

# Grid-based High Performance Computing.

R.J. Allan <sup>1</sup>, J.M. Brooke and F. Costen <sup>2</sup> and M. Westhead <sup>3</sup>

<sup>1</sup> Computational Science and Engineering Department,  
CLRC Daresbury Laboratory,  
Daresbury, Warrington WA4 4AD, UK

<sup>2</sup> Manchester Research Centre for Computational Science,  
University of Manchester,  
Oxford Road, Manchester M13 9PL, UK

<sup>3</sup> Edinburgh Parallel Computing Centre,  
University of Edinburgh, JCMB, King's Buildings, Mayfield Road,  
Edinburgh EH9 3JZ, Scotland

May 11, 2000

This is a Technical Report of the UKHEC Collaboration.

Available from [www.ukhec.ac.uk](http://www.ukhec.ac.uk)

© **UKHEC 2000**. Neither the UKHEC Collaboration nor its members separately accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Access to Heterogeneous Resources and Meta-computing Environments</b>	<b>2</b>
<b>3</b>	<b>Computational Grid Environments</b>	<b>4</b>
3.1	LSF . . . . .	4
3.2	UNICORE . . . . .	5
3.3	GLOBUS . . . . .	6
3.4	Seamless Thinking Aid . . . . .	8
3.5	Legion . . . . .	10
3.6	Jini . . . . .	11
<b>4</b>	<b>Relationship between Projects</b>	<b>13</b>
<b>5</b>	<b>Resource Management for Heterogeneous Clusters</b>	<b>13</b>
<b>6</b>	<b>Message-passing Libraries</b>	<b>14</b>
6.1	StaMPI . . . . .	14
6.2	PACX-MPI . . . . .	16
6.3	MPICH-G . . . . .	18
<b>7</b>	<b>SuperComputing'99 Experiment</b>	<b>18</b>
<b>8</b>	<b>Outstanding Issues</b>	<b>18</b>
<b>9</b>	<b>Further Information</b>	<b>19</b>
<b>10</b>	<b>Acknowledgements</b>	<b>19</b>

## 1 Introduction

Grid computing describes the linking together of distributed computational resources to provide flexible access and a common interface for the user. This is sometimes referred to as an “E-services” environment in commercial applications. Meta-computing extends this concept to enable distributed systems to compete with supercomputers in order to try to overcome the limitations of a single computing system. To achieve these goals software systems must be provided which use Internet technology, now common in E-commerce, for the benefit of the computational science community. This has recently been referred to as “E-science”.

Distributed computing systems offer more than just a large CPU resource. A software environment of un-precedented quality and functionality is emerging along with the use of the Internet for commercial and leisure purposes. This is driven by a combination of the computer industry and the loose collection of worldwide “freeware” programmers. Geoffrey Fox has referred to this as the “Distributed Commodity Computing and Information System” [7].

In the USA and Japan there are several alliances of computing centres separated by large distances. In Europe, Germany has taken a lead because of the regional computing centres. In the UK the JREI-funded centres may be (but are not yet) a source of similar resources.

The whole concept is often referred to as a “computational grid”. Computers on a grid can solve very large problems requiring, for instance, more main memory than is available on a single machine. However resources are typically heterogeneous in nature with fast internal message passing or shared memory but a slow wide area network connecting the separate computers. The implementation of a message passing standard (e.g. MPI or PVM) on a meta computer must therefore be able to take into account heterogeneous architectures and hierarchical networks with differing bandwidth and latency in a way transparent to the user. Issues of security, resource management and load balancing are also fundamental to successful exploitation.

A key concept is that of “ownership”. A grid is a “federation” of resources which can be accessed in a transparent way by authenticated grid users. This raises the fundamental question of “accounting” for resource usage, whether it be cpu time, disk, memory, licensed software or preserved data. Whilst this is perhaps the most important issue to be considered in implementing a national grid environment we do not consider it further in this report. Instead we focus on how the scientific user might benefit from such an ideal environment.

Grid-based computing is likely to become an important key technology for future UK High Performance Computing. We give a very brief description of some current international developments which were presented at SuperComputing’99 held in Portland, Oregon and the IEEE Workshop in Cluster Computing held in Melbourne, Australia. Further information is available from larger technical surveys [2, 4] and the book by I. Foster and C. Kesselman [6]. We do not attempt to provide an introduction to all the underlying distributed-computing techniques which are both complex and diverse. There are numerous discussions in the computer-science literature which should be consulted for background information (see e.g. [15, 10, 3]).

In addition to surveying the ongoing world-wide research activities the UKHEC Collaboration is involved in practical experiments to demonstrate the utility of a grid computing environment. CSAR, Manchester has already been actively involved in a meta-computing demonstration which won a prize

at SuperComputing'99 last November. This was summarised in CSAR Focus, edition 4 [1] and is briefly described in Section 7.

## 2 Access to Heterogeneous Resources and Meta-computing Environments

Computational scientists are usually greedy for computer resources. The use of a single very large computer, even if it is of Distributed Memory Multicomputer (DMM) architecture, is limited – more often by cost than by technology. There are however many more modest systems in daily use with a total power far out-weighting the few supercomputers available. It is this fact which has sparked an interest in harnessing distributed resources to create “computational power grids”. Such grids can be used for throughput (accessing idle resources), high performance computing (meta-computing) or data access.

At the departmental level grids are built from workstations or PC “commodity” clusters which would otherwise be infrequently used but can be harnessed to obviate the need to purchase mid-range servers. Many universities and commercial companies have hundreds of PCs or workstations which could be coupled in this way. At the national level meta computers may be built by collaborating computer centres or university research groups with mid-range resources. Activities at the international level involve government laboratories and large national facilities.

Distributed software tools, and especially those which facilitate very complex “coupled” applications to be constructed and used are likely to be of growing interest over the coming few years. They are however difficult to implement, and it is likely that data management or throughput services will be more common in the short term.

Some of the technology gaps which have become focus areas for computational grid and related research are:

- execution environments that are portable and scalable;
  - resource management
  - data storage and movement
  - security and authorisation
  
- tools that enable the use of the execution environment;
  - automated tools for porting legacy code
  - collaborative problem solving environments for complex scientific and engineering tasks to extend the capacity of teams
  - formal, portable programming paradigms, languages and tools that express parallelism and support software synthesis and re-use
  
- development and execution environments to support applications of the future;
  - application software that can make use of up to 10,000 processors
  - numerical algorithms for heterogeneous (NUMA) systems

- methods for coupling multiple physics applications for analysis and optimisation and multi-disciplinary research
  - access to inter-disciplinary data
- design and architecture to integrate execution environment, user environment and applications.

Software implementations for meta-computing are often described in distinct software layers. Typical of this approach is the Integrated Grid Architecture proposed by the Grid Forum [www.gridforum.org/iga.html](http://www.gridforum.org/iga.html). Its four components are:

1. Grid Fabric – the lowest level with primitive mechanisms provide support for high-speed network i/o, differential services, instrumentation, etc.
2. Grid Services – the typical middleware level with a suite of grid-aware services implementing authentication, authorisation, resource location, event services, etc.
3. Application Toolkit – provides more specialised services for diverse application classes, e.g. data-intensive, visualisation, distributed computing, collaborations, problem solving environments (PSE);
4. Grid-aware Applications – implemented in terms of grid services and application toolkit components.

The packages which are relevant to grid computing applications include, among others: Legion, GLOBUS, STA, LSF, UNICORE, AppLeS, CORBA, Nimrod, NetSolve, Synthetix, Chorus, InfoSpheres, Amoeba, MILAN, Arjuna, Apertos, GrassHopper, WAVE, Locust, HPVM, HPC++, C-C++, MIST, GA, Fortran-M, HPF, Java, Raja/RMI, Jini, ANDF, DQS, NQS, LSF, Condor, NQE, LoadLeveler, PBS and Cumulvs. We will describe Legion, GLOBUS, STA and UNICORE in this report. We do not discuss the middle layers of software (referred to as “middleware”) which are used to implement security mechanisms, resource management and low-level heterogeneous communications. However we do briefly mention PACX-MPI, StaMPI and MPICH-G which are implementations of MPI and MPI-2 for grid-based environments.

Finally, grid infrastructures may be, and are being, applied for data management enabling large data sets stored and indexed at remote sites to be analysed and re-used in inter-disciplinary projects.

Foremost in data grid developments are the particle physics, astrophysics and climate modelling communities. The first of these is stimulated by the imminent appearance of very large quantities of data from the CERN Large Hadron Collider (LHC). A large number of countries will participate in analysis of the data, with interacting grids organised as follows:

- Tier 0 – CERN, Geneva where the ATLAS, CMS and other experiments will be run on the LHC;
- Tier 1 – independent national centres in the USA and Europe;
- Tier 2 – a number of regional centres in each country, probably deployed at universities or national laboratories;
- Tier 3 – computing resources of an individual research group;
- Tier 4 – a personal workstation.

This structure is typical of any grid organisation, see [www.gridforum.org/iga.html](http://www.gridforum.org/iga.html).

User requirements in data management are discussed in a separate report [11].

### 3 Computational Grid Environments

Some projects are beginning to provide the full functionality of a computational infrastructure on a distributed grid of computers. We briefly describe the better-known examples to provide an idea of the type of application environments that are being developed and what the HPC user can do with them. More extensive surveys of computational grid and hierarchical clustering projects are available separately [2, 4].

#### 3.1 LSF

LSF **L**oad **S**haring **F**acility is a product of Platform Computing, widely used for corporate computational resource management, especially in the engineering industry, see [www.platform.com](http://www.platform.com). Platform aim to provide the best application resource management solutions for enterprise, allowing administrators to intelligently harness and leverage the maximum power from their existing computing systems by using idle cycles in a flexible and dynamic manner.

Some users who are working in this way include AMD (design of the 1 GHz Athlon chip), Sanger Centre, Cambridge (decoding human chromosome 22), NCSA Alliance (running world's largest NT cluster), Daimler-Chrysler (engine and car development and crash simulation), Ford (advanced vehicle design), Bombardier (aerospace), Lockheed-Martin (aerospace), Mill Film (computer graphics), Cine Studio (computer graphics), Ericsson (Electronic Design Analysis), Alcatel (EDA), Infineon (EDA), Shell (seismic data processing and oil reservoir modelling), RABA (defense analysis and research), etc. Partners working on software explicitly integrated with LSF include MSC (Nastran range of finite-element products), Fluent (CFD packages), Model Technology (Mentor graphics and EDA), StarCD (CFD), etc.

The current product is LSF Suite 4.0 which includes a number of modules (base, batch, analyzer, parallel, multicluster, jobscheduler and make). The base module maintains a load information manager which, for each host on the LSF network, collects data on indices such as cpu queue length and utilisation, available user memory, paging and disk i/o rate, number of users, host idle time, available swap space, available /tmp space, host status. Additional external indices may be configured such as machine type, software license availability, network load etc. The batch schedule engine uses this information and a description of the requested job and priority, to decide when and where to start jobs. The analyzer keeps records of this and is able to produce accounting information and usage reports as required. The parallel module provides integrated MPI libraries with appropriate accounting procedures. The jobscheduler is used for data processing, and the make module uses GNU make for remote compilations.

In addition to scheduling jobs according to the system indices and a priority, limits may be set on run time, number of jobs per host, job type or user. Chaining of jobs is possible for pre- and post-execution processes. Configuration is fully dynamic to account for changes in workload type, e.g. interactive to

batch at night and weekends.

Checkpointing and restart facilities are built in and can be activated by LSF at given periods or by user-provided hooks. Job suspension and migration can be carried out in this way.

Of relevance to grid technology, the concepts are extended in the MultiCluster module which is able to transfer work between locally-managed or remote systems, e.g. to access machines with particular software licenses. This can work over autonomous and widely separately sites, e.g. multiple departments or divisions of large companies, computer centres supporting many sites, multiple cooperating organisations. It also has support for loosely connected sites with long distances or slow links and WAN with possible time differences. This maintains autonomy with each cluster having its own LSF administrators and policies, but negotiation to set up inter-cluster resource sharing.

The first worldwide conference for LSF users is being held in Toronto, Canada, 10-12/9/2000, see [www.platform.com/puc2000](http://www.platform.com/puc2000).

## 3.2 UNICORE

UNICORE (**U**niform access to **C**omputing **R**esources) was originally a project funded by the German Federal Republic to connect together several important regional super-computing centres. The strong federal political structure of the DBR makes this grid-based model particularly relevant and provides a grid environment that is also a very suitable model for a grid connecting the supercomputing centres of the whole EU.

UNICORE provides computing “portals” for engineers and scientists to access supercomputer centres transparently from anywhere on the Internet. Strong authentication is performed in a uniform manner that is designed for ease of use. The differences between platforms are hidden from the user, thus creating a seamless interface for accessing supercomputers, compiling and running applications, and transferring input/output data. An important feature is that UNICORE does not require the participating Centres to alter their operational and security policies.

The main UNICORE components, shown in Figure 1, are the Job Preparation Agent (JPA), the Job Monitor Controller (JMC), the UNICORE https server, also called the Gateway, and the Network Job Supervisor (NJS).

The JPA and JMC feature an intuitive graphical user interface with online help and assistant functionality. They are implemented as Java applets – requiring only a Web browser for execution.

Users interact with the JPA to create UNICORE jobs and to submit them to a UNICORE site. Job descriptions, which are a portable script, can be saved to disk on the local system and can easily be modified and re-used. Descriptions are system-independent, all platform and site-specific parameters and procedures are filled in by the NJS at submission time.

A job contains a number of tasks, connected by dependencies indicating temporal relations or data transfer. Supported tasks include compile, link, execute and data transfer tasks. An execution resource associated with a job specifies where its tasks are to be executed. Tasks can be grouped into sub-tasks, creating a hierarchical job structure and allowing different steps to execute on different systems.

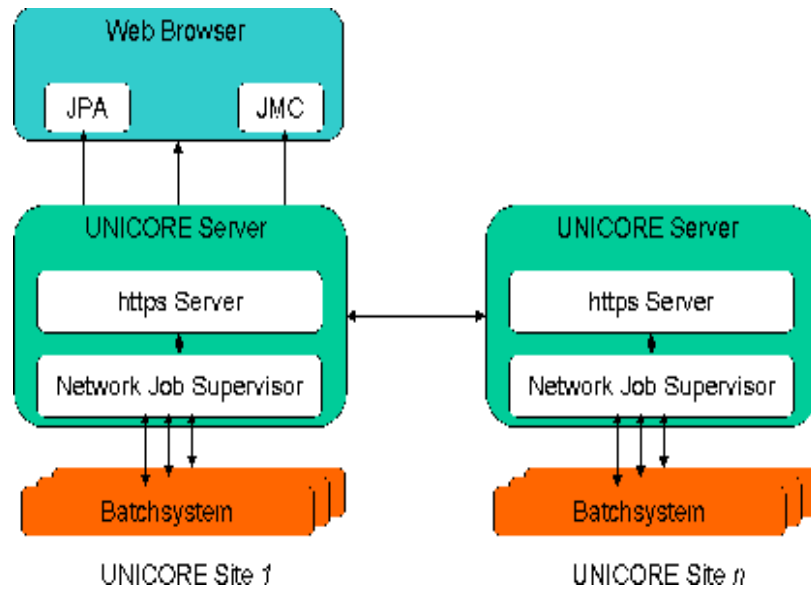


Figure 1: Unicore software architecture

The JMC provides a seamless interface that lets you monitor the progress and control the execution of UNICORE jobs, including access to the job's output and to result files.

In this way UNICORE lets the user compose and edit structured jobs with a graphical job preparation client on a local workstation or PC. Jobs can be submitted to run on any platform in the UNICORE grid, and the user can monitor and control the submitted jobs through the job monitor client.

For further information see [www.unicore.org](http://www.unicore.org).

### 3.3 GLOBUS

Globus is a research and development project focussed on enabling the application of grid concepts to scientific and engineering computing. Research is investigating areas of communication, scheduling, security, information, data access and fault detection. Development is providing the Globus toolkit, a set of services to support innovative high-performance applications in wide area, multi-institutional environments. Deployment activities are providing production grid infrastructures using combinations of Globus and other technologies.

GLOBUS is probably the largest current public-domain grid-software development project. It involves joint work of Argonne National Laboratory and the University of Southern California's Information Science Institute with many additional contributors. Researchers are developing a basic software infrastructure for computations that integrate geographically distributed computational and information resources. The main GLOBUS Web site is at [www.globus.org](http://www.globus.org). This includes information about how to download the software and join the GLOBUS project.

GLOBUS distributed and meta-computing concepts are being tested on a global scale by participants of the Globus Ubiquitous Supercomputing Testbed Organization (GUSTO). This is an agreement between US PACI sites to develop a grid-computing testbed. PACI is the National Science Founda-

tion (NSF) Partnerships for Advanced Computational Infrastructure programme funded by the NSF Advanced Scientific Computing Division. PACI has established two national centres at the Universities of California at San Diego (NPACI) and Illinois (NCSA) which involve over a hundred academic institutions in collaborative HPC projects. This costs around \$64M per year.

GUSTO is based at the main GLOBUS development sites and started in 1997. Users currently span over twenty institutions and contributing sites include some of the largest computers in the world. Both dedicated and commodity Internet services are used.

GUSTO is further described in the GLOBUS Web pages at URLs [www.globus.org](http://www.globus.org) and [www-fp.globus.org/testbeds](http://www-fp.globus.org/testbeds).

A useful list of papers describing GLOBUS and applications which use it can be found at [www.globus.org/documentation/papers.htm](http://www.globus.org/documentation/papers.htm).

The Globus grid programming toolkit is designed to help application developers and tool builders overcome the challenges in the construction of “grid-aware” scientific and engineering applications. It does so by providing a set of standard services for user authentication, resource location, resource allocation, configuration, communication, file access, fault detection, and executable management. These services can be incorporated into applications and/ or programming tools in a mix-and-match fashion to provide access to needed capabilities.

High-level services in GLOBUS include:

- MPICH-G and PAWS – grid-aware MPI communications libraries, see below. MPICH-G uses Nexus, which is a low-level heterogeneous communication layer;
- CC++ and HPC++ – parallel languages;
- grid-aware libraries to provide uniform programming environment;
- remote-access and visualisation;
- DUROC and Nimrod – resource brokers;
- graphical job status monitors.

Core middleware components in GLOBUS include:

- GRAM – Globus Resource Allocation and process Manager interfaces to a local RMS and provides uniform resource allocation, object creation, computation management and co-allocation mechanisms for diverse resource types;
- Nexus – heterogeneous communication infrastructure, supports unicast and multicast;
- MDS – Meta-computing Directory information Services, structure and state information;
- GSI – Grid Security Infrastructure, authentication and related security services, provides public key based single sign-on, run-anywhere capabilities for multi site environments. GSI supports proxy credentials, inter-operability with local security mechanisms, local control over access, and delegation. A wide range of GSI-based applications have been developed ranging from `ssh` and `ftp` to MPI, Condor and the SDSC Storage Resource Broker;
- GASS and GEM – Global Access to Secondary Storage and Global Executable Manager. GASS provides a uniform name space (via URLs) and access mechanisms for files accessed via different protocols and stored in diverse storage system types (HTTP, FTP, HPSS, DPSS etc.)
- Metacomputing Directory Service (MDS) – see below
- Heartbeat Monitor (HBM) – allows the Globus system administrator or ordinary user to detect

failure of Globus system components or application processes;

- Globus Network Performance Monitor (GloPerf) – service that provides on-line information about the latency and bandwidth observed on network links. This information is used to provide the "Network Weather Service".

One of the most important parts is the meta-computing directory service (MDS) which maintains lists of resource components in a distributed directory. This combines an integrated information service with data discovery mechanisms with a lightweight Directory Access Protocol (LDAP). Information can be updated from the GLOBUS system, other information providers and tools and from applications. Information is provided dynamically to tools and applications. A lightweight directory access protocol has been developed. MDS tools include an object class browser, explorer, various APIs and search tools and translators from GLOBUS object definition language.

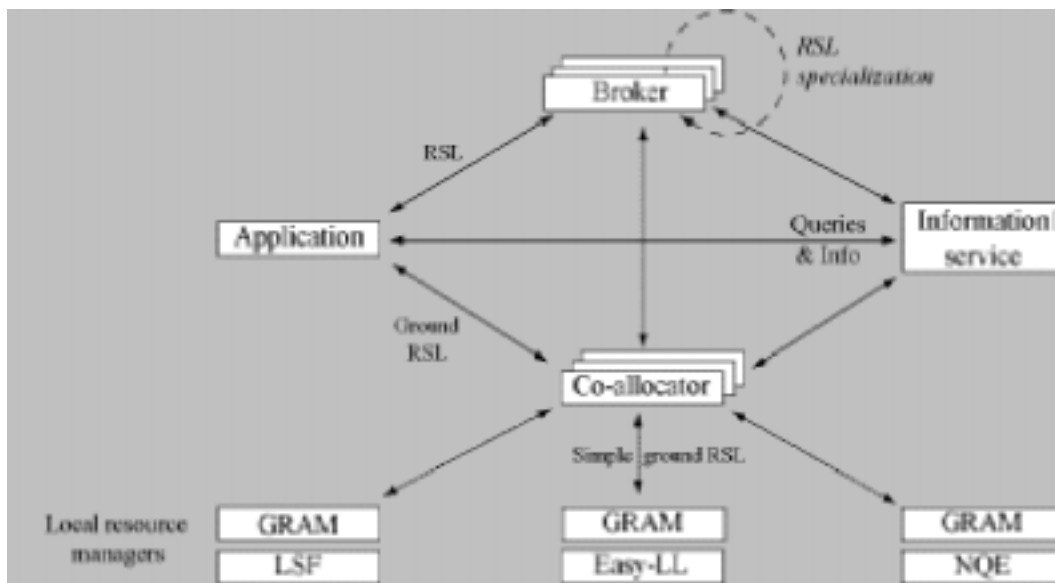


Figure 2: GLOBUS GRAM architecture

For each Globus service a C language, or in some cases a Java application programming interface (API) is defined for use by developers. Concerning the user interface, the GRAM is probably the most important component. This includes an extensible Resource Specification Language (RSL). The interaction between the user, who submits a job with a set of resource requirements via the RSL, and the local resource management software on individual grid platforms, is shown in figure 2. This figure shows examples of some of the supported local RMS software, such as IBM's LoadLeveler (Easy-LL is the modified system used at the Maui Supercomputer Center), LSF (the Load Sharing Facility from Platform Computing) and NQE. PBS, Condor, etc. are also currently supported.

Other APIs are provided, such as Java-based interfaces to produce and submit job templates.

### 3.4 Seamless Thinking Aid

Seamless Thinking Aid (STA) is a Web aware Java-based environment which includes a number of tools for assisting parallel programming. The goal is to allow larger calculations and to couple applications

with different memory or architectural requirements.

This is a project of the Japanese Center for Computational Science and Engineering (CCSE) which was established within the Japanese Atomic Energy Research Institute (JAERI) in 1995. CCSE is playing a leading role in the research and development of computational science and engineering in Japan. This is continuing the work started in the Science and Technology Agency (STA) and will continue to satisfy their requirements.

Principal strands of the research and development at CCSE are:

- development of parallel basic software;
- development of parallel algorithms;
- development of parallel processing tools;
- studies of numerical simulations on complex phenomena by particle and continuum methods;
- new computer architectures.

These feed into applications of special interest to the STA laboratories and Japanese Universities and software is available on JAERI and STA computers. Fortran 90 and MPI is used and the software is portable across many platforms including: Intel Paragon, Fujitsu VPP, Hitachi SR2201, Fujitsu AP3000, IBM SP, NEC SX4, Cray T90 etc.

Seamless Thinking Aid is a Scientific Simulation Platform (SSP) Project. SSP is composed of a communication environment, SCE and two service groups, PPDE/STA and PPExe. These are explained as follows:

- SCE (SSP Communication Environment) provides communication services between tools for program development, and applications distributed to network-based, heterogeneous computers. The communication libraries which SCE provides are:
  - Starpc supports communication between GUIs implemented as Java Applets on a client terminal and tools distributed on parallel computers, as well as communication between tools. Starpc employs the Nexus communication library developed at Argonne National Laboratory (part of GLOBUS) instead of using TCP/IP directory, it is available on networks supported by diverse communication protocols such as TCP/IP, AAL5, and MPL.
  - Stampi is an MPI-2 based communication library to realise distributed parallel computing on a heterogeneous parallel computer cluster
- PPDE/STA (Parallel Program Development Environment/ Seamless Thinking Aid) – integrates development tools under the editor and provides supports for parallel programming. Since PPDE/STA is a Java-based environment, it can be easily used from a Web browser on a personal computer. PPDE/STA has the following tools:
  - PPEditor (Parallel Programming Editor) – the editor that provides a program source listing browser and informational tools in the same window. PPEditor is a consistent GUI which supports user-friendly editing of large programs.
  - Compile Tool – executes a make command for compiling, analyses error messages from a compiler and sends analysis results to the PPEditor.
  - Debug Tool – analyses messages from a debugger and sends such information to PPEditor.

- Visualizing Debug, Vdebug – dynamically visualises values of arrays in a program with co-operative work between a debugger and the visualisation system, such as AVS.
- Tuning Tool – collects application tuning information by using kpx, mtool and kmtool developed by the CCSE and sends information to PPEditor.
- PPExe – executes programs distributed on a heterogeneous parallel computer cluster. PPExe consists of services to manage resource information such as CPU and network loading. This environment provides support for parallel distributed computing in co-operation with four tools as follows:
  - The Task Mapping Editor, TME – graphically defines data flow of tasks assigned to computers in network-based, heterogeneously shared environments.
  - The Meta Scheduler – estimates turnaround time of tasks taking into account loads of network and computers, and selects the computer to execute the task as fast as possible.
  - The Resource Information Monitor, RIM – accumulates and predicts data of loads of computers and network in the past, present and future, and provides data to the Meta Scheduler.
  - The Executing Manager Server, EMS – provides data files necessary for parallel programs and executes programs assigned to each parallel computer according to information from TME.

The STA testbed is a distributed system of parallel computers. Known as COMPACS the system comprises five computers with different architectures: Shared Memory Vector Parallel Cray T94/4; Scalar Parallel IBM SP2/48; Scalar Parallel Hitachi SR2201/64; Distributed Memory Vector Parallel Fujitsu VPP300/16; Distributed/Shared Memory Vector Parallel NEC SX4/2Cx3; and Graphics Server SGI POWER ONYX 10000. Graphics and data servers are also included. Each computer in the COMPACS network is connected through ATM and HIPPI. COMPACS is used to develop a common technology basis for parallel processing.

More information on STA is available from [stasrv1.koma.jaeri.go.jp/en](http://stasrv1.koma.jaeri.go.jp/en).

### 3.5 Legion

Legion is an integrated grid-computing system that has now been deployed at a number of sites. It arose from an object-based software project at the University of Virginia beginning in 1993.

The goal of the project is to develop a highly usable, efficient and scalable system designed to support a large degrees of parallelism in application codes and manage the complexities of the physical system for the user. This software is founded on solid principles, guided by work in object-oriented parallel processing, distributed computing and security. The Legion group has wide experience of distributed computing systems.

Legion supports existing codes written in MPI and PVM, as well as “legacy” binaries. Key capabilities include:

- eliminating the need to move and install binaries manually on multiple platforms;
- providing a shared, secure virtual file system that spans all the machines in a Legion system;

- providing strong private key based authentication and flexible access control for user objects; and
- supporting remote execution of legacy codes, and their use in parameter space studies.

In this way Legion attempts to address the issues of scalability, programming ease, fault tolerance, security and site autonomy. It is designed to support a large degrees of parallelism in application codes and manage the complexities of the physical system for the user.

Components include:

- method invocation service;
- file system;
- security system;
- context space directory services;
- resource management service;
- core-object management service.

These components link the local computer operating system to the application codes.

Legion is the second grid project which has been adopted by the US NSF at its National Partnership for Advanced Computational Infrastructure (NPACI) sites. These include SDSC, University of California San Diego, CalTech, U Texas, U Michigan, UC Berkeley, UC Davis, UCLA, UC Santa Barbara, U Houston/Keck, U Maryland and U Washington. NASA and the DoD are also running Legion testbeds.

A variety of applications have been ported to Legion, e.g. CHARMM, ocean models, CCM, particle-in-cell codes, and several parameter-space studies.

For more information on Legion see <http://legion.virginia.edu/overview.html>. Web pages contain historical information about the project, documentation, discussion of the key features and a download facility.

### 3.6 Jini

Jini is not a technology for HPC grids. It is designed to meet the more general requirements of federating network resources. This generality makes it very important in the broader view of the Grid as a ubiquitous, robust information analogue to electrical power grids. It has been included in this document because of its design elegance and simplicity as well as its growing importance as a community standard. Whilst Jini will not currently support an HPC grid it represents the natural direction for grid technology and it is likely that either Jini, or something that builds on its design concepts will be an integral part of the Grid of the future.

A Jini system is a distributed system based on the idea of federating groups of users and the resources required by those users. The overall goal is to turn the network into a flexible, easily administered tool with which resources can easily be found. Resources can be hardware devices or software programs. The focus is to make the network a more dynamic entity that enables the ability to add and remove services flexibly.

A Jini system consists of the following parts:

- A set of components that provides an infrastructure for federating services in a distributed system;
- A programming model that supports and encourages the production of reliable distributed services;
- Services that can be made part of a federated Jini system and that offer functionality to any other member of the federation.

The Jini system is Java technology centred. The Jini architecture gains much of its simplicity from assuming that the Java programming language is the implementation language for components. The ability to dynamically download and run code is central to a number of the features of the Jini architecture. Jini leverages the Java environment to provide systems which are far more dynamic than is currently possible in networked groups where configuring a network is a centralised function done by hand.

It is important to note that the Java centric nature of Jini does not imply that programs on a Jini-grid would have to be written in Java. The Java Native Interface (JNI) would allow performance codes in Fortran, C or C++ to be contained in Java wrappers so that they could benefit from the advantages of the Jini environment without suffering any performance impact.

Jini is part of the Sun Java standard and as such is a community standard, but one controlled by a single corporation. It has not been used, as yet, for constructing HPC grids because the current implementation has two key drawbacks:

- the security model is yet to be implemented although the standard is currently available for comment and an implementation should be available by 2002;
- the implementation is currently limited to local area networks, although several groups are working on wide area extensions.

The main contributions that Jini would make to a Grid environment are:

- platform independence;
- dynamic service registration/lookup facilities; and
- allowing the traditional client/server interface to be moved inside the client (through the use of downloadable Java client stub code), thus making it easier to revise or enhance low-level network technologies/ protocols while minimising the impact of such changes to applications.

Jini is not an immediate solution to grid problems. However it points the direction that federated systems of network technologies are likely to take. It could well be the basis for tomorrow's Grid. More information on Jini can be found at <http://www.javasoft.com/jini>.

## 4 Relationship between Projects

Condor is a tool for harnessing the capacity of idle workstations for computational tasks whereas Globus is primarily concerned with harnessing high-end computing resources. Condor and Globus are thus complementary and the groups are working together. Condor uses Globus security technologies and Globus supports the submission of Condor jobs.

The Legion project is developing an object-oriented framework for grid applications. The goal is to promote the “principled” design of distributed systems software by providing sets of standard objects (for processors, data systems, etc.). Applications are then developed in terms of those objects. Globus and Legion are to some extent complementary since Globus focuses on low-level services and Legion on higher-level programming models. There are however significant areas of overlap.

Java is becoming very widespread as a language for Internet-based software development, but it does not address many of the hard problems that arise when we try to achieve high performance execution in a heterogeneous distributed environment. Java in itself, although portable, cannot help us run transparently on different supercomputers or solve the other issues that are the focus of the Globus research and development project. Jini uses Java to provide portable client-side programs which will contribute to grid software.

The Common Object Request Broker (CORBA) is also widely used, as is its Microsoft equivalent, especially in commercial circles. CORBA defines a standard Interface Definition Language (IDL) for inter-language operability, a remote procedure call service and a variety of more specialised services such as a resource trader. CORBA, like Java, has important software engineering advantages, but also does not address many of the challenges that arise in grid environments such as specialised devices and high run-time performance. CORBA is therefore also complementary to grid technology and, for instance, a Globus/ CORBA interface is being developed.

## 5 Resource Management for Heterogeneous Clusters

Computational grid technology is potentially applicable to all HPC applications, and future work of UKHEC will investigate requirements and performance issues. In this report we concentrate on the capabilities of software and tools of immediate interest to scientific programmers.

With a large number of diverse applications and users, each of whom considers his own to be the top priority, issues like job scheduling, controlling access to specific machines or software and fault tolerance are very important. Resource Management Systems (RMS) are designed to simplify the management of clustered systems. The available RMS products provide transparent access to the most suitable resources for the end user, configurable options to control access as required: to limit specific users or groups of users to particular machines, software, times, number of concurrent jobs, etc, resilience to the loss of one or more systems; jobs can be automatically migrated and restarted elsewhere (with higher priority) if access to the machine on which it was running becomes unavailable, and dynamic load management; continual monitoring of all systems ensures that work can be shared evenly.

The most important middleware tools access the RMS which allocates resources to the jobs issued on a

distributed system. They enable users to access computers whose load is relatively low or those which satisfy a particular requirement (e.g. memory, vector processing, visualisation etc.). Examples are LSF (Load Sharing Facility) from Platform Computing, CODINE (**C**omputing in **D**istributed **N**etworked **E**nvironments) from GridWare, LoadLeveller and POE (from IBM) as commercial products with PBS (Portable Batch System) and DQS (Distributed Queuing System) as public-domain implementations. These systems are for instance integrated into the GLOBUS software.

More information about cluster tools is given in a separate report [4].

## 6 Message-passing Libraries

MRCCS has been working with HLRS, Stuttgart who have developed the PACX-MPI library that can address issues of message passing in heterogeneous systems. Issues of data portability must be considered, e.g. number representation, IEEE etc. big- or little-endian, as well as different network connectivity of the nodes in the cluster. It was this software that was demonstrated running meta-computer applications at SC'99 via inter-continental links between Europe, the USA and Japan. Other message-passing libraries include StaMPI from JAERI and MPICH-G from Argonne. PVM can also cope with heterogeneous clusters. Case studies involving coupled models on the different components of the CSAR service are being tested and the results will be made available to the community. PVM is particularly suitable for message-passing between different platforms (nodes) with, perhaps, MPI used within the node.

### 6.1 StaMPI

StaMPI is the application-layer communication interface for the Seamless Thinking Aid from JAERI. It is a meta-scheduling method which includes MPI-2 features to dynamically assign macro-tasks to heterogeneous computers using dynamic resource information and static compile time information. This kind of dynamic assignment can be done by PVM but not by MPI-1. An application program is invoked for execution as the main process on a host computer in the grid. Other processes are spawned by the main process on other computers to execute macro-tasks.

StaMPI automatically distinguishes inter- and intra-cluster messages using vendor-supplied MPI for intra-cluster messages and IP for inter-cluster messages. Additionally, StaMPI has implemented `MPI_Comm_Spawn` (from the proposed MPI-2 standard) for dynamic process creation.

Features of StaMPI are now described.

**Automatic switching from internal to external communication protocol.** In the case of internal communication between processes on the same parallel computer, StaMPI uses a vendor specific communication library that supports a high performance communication protocol. In the case of external communication between processes on different parallel computers, StaMPI uses IP (Internet Protocol). StaMPI automatically switches communication protocols by detecting whether communication is internal or external. This procedure is quite similar to that in MPI implementations on NUMA systems, e.g. the IBM SP Winterhawk or Nighthawk which has SMP nodes.

**Message relaying by message routers.** If processes on a parallel computer can not communicate directly with external processes through IP, StaMPI creates a message routing process that relays messages and enables indirect communication with them. These processes need not know whether the message routing process should be created, because this creation is done automatically. StaMPI creates as many routers as needed, that is, the number is not fixed to two as with PACX-MPI. For example, to connect two machines such as Fujitsu VPP300 and Hitachi SR2201, the following number of routers are made/ used for the connection.

machine1	machine2	Number of routers in PACX-MPI	Number of routers in StaMPI
VPP300	VPP300	2	2
SR2201	SR2201	2	0
VPP300	SR2201	2	1

The smaller the number of routers, the greater the possible efficiency of the communication.

**Dynamic process creation.** There are two ways to create processes: static and dynamic. It is difficult and inefficient to obtain computer resources at the beginning of a program execution, because these resources are shared by many users in general. StaMPI can create processes dynamically on parallel computers. As StaMPI is based on the MPI-2 specification the `MPI_Comm_spawn` function is used for the creation.

Currently, StaMPI works on MPPs, Fujitsu VPP300, NEC SX-4, SGI T94, IBM SP, Hitachi SR2201, Fujitsu AP3000, and Intel Paragon.

We include below the outline of a C program using StaMPI to show that the coding is almost identical to that used with MPI on a single machine.

```
#include <stdio.h>
#include "mpi.h"

#define NumMach 100
#define MaxNAME 100
#define MaxCPU 100
#define MachEnt 20
#define TaskSize 100

static MPI_Comm server_comm, client_comm[NumMach], mcw, meta_comm;
MPI_Status mpi_stat;
int me, size, err;
char localhost[MaxNAME];

char *machines[NumMach][MachEnt] = {
  { "host", "desr01", "-np", "7", "user", "fumie",
    "wdir", "/home002/ssp/fumie/wrk/stampi-test",
    "-cmd", "desr01", NULL, NULL },
  { "host", "stasrv0", "-np", "2", "user", "fumie",
    "wdir", "/home/pos/fumie/wrk/stampi-test",
    "-cmd", "stasrv0", NULL, NULL },
  { "host", "stasrv1", "-np", "3", "user", "fumie",
    "wdir", "/home/pos/fumie/wrk/stampi-test",
```

```

    "-cmd", "stasrv1", NULL, NULL },
    { NULL, NULL }
};

int main(int ac, char **av) {
    int i;
    /*-- Initialize -----*/
    MPI_Init(&ac, &av);
    MPI_Comm_rank(MPI_COMM_WORLD, &me);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_get_parent(&server_comm);
    gethostname(localhost, MaxNAME);
    if (server_comm==MPI_COMM_NULL) {
    /*----- Server part -----*/
        invoke_client();
        server_loop();
        server_finalize();
    } else {
    /*----- Client part -----*/
        client_loop();
    }
    /*--- Finalize -----*/
    MPI_Finalize();
    exit(0);
    return 0;
}

```

## 6.2 PACX-MPI

PACX-MPI (**PA**rallel **C**omputer **eX**tension to MPI) [12, 13] is an on-going project of the HLRS, Stuttgart. It enables an MPI application to run on a meta computer consisting of several, possibly heterogeneous, machines, each of which may itself be massively parallel. Further documentation is available on-line at [www.hlrs.de/structure/organisation/par/projects/pacx-mpi](http://www.hlrs.de/structure/organisation/par/projects/pacx-mpi).

From the programmer's point of view, the PACX-MPI approach is very convenient. Provided that the programmer does not step outside the bounds of the (increasingly) large subset of MPI-1.2 that PACX-MPI supports, a correct MPI application can be expected to run without alteration under PACX-MPI. The details of the configuration of the meta computer are hidden from the application code, which sees a single `MPI_COMM_WORLD` consisting of processes on all hosts. In fact, PACX-MPI does not provide a mechanism to distinguish between processes running on different hosts, at least not yet.

PACX-MPI achieves this by reserving an additional two processes on each host to mediate inter-host communications. All inter-host communications are routed through these processes, a feature which permits message bundling and minimises the number of connections between processes on different hosts. All intra-host communications are delegated to the host's native MPI library.

Figure 3 illustrates how the processes in local communicators on two systems are combined to form

a single `MPI_COMM_WORLD`. Two processors on each system are hidden from the application code; these serve as daemons for external communication.

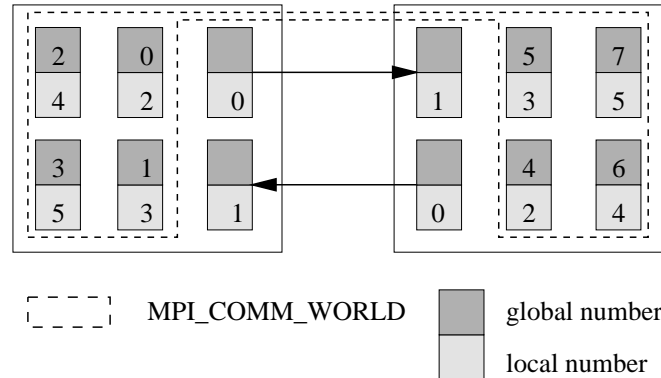


Figure 3: Process numbering in PACX-MPI – two systems integrated into a single `MPI_COMM_WORLD`

Complexities arise in the configuration of the meta computer and in the launching of jobs. The user defines the meta computer configuration in a short text file `.hostfile`, which must be replicated in working directory of the application on each host. The application is launched via a standard mechanism (e.g. `mpirun`) on the first host listed in the hostfile, with the notable exception that provision must be made for the two daemon processes. What happens next depends on whether the *automatic* or *manual* startup procedure is employed.

We first describe manual startup with reference to an example taken from the SC'99 meta computing demonstration (see Section 7) in which the following hostfile was used.

```
turing.cfs.ac.uk tcp 512
129.69.200.195 tcp 128 #
jaromir.psc.edu tcp 128 #
```

This describes a meta computer consisting of 512+2 nodes on `turing.cfs.ac.uk`, 128+2 nodes on `129.69.200.195` and 128+2 nodes on `jaromir.psc.edu`. The user starts by launching the application on the first host (here `turing.cfs.ac.uk`), requesting 514 (not 512) processors. When the application begins, PACX reports additional command line arguments (containing information about socket numbers) that the user must supply when launching the application on the second host. By the time the user comes to launch the application on the third host, there are two additional sets of socket numbers to specify on the command line.

The automatic startup facility is far more convenient. The user provides, in the hostfile, additional information that the PACX daemons use to start the application on the remote systems. Unfortunately, the automatic startup facility does not work on all important platforms; the Cray-T3E is a noticeable exception.

### 6.3 MPICH-G

MPICH-G is the grid-aware version of the free MPI implementation from Argonne National Laboratory. It is used with GLOBUS, see [www.globus.org/mpi](http://www.globus.org/mpi). MPICH-G enables a MPI program to run in a grid environment without change.

## 7 SuperComputing'99 Experiment

During SuperComputing'99, four platforms were linked together by PACX-MPI and ATM Private Virtual Circuits (PVCs). These included the Hitachi SR8000 at Tsukuba, Japan with 512 processors (64 nodes), Cray T3E-900/512 at Pittsburgh, USA with 512 processors, Cray T3E-1200E/576 at Manchester, UK with 576 processors, Cray T3E-900/512 at Stuttgart, Germany with 512 processors. PVCs were set up with bandwidth of 10 Mbit/s in European networks. On all other connections bandwidth was shared with other users. The sustained bandwidth as well as the latency was as follows:

	latency	bandwidth
HLRS-CSAR	20 ms	0.8 MBit/s
HLRS-PSC	78 ms	1.6 MBit /s
CSAR-PSC	58 ms	0.5 MBit /s

The usability of this meta computer was demonstrated by several demanding applications. Further information on this experiment is provided by Fumie Costen [www.csar.cfs.ac.uk/staff/costen/sc99.html](http://www.csar.cfs.ac.uk/staff/costen/sc99.html). We have shown that it is possible to link together different types of machine, in this case Cray-T3E and Hitachi SR8000 and that even data-intensive applications, such as processing output from experimental facilities, can be successfully undertaken on such a meta computer.

## 8 Outstanding Issues

We have described a number of software projects aimed at providing heterogeneous distributed computational grids. These are intended to overcome some of the barriers to remote super-computing and provide a flexible transparent means of access. However, to the software developers, many challenges remain. These mean that deployment of a “useable” grid to satisfy day-to-day computing requirements may be some time off. Among the major research work which is still ongoing is:

- end-to-end resource management and adaptation techniques able to provide application-level performance guarantees despite dynamic resource properties;
- automated techniques for negotiation of resource usage, policy and accounting in large-scale grid environments;
- high-performance communication methods and protocols;

- infra-structure and tools to support data-intensive applications, advanced tele-immersion concepts and new problem solving environment techniques;
- meta-schedulers;
- resource accounting and billing mechanisms;
- how to involve IT experts in enhancing the grid infrastructure: new s/w, tools, algorithms, etc.
- how to adapt applications to use it: steering, visualisation, etc.
- how to encourage E-service providers to make facilities available: cpu, storage, VR, instrumentation (JREI ?)
- what networking implications for data retrieval, computational steering, remote visualisation and "collaborative working"
- role of "top end" system: flexible access
- role of 3rd party software providers, E-services and licenses on demand ?

In the commercial sector LSF from Platform Computing already solves many of these problems.

## 9 Further Information

The authors are currently involved in surveying grid computing and heterogeneous cluster developments, see separate reports [2, 4] which will also be available from the UKHEC Web page [www.ukhec.ac.uk](http://www.ukhec.ac.uk).

For more general information on grid-based computing see the book by Ian Foster and Carl Kesselman [6].

## 10 Acknowledgements

The preparation of this report was funded by EPSRC through grant GR/N09688 to the UKHEC Collaboration at Daresbury Laboratory and the Universities of Edinburgh and Manchester. The authors wish to thank Michael Resch of HLRS Stuttgart and Stephen Pickles of CSAR.

## References

- [1] See CSAR Web pages [www.csar.cfs.ac.uk](http://www.csar.cfs.ac.uk)
- [2] R.J. Allan *Survey of Computational Grid, Meta-computing and Network Information Tools* Edition 2. Technical Report DL-TR-99-01 (Daresbury Laboratory, 2000)

- [3] R. Buyya (ed.) *High Performance Cluster Computing: Architectures and Systems, Volume 1* (Prentice Hall, NJ, USA, 1999)
- R. Buyya (ed.) *High Performance Cluster Computing: Programming and Applications, Volume 2* (Prentice Hall, NJ, USA, 1999)
- For more information, please see:  
[www.dgs.monash.edu.au/~rajkumar/cluster](http://www.dgs.monash.edu.au/~rajkumar/cluster)  
[www.phptr.com/ptrbooks/ptr\\_0130137847.html](http://www.phptr.com/ptrbooks/ptr_0130137847.html)  
[www.phptr.com/ptrbooks/ptr\\_0130137855.html](http://www.phptr.com/ptrbooks/ptr_0130137855.html)
- [4] F. Costen, M. Foster and M. Pettipher *Hierarchical Clustering* Technical Report, Manchester Computing (University of Manchester, 2000)
- [5] I. Foster and K. Kesselman *GLOBUS: a Meta-computing Infrastructure Toolkit* Int. J. Supercomputing Applications (1997) 115-28
- I. Foster and K. Kesselman *The Globus Project: a status report* IPPS/SPDP'98 Heterogeneous Computing Workshop S.4-18 (1998) [www-fp.globus.org/documentation/papers.html](http://www-fp.globus.org/documentation/papers.html)
- [6] I. Foster and C. Kesselman (eds.) *The Grid: Blueprint for a new Computing Infrastructure* (Morgan Kaufmann Publishers, 1998) ISBN 1-55860-475-8. Abstracts of chapters and ordering information from URL [www.mkp.com/grids](http://www.mkp.com/grids)
- [7] G.C. Fox and W. Furmanski *Petaops and Exaops: Supercomputing on the Web* IEEE Internet Computing 1 (1997) 38-46. Also available via the Web at URL <http://www.npac.syr.edu/users/gcftpastuff/petaweb>
- G.C. Fox, W. Furmanski, T. Haupt, E. Akarsu and H. Ozdemir *HPcc as High-Performance Commodity Computing on top of Integrated Java, CORBA, COM and Web Standards* in "EuroPar'98: Parallel Processing" D. Pritchard and J. Reeve (eds.) LCNS 1470 pp55-74 (Springer 1998) ISBN 3-540-64952-2
- G. Fox and W. Furmanski *Java for Parallel Computing and as a general Language for Scientific and Engineering Simulation and Modelling* Concurrency: Practice and Experience 9 (1997) 415-26
- G. Fox et al. *Using WebHLA to integrate HPC FMS Modules with Web/ Commodity based Distributed Objects Technologies of CORBA, Java, COM and XML* in "High Performance Computing 1999" A. Tentner (ed.) (Society for Computer Simulation International, 1999) pp 273-8 ISBN 1-56555-166-4
- [8] E. Gabriel et al. *Implementing MPI with optimised algorithms for meta-computing* in "Proc. 3rd MPI Developers' and Users' Conference" (MPI Software Technology Press, Mississippi, 1999) see [www.hlrs.de/people/resch/PAPER/pubs.html](http://www.hlrs.de/people/resch/PAPER/pubs.html)
- [9] A.S. Grimshaw, W.A. Wulf, J.C. French, A.C. Weaver and P.F. Reynolds Jr. *A Synopsis of the Legion Project* Technical Report CS-94-20 (University of Virginia, 1994)
- A.S. Grimshaw, W.A. Wulf, J.C. French, A.C. Weaver and P.F. Reynolds *Legion: the next Logical Step toward a Nationwide Virtual Computer* Technical Report CS-94-21 (University of Virginia, 1994)
- A.S. Grimshaw and W.A. Wulf *Legion - a view from 20,000 feet* in "Proc. 5th IEEE International Symposium on HPDC" IEEE Computer Society Press (1996)

- A.S. Grimshaw, W.A. Wulf, J.C. French, A.C. Weaver and P.F. Reynolds Jr. *The Legion Vision of a Worldwide Virtual Computer* CACM 40 (1997)
- [10] K. Hwang and Z. Xu *Scalable Parallel Computing: Technology, Architecture, Programming* (WBC/McGraw Hill, 1998) ISBN 0-07-031798-4
- [11] K. Kleese *Requirements for a Data Management Infrastructure to support UK High-End Computing* Technical Report DL-TR-99-04 (Daresbury Laboratory, 1999)
- [12] Edgar Gabriel, Michael Resch, Thomas Beisel, Rainer Keller, *Distributed Computing in a heterogeneous computing environment* in "Recent Advances in Parallel Virtual Machine and Message Passing Interface" Vassil Alexandrov, Jack Dongarra (Eds.) Lecture Notes in Computer Science (Springer, 1998) pp 180-8
- [13] Matthias A. Brune, Graham E. Fagg, Michael Resch, *Message-Passing Environments for Meta-computing* Future Generation Computer Systems 15 (1999) 5-6 pp. 699-712
- [14] H. Koide *STAMPI* see [www.globus.org/mpi/related.html](http://www.globus.org/mpi/related.html)  
H. Koide et al. *MPI-based communication library for a heterogeneous parallel computer cluster, STAMPI* JAERI <http://ssp.koma.jaeri.go.jp/en/stampi.html>
- [15] R. Orfali and D. Harkey *Client/Server Programming with Java and CORBA* (Wiley, 1997) ISBN 0-471-16351-1
- [16] L.H. Turcotte *A Survey of Software Environments for Exploiting Networked Computing* Technical Report MSU-EIRS-ERC-93-2 (Mississippi State University, 1993)