The Grid: International Efforts in Global Computing

Mark Baker, Rajkumar Buyya and Domenico Laforenza

Abstract

The last decade has seen a considerable increase in commodity computer and network performance, mainly as a result of faster hardware and more sophisticated software. Nevertheless, there are still problems, in the fields of science, engineering and business, which cannot be dealt effectively with the current generation of supercomputers. In fact, due to their size and complexity, these problems are often numerically and/or data intensive and require a variety of heterogeneous resources that are not available from a single machine. A number of teams have conducted experimental studies on the cooperative use of geographically distributed resources conceived as a single powerful computer. This new approach is known by several names, such as, metacomputing, seamless scalable computing, global computing, and more recently grid computing. The early efforts in grid computing started as a project to link supercomputing sites, but now it has grown far beyond its original intent. In fact, there are many applications that can benefit from the grid infrastructure, including collaborative engineering, data exploration, high throughput computing, and of course distributed supercomputing. Moreover, the rapid and impressive growth of the Internet, there has been a rising interest in web-based parallel computing. In fact, many projected have been incepted to exploit the Web as an infrastructure for running coarse-grained distributed parallel applications. In this context, the web has the capability to become a suitable and potentially infinite scalable metacomputer for parallel and collaborative work as well as a key technology to create a pervasive and ubiquitous grid infrastructure. This paper aims to present the state-of-the-art of grid computing and attempts to survey the major international adventures in developing this upcoming technology.

Keywords: Metacomputing, Grids, Middleware, Resource Management and Scheduling, Internet Computing, Computing Portals.

I. Introduction

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we use computers today. This technology opportunity has led to the possibility of using networks of computers as a single, unified computing resource. It is possible to cluster or couple a wide variety of resources including supercomputers, storage systems, data sources, and special classes of devices distributed geographically and use them as a single unified resource, thus forming what is popularly known as a "computational grid".

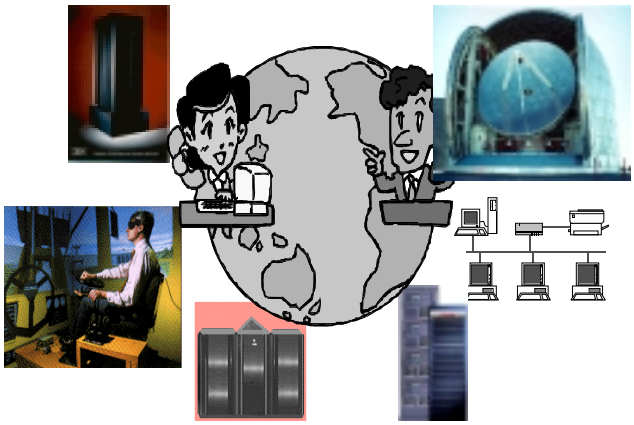

Figure 1. Towards Grid Computing: A Conceptual

The origin of the terms metacomputer and metacomputing are believed to have come out of the CASA project [1] one of several U.S. Gigabit testbeds around in late 1980's. Larry Smarr, the NCSA Director, is generally accredited with popularizing the term thereafter. In particular, Catlett and Smarr have related the term metacomputing to "the use of powerful computing resources transparently available to the user via a networked environment" [2]. Their view is that a metacomputer is a networked virtual supercomputer. To an extent our usage of the term metacomputing still holds true to this definition apart from explicitly referring to "powerful" computing resources. Other terms have also been used to describe this computing paradigm, such as seamless, scalable or global computing and more recently grid computing.

The concept of grid computing started as a project to link supercomputing sites, but now it has grown far beyond its original intent. In fact, there are many several applications that can benefit from the grid infrastructure, including collaborative engineering, data exploration, high throughput computing, and of course distributed supercomputing. According to Larry Smarr, a grid is a seamless, integrated computational and collaborative environment (see Figure 1). Grid functions can be bisected into two logical grids: the *computational* grid and the *access* grid. Through the computational grid the scientists will be able to access virtually unlimited computing and distributed data resources. The access grid will provide a group

Mark Baker, School of Computer Science, University of Portsmouth, Mercantile House, Portsmouth, Hants, PO1 2EG, UK – Mark.Baker@port.ac.uk
Rajkumar Buyya, School of Computer Science and software Eng., Monash University, Caulfield Campus, Melbourne, Australia,
rajkumar@csse.monash.edu.au
Domenico Laforenza, Advanced Computing Department, CNUCE-Institute of the Italian National Research Council, via Vittorio Alfieri, 1, I-56010 Ghezzano, Pisa, Italy
Domenico.Laforenza@cnuce.cnr.it

collaboration environment. Through a Web browser, users will be able to view and select all the grid resources and services in a *virtual infinite machine room* [3][4]. To build a grid requires the development and deployment of a number of services, including those for: resource discovery, scheduling configuration management, security, and payment mechanisms in an open environment [5][6][10].

Grid applications (multi-disciplinary applications) couple resources that cannot be replicated at a single site even or may be globally located for other practical reasons. These are some of the driving forces behind the inception of grids. In this light, grids let users solve larger or new problems by pooling together resources that could not be coupled easily before.

Hence the *Grid* is not only a computing paradigm for just providing computational resources for grand-challenge applications. It is an infrastructure that can bond and unify globally remote and diverse resources ranging from meteorological sensors to data vaults, from parallel supercomputers to personal digital organizers. As such, it will provide pervasive services to all users that need them.

This paper aims to present the state-of-the-art of grid computing and attempts to survey the major international efforts in this area. A set of general principles and design criteria that can be followed in the grid construction are given in Section 2. Some of the current grid experiments selected as representative of the possible technologies are presented in Section 3. We conclude and then discuss future trends in Section 4.

II. Grid Construction: General Principles

This section briefly highlights some of the general principles that underlie the construction of the grid. In particular, the idealized design features that are required by a grid to provide users with a seamless computing environment are discussed. There are three main issues that characterize computational grids:

- *Heterogeneity*: a grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across wide geographical distances.
- *Scalability*: a grid might grow from few resources to millions. This raises the problem of potential performance degradation as a Grids size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant.
- *Dynamicity or Adaptability*: in a grid, a resource failure is the rule, not the exception. In fact, with so many resources in a Grid, the probability of some resource failing is naturally high. The resource managers or applications must tailor their behaviour dynamically so as to extract the maximum performance from the available resources and services.

The steps necessary to realize a computational grid include [6]:

- The integration of individual software and hardware components into a combined networked resource.
- The implementation of *middleware* to provide a transparent view of the resources available.
- The development of tools that allows management and control of grid applications and infrastructure.
- The development and optimization of distributed applications to take advantage of the resources.

The components that are necessary to form a grid are shown in Figure 2 and they are briefly discussed below:

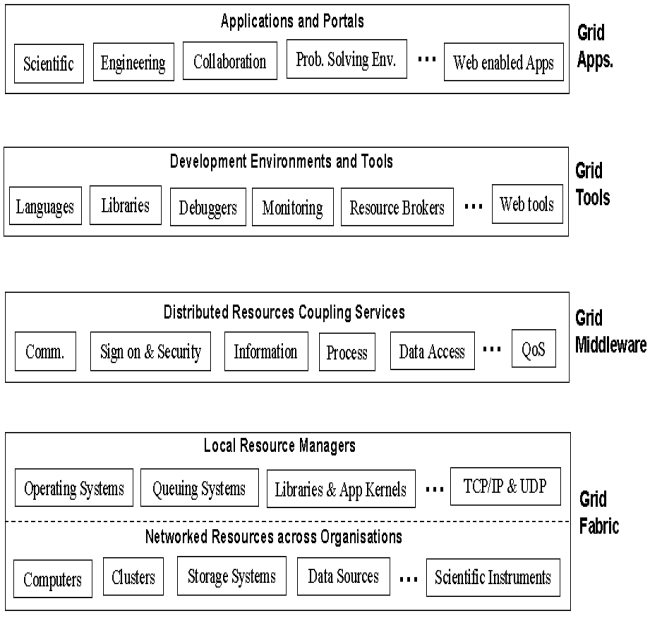

Figure 2: Grid Components.

- Grid *Fabric*: It comprises all the resources geographically distributed (across the globe) and accessible from anywhere on the Internet. They could be computers (such as PCs or Workstations running operating systems such as UNIX or NT), clusters (running cluster operating systems or resource management systems such as LSF, Condor or PBS), storage devices, databases, and special scientific instruments such as a radio telescope.
- G*rid Middleware*: It offers core services such as remote process management, co-allocation of resources, storage access, information (registry), security, authentication, and Quality of Service (QoS) such as resource reservation and trading.
- G*rid Development Environments and Tools*: These offer high-level services that allows programmers to develop applications and *brokers* that act as user agents that can manage or schedule computations across global resources.
- G*rid Applications and Portals*: They are developed using grid-enabled languages such as HPC++, and message-passing systems such as MPI. Applications, such as parameter simulations and grand-challenge

problems often require considerable computational power, require access to remote data sets, and may need to interact with scientific instruments. *Grid portals* offer web-enabled application services — i.e., users can submit and collect results for their jobs on remote resources through a web interface.

In attempting to facilitate the collaboration of multiple organizations running diverse autonomous heterogeneous resources, a number of basic principles should be followed so that the grid environment:

- Does not interfere with the existing site administration or autonomy;
- Does not compromise existing security of users or remote sites;
- Does not need to replace existing operating systems, network protocols, or services;
- Allows remote sites to join or leave the environment whenever they choose;
- Does not mandate the programming paradigms, languages, tools, or libraries that a user wants;
- Provides a reliable and fault tolerance infrastructure with no single point of failure;
- Provides support for heterogeneous components;
- Uses standards, and existing technologies, and is able to interact with legacy applications;
- Provides appropriate synchronization and component program linkage.

Initiative	Focus and Technologies Developed
Computing Portals	A collaborative effort between different Computer Science projects to enable desktop access to remote resources including, supercomputers, network of workstations, smart instruments, data resources, and more – `www.computingportals.org`
Grid Forum	This is a community-initiated forum of individual researchers and practitioners working on distributed computing, or "grid" technologies. This forum focuses on the promotion and development of grid technologies and applications via the development and documentation of "best practices," implementation guidelines, and standards with an emphasis on rough consensus and running code – www.gridforum.org
European Grid Forum	EGRID, aims to foster the cooperative use of distributed computing resources that are accessible via WANs. EGRID is an open forum; the community includes individuals from European research institutes, universities and companies working in the field of wide area computing and computational grids - www.egrid.org

Table 1: Major Grid Forums

As one would expect, a grid environment must be able to operate on top of the whole spectrum of current and emerging hardware and software technologies. An obvious analogy is the Web. Users of the Web do not care if the server they are accessing is on a UNIX or NT platform. From the client browser's point of view, they "just" want their requests to Web services handled quickly and efficiently. In the same way, a user of a grid does not want to be bothered with details of its underlying hardware and software infrastructure. A user is really only interested in submitting their application to the appropriate resources and getting correct results back in a timely fashion.

An ideal grid environment will therefore provide access to the available resources in a seamless manner such that physical discontinuities such as differences between platforms, network protocols, and administrative boundaries become completely transparent. In essence, the grid middleware turns a radically heterogeneous environment into a virtual homogeneous one.

The following are the main design features required by a grid environment:

- *Administrative Hierarchy* - An administrative hierarchy is the way that each grid environment divides itself up to cope with a potentially global extent. The administrative hierarchy determines how administrative information flows through the grid.
- *Communication Services* - The communication needs of applications using a grid environment are diverse, ranging from reliable point-to-point to unreliable multicast communications. The communications infrastructure needs to support protocols that are used for bulk-data transport, streaming data, group communications, and those used by distributed objects. The network services used also provide the grid with important Quality of Service parameters such as latency, bandwidth, reliability, fault-tolerance, and jitter control.
- *Information Services* - A grid is a dynamic environment where the location and type of services available are constantly changing. A major goal is to make all resources accessible to any process in the system, without regard to the relative location of the resource user. It is necessary to provide mechanisms to enable a rich environment in which information about grid is reliably and easily obtained by those services requesting the information. The grid information (registration and directory) services components provide the mechanisms for registering and obtaining information about the grid structure, resources, services, and status.
- *Naming Services* – In a grid, like in any distributed system, names are used to refer to a wide variety of resources such as computers, services, or data objects. The naming service provides a uniform name space across the complete metacomputing environment. Typical naming services are provided by the international X.500 naming scheme or DNS, the Internet's scheme.

- *Distributed File Systems and Caching* – Distributed applications, more often than not, require access to files distributed among many servers. A distributed file system is therefore a key component in a distributed system. From an applications point of view it is important that a distributed file system can provide a uniform global namespace, support a range of file I/O protocols, require little or no program modification, and provide means that enable performance optimizations to be implemented, such as the usage of caches.
- *Security and Authorization* – Any distributed system involves all four aspects of security: confidentiality, integrity, authentication and accountability. Security within a grid environment is a complex issue requiring diverse resources autonomously administered to interact in a manner that does not impact the usability of the resources or introduces security holes/lapses in individual systems or the environments as a whole. A security infrastructure is key to the success or failure of a grid environment.
- *System Status and Fault Tolerance* – To provide a reliable and robust environment it is important that a means of monitoring resources and applications is provided. To accomplish this task, tools that monitor resources and application need to be deployed.

Initiative	Focus and Technologies Developed
DISCWorld	It is an infrastructure for service-based metacomputing across LAN and WAN clusters. It allows remote users to login to this environment over the WWW and request access to data, and also to invoke services or operations on the available data [25] – dhpc.adelaide.edu.au/Projects/DISCWorld/
Nimrod/G & GRACE	A global scheduler (resource broker) for parametric computing over a enterprise wide clusters or computational grids – www.dgs.monash.edu.au/~rajkumar/ecogrid

Table 2: Major Australian Grid Computing Efforts

- *Resource Management and Scheduling* – The management of processor time, memory, network, storage, and other components in a grid is clearly very important. The overall aim is to efficiently and effectively schedule the applications that need to utilize the available resources in the metacomputing environment. From a user's point of view, resource management and scheduling should be transparent; their interaction with it being confined to a manipulating mechanism for submitting their application. It is important in a grid that a resource management and scheduling service can interact with those that may be installed locally.
- *Computational Economy and Resource Trading* – The grid is constructed by coupling resources distributed across various organizations and administrative domains and may be owned by different organisations. The motivations or incentives for contributing resources towards building grid, to date, has been driven by public good, prizes, fun, fame, or collaborative advantage. This is clearly evident from the construction of public or research test-beds such as Distributed.net [27], SETI@Home [26], GUSTO [14], and DAS [30]. The computational resource contributors to these test-beds are mostly motivated by the aforementioned reasons. The chances for getting access to such computational test-beds for solving commercial problems are rarely possible. This necessitates the need for a mechanism where one can buy compute power on-demand from computational grids or resource owners.

  In order to push the concept of grid into mainstream computing, we need a mechanism that motivates everyone on the Internet to contribute their machines (idle) resources. One of the best mechanisms for achieving this is supporting the concept of computational economy in building and management of grid resources [10]. It allows resource owners to earn money by letting others use their (idle) computational resources for solving their problems. In such industrial strength, commercial computational grid, the resource owners act as sellers and the users act as buyers. The pricing of resources will be driven by demand and supply and this is one of the best mechanisms to regulate and control access to computational resources.

Initiative	Focus and Technologies Developed
distributed.net	An experiment that uses Internet-connected computers to crack RSA encryption algorithms – `www.distributed.net`
SETI@Home	A scientific experiment that uses Internet-connected computers in the search for extraterrestrial intelligence - setiathome.ssl.berkeley.edu
Compute Power Grid	A portal and an economic based resource management infrastructure for computing on internet-wide resources that enables portal supercomputing – www.computepower.net

Table 3: Major Public Grid Computing Efforts

  The grid resource management systems must dynamically trade for the best resources based on a metric of the price and performance available and schedule computations on these resources such that they meet user requirements. The grid middleware needs to offer services that help resource brokers and resource owners to trade for resource access [10].
- *Programming Tools and Paradigms* – Grid applications (multi-disciplinary applications) couple resources that cannot be replicated at a single site even or may be globally located for other practical reasons. A grid should include interfaces, APIs, utilities and tools so as to provide a rich development environment. Common scientific languages such as C, C++, and Fortran should be available, as should application-level interfaces like MPI and PVM. A range of programming

paradigms should be supported, such as message passing and distributed shared memory. In addition, a suite of numerical and other commonly used libraries should be available.

- *User and Administrative GUI* – The interfaces to the services and resources available should be intuitive and easy to use. In addition, they should work on a range of different platforms and operating systems. They also need take advantage of web technologies to offer a view of portal supercomputing. The web-centric approach to access supercomputing resources should enable users to access any resource from anywhere over any platform at any time. That means, the users should be allowed to submit their jobs to computational resources through a web interface from any of the accessible platforms such as PCs, laptops, PDA, etc. thus supporting the ubiquitous access to the grid. The provision of access to scientific applications through the web (e.g., RWCP's PAPIA (Parallel Protein Information Analysis) system [29]) leads to the creation science portals.

III. Grid Computing Projects

There are many grid projects worldwide. Table 1-6 (this is not an exhaustive) lists some of the most significant projects. Due to the limited on the size of this paper it is impossible to describe all here. A more complete listing can be found in [11] [12]. This section presents some of the current grid projects representative of the grid technology approaches. Moreover, a short description of Grid and E-Grid Forums, two initiatives intended to promote and develop grid technologies and applications is given at the end of this section. The projects briefly detailed and reviewed in this paper include the following:

- USA: Globus, Legion, WebFlow, NetSolve, and NASA IPG.
- Asia/Japan: Ninf and Bricks
- Australia: Nimrod/G and DISCWorld.
- Europe: UNICORE, CERN Data Grid, MOL, Globe, DAS, MetaMPI.

Initiative	Focus and Technologies Developed
UNICORE	The UNiform Interface to Computer Resources aims to deliver software that allows users to submit jobs to remote high performance computing resources – `www.fz-juelich.de/unicore`
MOL	Metacomputer OnLine is a toolbox for the coordinated use of WAN/LAN connected systems. MOL aims at utilizing multiple WAN-connected high performance systems for solving large-scale problems that are intractable on a single supercomputer – www.uni-paderborn.de/pc2/projects/mol
METODIS	Metacomputing Tools for Distributed Systems – www.hlrs.de/structure/organisation/par/projects/metodis/
Globe	Globe is a research project aiming to study and implement a powerful unifying paradigm for the construction of large-scale wide area distributed systems: distributed shared objects – www.cs.vu.nl/~steen/globe
Poznan Metacomputing	Poznan Centre works on development of tools and methods for metacomputing - www.man.poznan.pl/metacomputing/
CERN Data Grid	This project aims to develop middleware and tools necessary for the data-intensive applications of high-energy physics - grid.web.cern.ch/grid/
MetaMPI	MetaMPI supports the coupling of heterogeneous MPI systems, thus allowing parallel applications developed using MPI to be run on grids without alteration – www.lfbs.rwth-aachen.de/~martin/MetaMPICH/
DAS	This is a wide-area distributed cluster, used for research on parallel and distributed computing by five Dutch universities – www.cs.vu.nl/das
JaWs	JaWS is an economy-based computing model where both resource owners and programs using these resources place bids to a central marketplace that generates leases of use – roadrunner.ics.forth.gr:8080/

Table 4: Major European Grid Computing Efforts

A. Globus

Globus [13][15] provides a software infrastructure that enables applications to handle distributed, heterogeneous computing resources as a single virtual machine. The Globus project is a U.S. multi-institutional research effort that seeks to enable the construction of computational grids. A computational grid, in this context, is a hardware and software infrastructure that provides dependable, consistent, and pervasive access to high-end computational capabilities, despite the geographical distribution of both resources and users. A central element of the Globus system is the Globus Metacomputing Toolkit (GMT), which defines the basic services and capabilities required to construct a computational grid. The toolkit consists of a set of components that implement basic services, such as security, resource location, resource management, and communications.

It is necessary for computational grids to support a wide variety of applications and programming paradigms. Consequently, rather than providing a uniform programming model, such as the object-oriented model, the GMT provides a bag of services from which developers of specific tools or applications can use to meet their own particular needs. This methodology is only possible when the services are distinct and have well-defined interfaces (API) that can be incorporated into applications or tools in an incremental fashion.

Globus is constructed as a layered architecture in which high-level global services are built upon essential low-level core local services. The Globus toolkit is modular, and an application can exploit Globus features, such as resource

management or information infrastructure, without using the Globus communication libraries. The GMT currently consists of the following:

- Resource allocation and process management (GRAM)
- Unicast and multicast communications services (Nexus)
- Authentication and related security services (GSI)
- Distributed access to structure and state information (MDS)
- Monitoring of health and status of system components (HBM)
- Remote access to data via sequential and parallel interfaces (GASS)
- Construction, caching, and location of executables (GEM)
- Advanced Resource Reservation and Allocation (GARA)

Initiative	Focus and Technologies Developed
Globus	This project is developing basic software infrastructure for computations that integrate geographically distributed computational and information resources – www.globus.org
Legion	Legion is an object-based metasystem. Legion supports transparent scheduling, data management, fault tolerance, site autonomy, and a wide range of security options – legion.virginia.edu
JAVELIN	Javelin: Internet-Based Parallel Computing Using Java – www.cs.ucsb.edu/research/javelin/
AppLes	This is an application-specific approach to scheduling individual parallel applications on production heterogeneous systems – www.infospheres.caltech.edu/
NASA IPG	The Information Power Grid is a testbed that provides access to a grid – a widely distributed network of high performance computers, stored data, instruments, and collaboration environments – www.ipg.nasa.gov
Condor	The Condor project aims is to develop and deploy, and evaluate mechanisms and policies that support high throughput computing (HTC) on large collections of distributed computing resources – www.cs.wisc.edu/condor/
Harness	Harness builds on the concept of the virtual machine and explores dynamic capabilities beyond what PVM can supply. It focused on developing three key capabilities: Parallel plug-ins, Peer-to-peer distributed control, and multiple virtual machines – www.cs.wisc.edu/condor/
NetSolve	NetSolve is a project that aims to bring together disparate computational resources connected by computer networks. It is a RPC based client/agent/server system that allows one to remotely access both hardware and software components – www.cs.utk.edu/netsolve/

Table 5: Major American (USA) Grid Computing Efforts

Globus can be viewed as a metacomputing framework based on a set of APIs to the underlying services. Globus provides application developers with a pragmatic means of implementing a range of services to provide a wide-area application execution environment.

B. Legion

Legion [16][17] is an object-based metasystem developed at the University of Virginia. Legion provides the software infrastructure so that a system of heterogeneous, geographically distributed, high performance machines can interact seamlessly. Legion attempts to provide users, at their workstations, with a single, coherent, virtual machine. Legion is organized by classes and metaclasses (classes of classes). In the Legion system:

- *Everything is an object* - Objects represent all hardware and software components. Each object is an active process that responds to method invocations from other objects within the system. Legion defines an API for object interaction, but not the programming language or communication protocol.
- *Classes manage their instances* - Every Legion object is defined and managed by its own active class object. Class objects are given system-level capabilities; they can create new instances, schedule them for execution, activate or deactivate an object, as well as provide state information to client objects.
- *Users can define their own classes* - As in other object-oriented systems users can override or redefine the functionality of a class.

This feature allows functionality to be added or removed to meet a user's needs. Core objects - Legion defines the API to a set of core objects that support the basic services needed by the metasystem. The Legion system has the following set of core object types:

- Classes and Metaclasses – Classes can be considered managers and policy makers. Metaclasses are classes of classes.
- Host objects – Host objects are abstractions of processing resources, they may represent a single processor or multiple hosts and processors.
- Vault objects – Vault objects represents persistent storage, but only for the purpose of maintaining the state of Object Persistent Representation (OPR).
- Implementation Objects and Caches – Implementation objects hide the storage details of object implementations and can be thought of as equivalent to executable files in UNIX. Implementation cache objects provide objects with a cache of frequently used data.
- Binding Agents – A binding agent maps object IDs to physical address. Binding agents can cache bindings and organize themselves in hierarchies and software combining trees.
- Context objects and Context spaces – Context objects map context names to Legion object IDs, allowing users to name objects with arbitrary-length string names. Context spaces consist of directed graphs of context objects that name and organize information.

A Legion object is an instance of its class. Objects are independent, active, and capable of communicating with each other via unordered non-blocking calls. Like other object-oriented systems, the set of methods of an object describes its interface. The Legion interfaces are described in an Interface Definition Language (IDL).

Legion takes a different approach to provide a metacomputing environment: it encapsulates all its components as objects. The methodology used has all the normal advantages of an object-oriented approach, such as data abstraction, encapsulation, inheritance, and polymorphism. It can be argued that many aspects of this object-oriented approach potentially make it ideal for designing and implementing a complex environment such as a metacomputer. However, using an object-oriented methodology does not come without a raft of problems, many of these is tied-up with the need for Legion to interact with legacy applications and services. In addition, as Legion is written in Mentat Programming Language (MPL), it is necessary to "port" MPL onto each platform before Legion can be installed.

C. WebFlow

WebFlow [18][19] is a computational extension of the Web model that can act as a framework for the wide-area distributed computing and metacomputing. The main goal of the WebFlow design was to build a seamless framework for publishing and reusing computational modules on the Web so that end users, via a Web browser, can engage in composing distributed applications using WebFlow modules as visual components and editors as visual authoring tools. Webflow has a three-tier Java-based architecture that can be considered a visual dataflow system. The front-end uses applets for authoring, visualization, and control of the environment. WebFlow uses servlet-based middleware layer to manage and interact with backend modules such as legacy codes for databases or high performance simulations. Webflow is analogous to the Web. Web pages can be compared to WebFlow modules and hyperlinks that connect Web pages to inter-modular dataflow channels. WebFlow content developers build and publish modules by attaching them to Web servers. Application integrators use visual tools to link outputs of the source modules with inputs of the destination modules, thereby forming distributed computational graphs (or compute-webs) and publishing them as composite WebFlow modules. A user activates these compute-webs by clicking suitable hyperlinks, or customizing the computation either in terms of available parameters or by employing some high-level commodity tools for visual graph authoring. The high performance backend tier is implemented using the Globus toolkit:

- The Metacomputing Directory Services (MDS) is used to map and identify resources.
- The Globus Resource Allocation Manager (GRAM) is used to allocate resources.
- The Global Access to Secondary Storage (GASS) is used for a high performance data transfer.

With WebFlow, new applications can be composed dynamically from reusable components just by clicking on visual module icons, dragging them into the active WebFlow editor area, and linking them by drawing the required connection lines. The modules are executed using Globus components combined with the pervasive commodity services where native high performance versions are not available. The prototype WebFlow system is based on a mesh of Java-enhanced Web Servers (Apache), running servlets that manage and coordinate distributed computation. This management infrastructure is implemented by three servlets: Session Manager, Module Manager, and Connection Manager. These servlets use URL addresses and can offer dynamic information about their services and current state. Each management servlet can communicate with others via sockets. The servlets are persistent and application-independent. Future implementations of WebFlow will use emerging standards for distributed objects and take advantage of commercial technologies, such as the CORBA [32] as the base distributed object model. WebFlow takes a different approach to both Globus and Legion. It is implemented in a hybrid manner using a three-tier architecture that encompasses both the Web and third party backend services. This approach has a number of advantages, including the ability to ``plug-in" to a diverse set of backend services. For example, many of these services are currently supplied by the Globus toolkit, but they could be replaced with components from CORBA or Legion. WebFlow also has the advantage that it is more portable and can be installed anywhere a Web server supporting servlets is capable of running.

D. NetSolve

NetSolve [20][21][22] is a client/server application designed to solve computational science problems in a distributed environment. The Netsolve system is based around loosely coupled distributed systems, connected via a LAN or WAN. Netsolve clients can be written in C and Fortran, use Matlab or the Web to interact with the server. A Netsolve server can use any scientific package to provide its computational software. Communications within Netsolve is via sockets. Good performance is ensured by a load-balancing policy that enables NetSolve to use the computational resources available as efficiently as possible. NetSolve offers the ability to search for computational resources on a network, choose the best one available, solve a problem (with retry for fault-tolerance), and return the answer to the user.

E. NASA Information Power Grid (IPG)

The NAS Systems Division is leading the effort to build and test NASA's Information Power Grid (lPG) [38], a network of high performance computers, data storage devices, scientific instruments, and advanced user interfaces. The overall mission of the IPG is to provide NASA's scientific and engineering communities a substantial increase in their ability to solve problems that depend on use of large-scale and/or distributed resources.

The project team is focused on creating an infrastructure and services to locate, combine, integrate, and manage resources from across NASA centers. An important goal of the IPG is to produce a common view of these resources, and at the same time provide for distributed management and local control. The IPG team at NAS is working to develop:

- Independent but consistent tools and services that support a range of programming environments used to build applications in widely distributed systems.
- Tools, services, and infrastructure for managing and aggregating dynamic collections of resources: processors, data storage/information systems, communications systems, real-time data sources and instruments, as well as human collaborators.
- Facilities for constructing collaborative, application-oriented workbenches and problem solving environments across NASA, based on the IPG infrastructure and applications.
- A common resource management approach that addresses areas such as systems management, user identification, resource allocations, accounting, and security.
- An operational grid environment that incorporates major computing and data resources at multiple NASA sites in order to provide an infrastructure capable of routinely addressing larger scale, more diverse, and more transient problems than is currently possible.

The starting point for IPG "middleware" is the Globus Metacomputing Toolkit. This IPG middleware will make its systems interoperable by providing a set of commands that lets researchers execute computational jobs on remote systems.

Initiative	Focus and Technologies Developed
Ninf	Ninf allows users to access computational resources including hardware, software and scientific data distributed across a wide area network with an easy-to-use interface – ninf.etl.go.jp
Bricks	Bricks is a performance evaluation system that allows analysis and comparison of various scheduling schemes on a typical high-performance global computing setting – matsu-www.is.titech.ac.jp/~takefusa/bricks/

Table 6: Major Japanese Grid Computing Efforts

F. NINF

The Network Infrastructure [36][37] for global computing (Ninf) is a client/server- based system that allows access to multiple remote compute and database servers. Ninf clients can semi-transparently access remote computational resources from languages such as C and Fortran. A programmer is able to build a global computing application by using the Ninf remote libraries as its components, without being aware of the complexities of the underlying system they are programming.

G. Nimrod/G Resource Broker and GRACE

Nimrod is a tool for parametric computing on clusters and it provides a simple *declarative* parametric modeling language for expressing a parametric experiment [8]. Domain experts can easily create a *plan* for a parametric computing (task farming) and use the Nimrod runtime system to submit, run, and collect the results from multiple computers (cluster nodes). Nimrod has been used to run applications ranging from bio-informatics and operations research, to the simulation of business processes. A reengineered version of Nimrod, called Clustor, has been commercialized by Active Tools [28]. However, research on Nimrod has been continued, to address its use in the global computational grid environment and to overcome shortcomings of the earlier system.

Nimrod has been used successfully with a static set of computational clusters, but is unsuitable as implemented in the large-scale dynamic context of computational grids, where resources are scattered across several administrative domains, each with their own user policies, employing their own queuing system, varying access cost and processing power. These shortcomings are addressed by a new system called Nimrod/G [7][9] that uses the Globus [13] middleware services for dynamic resource discovery and dispatching jobs over wide-area distributed systems called computational grids.

Nimrod/G allows scientists and engineers to model whole parametric experiments and transparently stage the data and program at remote sites, and run the program on each element of a data set on different machines and finally gather results from remote sites to the user site. The user need not worry about the way in which the complete experiment is set up, data or executable staging, or management. The user can also set the deadline by which the results are needed and the Nimrod/G broker tries to find the cheapest computational resources available in the grid and use them so that the user deadline is met and cost of computation is kept to a minimum.

The current focus of the Nimrod/G project team is on the use of economic theories in grid resource management and scheduling as part of a new framework called GRACE (Grid Architecture for Computational Economy) [10]. The components that make up GRACE include global scheduler (broker), bid-manager, directory server, and bid-server working closely with grid middleware and fabrics. The GRACE infrastructure also offers generic interfaces (APIs) that the grid tools and applications programmers can use to develop software supporting the computational economy. The grid resource brokers such as (Nimrod/G) uses GRACE services to dynamically trade with resources owner agents to select those resources that offer low-cost access services yet meet the user requirements.

H. UNICORE

UNICORE (UNiform Interface to COmputer REsources) [23][24] is a project funded by the German Ministry of Education and Research. A consortium of people from

universities, national research laboratories, software industry, and computer vendors develops UNICORE. Initially, it was a two years project ending in December 1999 but there is a plan to retarget it and extend it for another two/three years. UNICORE main focus is in providing a uniform interface for job preparation and control that offers seamless and secure access to supercomputer resources. The idea behind UNICORE is to support the users by hiding the system and site-specific idiosyncrasies and by helping to develop distributed applications.

Distributed applications within UNICORE are defined as multi-part applications where the different parts may run on different computer systems asynchronously or sequentially synchronized. A UNICORE job contains a multi-part application as described above augmented by the information about the destination systems, the resource requirements, and the dependencies between the different parts. From a structural viewpoint a UNICORE job is a recursive object containing job groups and tasks. Job groups themselves consist of other job groups and tasks. UNICORE jobs and job groups carry the information of the destination system for the included tasks. A task is the unit, which boils down to a batch job for the destination system.

The design goals for UNICORE include a uniform and easy to use GUI, an open architecture based on the concept of an abstract job, a consistent security architecture, minimal interference with local administrative procedures, exploitation of existing and emerging technologies, zero-administration user interface through standard Web browser and Java applets, and a production ready prototype within two years. UNICORE is designed to support batch jobs, it does not allow for interactive processes. At the application level asynchronous metacomputing is supported allowing for independent and dependent parts of a UNICORE job to be executed on a set of distributed systems. The user is provided with a unique UNICORE user-id to uniformly get access to all UNICORE sites. An intuitive GUI allows job preparation and control. It should be noted that the prototype excludes metacomputing at the application level (synchronous metacomputing), resource brokerage, and interactive applications including application steering. UNICORE has laid a solid basis for seamless computing and is well accepted by non-expert users. Future developments will integrate features to support more users with more sophisticated applications.

I. Grid Applications

Although wide-area distributed supercomputing has been a popular application of the grid, there are a number of other applications that can benefit from it [4]. These include collaborative engineering, high-throughput computing (large-scale simulation and parameter studies), remote software access, data-intensive computing, and on-demand computing. Some of these applications were demonstrated at SC'98 [41] by the international computational science community.

Some applications that have been designed and developed using message passing interfaces, and to run on parallel platforms can be executed on computational grids without porting – message passing interfaces are available for grid environments. A number of computational physics applications are not grid-enabled [39]. Today, large-scale parameter-study (embarrassingly parallel) applications are exploiting computational grid resources heavily and have been termed as killer applications [7][40]. Projects, such as SETI@Home [26] and Distributed.Net [27], build grids by linking multiple low-end computational resources, such as PCs, across the Internet to detect extraterrestrial intelligence and crack security algorithms respectively. The nodes in these grids work simultaneously on different parts of the problem and pass results to central system for post-processing.

Grid resources can be used to solve grand challenge problems in areas such as biophysics, chemistry, biology, high energy physics, data mining, financial analysis, nuclear simulations, material science, chemical engineering, environmental studies, molecular biology, structural analysis, mechanical CAD/CAM, weather prediction, astrophysics, scientific instrumentation, and so on.

IV. Conclusions and Future Trends

There are currently a large number of projects and diverse range of emerging grid developmental approaches being pursued. These systems range from metacomputing frameworks to application testbeds, and from collaborative environments to batch submission mechanisms.

It is very difficult to predict the future and this is particularly true in a field such as information technology where the technological advances are moving very fast. Hence, it is not an easy task to forecast what will be the "dominant" grid approach in the next future. Windows of opportunity for ideas and products seem to open and close in the seeming "blink of the eye". However, some trends are evident. One of those is growing interest in the use of Java [31] for network computing. In fact, it is interesting to note some that some of the projects described in this paper are using Java and the Web as the communications infrastructure.

The Java programming language successfully addresses several key issues that plague the development of grid environments, such as heterogeneity and security. It also removes the need to install programs remotely; the minimum execution environment is a Java-enabled Web browser. Java, with its related technologies and growing repository of tools and utilities, is having a huge impact on the growth and development of grid environments. From a relatively slow start, the development of metacomputers is accelerating fast with the advent of these new and emerging technologies. It is very hard to ignore the presence of the sleepy giant CORBA [32] in the background. We believe that frameworks incorporating CORBA services will be

very influential on the design of grid environments in the future.

Another promising Java technology is Jini [33][34]. The Jini architecture exemplifies a new approach to computing systems that is netcentric. By replacing the notion of peripherals and applications with that of network-available devices and services with clients that use those services. Jini helps breakdown the conventional view of what a computer is, while including new classes of devices working together in a federated architecture. The ability to move code from the service to its client is the core difference between the Jini environment and other distributed systems, such as the Common Object Request Broker Architecture (CORBA) and the Distributed Common Object Model (DCOM) [35].

Whatever the technology or computing paradigm that becomes influential or most popular, it can be guaranteed that at some stage in the future its star will wane. Historically, in the computing field, this fact can be repeatedly observed. The lesson from this observation must therefore be drawn that, in the long term, backing only one technology can be an expensive mistake. The framework that provides a grid environment must be adaptable, malleable, and extensible. As technology and fashions change it is crucial that a grid computing environment evolves with them.

Larry Smarr observes in [5] that grid computing has serious social consequences and is going to have as revolutionary an effect as railroads did in the American mid-West in the early nineteenth century. Instead of a 30 to 40 year lead-time to see its effects, however, its impact is going to be much faster. He concludes that the effects of computational grids are going to change the world so quickly that mankind will struggle to react and change in the face of the challenges and issues they present. So, at some stage in the future, our computing needs will be satisfied in the same pervasive and ubiquitous manner that we use the electricity power grid. The analogies with the generation and delivery of electricity are hard to ignore, and the implications are enormous. In fact, the computational grid is analogous to electricity (power) grid and the vision is to offer a (almost) dependable, consistent, pervasive, and inexpensive access to high-end resources irrespective their location of physical existence and the location of access.

ACKNOWLEDGMENTS

The authors would like to acknowledge all developers of the systems or projects described in this paper. In the past we had intellectual communication and exchanged views on this upcoming technology with David Abramson (Monash), Fran Berman (UCSD), David C. DiNucci (Elepar), Jack Dongarra (UTK/ORNL), Ian Foster (ANL), Geoffrey Fox (Syracuse), Wolfgang Gentzsch (GRIDware), Jon Giddy (DSTC), Al Geist (ORNL), and Tom Haupt (Syracuse). We thank all of them for sharing their knowledge with us.

REFERENCES

[1] Lyster P., Bergman L., Li P., Stanfill D., Crippe B., Blom R., Pardo C., Okaya D., CASA Gigabit Supercomputing Network: CALCRUST three-dimensional real-time multi-dataset rendering, *Proceedings of Supercomputing '92*

[2] Catlett C., Smarr L., Metacomputing, *Communications of the ACM*, vol. 35(6), pages 44-52, 1992.

[3] Smarr L., Infrastructure for Science Portals, *IEEE Internet Computing*, January/February 2000, 71-73.

[4] Leinberger W., Kumar V., Information Power Grid: The new frontier in parallel computing? *IEEE Concurrency*, October-December 1999, 75-84

[5] Foster I. and Kesselman C. (editors), The Grid: Blueprint for a Future Computing Infrastructure, *Morgan Kaufmann Publishers*, USA, 1999.

[6] Baker M., Fox G., Metacomputing: Harnessing Informal Supercomputers, *High Performance Cluster Computing: Architectures and Systems*, Buyya, R. (ed.), Volume 1, Prentice Hall PTR, NJ, USA, 1999.

[7] Abramson D., Giddy J., and Kotler L., High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? *International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE Computer Society Press, 2000.

[8] Nimrod/G - http://www.dgs.monash.edu.au/~davida/nimrod.html

[9] Buyya R, Abramson D, and Giddy J, Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid, *The 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia'2000)*, Beijing, China. IEEE Computer Society Press, USA, 2000.

[10] Buyya R, Abramson D, and Giddy J, Economy Driven Resource Management Architecture for Computational Power Grids, *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000)*, Las Vegas, USA, 2000.

[11] Gentzsch W. (editor), Special Issue on Metacomputing: From Workstation Clusters to Internet computing, *Future Generation Computer Systems*, No. 15, North Holland, 1999.

[12] Grid Computing Infoware - http://www.gridcomputing.com/

[13] Globus - http://www.globus.org/

[14] Globus Testbeds - http://www-fp.globus.org/testbeds/

[15] Foster I. and Kesselman C., Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications*, 11(2): 115-128, 1997.

[16] Legion - http://legion.virginia.edu/

[17] Grimshaw A., Wulf W. et al., The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, vol. 40(1), January 1997.

[18] WebFlow – http://osprey7.npac.syr.edu:1998/iwt98/products/webflow/

[19] Haupt T., Akarsu E., and Fox G., Furmanski W, Web Based Metacomputing, Special Issue on Metacomputing, *Future Generation Computer Systems*, North Holland 1999.

[20] NetSolve – http://www.cs.utk.edu/~casanova/NetSolve/

[21] Casanova H. and Dongarra, J., *NetSolve: A Network Server for Solving Computational Science Problems*, *Inernational Journal of Supercomputing Applications and High Performance Computing*, Vol. 11, No. 3, 1997.

[22] Casanova H., Kim M., Plank J., and Dongarra J., Adaptive Scheduling for Task Farming with Grid Middleware, *International Journal of Supercomputer Applications and High-Performance Computing*, 1999.

[23] Almond J., Snelling D., UNICORE: uniform access to supercomputing as an element of electronic commerce, *Future Generation Computer Systems*, 15(1999) 539-548, NH-Elsevier.

[24] UNICORE - http://www.unicore.org

[25] Hawick K., James H., Silis A, Grove D., Kerry K., Mathew J., Coddington P., Patten C., Hercus J., Vaughan F., DISCWorld: An Environment for Service-Based Metacomputing, *Future Generation Computing Systems (FGCS)*, Vol. 15, 1999.

[26] SETI@Home – http://setiathome.ssl.berkeley.edu/

[27] Distributed.Net – http://www.distributed.net/

[28] Active Tools – http://www.activetools.com

[29] PAPIA: Parallel Protein Information Analysis system – http://www.rwcp.or.jp/papia/
[30] Distributed ASCI Supercomputer (DAS) – http://www.cs.vu.nl/das/
[31] Arnold, K., and Gosling, J. The Java Programming Language, *Addison-Wesley, Longman*, Reading, Mass., 1996.
[32] Object Management Group, Common Object Request Broker: Architecture and Specification, *OMG Doc*. No. 91.12.1, 1991.
[33] Waldo J., The JINI Architecture for Network-Centric Computing, *Communications of the ACM*, Vol. 42, No. 7, July 1999.
[34] Sun Microsystems, Inc., *Jini architectural overview* – http://www.sun.com/jini/whitepapers/
[35] Rogerson, D. Inside COM, *Microsoft Press*, Redmond, Wash., 1997.
[36] Ninf – `http://ninf.etl.go.jp/`
[37] M. Sato, H. Nakada, S. Sekiguchi, S. Matsuoka, U. Nagashima, and H. Takagi, Ninf: A Network based Information Library for a Global World-Wide Computing Infrastructure, *Lecture Notes in Computer Science, High-Performance Computing and Networking*, Springer Verlag, pp. 491-502, 1997.
[38] NASA Information Power Grid (IPG) – http://www.ipg.nasa.gov
[39] Smarr L, Computational Physics in the Grid Computing Era, http://doug-pc.itp.ucsb.edu/online/numrel00/smarr1/
[40] Smallen, S., et al., Combining Workstations and Supercomputers to Support Grid Applications: The Parallel Tomography Experience, *9th Heterogenous Computing Workshop (HCW 2000@IPDPS)*, Cancun, Mexico.
[41] Brown, M., et al, The International Grid (iGrid): Empowering Global Research Community Networking Using High Performance International Internet Services, April 1999, http://www-fp.globus.org/documentation/papers.html.