networks with a peak capacity of 10 Mbps per network. Two of the PCs have separate Ethernet interfaces to external LAN's for remote access and data transfer. Two additional subsystems have high resolution video controller/drivers, one of these also provides a keyboard and mouse interface for conventional single−user workstation access.

Most of the currently running parallel applications are written using PVM to communicate, others use RPC. Future work is expected to include:

· MPI support.

· Implementing a Network Virtual Memory system − like distributed shared memory.

· A distributed I/O file server.

5.3 The Oxford BSP Library

URL http://www.comlab.ox.ac.uk/oucl/oxpara/bsplib1.html

The Oxford BSP library provides a mechanism for developing portable and predictably−efficient code across a wide range of distributed platforms. The programming model is based on the Bulk Synchronous Parallel (BSP) architectural model [30 & 31]. This abstract general model parameterises system characteristics in order to realise performance across distributed platforms without tying application code to a specific architecture. Predicting performance and determining optimal parallel strategies for applications are based on a corresponding cost model.

The library [32] consists of a small number of subroutines to implement process creation, remote data access, and bulk synchronisation. The library can be linked to programs written in standard sequential languages such as Fortran, C, and Pascal.

Portable Interface − Single−source Application Code

The library offers the same interface and semantics to all applications. The specification of the library routines is portable, but their implementation is not necessarily so. Machine−specific instances of the library can be created to utilise the disparate facilities for parallelism native to each environment. Thus, the library is tuned for a machine rather than individually for the program, thereby achieving single−source application codes.

The BSP library embodies a static SPMD (Single Program, Multiple Data) programming model, in which every parallel process executes the same program text. Processes proceed together through a sequence of supersteps, all processes must reach the end of the superstep before anyone can proceed to the next. Each process has its own private data space; there is no globally shared data, but a process can explicitly access the variables of any other process. Remote data access is asynchronous; but is not guaranteed to be satisfied until the end of the current superstep, when all the processes are synchronised.

Predicting performance of programs is based on system parameters which characterise processor speed, bulk−synchronisation time and global communication bandwidth. The first prototype of the library was built to use either the PVM or PARMACS message−passing library, this implementation was portable but not particularly efficient. Higher−performance generic versions have been implemented using TCP/IP sockets (for workstation clusters), and System V shared−memory primitives (for shared−memory multiprocessors). For the highest performance, there are native library versions for specific machines such as the IBM SP1/SP2, SGI Power Challenge, Meiko CS−2 and Cray T3D.

5.4 Distributed Computing Environment (DCE)

URL http://www.osf.org/dce/index.html

The OSF Distributed Computing Environment (DCE) [33] is a comprehensive, integrated set of services that supports the development, use and maintenance of distributed applications. It provides a uniform set of services, anywhere in the network, enabling applications to utilise the power of a heterogeneous network of computers.

The DCE aims to be operating system and network−independent, providing compatibility with users’ existing environments. DCE includes a comprehensive set of distributed services which provides commercial−quality, inter−operable distributed computing environment.

Distributed Computing Environment Overview

The architecture of DCE tries to mask the physical complexity of the networked environment and provides a layer of logical simplicity for the user. This layer is composed of a set of services that can be used, separately or in combination, to form a DCE. The services are organised into two categories:

Fundamental Distributed Services (FDS) provide tools for software developers to create the user application needed for distributed computing. They include:

· Remote Procedure Call.

· Directory Service.

· Time Service.

· Security Service.

· Threads Service.

Data−Sharing Services provide users with capabilities built upon the FDS. These services require no programming on the part of the user and facilitate better use of information. They include:

· Distributed File System.

· Diskless Support.

DCE and the Desktop

DCE allows the processing power inherent in a network to be extended to large numbers of nodes. DCE extends the power of the network − and all its resources − to individual users on PCs and Macintosh computers. As a result, PCs and Macintosh machines interconnected by LANs or network servers are no longer isolated. With DCE, they become trusted peer−level client systems that can reach out to servers for information and processing services.

Operating System and Network Independence

DCE can be used on a variety of networks and operating systems by system vendors, software developers or end−users. It can be used with any network hardware and transport software, including TCP/IP, OSI, X.25, and other similar products.

The DCE is a portable implementation, written in standard C, which makes use of standard interfaces for operating system services, such as POSIX and X/Open guidelines. It can be ported easily to OSF/1, AIX, DOMAIN OS, ULTRIX, HP−UX, SINIX, SunOS, and UNIX System V operating systems.

5.5 Dome

`URL http://parsys.cs.cmu.edu/dome/Dome.html`

Dome (Distributed Object Migration Environment) [34 & 35] is being used to develop Grand Challenge applications, such as molecular dynamics, nuclear physics, distributed simulation, gene sequencing, and speech recognition, on heterogeneous networks of workstations. Dome makes it possible to take advantage of multicomputers (workstation networks), MPPs (massively parallel processors), advanced operating systems, and gigabit networks being developed under the US HPCC program.

Dome provides this capability through libraries of distributed objects (i.e., scalable software) which can be used to program heterogeneous networks of computers as a single resource, obtaining performance that cannot be achieved on the individual machines.

Some Features of Dome:

· Addresses the problems of load balancing, ease of programming, and fault tolerance.

· Objects distribute themselves automatically over multiple computers.

· Attempts to load balance the applications being run.

· Supports mechanisms to recover from resource failures.

Dome is available on eight platforms: DEC Alpha OSF/1, HP HP−UX, Intel Paragon, Sparc SunOS, SGI Irix, DEC Ultrix, IBM AIX, and Cray C90 Unicos.

Other Features of Dome

Adaptive Collective Communications − A technique for performing collective communications operations is incorporated in Dome. The technique automatically derives network topology and uses this information to optimise collective operations. This is achieved by choosing message routing schemes that efficiently map to the underlying network.

Checkpointing − Dome has architecture independent checkpoint and restart to include arbitrarily deep call sequences, making the mechanism more flexible. The current system allows checkpoints to be initiated from almost anywhere in a call sequence.

Load balancing − Dome uses dynamic load balancing techniques to include both global and local load information.

Future work:

· Further development of the adaptive collective communications − based on the MPI collective communications operations

· Develop new distributed object classes and applications.

· Implement a streamlined distributed I/O interface that utilises the Scotch Parallel File system.

5.6 GLU Parallel Programming System

URL http://www.csl.sri.com/GLU.html

GLU (Granular Lucid) [36] is a high−level programming system for constructing parallel and distributed applications to run on diverse high−performance computing systems. GLU is based on a hybrid programming model that combines multi−dimensional dataflow (Lucid) with imperative models. GLU enables rapid parallel program development using existing sequential code, it helps produce programs that are portable, efficient, and fault tolerant.

The GLU is toolkit is for constructing and executing high−performance applications on distributed systems including heterogeneous workstation clusters (over WAN, LAN and wireless networks), multiprocessors, and massively parallel processors. The GLU toolkit consists of a high−level application description notation, a compiler, a graphical application execution user−interface, and a comprehensive performance analysis tool. GLU can be used to develop new parallel applications or convert sequential, legacy, applications to parallel equivalents.

The GLU toolkit has been used to develop high performance applications from existing sequential ones, where the original applications were written in C, Fortran and C++. The GLU system is currently available for the following systems:

· *Workstations:* Sparc/Solaris 1.1 and 2.3, SGI/IRIX v4.0.5F, IBM RS6000/AIX v3.2, DEC AXP/OSF1 v1.3 and x86/Linux v1.1.58

· *Symmetric Multiprocessors:* SparcMP/Solaris 1.1 and SGI ONYX/IRIX v4.0.5F

· *Massively Parallel Processors:* CM5/SunOS 4.1.3

See URL http://www.csl.sri.com/Lucid.html for a description of the Lucid language.

5.7 Local area Multicomputing (LAM)

URL http://www.osc.edu/lam.html

LAM [37] runs on each computer as a single Unix daemon structured as a micro−kernel and hand−threaded virtual processes. It provides a simple message−passing and rendezvous service to local processes. Some of the in−daemon processes form a network communication subsystem, which transfers messages to and from other LAM daemons on other machines. The network subsystem adds features like packetisation and buffering to the base synchronisation. Other in−daemon processes are servers for remote capabilities, such as program execution and parallel file access. Users can configure in or out services as necessary.

The features of LAM are transparent to users and system administrators, who only see a conventional Unix daemon. System developers can de−cluster the daemon into a daemon containing only the nano−kernel and several full Unix client processes. The network layer in LAM is a documented, primitive and abstract layer on which to implement a more powerful communication standard like MPI (PVM has also been implemented).

An important feature of LAM is hands−on control of the multicomputer. There is very little that cannot be seen or changed at runtime. Programs residing anywhere can be executed anywhere, stopped, resumed, killed, and watched the whole time. Messages can be viewed anywhere on the multicomputer and buffer constraints tuned as experience with the application dictates. If the synchronisation of a process and a message can be easily displayed, mismatches resulting in bugs can easily be found. These and other services are available both as a programming library and as utility programs run from any shell.

LAM installs anywhere and uses the shell's search path at all times to find LAM and application executables. A cluster is specified as a simple list of machine names in a file, which LAM uses to verify access, start the environment, and remove it.

MPI Implementation

The four main MPI synchronisation variables: context, tag, source rank, destination rank are mapped to LAM's abstract synchronisation at the network layer. MPI debugging tools interpret the LAM information with the knowledge of the LAM/MPI mapping and present detailed information to MPI programmers. A significant portion of the MPI specification is being implemented completely within the runtime system and independent of the underlying environment. MPI programs developed on LAM can be moved without source code changes to any other platform that supports MPI.

5.8 Networks Of Workstations (NOW)

`URL http://now.cs.berkeley.edu/`

The Berkeley NOW project [38] is building system support for using a network of workstations (NOW) to act as a distributed supercomputer on a building−wide scale. It aims to utilise volume produced commercial workstations and switch−based networks, such as ATM, to show that NOW can be used for both the tasks traditionally run on workstations and large parallel programs.

In conjunction with complementary research efforts in operating systems and communication architecture, it is planned that 100 processor system will be demonstrated during the project lifetime and show that it will have better cost−performance for parallel applications than a massively parallel processing (MPP) machine, and better performance for sequential applications than an individual workstation. To attain these goals the NOW project is combining elements of workstation and MPP technology into a single system. The NOW project includes research and development into network interface hardware, fast communication protocols, distributed file systems, distributed scheduling and job control. The NOW system will consist of the following five packages:

- *Active Messages* − This interface generalises previous active messages interfaces to support a broader spectrum of applications such as client/server programs, file systems, operating systems, as well continuing support for parallel programs.
- *GLUnix* − the Unix co−ordinator − The operating system must support gang−scheduling of parallel programs, identify idle resources in the network (CPU, disk capacity/bandwidth, memory capacity, network bandwidth), allow for process migration to support dynamic load balancing, and provide support for fast inter−process communication for both the operating system and user−level applications. GLUnix is built as a layer on top of existing operating systems and will provide the functionality mentioned above.
- *Fast communication implementations* − The system needs a collection of low−latency, parallel communication primitives. These will include extensions of current ones, including sockets, remote procedure calls, and general messaging primitives on top of Active Messages. These communications layers will provide the link between the large existing base of MPP programs and fast but primitive communication layers.
- *xFS* − A serverless, distributed file system which attempts to have low latency, high bandwidth access to file system data by distributing the functionality of the server among the clients. The typical duties of a server include maintaining cache coherence, locating data, and servicing disk requests. The function of locating data in xFS is distributed by having each client responsible for servicing requests on a subset of the files. File data is striped across multiple clients to provide high bandwidth.
- *Network RAM* − On a fast network with Active Messages, fetching a page from a remote machine’s main memory can be more than an order of magnitude faster than getting the same amount of data from the local disk. The system will be able to utilise the memory on idle machines as a paging devices for busy machines. This memory, called Network RAM, will appear as an additional memory hierarchy between the local RAM and the paging space on disk. The Network RAM system being designed is serverless, in that there is no dedicated Network RAM. Any machine can be a server when it is idle, or a client when it needs more memory than the physical memory it has.

5.9 Parallel Virtual Machine (PVM)

URL http://www.epm.ornl.gov/pvm/

PVM [39] is an integrated set of software tools and libraries that emulates a general−purpose, flexible, heterogeneous concurrent computing framework on interconnected computers of varied architecture. The overall objective of the PVM system is to enable such a collection of computers to be used cooperatively for concurrent or parallel computation. The principles upon which PVM is based include the following:

· User−configured host pool: A PVM application run on a set of machines that are selected by the user. Both shared and distributed−memory computers may be part of the host pool.

· Definable access to hardware: A PVM application views the hardware environment as an attributeless collection of virtual processing elements or user specified to take advantage of different machine capabilities.

· Process−based computation: The unit of parallelism in PVM is a an independent sequential thread of control that alternates between communication and computation. Multiple tasks may execute on a single processor.

· Explicit message−passing model: Collections of computational tasks, each performing a part of an application's workload using data, functional, or hybrid decomposition, cooperate by explicitly sending and receiving messages to one another.

· Heterogeneity support: The PVM system supports heterogeneity in terms of machines, networks, and applications.

· Multiprocessor support: PVM uses the native message−passing facilities on multiprocessors to take advantage of the underlying hardware. Vendors often supply their own optimized PVM for their systems, which can still communicate with the public PVM version.

The PVM system is composed of two parts. The first part is a daemon , that resides on all the computers making up the virtual machine. The daemon is designed so any user with a valid login can install this daemon on a machine. When a user wishes to run a PVM application, he/she first creates a virtual machine by starting up PVM. The PVM application can then be started from a Unix prompt on any of the hosts. Multiple users can configure overlapping virtual machines, and each user can execute several PVM applications simultaneously.

The second part of the system is a library of PVM interface routines. It contains a functionally complete repertoire of primitives that are needed for cooperation between tasks of an application. This library contains user−callable routines for message passing, spawning processes, coordinating tasks, and modifying the virtual machine.

5.10 The SHRIMP Project

`URL http://www.cs.princeton.edu/shrimp/`

SHRIMP (Scalable, High−Performance, Really Inexpensive Multi−Processor) [40] is a parallel machine being designed and built in the Computer Science Department at Princeton University. Shrimp is built from highly−integrated, commodity parts. The computing nodes of SHRIMP are Pentium PCs, and the routing network is the same one used in the Intel Paragon. A network interface card that connects the PCs to the routing network and the software to make SHRIMP a fully usable multicomputer is being designed and implemented.

SHRIMP uses an innovative model of communication called *virtual memory mapped communication* (see [38] for full description). This has several advantages over existing systems: it allows communication directly from user−level without violating protection, it enables very−low−overhead implementations of message−passing, and it is only a little bit harder to design and build than conventional DMA−based interfaces.

5.11 The THESIS Distributed Operating System Project

`URL http://info.mcc.ac.uk/MultiComp/dosp.html`

The aim of the THESIS (Transparent HEterogeneous Single Image System) [41 & 42] project is to exploit the fact that many commercial operating systems are now becoming micro−kernel based (i.e. using Mach), in order to create heterogeneous clusters of workstations. These workstations would be largely unchanged systems, requiring the replacement of existing pagers with distributed pagers, the introduction of distributed process managers and also translation mechanisms to enable the migration of text and data between heterogeneous processors.

The primary problems in the design of a distributed operating system for workstations which must be addressed are:

· Network latency. The advantages of the resources gained through collaboration must be offset against the overhead of the increase in communication times between machines.

· Coherency. The workstations must conspire together to produce the illusion of a single virtual memory.

· Heterogeneity. The collaborating machines will probably not all be code or data compatible. The translation of data and the use of ‘fat8’ binaries can be used to address this problem, but every effort must be made to avoid their costly use at runtime.

· Fault tolerance. Distributing a process and its memory over a number of machines increases its vulnerability to host and network failures. Provision for migrating processes away from failing or heavily loaded machines should be made.

5.12 WANE

URL http://www−wane.scri.fsu.edu/

WANE is a comprehensive Wide Area Networked Environment being developed at the Supercomputer Computations Research Institute as part of a three year project funded by the US Department of Energy. It is designed to be a highly scalable and fault tolerant computing and information suite. WANE draws heavily from existing and ongoing software projects and attempts to integrate them into a single entity. Some of the key technologies exploited by WANE include:

· DQS − CPU and device scheduling (batch and interactive).

· PostGres − database accesses.

· Tcl/Tkl − X−Window development.

· Mosaic/WWW/gopher − internet information services.

· FreeNet − information services.

The Wane Server

The WANE Server software package is a "turnkey" solution targeted at a diverse audience, which includes commercial, governmental, educational, and individual users. The system is designed to support thousands of users, is incrementally scalable, and fault tolerant. The package is modular, with services ranging from a simple mail or WWW server to a full fledged free net service package. The server package contains:

· Extensive user access controls.

· Hierarchical administrative levels.

· Multiple user domains.

· Optional support of separate administrative groups, allowing delegation of administrative tasks.

· User friendly graphical interface for user administration.

· A complete distribution of Linux.

· Multi−user operating system on inexpensive PC clones.

· Mail Hub service.

· World Wide Web multimedia server software.

· Internet connection tools/software.

· Internet Newsgroups.

· Comprehensive suite of interactive personal communication tools (Text, Audio, Video).

· Access to vast archives of software and data.

Client Software:

The client software includes a variety of public domain and shareware packages enabling users to connect to the Internet via modem or LAN. The WANE client distribution provides support for MacintoshOS, DOS, Windows, Windows NT, OS/2, and Linux.

Chapter6

6. Conclusions

6.1 General Comments

The information collected and presented in this review is just a "snap−shot" of the CMS and CCE packages available and found, via the Internet, during the Summer of 1995. Without actually physically setting up, installing and testing these software packages on a number of different platforms it is impossible to objectively determine which is the best amongst them. Even if there was the time to do so, a number issues, such as ease of use, configurability, user support, etc., would also need to be assessed, making the whole exercise rather subjective and thus difficult to quantify.

Software for managing clusters and environments that utilise clusters have become very popular in the last decade, as this review can verify. Anyone taking a brief look through this review will quickly establish that a large percentage of the packages described are interesting research projects that will probably go nowhere. But work on these projects will feed into other projects which will further our knowledge and understanding of the problems and needs associated with cluster computing. The projects described in the Chapter 5 under CCE are not the main focus of this review and are therefore only mentioned in passing in this chapter.

The importance of CMS packages for the commercial world can be readily seen by the fact that most of the major computer vendors are now involved, or support, one or more of the packages available. It is also clear that the limited finances of many institutions have forced them to reassess how their present computing facilities are being used and is making them look at ways of utilising their resources more efficiently and effectively. There is clear evidence that both commercial and research/academic communities are becoming increasingly interested in CMS.

It is not clear that CMS is actually being used to take advantage of spare CPU cycles, but it is evident that large efforts are being expended in trying to increase the throughput on networks of workstations by load balancing the work that needs to be done. Thus increasing the overall throughput of the cluster.

An aim of this report has been to highlight the issues to be considered when choosing a CMS. This knowledge can then be cross−checked against the features provided by each CMS package. Choosing, the best CMS package for a particular site, is not the easiest task. A pragmatic look at a sites needs and a profile of typical applications that run must be made. The administrator of a site can then use these to best judge which package would be the most appropriate for his/her site.

It is clear that, without testing, just because a package supports a particular feature, it does not necessarily mean that its functionality will meet the demands of the administrator of a site. For example, a package may support parallel jobs, but it may be impossible to configure the queues to sensibly run the jobs − due to the need to wait for sufficient resources or for numerous other reasons.

In Chapter 4.3 the nineteen CMS packages were assessed by comparing their functionality against the authors highly desirable criteria list. This narrowed the list down to six, two public domain and four commercial packages. It was recommended that If finances permit, it would be wise to choose one of the commercial packages. The reason for this choice was that this path should minimise the amount of time and effort that is expended by staff on a site to understand, set up, install and run the package. The onus will be on the vendors to ensure a site has a smooth passage with the their software. It was also stated that the final decision as to which package to purchase would be an economic one, as each vendor is trying sell software that provides the same cluster services and therefore price performance would be the deciding factor.

6.2 Ommissions in the CMS Packages

The CMS looked at in this review exhibited a number of rather important omissions, namely:

− Limited support for Linux − so the ability to utilise PC’s is limited.

− No support for Microsoft Windows−95 or NT.

− No support for Apple Macintosh computers.

− Only one, NQE from CraySoft Corp., has a WWW interface .

− No mentions of support for MPI.

− Only one package supports HPF (**far** from the University of Liverpool).

− No package has the ability to manage forked child processes.

6.3 A Step−by−Step Guide to Choosing a CMS package

1. Assess the needs of your site − class them as mandatory, desirable and useful.

2. Produce a list of commonly used packages and applications that would be typical jobs on your designated cluster.

3. Cross match the chosen criteria (see 1.) against the ones shown for the CMS packages being assessed.

4. Contact the authors or vendors of the packages you have come up with and ask them for further details and contacts at reference sites.

5. Read and digest details.

6. Compose a list of questions to ask the reference site and contact reference site − maybe visit for demonstration and to ask further questions.

7. Compose a list of questions for the authors/vendors and contact them.

8. When satisfied with a particular package, negotiate a demonstration software license for 30−60 days.

9. Install software on a designated test cluster − make full use of the User support provided to get an idea of their efficiency and effectiveness at solving your problems.

10. Test the CMS package rigorously by running your typical applications (see 2).

11. Test its fault tolerance and configurability.

12. Assess the features, functionality and usability of the CMS and decide whether it fits your needs.

13. Try to test alternative packages if unhappy.

14. Purchase the package that fulfills your sites needs (obviously).

6.4 Some Personal Views

DQS and Codine are probably the most comprehensive, functional and commonly used CMS packages available. Both are available on most major platforms and support sequential and parallel jobs. Codine is a well supported commercial package, whereas DQS is still a research project and the level of support that may be needed cannot be guaranteed.

Many of the commercial CMS packages will provide better user support for one particular vendor, than the other platforms that is supposed to support equally.

Generic NQS, now supported by the University of Sheffield, is a very mature package which has been widely used by numerous large sites − CERN in particular. NQS protocols are also well established − packages that support these protocols exhibit additional functionality and an degree of compatibility.

Many interesting projects are emerging. In particular the resources and effort being put into NOW can only be admired. The NOW project is addressing many of the key areas that are seen as problem areas with CMS.

The WANE project, which is integrating a number of packages into ones environment, has high ambitions and should be worth watching.

There does not seem to be enough emphasis on utilising commodity PCs. It is obvious that these are the most common machines available and that they are a huge untapped source of additional CPU power. Projects involving multi−taking PC operating systems such as Windows−NT and Linux should certainly be encouraged.

6.5 The Future ?

The popularity of the WWW along with the increasingly functional and maturing tools [43] indicates that future successful cluster management systems will be based on this technology.

Presently no integrated WWW−based cluster management systems exists. But the fundamental infrastructure needed to produce such a system is already in−place. It should be a relatively simple exercise to use the WWW as a uniform interface to administer and run applications on a heterogeneous computing environment.

· Communications protocols are established (`http`, `ftp`),

· Distributed management daemons already run on most machines (`httpd`) − including PC and MAC servers.

· Well known user interfaces are already available (`netscape` and `mosaic`).

· A powerful scripting language for these systems is available (Perl5).

· User I/O is simple to set up (forms).

· Other tools are being added on a daily basis.

Finally, it seems likely that the experience learned from the packages reviewed in this document will used to produced a WWW based system in the very near future.

Chapter 7

7. Glossary of Terms

AFS

AFS is a distributed filesystem that enables co−operating hosts (clients and servers) to share filesystem resources across both local area and wide area networks. AFS is based on a distributed file system originally developed at the Information Technology Center at Carnegie−Mellon University that was called the "Andrew File System". AFS is marketed, maintained, and extended by Transarc Corporation.

Amdahl's Law

A rule first formalised by Gene Amdahl in 1967, which states that if F is the fraction of a calculation that is serial and 1−F the fraction that can be parallelised, then the speedup that can be achieved using P processors is: 1/(F + (1−F)/P) which has a limiting value of 1/F for an infinite number of processors. This no matter how many processors are employed, if a calculation has a 10% serial component, the maximum speedup obtainable is 10.

Application Programming Interface (API)

A set of library routine definitions with which third party software developers can write portable programs. Examples are the Berkeley Sockets for applications to transfer data over networks, those published by Microsoft for their Windows graphical user interface, and the Open/GL graphics library initiated by Silicon Graphics Inc. for displaying three dimensional rendered objects.

Bandwidth

The communications capacity (measured in bits per second) of a transmission line or of a specific path through the network. Contiguous bandwidth is a synonym for consecutive grouped channels in multiplexer, switch or DACS; i.e. 256kbps (4 x 64kbps channels).

Batching System

A batching system is one that controls the access to computing resources of applications. Typically a user will send a request the batch manager agent to run an application. The batch manager agent will then place the job (see definition below) in a queue (normally FIFO).

Cluster Computing

A commonly found computing environment consists of many workstations connected together by a local area network. The workstations, which have become increasingly powerful over the years, can together, be viewed as a significant computing resource. This resource is commonly know as cluster of workstations.

Distributed Computing Environment

The OSF Distributed Computing Environment (DCE) [32] is a comprehensive, integrated set of services that supports the development, use and maintenance of distributed applications. It provides a uniform set of services, anywhere in the network, enabling applications to utilise the power of a heterogeneous network of computers.

flops

Floating point operations per second; a measure of memory access performance, equal to the rate which a machine can perform single precision floating−point calculations.

Heterogeneous

Containing components of more than one kind. A heterogeneous architecture may be one in which some components are processors, and others memories, or it may be one that uses different types of processor together.

HPF

A language specification published in 1993 by experts in compiler writing and parallel computation, the aim of which is to define a set of directives which will allow a Fortran 90 program to run efficiently on a distributed memory machine. At the time of writing, many hardware vendors have expressed interests, a few have preliminary compilers, and a few independent compiler producers also have early releases. If successful, HPF would mean data parallel programs can be written portably for various multiprocessor platforms.

Homogeneous

Made up of identical components. A homogeneous architecture is one in which each element is of the same type − processor arrays and multicomputers are usually homogeneous. See also heterogeneous.

Homogeneous vs Heterogeneous

Often a cluster of workstations is viewed as either homogenous or heterogeneous. These terms are ambiguous, as they refer to not only the make of the workstation but also to the operating system being used on them. For example, it is possible to have a homogenous cluster running various operating systems (SunOS and Solaris, or Irix 4 and 5).

In this review we define homogenous as a cluster of workstations of the same make (i.e. Sun, HP, IBM, etc.). In contrast everything else is referred to as heterogeneous, i.e. a mixture of different makes of workstations.

I/O

Refers to the hardware and software mechanisms connecting a computer with its environment. This includes connections between the computer and its disk and bulk storage system, connections to user terminals, graphics systems, and networks to other computer systems or devices.

Interconnection network

The system of logic and conductors that connects the processors in a parallel computer system. Some examples are bus, mesh, hypercube and Omega networks.

Internet Protocol (IP)

The network−layer communication protocol used in the DARPA Internet. IP is responsible for host−to−host addressing and routing, packet forwarding, and packet fragmentation and reassembly.

Interprocessor communication

The passing of data and information among the processors of a parallel computer during the execution of a parallel program.

Job

This term generally refers to an application sent to a batching system − a job finishes when the application has completed its run.

Latency

The time taken to service a request or deliver a message which is independent of the size or nature of the operation. The latency of a message passing system is the minimum time to deliver a message, even one of zero length that does not have to leave the source processor. The latency of a file system is the time required to decode and execute a null operation.

Load balance

The degree to which work is evenly distributed among available processors. A program executes most quickly when it is perfectly load balanced, that is when every processor has a share of the total amount of work to perform so that all processors complete their assigned tasks at the same time. One measure of load imbalance is the ratio of the difference between the finishing times of the first and last processors to complete their portion of the calculation, to the time taken by the last processor.

Memory Hierarchy

Due to the rapid acceleration of microprocessor clock speeds, largely driven by the introduction of RISC technology, most vendors no longer supply machines with the same main memory access speeds as the CPU clock. Instead, the idea of having hierarchies of memory subsystems, each with different access speeds is used to make more effective use of smaller, expensive fast (so called cache) memories. In the context of computer networks, this hierarchy extends to storage devices, such as the local disk on a workstation (fast), to the disk servers, to off−line secondary storage devices (slow).

Message passing

A style of inter−process communication in which processes send discrete messages to one another. Some computer architectures are called message passing architectures because they support this model in hardware, although message passing has often been used to construct operating systems and network software for uni−processors and distributed computers..

Metacomputer

A term invented by Paul Messina to describe a collection of heterogeneous computers networked by a high speed wide area network. Such an environment would recognise the strengths of each machine in the Metacomputer, and use it accordingly to efficiently solve so−called Metaproblems. The World Wide Web has the potential to be a physical realisation of a Metacomputer. Also, see Metaproblem.

Metaproblem

A term invented by Geoffrey Fox for a class of problem which is outside the scope of a single computer architecture, but is instead best run on a Metacomputer with many disparate designs. An example is the design and manufacture of a modern aircraft, which presents problems in geometry, grid generation, fluid flow, acoustics, structural analysis, operational research, visualisation, and database management. The Metacomputer for such a Metaproblem would be networked workstations, array processors, vector supercomputers, massively parallel processors, and visualisation engines.

MIPS

One Million Instructions Per Second. A performance rating usually referring to integer or non−floating point instructions. See also MOPS.

Message Passing

A style of interprocess communication in which processes send discrete messages to one another. Some computer architectures are called message passing architectures because they support this model in hardware, although message passing has often been used to construct operating systems and network software for uniprocessors and distributed computers.

Message Passing Interface (MPI)

The parallel programming community recently organised an effort to standardise the communication subroutine libraries used for programming on massively parallel computers such as Intel’s Paragon, Cray’s T3D, as well as networks of workstations. MPI not only unifies within a common framework programs written in a variety of existing (and currently incompatible) parallel languages but allows for future portability of programs between machines.

Massively Parallel Processing (MPP)

The strict definition of MPP is a machine with many interconnected processors, where ‘many’ is dependent on the state of the art. Currently, the majority of high−end machines have fewer than 256 processors. A more practical definition of an MPP is a machine whose architecture is capable of having many processors − that is, it is scalable. In particular, machines with a distributed memory design (in comparison with shared memory designs) are usually synonymous with MPPs since they are not limited to a certain number of processors. In this sense, "many" is a number larger than the current largest number of processors in a shared−memory machine.

Multicomputer

A computer in which processors can execute separate instruction streams, have their own private memories and cannot directly access one another’s memories. Most multicomputers are disjoint memory machines, constructed by joining nodes (each containing a microprocessor and some memory) via links.

Multitasking

Executing many processes on a single processor. This is usually done by time−slicing the execution of individual processes and performing a context switch each time a process is swapped in or out − supported by special−purpose hardware in some computers. Most operating systems support multitasking, but it can be costly if the need to switch large caches or execution pipelines makes context switching expensive in time.

Network

A physical communication medium. A network may consist of one or more buses, a switch, or the links joining processors in a multicomputer.

NFS

Network Filing System is a protocol developed to use IP and allow a set of computers to access each other’s file systems as if they were on the local host.

Network Information Services − NIS (former Yellow Pages)

Developed by Sun Microsystems, NIS is a means of storing network wide information in central databases (NIS servers), where they can be accessed by any of the clients. Typically, an NIS database will be used to store the user password file, mail aliases, group identification numbers, and network resources. The use of a single server avoids the problem of data synchronisation.

Parallel Job

This can be defined as a single application (job) that has multiple processes that run concurrently . Generally each process will run on a different processor (workstation) and communicate boundary, or other data, between the processes at regular intervals. Typically a parallel job would utilise a message passing interface, such as MPI or PVM, to pass data between the processes.

Process

The fundamental entity of the software implementation on a computer system. A process is a sequentially executing piece of code that runs on one processing unit of the system.

Queuing

Queuing is the method by which jobs are ordered to access some computer resource. Typically the batch manager will place a job the queue. A particular compute resource could possibly have more than one queue, for example queues could be set up for sequential and parallel jobs or short and long job runs.

Remote Procedural Call (RPC)

A mechanism to allow the execution of individual routines on remote computers across a network. Communication to these routines are via passing arguments, so that in contrast to using Sockets, the communication itself is hidden from the application. The programming model is that of the clients−servers.

Sequential computer

Synonymous with a Von Neumann architecture computer and is a "conventional" computer in which only one processing element works on a problem at a given time.

Sequential Job

This can de defined as a job that does not pass data to remote processes. Typically such a job would run on a single workstation − it is possible that for a sequential process to spawn multiple threads on its processor.

Single Point of Failure

This is where one part of a system will make the whole system fail. In cluster computing this is typically the batch manager, which if it fails the compute resource are no longer accessible by users.

Sockets

Also commonly known as Unix Berkeley Sockets, these were developed in the early 1980s as a means of providing application writers a portable means of accessing the communications hardware of the network. Since sockets allow point to point communications between processes, it is used in most of the networked workstation implementations of message passing libraries.

Speedup

The ratio of two program execution times, particularly when the times are from execution on 1 and P nodes of the same computer. Speedup is usually discussed as a function of the number of processors, but is also a function (implicitly) of the problem size.

Supercomputer

A time dependent term which refers to the class of most powerful computer systems world−wide at the time of reference.

Transmission Control Protocol (TCP)

TCP is a connection−oriented transport protocol used in the DARPA Internet. TCP provides for the reliable transfer of data as well as the out−of−band indication of urgent data.

Chapter 8

8. References

[1] D. Hutchinson and P. Lever, *Uses of Computing Clusters for Parallel and Cooperative Computing*, A Report for JISC NTSC, Manchester University, December 1993.

[2] M. Baker, T. Hey and L. Robertson, *Cluster Computing Report*, A Report for the JISC NTSC, The University of Southampton, January 1995.

[3] J.S. Kaplan and M.L. Nelson, *A comparison of Queuing, Cluster and Distributed Computer System*, NASA Langley Research Center, Technical Report, October 1993.

[4] J.S. Kaplan and M.L. Nelson, *A comparison of Queuing, Cluster and Distributed Computer System*, NASA Langley Research Center, Technical Report, June 1994.

[5] L.H. Turcotte, *A Survey of Software Environments for Exploiting Networked Computing Resources*, Mississippi State University Report, June 1993.

[6] G.C. Fox and W. Furmanski, *The Use of the National Information Infrastructure and High Performance Computer in Industry*, NPAC Technical Paper, Syracuse University, USA, July 1995.

[7] W. Saphir, L. Tanner and B. Traversat, *Job Management Requirement for NAS*

Parallel Systems and Clusters NAS Technical Report NAS−95−006, February 1995

[8] M.A. Baker and R.J. Cloke, *Preliminary Evaluation of Batch/Cluster Software*, The University of Southampton, Computing Services, Technical Report, July 1994.

[9] S. Herbert, *Systems Analysis: Batch Processing Systems*, The University of Sheffield, Academic Computing Services, Technical Report, May 1995.

[10] P. Rousselle, S. Hariri and P. Tymann, *Redundant Task Scheduling in a Heterogeneous Distributed Computing Environment*, Department of Electrical and Computer Engineering, Syracuse University, US, Technical Report SCCS 658, 1994.

[11] CODINE: Computing in Distributed Networked Environments, GENIAS Software GmbH on−line −URL `http://www.genias.de/genias/english/codine.html`, September 1995.

[12] J. Suplick, *An Analysis of Load Balancing Technology*, CXSOFT Technical Report, Richardson, Texas, January 1994.

[13] *Introducing NQE − IN−2153 2.0*, Craysoft Publications, Cray Research Inc, May 1995.

[14] S.R. Presnell, *The Batch Reference Guide*, 3rd Edition, Batch Version 4.0, Dept. Pharmaceutical Chemistry, University of California, March 1994.

[15] F. Ramme, T. R˚mke and K. Kremer, *A Distributed Computing Center Software for the Efficient Use of Parallel Computer Systems*, Int. Conf. on High−Performance Computing and Networking, Proceedings of the HPCN Europe, Springer−Verlag 1994, LNCS No. 797, Vol. II, pp 129−136

[16] F. Ramme and K. Kremer, *Scheduling a Metacomputer by an Implicit Voting System*, 3rd IEEE Int. Symp. on High−Performance Distributed Computing, San Francisco, 1994, pp 106−113.

[17] J. Pruyne & M. Livny, *Parallel Processing on Dynamic Resources with CARMI*, Workshop on Job Scheduling Strategies for Parallel Processing, IPPS ’95, April 25, 1995.

[18] M. Livny, *The Condor Distributed Processing System*, Dr Dobbs Journal, Feb 1995 pp 40−48

[19] T.P. Green, *DQS 3.0.2 Readme/Installation Manual*, Supercomputer Computations Research Institute, Florida State University, Tallassee, July 1995 (URL `http://www.scri.fsu.edu/~pasko/DQS.ps`).

[20] D.A. Lifka, M.W. Henderson, and K. Rayl, *Users Guide to the Argonne Sp Scheduling System*, Mathematics and Computer Science Division Technical Memorandum No. ANL/MCS−TM−201, Argonne National Laboratory, USA, May 1995

[21] C.J. Gittings, J.S. Morgan and J. Wetherall, *far − A Tool for Exploiting Space Workstation Capacity*, A draft paper available from `ftp://ftp.liv.ac.uk/pub/far/farpaper.ps`

[22] S. Herbert, *Systems Analysis− Batch Processing Systems*, Academic Computing Services, The University Of Sheffield, UK URL `http://www.shef.ac.uk/uni/projects/nqs/Reports/eval_9410/`

[23] S. Herbert, *Generic NQS − Free Batch Processing For UNIX*, Academic Computing Services, The University Of Sheffield, UK, URL `http://www.shef.ac.uk/uni/projects/nqs/Product/Generic−NQS/v3.`

[24] D.P. Kingston, *A Tour Through the Multi−Device Queueing Systems Revised for MDQS 2.12*, Technical Paper, Ballistics Research Laboratory, USA, February 1989.

[25] R.L. Henderson and D. Tweten, *Portable Batch System: Requirement Specification*, NAS Technical Report, NASA Ames Research Center, April 1995.

[26] B. C. Neuman and S Roa, *The Prospero Resource Manager: A Scalable Framework for Processor Allocation in Distributed Systems*, Concurrency: Practice and Experience, Vol 6(4), 339−355, June 1994.

[27] S. Roa, *Prospero Resource Manager 1.0 User Guide*, Technical Paper, Information Sciences Institute, University of Southern California, 1995.

[28] A.S. Tanebaum, *The Amoeba Distribute Operating System*, Vrije Universiteit Technical Report, URL `ftp://ftp.cs.vu.nl/pub/amoeba/Intro.ps.Z`

[29] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawake and C. V. Packer *Beowulf: A Parallel Workstation for Scientific Computation*, ICPP Paper, August 1995.

[30] W McColl, *Bulk Synchronous Parallel Computing*, 2nd Workshop on

Abstract Machines for Highly Parallel Computing, Univ of Leeds, 14th−16th April 1993.

[31] R Miller, *A Library for Bulk−synchronous Parallel Computing*, British Computer Society Parallel Processing Specialist Group, General Purpose Parallel Computing, December 1993.

[32] R Miller and J Reed, *The Oxford BSP Library: Users' Guide*, version 1.0, Oxford Parallel, 1993.

[33] The OSF Distributed Computing Environment, Overview − URL `http://www.osf.org/comm/lit/OSF-DCE-PD-1090-4.html`

[34] J. Arabe, A. Beguelin, B. Lowekamp, E. Seligman, M. Starkey, and P. Stephan *Dome: Parallel programming in a heterogeneous multi−user environment*, Carnegie Mellon University, March 1995. Submitted for presentation at Supercomputing '95.

[35] E. Seligman and A. Beguelin, *High−Level Fault Tolerance in Distributed Programs*, Technical Report CMU−CS−94−223, School of Computer, Science, Carnegie Mellon University, December 199

[36] *GLU Programmer's Guide v0.9*, SRI CSL Technical Report 94−06, 1994.

[37] LAM Overview − URL `http://www.osc.edu/Lam/lam/overview.html`

[38] T. Anderson, D. Culler, D. Paterson, et. at., *A Case for Networks of Workstations (NOW)*, Computer Science Division, EECS Department, University of California, Berkeley, 1994.

[39] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, PVM: *Parallel Virtual Machine − A Users' Guide and Tutorial for Networked Parallel Computing*, Scientific and Engineering Series, The MIT Press, ISBN 0−262−57108−0

[40] M. A. Blumrich, K. Li, R. Alpert, C. Dubnicki, E. W. Felten, and J. Sandberg, *Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer*, Int'l Symposium on Computer Architecture, April 1994, pp. 142 − 153.

[41] I. McLean, *A Multiple pager approach to distributed memory coherence*. Dec 1994. URL `http://info.mcc.ac.uk/MultiComp/reports.html`

[42] M. Kelly, *The Method Of Imaginary Machines, a Systems' Analysis Tool*.

Dec 1994. URL `http://info.mcc.ac.uk/MultiComp/reports.html`

[43] G.C. Fox, W. Furmanski, M. Chen, C. Rebbi, J. Cowie, *WebWork: Integrated Programming Environment Tools for National and Grand Challenges*, NPAC Technical Report SCCS−715, Syracuse University, June 14 1995.