The Evolution of the Grid

David De Roure[1], Mark A. Baker[2],
Nicholas R. Jennings[1] and Nigel R. Shadbolt[1]

[1]University of Southampton, UK
[2]University of Portsmouth, UK

Correspondence to David De Roure,
Department of Electronics and Computer Science,
University of Southampton, Southampton SO17 1BJ, UK

Tel +44 23 8059 2418, email dder@ecs.soton.ac.uk

Abstract

In this paper we describe the evolution of grid systems, identifying three generations: first generation systems which were the forerunners of the Grid as we recognise it today; second generation systems with a focus on middleware to support large scale data and computation; and third generation systems where the emphasis shifts to distributed global collaboration, a service oriented approach and information layer issues. In particular, we discuss the relationship between the Grid and the World Wide Web, and suggest that evolving web technologies will provide the basis for the next generation of the Grid. The latter aspect – which we define as the Semantic Grid – is explored in a companion paper.

1. Introduction

The last decade has seen a substantial change in the way we perceive and use computing resources and services. A decade ago, it was normal to expect ones computing needs to be serviced by localised computing platforms and infrastructures. This situation has changed; the change has been caused by, among other factors, the take-up of commodity computer and network components, the result of faster and more capable hardware and increasingly sophisticated software. A consequence of these changes has been the capability for effective and efficient utilization of widely distributed resources to fulfil a range of application needs.

As soon as computers are interconnected and communicating we have a distributed system, and the issues in designing, building and deploying distributed computer systems have now been explored over many years. An increasing number of research groups have been working in the field of wide-area distributed computing. These groups have implemented middleware, libraries and tools that allow the cooperative use of geographically distributed resources unified to act as a single powerful platform for the execution of a range of parallel and distributed applications. This approach to computing has been known by several names, such as metacomputing, scalable computing, global computing, Internet computing and

lately as grid computing.

More recently there has been a shift in emphasis. In [Foster01], the 'Grid problem' is defined as "Flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources". This view emphasizes the importance of information aspects, essential for resource discovery and interoperability. Current grid projects are beginning to take this further, from information to knowledge. These aspects of the grid are related to the evolution of web technologies and standards, such as XML to support machine-to-machine communication and the Resource Description Framework (RDF) to represent interchangeable metadata.

The next three sections identify three stages of grid evolution: first generation systems which were the forerunners of grid computing as we recognise it today; second generation systems with a focus on middleware to support large scale data and computation; and current third generation systems where the emphasis shifts to distributed global collaboration, a service oriented approach and information layer issues. Of course, the evolution is a continuous process and distinctions are not always clear-cut, but characterising the evolution helps identify issues and suggests the beginnings of a grid roadmap. In section 5 we draw parallels with the evolution of the World Wide Web and introduce the notion of the 'Semantic Grid' in which semantic web technologies provide the infrastructure for grid applications. A research agenda for future evolution is discussed in a companion paper [DeRoure02].

2. The Evolution of the Grid: The First Generation

The early Grid efforts started as projects to link supercomputing sites; at this time this approach was known as metacomputing. The origin of the term is believed to have been the CASA project, one of several US Gigabit testbeds around in 1989. Larry Smarr, the former NCSA Director, is generally accredited with popularising the term thereafter [Catlett92].

The early to mid 1990s mark the emergence of the early metacomputing or grid environments. Typically, the objective of these early metacomputing projects was to provide computational resources to a range of high performance applications. Two representative projects in the vanguard of this type of technology were FAFNER [FAFNER] and I-WAY [Foster97a]. These projects differ in many ways, but both had to overcome a number of similar hurdles, including communications, resource management, and the manipulation of remote data, to be able to work efficiently and effectively. The two projects also attempted to provide metacomputing resources from opposite ends of the computing spectrum. Whereas FAFNER was capable of running on any workstation with more than 4 Mbytes of memory, I-WAY was a means of unifying the resources of large US supercomputing centres.

2.1 FAFNER

The RSA public key encryption algorithm, invented by Rivest, Shamri and Adelman at MIT's Laboratory for Computer Science in 1976-77 [Rivest77], is widely used; for example, in the Secure Sockets Layer (SSL). The security of RSA is based on the premise that it is very difficult to factor extremely large numbers, in particular those with hundreds of digits. To keep abreast of the state of the art in factoring, RSA Data Security Inc. initiated the RSA

Factoring Challenge in March 1991. The Factoring Challenge provides a test-bed for factoring implementations and provides one of the largest collections of factoring results from many different experts worldwide.

Factoring is computationally very expensive. For this reason parallel factoring algorithms have been developed so that factoring can be distributed. The algorithms used are trivially parallel and require no communications after the initial setup. With this setup, it is possible that many contributors can provide a small part of a larger factoring effort. Early efforts relied on electronic mail to distribute and receive factoring code and information. In 1995, a consortium led by Bellcore Labs., Syracuse University and Co-Operating Systems started a project, factoring via the Web, known as FAFNER (Factoring via Network-Enabled Recursion).

FAFNER was set up to factor RSA130 using a new numerical technique called the Number Field Sieve (NFS) factoring method using computational Web servers. The consortium produced a Web interface to NFS. A contributor then used a Web form to invoke server side CGI (Common Gateway Interface) scripts written in Perl. Contributors could, from one set of Web pages, access a wide range of support services for the sieving step of the factorisation: NFS software distribution, project documentation, anonymous user registration, dissemination of sieving tasks, collection of relations, relation archival services and real-time sieving status reports. The CGI scripts produced supported cluster management, directing individual sieving workstations through appropriate day/night sleep cycles to minimize the impact on their owners. Contributors down-loaded and built a sieving software daemon. This then became their Web client that used HTTP protocol to GET values from and POST the resulting results back to a CGI script on the Web server.

Three factors combined to make this approach successful:

- The NFS implementation allowed even workstations with 4 Mbytes of memory to perform useful work using small bounds and a small sieve.
- FAFNER supported anonymous registration; users could contribute their hardware resources to the sieving effort without revealing their identity to anyone other than the local server administrator.
- A consortium of sites was recruited to run the CGI script package locally, forming a hierarchical network of RSA130 Web servers, which reduced the potential administration bottleneck and allowed sieving to proceed around the clock with minimal human intervention.

The FAFNER project won an award in TeraFlop challenge at Supercomputing 95 (SC95) in San Diego. It paved the way for a wave of web based metacomputing projects.

2.2 I-WAY

The Information Wide Area Year (I-WAY) was an experimental high performance network linking many high performance computers and advanced visualization environments. The I-WAY project was conceived in early 1995 with the idea not to build a network but to integrate existing high bandwidth networks. The virtual environments, datasets, and computers used resided at seventeen different U.S. sites and were connected by ten networks of varying bandwidths and protocols, using different routing and switching technologies.

The network was based on ATM technology, which at the time was an emerging standard. This network provided the wide area backbone for various experimental activities at SC95, supporting both TCP/IP over ATM and direct ATM-oriented protocols.

To help standardize the I-WAY software interface and management, the key sites installed point-of-presence (I-POP) servers to act as gateways to I-WAY. The I-POP servers were UNIX workstations configured uniformly and possessing a standard software environment called I-Soft. I-Soft attempted to overcome issues concerning heterogeneity, scalability, performance, and security. Each site participating in I-WAY ran an I-POP server. The I-POP server mechanisms that allowed uniform I-WAY authentication, resource reservation, process creation, and communication functions. Each I-POP server was accessible via the Internet and operated within its site's firewall. It also had an ATM interface that allowed monitoring and potential management of the site's ATM switch.

The I-WAY project developed a resource scheduler known as the Computational Resource Broker (CRB). The CRB consisted of user-to-CRB and CRB-to-local-scheduler protocols. The actual CRB implementation was structured in terms of a single central scheduler and multiple local scheduler daemons – one per I-POP server. The central scheduler maintained queues of jobs and tables representing the state of local machines, allocating jobs to machine and maintaining state information on the AFS file system (a distributed filesystem that enables co-operating hosts to share resources across both local area and wide area networks, based on the 'Andrew File System' originally developed at Carnegie-Mellon University).

In I-POP, security was handled by using a telnet client modified to use Kerberos authentication and encryption. In addition, the CRB acted as an authentication proxy, performing subsequent authentication to I-WAY resources on a user's behalf. With regards to filesystems, I-WAY used AFS to provide a shared repository for software and scheduler information. An AFS cell was set up and made accessible from only I-POPs. To move data between machines where AFS was unavailable, a version of remote copy was adapted for I-WAY.

To support user-level tools, a low-level communications library, Nexus [Foster96], was adapted to execute in the I-WAY environment. Nexus supported automatic configuration mechanisms that enabled it to choose the appropriate configuration depending on the technology being used, for example, communications via TCP/IP or AAL5 (the ATM adaptation layer for framed traffic) when using the Internet or ATM. The MPICH library (a portable implementation of the MPI message passing standard) and CAVEcomm (networking for the CAVE virtual reality system) were also extended to use Nexus.

The I-WAY project was application driven and defined several types of applications:

- Supercomputing,
- Access to Remote Resources,
- Virtual Reality,
- Video, Web, GII-Windows.

The I-WAY project was successfully demonstrated at SC'95 in San Diego. The I-POP servers were shown to simplify the configuration, usage and management of this type of wide area

computational testbed. I-Soft was a success in terms that most applications ran, most of the time. More importantly, the experiences and software developed as part of the I-WAY project have been fed into the Globus project (which we discuss in section 3.2.1).

2.3 A Summary of Early Experiences

Both FAFNER and I-Way attempted to produce metacomputing environments by integrating resources from opposite ends of the computing spectrum. FAFNER was a ubiquitous system that worked on any platform with a Web server. Typically, its clients were low end computers, whereas I-WAY unified the resources at multiple supercomputing centres.

The two projects also differed in the types of applications that could utilise their environments. FAFNER was tailored to a particular factoring application that was in itself trivially parallel and was not dependent on a fast interconnect. I-WAY, on the other hand, was designed to cope with a range of diverse high performance applications that typically needed a fast interconnect and powerful resources. Both projects, in their way, lacked scalability. For example, FAFNER was dependent on a lot of human intervention to distribute and collect sieving results, and I-WAY was limited by the design of components that made up I-POP and I-Soft.

FAFNER lacked a number of features that would now be considered obvious. For example, every client had to compile, link, and run a FAFNER daemon in order to contribute to the factoring exercise. FAFNER was really a means of task-farming a large number of fine-grain computations. Individual computational tasks were unable to communicate with one another, or with their parent Web-server. Likewise, I-WAY embodied a number of features that would today seem inappropriate. The installation of an I-POP platform made it easier to set up I-WAY services in a uniform manner, but it meant that each site needed to be specially set up to participate in I-WAY. In addition, the I-POP platform and server created one, of many, single points of failure in the design of the I-WAY. Even though this was not reported to be a problem, the failure of an I-POP would mean that a site would drop out of the I-WAY environment.

Notwithstanding the aforementioned features, both FAFNER and I-WAY were highly innovative and successful. Each project was in the vanguard of metacomputing and helped pave the way for many of the succeeding second generation Grid projects. FAFNER was the forerunner of the likes of SETI@home [SETI] and Distributed.Net [NET], and I-WAY for Globus [Foster97b] and Legion [Grimshaw97].

3. The Evolution of the Grid: The Second Generation

The emphasis of the early efforts in grid computing was in part driven by the need to link a number of US national supercomputing centres. The I-WAY project (see section 2.2) successfully achieved this goal. Today the grid infrastructure is capable of binding together more than just a few specialised supercomputing centres. A number of key enablers have helped make the Grid more ubiquitous, including the take up of high bandwidth network technologies and adoption of standards, allowing the Grid to be viewed as a viable distributed infrastructure on a global scale that can support diverse applications requiring

large-scale computation and data. This vision of the Grid was presented in [Foster98] and we regard this as the second generation, typified by many of today's grid applications.

There are three main issues that had to be confronted:

- Heterogeneity: a Grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across a potentially global expanse. As any cluster manager knows, their only truly homogeneous cluster is their first one!
- Scalability: a Grid might grow from few resources to millions. This raises the problem of potential performance degradation as the size of a Grid increases. Consequently, applications that require a large number of geographically located resources must be designed to be latency tolerant and exploit the locality of accessed resources. Furthermore, increasing scale also involves crossing an increasing number of organisational boundaries, which emphasises heterogeneity and the need to address authentication and trust issues. Larger scale applications may also result from the composition of other applications, which increases the 'intellectual complexity' of systems.
- 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. Resource managers or applications must tailor their behaviour dynamically so that they can extract the maximum performance from the available resources and services.

Middleware is generally considered to be the layer of software sandwiched between the operating system and applications, providing a variety of services required by an application to function correctly. Recently, middleware has re-emerged as a means of integrating software applications running in distributed heterogeneous environments. In a Grid, the middleware is used to hide the heterogeneous nature and provide users and applications with a homogeneous and seamless environment by providing a set of standardised interfaces to a variety of services.

Setting and using standards is also key to tackling heterogeneity. Systems use varying standards and system APIs, resulting in the need to port services and applications to the plethora of computer systems used in a grid environment. As a general principle, agreed interchange formats help reduce complexity, because n converters are needed to enable n components to interoperate via one standard, as opposed to n^2 converters for them to interoperate with each other.

In this section, we consider the second generation requirements, followed by representatives of the key second generation grid technologies: core technologies, distributed object systems, resource brokers and schedulers, complete integrated systems and peer-to-peer systems.

3.1 Requirements for the data and computation infrastructure

The data infrastructure can consist of all manner of networked resources ranging from computers and mass storage devices to databases and special scientific instruments. Additionally there are computational resources, such as supercomputers and clusters. Traditionally, it is the huge scale of the data and computation which characterises grid

applications.

The main design features required at the data and computational fabric of the Grid are:

- 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, for example, 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. 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 QoS 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 the Grid is reliably and easily obtained by those services requesting the information. The Grid information (registration and directory) services provide the mechanisms for registering and obtaining information about the structure, resources, services, status and nature of the environment.
- Naming Services – In a Grid, like in any other distributed system, names are used to refer to a wide variety of objects such as computers, services or data. The naming service provides a uniform name space across the complete distributed 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 optimisations to be implemented (such as the usage of caches).
- Security and Authorisation – 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 and that does not introduce 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, tools that monitor resources and applications need to be deployed.
- Resource Management and Scheduling – The management of processor time, memory, network, storage, and other components in a Grid are clearly important. The overall aim is the efficient and effective scheduling of the applications that need to utilise the available resources in the distributed environment. From a user's point of view, resource management and scheduling should be transparent and their

interaction with it should be confined to application submission. It is important in a grid that a resource management and scheduling service can interact with those that may be installed locally.

- User and Administrative GUI – The interfaces to the services and resources available should be intuitive and easy to use as well as being heterogeneous in nature. Typically, user and administrative access to grid applications and services are Web based interfaces.

3.2 Second Generation Core Technologies

There are growing numbers of Grid-related projects, dealing with areas such as infrastructure, key services, collaborations, specific applications and domain portals. Here we identify some of the most significant to date.

3.2.1 Globus

Globus [Foster97b] 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 Toolkit, 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 Globus Toolkit provides a bag of services 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 (APIs) 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 Globus Toolkit currently consists of the following (the precise set depends on Globus version):

- An HTTP-based 'Globus Toolkit Resource Allocation Manager' (GRAM) protocol is used for allocation of computational resources and for monitoring and control of computation on those resources.
- An extended version of the File Transfer Protocol, GridFTP, is used for data access; extensions include use of connectivity layer security protocols, partial file access, and management of parallelism for high-speed transfers.
- Authentication and related security services (GSI - Grid Security Infrastructure)
- Distributed access to structure and state information that is based on the Lightweight

Directory Access Protocol (LDAP). This service is used to define a standard resource information protocol and associated information model.
- Remote access to data via sequential and parallel interfaces (GASS - Global Access to Secondary Storage) including an interface to GridFTP.
- The construction, caching and location of executables (GEM - Globus Executable Management).
- Resource reservation and allocation (GARA - Globus Advanced Reservation and Allocation).

Globus has evolved from its original first generation incarnation as I-WAY, through version 1 (GT1) to version 2 (GT2). The protocols and services that Globus provided have changed as it has evolved. The emphasis of Globus has moved away from supporting just high performance applications towards more pervasive services that can support virtual organisations. The evolution of Globus is continuing with the introduction of the Open Grid Service Architecture (OGSA) [Foster02], a Grid architecture based on Web Services and Globus (see section 4.1 for details).

3.2.2 Legion

Legion [Grimshaw97] is an object-based 'metasystem', developed at the University of Virginia. Legion provided the software infrastructure so that a system of heterogeneous, geographically distributed, high performance machines could interact seamlessly. Legion attempted to provide users, at their workstations, with a single integrated infrastructure regardless of scale, physical location, language and underlying operating system.

Legion differed from Globus in its approach to providing to a Grid environment: it encapsulated all of its components as objects. This methodology has all the normal advantages of an object-oriented approach, such as data abstraction, encapsulation, inheritance and polymorphism.

Legion defined the APIs to a set of core objects that support the basic services needed by the metasystem. The Legion system had 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 represent persistent storage, but only for the purpose of maintaining the state of object persistent representation.
- Implementation Objects and Caches – Implementation objects hide details of storage object implementations and can be thought of as equivalent to an executable in UNIX.
- Binding Agents – A binding agent maps object IDs to physical addressees.
- Context objects and Context spaces – Context objects map context names to Legion object IDs, allowing users to name objects with arbitrary-length string names.

Legion was first released in November 1997. Since then the components that make up Legion have continued to evolve. In August 1998, Applied Metacomputing, was established to exploit Legion commercially. In June 2001, Applied Metacomputing was relaunched as Avaki

Corporation [Avaki].

3.3 Distributed Object Systems

The Common Object Request Broker Architecture (CORBA) is an open distributed object-computing infrastructure being standardised by the Object Management Group (OMG) [OMG]. CORBA automates many common network programming tasks such as object registration, location, and activation; request de-multiplexing; framing and error-handling; parameter marshalling and de-marshalling; and operation dispatching. Although CORBA provides a rich set of services, it does not contain the Grid level allocation and scheduling services found in Globus (see section 3.2.1), however, it is possible to integrate CORBA with the Grid.

The OMG has been quick to demonstrate the role of CORBA in the Grid infrastructure; e.g., through the 'Software Services Grid Workshop' held in 2001. Apart from providing a well-established set of technologies that can be applied to e-Science, CORBA is also a candidate for a higher level conceptual model. It is language-neutral and targeted to provide benefits on the enterprise scale, and is closely associated with the Unified Modelling Language (UML). One of the concerns about CORBA is reflected by the evidence of intranet rather than Internet deployment, indicating difficulty crossing organisational boundaries; e.g. operation through firewalls. Furthermore, real-time and multimedia support was not part of the original design.

While CORBA provides a higher layer model and standards to deal with heterogeneity, Java provides a single implementation framework for realising distributed object systems. To a certain extent the Java Virtual Machine (JVM) with Java-based applications and services are overcoming the problems associated with heterogeneous systems, providing portable programs and a distributed object model through remote method invocation (RMI). Where legacy code needs to be integrated, it can be 'wrapped' by Java code.

However, the use of Java in itself has its drawbacks, the main one being computational speed. This and other problems associated with Java (e.g. numerics and concurrency) are being addressed by the likes of the Java Grande Forum (a 'Grande Application' is 'any application, scientific or industrial, that requires a large number of computing resources, such as those found on the Internet, to solve one or more problems') [JavaGrande]. Java has also been chosen for UNICORE (see section 3.6.3). Thus what is lost in computational speed might be gained in terms of software development and maintenance times when taking a broader view of the engineering of Grid applications.

3.3.1 Jini and RMI

Jini [Jini] is designed to provide a software infrastructure that can form a distributed computing environment that offers network plug and play. A collection of Jini enabled processes constitutes a Jini community – a collection of clients and services all communicating by the Jini protocols. In Jini, applications will normally be written in Java and communicate using the Java Remote Method Invocation (RMI) mechanism. Even though Jini is written in pure Java, neither Jini clients nor services are constrained to be pure Java. They may include Java wrappers around non-Java code, or even be written in some other language altogether. This enables a Jini community to extend beyond the normal Java framework and link services

and clients from a variety of sources.

More fundamentally, Jini is primarily concerned with communications between devices (not what devices do). The abstraction is the service and an interface that defines a service. The actual implementation of the service can be in hardware, software, or both. Services in a Jini community are mutually aware and the size of a community is generally considered that of a workgroup. A community's lookup service (LUS) can be exported to other communities, thus providing interaction between two or more isolated communities.

In Jini a device or software service can be connected to a network and can announce its presence. Clients that wish to use such a service can then locate it and call it to perform tasks. Jini is built on RMI, which introduces some constraints. Furthermore, Jini is not a distributed operating system, as an operating system provides services such as file access, processor scheduling and user logins. The five key concepts of Jini are:

- Lookup – search for a service and download the code needed to access it;
- Discovery – spontaneously find a community and join;
- Leasing – time-bounded access to a service;
- Remote Events – service A notifies service B of A's state change. Lookup can notify all services of a new service;
- Transactions – used to ensure that a system's distributed state stays consistent.

3.3.2 The Common Component Architecture Forum

The Common Component Architecture Forum [Armstrong99] is attempting to define a minimal set of standard features that a high performance component framework would need to provide, or can expect, in order to be able to use components developed within different frameworks. Like CORBA it supports component programming, but it is distinguished from other component programming approaches by the emphasis on supporting the abstractions necessary for high-performance programming. The core technologies described in the previous section, Globus or Legion, could be used to implement services within a component framework.

The idea of using component frameworks to deal with the complexity of developing interdisciplinary high performance computing applications is becoming increasingly popular. Such systems enable programmers to accelerate project development by introducing higher-level abstractions and allowing code reusability. They also provide clearly defined component interfaces, which facilitate the task of team interaction: such a standard will promote interoperability between components developed by different teams across different institutions. These potential benefits have encouraged research groups within a number of laboratories and universities to develop, and experiment with prototype systems. There is a need for interoperability standards to avoid fragmentation.

3.4 Grid Resource Brokers and Schedulers

3.4.1 Batch and Scheduling Systems

There are several systems available whose primary focus is batching and resource scheduling.

It should be noted that all the packages listed here started life as systems for managing jobs or tasks on locally distributed computing platforms. A fuller list of the available software can be found elsewhere [Baker96][Jones96].

- Condor [Condor] is a software package for executing batch jobs on a variety of UNIX platforms, in particular those that would otherwise be idle. The major features of Condor are automatic resource location and job allocation, check pointing, and the migration of processes. These features are implemented without modification to the underlying UNIX kernel. However, it is necessary for a user to link their source code with Condor libraries. Condor monitors the activity on all the participating computing resources, those machines, which are determined to be available, are placed into a resource pool. Machines are then allocated from the pool for the execution of jobs. The pool is a dynamic entity – workstations enter when they become idle, and leave when they get busy.
- The Portable Batch System (PBS) [PBS] is a batch queuing and workload management system (originally developed for NASA). It operates on a variety of UNIX platforms, from clusters to supercomputers. The PBS job scheduler allows sites to establish their own scheduling policies for running jobs in both time and space. PBS is adaptable to a wide variety of administrative policies, and provides an extensible authentication and security model. PBS provides a GUI for job submission, tracking, and administrative purposes.
- The Sun Grid Engine (SGE) [SGE] is based on the software developed by Genias known as Codine/GRM. In the SGE, jobs wait in a holding area and queues located on servers provide the services for jobs. A user submits a job to the SGE, and declares a requirements profile for the job. When a queue is ready for a new job, the SGE determines suitable jobs for that queue and then dispatches the job with the highest priority or longest waiting time; it will try to start new jobs on the most suitable or least loaded queue.
- The Load Sharing Facility (LSF) is a commercial system from Platform Computing Corp. [PLATFORM]. LSF evolved from the Utopia system developed at the University of Toronto [Zhou93] and is currently the most widely used commercial job management system. LSF comprises distributed load sharing and batch queuing software that manages, monitors and analyses the resources and workloads on a network of heterogeneous computers, and has fault tolerance capabilities.

3.4.2 Storage Resource Broker

The Storage Resource Broker (SRB) [Rajasekar01] has been developed at San Diego Supercomputer Centre (SDSC) to provide "uniform access to distributed storage" across a range of storage devices, via a well-defined API. The SRB supports file replication, and this can occur either offline or on-the-fly. Interaction with the SRB is via a GUI. The SRB servers can be federated. The SRB is managed by an administrator, with authority to create user groups. A key feature of the SRB is that it supports metadata associated with a distributed file system, such as location, size and creation date information. It also supports the notion of application-level (or domain-dependent) metadata, specific to the content, which cannot be generalised across all data sets. In contrast with traditional network file systems, SRB is attractive for Grid applications in that it deals with large volumes of data, which can transcend individual storage devices, because it deals with metadata and takes advantage of

file replication.

3.4.3 Nimrod/G Resource Broker and GRACE

Nimrod-G is a Grid broker that performs resource management and scheduling of parameter sweep and task-farming applications [Buyya00a][Abramson00]. It consists of four components:

- A task-farming engine,
- A scheduler,
- A dispatcher,
- Resource agents.

A Nimrod-G task farming engine allows user-defined schedulers, customised applications or problem solving environments (e.g., ActiveSheets [Abramson01]) to be "plugged in", in place of default components. The dispatcher uses Globus for deploying Nimrod-G agents on remote resources in order to manage the execution of assigned jobs. The Nimrod-G scheduler has the ability to lease grid resources and services depending on their capability, cost, and availability. The scheduler supports resource discovery, selection, scheduling, and the execution of user jobs on remote resources. The users can set the deadline by which time their results are needed and the Nimrod-G broker tries to find the best resources available in the Grid and use them to meet the user's deadline and attempt to minimize the costs of the execution of the task.

Nimrod-G supports user-defined deadline and budget constraints for scheduling optimisations and manages the supply and demand of resources in the Grid using a set of resource trading services called GRACE (Grid Architecture for Computational Economy) [Buyya00b]. There are four scheduling algorithms in Nimrod-G [Abramson01]:

- The cost optimisation uses the cheapest resources to ensure that the deadline can be met and the computational cost is minimized.
- The time optimisation uses all the affordable resources to process jobs in parallel as early as possible.
- The cost-time optimisation is similar to cost optimisation, but if there are multiple resources with the same cost, it applies time optimisation strategy while scheduling jobs on them.
- The conservative time strategy is similar to the time-optimisation, but it guarantees that each unprocessed job has a minimum budget-per-job.

The Nimrod-G broker with these scheduling strategies has been used in solving large-scale data-intensive computing applications such as the simulation of ionisation chamber calibration [Abramson00a] and the molecular modelling for drug design [Buyya01].

3.5 Grid Portals

A Web portal allows application scientists and researchers to access resources specific to a particular domain of interest via a Web interface. Unlike typical Web subject portals, a Grid portal may also provide access to grid resources. For example, a grid portal may authenticate users, permit them to access remote resources, help them make decisions about scheduling jobs, and allow users to access and manipulate resource information obtained and stored on a

remote database. Grid portal access can also be personalised by the use of profiles, which are created and stored for each portal user. These attributes, and others, make grid portals the appropriate means for grid application users to access grid resources.

3.5.1 The NPACI HotPage

The NPACI HotPage [Hotpage] is a user portal that has been designed to be a single point-of-access to computer-based resources, to simplify access to resources that are distributed across member organisations and allows them to be viewed either as an integrated Grid system or as individual machines.

The two key services provided by the HotPage are information and resource access and management services. The information services are designed to increase the effectiveness of users. It provides links to:

- User documentation and navigation;
- News items of current interest;
- Training and consulting information;
- Data on platforms and software applications;
- Information resources, such as user allocations and accounts.

The above are characteristic of web portals. HotPage's interactive Web-based service also offers secure transactions for accessing resources and allows the user to perform tasks such as command execution, compilation, and running programs. Another key service offered by HotPage is that it provides status of resources and supports an easy mechanism for submitting jobs to resources. The status information includes:

- CPU load/percent usage;
- Processor node maps;
- Queue usage summaries;
- Current queue information for all participating platforms.

3.5.2 The SDSC Grid Port Toolkit

The SDSC grid port toolkit [GridPort] is a reusable portal toolkit that uses HotPage infrastructure. The two key components of GridPort are the web portal services and the application APIs. The web portal module runs on a web server and provides secure (authenticated) connectivity to the Grid. The application APIs provide a Web interface that helps end-users develop customised portals (without having to know the underlying portal infrastructure). GridPort is designed to allow the execution of portal services and the client applications on separate web servers. The GridPortal toolkit modules have been used to develop science portals for applications areas such as pharmacokinetic modelling, molecular modelling, cardiac physiology and tomography.

3.5.3 Grid Portal Development Kit

The Grid Portal Collaboration is an alliance between NCSA, SDSC and NASA IPG [NLANR]. The purpose of the Collaboration is to support a common set of components and utilities to make portal development easier and allow various portals to interoperate by using the same core infrastructure (namely the Grid Security Infrastructure (GSI) and Globus).

Example portal capabilities include the following:

- Either running simulations interactively or submitted to a batch queue.
- File transfer including: file upload, file download, and third party file transfers (migrating files between various storage systems).
- Querying databases for resource/job specific information.
- Maintaining user profiles that contain information about past jobs submitted, resources used, results information and user preferences.

The portal architecture is based on a three-tier model, where a client browser securely communicates to a Web server over a secure sockets (via https) connection. The Web server is capable of accessing various grid services using the Globus infrastructure. The Globus toolkit provides mechanisms for securely submitting jobs to a Globus gatekeeper, querying for hardware/software information using LDAP, and a secure PKI infrastructure using GSI.

The portals discussion in this sub-section highlights the characteristics and capabilities that are required in Grid environments.

3.6 Integrated Systems

As the second generation of grid components emerged, a number of international groups started projects that integrated these components into coherent systems. These projects were dedicated to a number of exemplar high-performance wide area applications. This section of the paper discusses a representative set of these projects.

3.6.1 Cactus

Cactus [Allen01] is an open source problem-solving environment designed for scientists and engineers. Cactus has a modular structure, which enables the execution of parallel applications across a range of architectures and collaborative code development between distributed groups. Cactus originated in the academic research community, where it was developed and used by a large international collaboration of physicists and computational scientists for black hole simulations.

Cactus provides a front-end to many core backend services, provided by, for example, Globus, the HDF5 parallel file I/O (HDF5 is a general purpose library and file format for storing scientific data), the PETSc scientific library (a suite of data structures and routines for parallel solution of scientific applications modelled by partial differential equations, using MPI) and advanced visualisation tools. The portal contains options to compile and deploy applications across distributed resources.

3.6.2 DataGrid

The European DataGrid project [DATAGRID], led by CERN, is funded by the European Union with the aim of setting up a computational and data-intensive Grid of resources for the analysis of data coming from scientific exploration [Hoschek00]. The primary driving application of the DataGrid project is the Large Hadron Collider (LHC), which will operate at CERN from about 2005 to 2015 and represents a leap forward in particle beam energy,

density, and collision frequency. This leap is necessary in order to produce some examples of previously undiscovered particles, such as the Higgs boson or perhaps super-symmetric quarks and leptons.

The LHC will present a number of challenges in terms of computing. The project is designing and developing scalable software solutions and testbeds in order to handle many Peta-bytes of distributed data, tens of thousands of computing resources (processors, disks, etc.), and thousands of simultaneous users from multiple research institutions. The main challenge facing the project is providing the means to share huge amounts of distributed data over the current network infrastructure. The DataGrid relies upon emerging Grid technologies that are expected to enable the deployment of a large-scale computational environment consisting of distributed collections of files, databases, computers, scientific instruments, and devices.

The objectives of the DataGrid project are:

- Implement middleware for fabric and grid management, including the evaluation, test, and integration of existing middleware and research and development of new software as appropriate.
- Deploy a large scale testbed.
- Provide production quality demonstrations.

The DataGrid project is divided into twelve work packages distributed over four working groups: testbed and infrastructure, applications, computational and DataGrid middleware, management and dissemination. The work emphasizes on enabling the distributed processing of data-intensive applications in the area of high-energy physics, earth observation, and bio-informatics.

The DataGrid is build on top of Globus and includes the following components:

- Job Description Language (JDL) – a script to describe the job parameters.
- User Interface (UI) – sends the job to the RB and receives the results.
- Resource Broker (RB) – locates and selects the target Computing Element (CE).
- Job Submission Service (JSS) – submits the job to the target CE.
- Logging and Book keeping (L&B) – records job status information.
- Grid Information Service (GIS) – Information Index about state of Grid fabric.
- Replica Catalogue – list of data sets and their duplicates held on Storage Elements (SE).

The DataGrid testbed 1 is currently available and being used for high energy physics experiments, and applications in Biology and Earth Observation. The final version of the DataGrid software is scheduled for release by the end of 2003

3.6.3 UNICORE

UNICORE (UNIform Interface to COmputer REsources) [Almond99] is a project funded by the German Ministry of Education and Research. The design goals of 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 through standard Java and Web technologies. UNICORE provides an interface for job preparation and secure submission to distributed supercomputer resources.

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 augmented by the information about 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. 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 main UNICORE components are:

- The Job Preparation Agent (JPA),
- The Job Monitor Controller (JMC),
- The UNICORE https server, also called the Gateway,
- The Network Job Supervisor (NJS),
- A Java applet-based GUI with an online help and assistance facility.

The UNICORE client enables the user to create, submit, and control jobs from any workstation or PC on the Internet. The client connects to a UNICORE gateway, which authenticates both the client and user, before contacting the UNICORE servers, which in turn manage the submitted UNICORE jobs. Tasks destined for local hosts are executed via the native batch sub-system. Tasks to be run at a remote site are transferred to peer UNICORE gateways. All necessary data transfers and synchronisations are performed by the servers. These servers also retain status information and job output, passing it to the client upon user request.

The protocol between the components is defined in terms of Java objects. A low-level layer called the UNICORE Protocol Layer (UPL) handles authentication, SSL (secure socket layer) communication and transfer of data as in-lined byte-streams and a high-level layer (the Abstract Job Object or AJO class library) contains the classes to define UNICORE jobs, tasks and resource requests. Third-party components, such as Globus, can be integrated into the UNICORE framework to extend its functionality.

UNICORE is being extensively used and developed for the EuroGrid [EuroGrid] project. EuroGrid is a project being funded by the European Commission. It aims to demonstrate the use of grids in selected scientific and industrial communities, address the specific requirements of these communities, and highlight their use in the areas of biology, meteorology and computer-aided engineering. The objectives of the EuroGrid project include the support of the EuroGrid software infrastructure, the development of software components, and demonstrations of distributed simulation codes from different application areas (biomolecular simulations, weather prediction, coupled CAE simulations, structural analysis and real-time data processing).

3.6.4 WebFlow

WebFlow [Akarsu98] [Haupt99] is a computational extension of the Web model that can act as a framework for wide-area distributed computing. The main design goal of WebFlow 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 could be considered a visual dataflow system. The front-end uses applets for authoring, visualization, and control of the environment. WebFlow uses a 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 built and published modules by attaching them to web servers. Application integrators used 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 activated 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 backend of WebFlow was implemented using the Globus Toolkit, in particular MDS, GRAM, and GASS.

WebFlow was based on a mesh of Java-enhanced web servers (Apache), running servlets that managed and coordinated distributed computation. This management infrastructure was implemented by three servlets: session manager, module manager, and connection manager. These servlets use URLs 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.

Various implementations of WebFlow were developed; one version used CORBA, as the base distributed object model. WebFlow has evolved into the Gateway Computational Web portal [Pierce02] and the Mississippi Computational Web Portal [Haupt02].

3.7 Peer-to-Peer Computing

One very plausible approach to address the concerns of scalability can be described as decentralisation, though this is not a simple solution. The traditional client-server model can be a performance bottleneck and a single point of failure, but is still prevalent because decentralisation brings its own challenges. Peer-to-Peer (P2P) computing [Clark01a] (as implemented, for example, by Napster [Napster], Gnutella [Gnutella], Freenet [Clarke01b] and JXTA [JXTA]) and Internet computing (as implemented, for example, by the SETI@home [SETI], Parabon [Parabon], and Entropia systems [Entropia]) are examples of the more general computational structures that are taking advantage of globally distributed resources.

In P2P computing, machines share data and resources, such as spare computing cycles and storage capacity, via the Internet or private networks. Machines can also communicate directly and manage computing tasks without using central servers. This permits P2P

computing to scale more effectively than traditional client-server systems that must expand a server's infrastructure in order to grow, and this 'clients as servers' decentralisation is attractive with respect to scalability and fault tolerance for the reasons discussed above. However, there are some obstacles to P2P computing:

- PCs and workstations used in complex P2P applications will require more computing power to carry the communications and security overhead that servers would otherwise handle.
- Security is an issue as P2P applications give computers access to other machines' resources (memory, hard drives, etc). Downloading files from other computers makes the systems vulnerable to viruses. For example, Gnutella users were exposed to the VBS_GNUTELWORM virus. Another issue is the ability to authenticate the identity of the machines with their peers.
- P2P systems have to cope with heterogeneous resources, such as computers using a variety of operating systems and networking components. Technologies such as Java and XML will help overcome some of these interoperability problems.
- One of the biggest challenges with P2P computing is enabling devices to find one another in a computing paradigm that lacks a central point of control. P2P computing needs the ability to find resources and services from a potentially huge number of globally based decentralised peers.

A number of P2P storage systems are being developed within the research community. In the grid context, these raise important issues of security and anonymity:

- The Federated, Available, and Reliable Storage for an Incompletely Trusted Environment (FARSITE) serverless distributed file system [Bolosky00].
- OceanStore, a global persistent data store which aims to provide a 'consistent, highly-available, and durable storage utility atop an infrastructure comprised of untrusted servers' and scale to very large numbers of users. [Kubiatowicz00; Zhuang01]
- The Self-Certifying File System (SFS), a network file system that aims to provide strong security over untrusted networks without significant performance costs [Fu00].
- PAST is a 'large-scale peer-to-peer storage utility' that aims to provide high availability, scalability, and anonymity. Documents are immutable and the identities of content owners, readers, and storage providers are protected. [Druschel01]

See [Fox02b] for further discussion of P2P in the grid context.

3.7.1 JXTA

In April 2001, Sun Microsystems announced the creation of Project JXTA, an open source development community for P2P infrastructure and applications. JXTA describes a network system of five layers:

- Device, the bottom-most layer, means anything with an electronic pulse - desktop PCs, laptops, palm-tops, set-tops, game consoles, cellular telephones, pagers, embedded controllers, et al.
- Network-Enabled Operating Environment consists of the JVM (Java Virtual

machine), which provides TCP/IP services and other resources to participate in a network.

- Platform is made up of functions that allow peers to identify each other (peering, and peer groups), know when a particular peer is available (presence), work out what resources (discovery) they have to offer, and communicate with them.
- Services provide the basic building blocks of applications, such as storage, resource aggregation, security, indexing, and searching.
- Application accommodates the likes of Napster, Groove, AIM, and SETI@home.

Much of what is in JXTA corresponds with the top three layers, leaving the JVM to handle basic device and network needs. It describes a suite of six protocols governing discovery, organization, monitoring, and intra-peer communication. A common XML-based messaging layer binds the protocols to the appropriate underlying network transports.

JXTA is a specification, rather than software. The concepts and specification are separated from its reference implementation, a set of Java class libraries. Apart from the core JXTA components, the project provides rudimentary applications, including JXTA Shell, which provides a view of the JXTA environment via a UNIX-like command-line interface, a file-sharing/chat GUI interface and a content management system.

3.8 A Summary of Experiences of the Second Generation

In the second generation the core software for the Grid has evolved from that provided by the early vanguard offerings, such as Globus (GT1) and Legion, which were dedicated to provision of proprietary services to large and computationally intensive high performance applications, through to the more generic and open deployment of Globus (GT2) and Avaki. Alongside this core software, the second generation also saw the development of a range of accompanying tools and utilities, which were developed to provide higher-level services to both users and applications, and spans resource schedulers and brokers as well as domain specific users interfaces and portals. Peer-to-peer techniques have also emerged during this period.

4. The Evolution of the Grid: The Third Generation

The second generation provided the interoperability that was essential to achieve large-scale computation. As further grid solutions were explored, other aspects of the engineering of the Grid became apparent. In order to build new grid applications it was desirable to be able to reuse existing components and information resources, and to assemble these components in a flexible manner. The solutions involved increasing adoption of a service-oriented model and increasing attention to metadata – these are two key characteristics of third generation systems. In fact the service-oriented approach itself has implications for the information fabric: the flexible assembly of grid resources into a grid application requires information about the functionality, availability and interfaces of the various components, and this information must have an agreed interpretation which can be processed by machine. For further discussion of the service oriented approach see the companion 'Semantic Grid' paper.

Whereas the Grid had traditionally been described in terms of large scale data and

computation, the shift in focus in the third generation was apparent from new descriptions. In particular, the terms 'distributed collaboration' and 'virtual organisation' were adopted in the 'anatomy' paper [Foster01]. The third generation is a more holistic view of grid computing and can be said to address the infrastructure for e-Science – a term which reminds us of the requirements (of doing new science, and of the e-Scientist) rather than the enabling technology. As Fox notes [Fox02a], the anticipated use of massively parallel computing facilities is only part of the picture that has emerged: there are also a lot more users, hence loosely coupled distributed computing has not been dominated by deployment of massively parallel machines.

There is a strong sense of automation in third generation systems; for example, when humans can no longer deal with the scale and heterogeneity but delegate to processes to do so (e.g. through scripting), which leads to autonomy within the systems. This implies a need for coordination, which, in turn, needs to be specified programmatically at various levels – including process descriptions. Similarly, the increased likelihood of failure implies a need for automatic recovery: configuration and repair cannot remain manual tasks. These requirements resemble the self-organising and healing properties of biological systems, and have been termed 'autonomic' after the autonomic nervous system. According to the definition in [Autonomic], an autonomic system has the following eight properties:

1. Needs detailed knowledge of its components and status;
2. Must configure and reconfigure itself dynamically
3. Seeks to optimise its behaviour to achieve its goal
4. Is able to recover from malfunction
5. Protect itself against attack
6. Be aware of its environment
7. Implement open standards
8. Make optimised use of resources

The third generation grid systems now under development are beginning to exhibit many of these features.

In this section we consider first the service oriented approach, looking a web services and agent based computing, and then the information layer issues.

4.1 Service-oriented architectures

By 2001, a number of grid architectures were apparent in a variety of projects. For example, the 'anatomy' paper proposed a layered model, and the Information Power Grid project [IPG] featured an extensive set of services again arranged in layers. Around this time the Web Services model was also gaining popularity, promising standards to support a service-oriented approach. In fact one research community, agent based computing, had already undertaken extensive work in this area: software agents can be seen as producers, consumers and indeed brokers of services. Altogether, it was apparent that the service-oriented paradigm provided the flexibility required for the third generation Grid. Figure 1 depicts the three technologies.

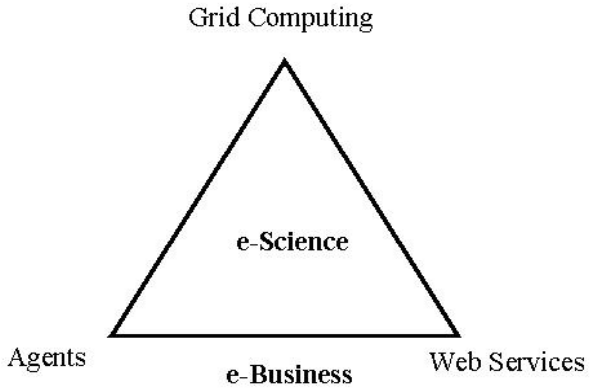

Figure 1: A View of e-Science Technologies

4.1.1 Web Services

The creation of Web Services standards is an industry-led initiative, with some of the emerging standards in various states of progress through the World Wide Web Consortium (W3C). The established standards include:

- SOAP (XML Protocol). SOAP provides an envelope which encapsulates XML data for transfer through the Web infrastructure (e.g. over HTTP, through caches and proxies), with a convention for Remote Procedure Calls (RPCs) and a serialisation mechanism based on XML Schema datatypes. SOAP is being developed by W3C in cooperation with the Internet Engineering Task Force (IETF).
- Web Services Description Language (WSDL). Describes a service in XML, using an XML Schema; there is also a mapping to the Resource Description Framework (RDF). In some ways WSDL is similar to an interface definition language IDL. WSDL is available as a W3C note [WSDL].
- Universal Description Discovery and Integration (UDDI). This is a specification for distributed registries of web services, similar to yellow and white pages services. UDDI supports 'publish, find and bind': a service provider describes and publishes the service details to the directory; service requestors make requests to the registry to find the providers of a service; the services 'bind' using the technical details provided by UDDI. It also builds on XML and SOAP [UDDI].

The next web service standards attracting interest are at the process level. For example, Web Services Flow Language (WSFL) [WSFL] is an IBM proposal that defines workflows as combinations of Web Services and enables workflows themselves to appear as services; XLANG [XLANG] from Microsoft supports complex transactions that may involve multiple Web Services. A combined proposal is anticipated.

There are several other proposals for standards that address various aspects of Web Services. In addition to the necessary machinery for Web Services, there are important efforts that address the design of Web Services systems. For example, the Web Services Modelling

Framework (WSMF) provides a conceptual model for developing and describing web services based on the principles of maximal decoupling and a scalable mediation service [WSMF].

Web Services are closely aligned to the third generation grid requirements: they support a service-oriented approach and they adopt standards to facilitate the information aspects such as service description. In fact, WSDL describes how to interface with a service rather than the functionality of that service. Further work is required on service descriptions; one such activity is DAML-S [DAML02].

4.1.2 The Open Grid Services Architecture (OGSA) Framework

The Open Grid Services Architecture (OGSA) Framework, the Globus-IBM vision for the convergence of Web services and Grid computing was presented at the Global Grid Forum (GGF) meeting held in Toronto in February 2002. OGSA is described in the 'physiology' paper [Foster02]. The GGF has set up an Open Grid Services working group to review and refine the Grid Services architecture and documents that form the technical specification.

The OGSA supports the creation, maintenance, and application of ensembles of services maintained by Virtual Organizations (VOs). Here a service is defined as a network-enabled entity that provides some capability, such as computational resources, storage resources, networks, programs and databases. It tailors the Web Services approach to meet some grid-specific requirements. For example, these are the standard interfaces defined in OGSA:

- Discovery: Clients require mechanisms for discovering available services and for determining the characteristics of those services so that they can configure themselves and their requests to those services appropriately.
- Dynamic service creation: A standard interface (Factory) and semantics that any service creation service must provide.
- Lifetime management: In a system that incorporates transient and stateful service instances, mechanisms must be provided for reclaiming services and state associated with failed operations.
- Notification: A collection of dynamic, distributed services must be able to notify each other asynchronously of interesting changes to their state.
- Manageability: The operations relevant to the management and monitoring of large numbers of Grid service instances are provided.
- Simple hosting environment: A simple execution environment is a set of resources located within a single administrative domain and supporting native facilities for service management: for example, a J2EE application server, Microsoft .NET system, or Linux cluster.

The parts of Globus that are impacted most by the OGSA are:

- The Grid Resource Allocation and Management (GRAM) protocol.
- The information infrastructure, Meta Directory Service (MDS-2), used for information discovery, registration, data modelling, and a local registry.
- The Grid Security Infrastructure (GSI), which supports single sign-on, restricted delegation, and credential mapping.

It is expected that the future implementation of Globus toolkit will be based on the OGSA architecture. Core services will implement the interfaces and behaviour described in the Grid Service specification. Base services will use the Core services to implement both existing Globus capabilities, such as resource management, data transfer and information services, as well as new capabilities such as resource reservation and monitoring. A range of higher-level services will use the Core and Base services to provide data management, workload management and diagnostics services.

4.1.3 Agents

Web Services provide a means of interoperability, key to grid computing, and OGSA is an important innovation which adapts Web Services to the Grid and quite probably anticipates needs in other applications also. However, Web Services do not provide a new solution to many of the challenges of large scale distributed systems, nor do they yet provide new techniques for the engineering of these systems. Hence it is valuable to look at other service-oriented models. The OGSA activity sits on one side of the triangle in figure 1, and we suggest agent based computing [Jennings01a] as another important input to inform the service-oriented grid vision.

The agent based computing paradigm provides a perspective on software systems in which entities typically have the following properties, known as *weak agency* [Wooldridge95]. Note the close relationship between these characteristics and those of autonomic computing listed above.

1. Autonomy – agents operate without intervention and have some control over their actions and internal state;
2. Social ability - agents interact with other agents using an agent-communication language;
3. Reactivity - agents perceive and respond to their environment;
4. Pro-activeness - agents exhibit goal-directed behaviour.

Agent based computing is particularly well suited to a dynamically changing environment, where the autonomy of agents enables the computation to adapt to changing circumstances. This is an important property for the third generation Grid. One of the techniques for achieving this is on-the-fly negotiation between agents, and there is an significant body of research in negotiation techniques [Jennings01b]. Market-based approaches in particular provide an important approach to the computational economies required in grid applications.

Hence we can view the Grid as a number of interacting components, and the information that is conveyed in these interactions falls into a number of categories. One of those is the domain specific content that is being processed. Additional types include:

- Information about components and their functionalities within the domain,
- Information about communication with the components,
- Information about the overall workflow and individual flows within it.

These must be tied down in a standard way to promote interoperability between

components, with agreed common vocabularies. Agent Communication Languages (ACLs) address exactly these issues. In particular, the Foundation for Intelligent Physical Agents (FIPA) activity [FIPA] provides approaches to establishing a semantics for this information in an interoperable manner. FIPA produces software standards for heterogeneous and interacting agents and agent-based systems, including extensive specifications. In the FIPA abstract architecture:

- Agents communicate by exchanging messages which represent speech acts, and which are encoded in an agent-communication-language.
- Services provide support agents, including directory-services and message-transport-services.
- Services may be implemented either as agents or as software that is accessed via method invocation, using programming interfaces (e.g. in Java, C++ or IDL).

Again, we can identify agent-to-agent information exchange and directory entries as information formats which are required by the infrastructure 'machinery'.

4.2 Information aspects: relationship with the World Wide Web

In this section, we focus firstly on the Web. The Web's information handling capabilities are clearly an important component of the e-Science infrastructure, and the web infrastructure is itself of interest as an example of a distributed system that has achieved global deployment. The second aspect addressed in this section is support for collaboration, something which is key to the third generation Grid. We show that the web infrastructure itself lacks support for synchronous collaboration between users, and we discuss technologies that do provide such support.

It is interesting to consider the rapid uptake of the Web and how this might inform the design of the Grid, which has similar aspirations in terms of scale and deployment. One principle is clearly simplicity – there was little new in HTTP and HTML, and this facilitated massive deployment. We should however be aware of a dramatic contrast between Web and Grid: despite the large scale of the Internet, the number of hosts involved in a typical web transaction is still small, significantly lower than that envisaged for many grid applications.

4.2.1 The Web as a Grid Information Infrastructure

The Web was originally created for distribution of information in an e-Science context at CERN. So an obvious question to ask is, does this information distribution architecture described in section 4.1 meet grid requirements? A number of concerns arise:

- Version control. The popular publishing paradigm of the Web involves continually updating pages without version control. In itself the web infrastructure does not explicitly support versioning.
- Quality of service. Links are embedded, hardwired global references and they are fragile, rendered useless by changing the server, location, name or content of the destination document. Expectations of link consistency are low and e-Science may demand a higher quality of service.
- Provenance. There is no standard mechanism to provide legally significant evidence that a document has been published on the Web at a particular time [Haber91].

- Digital Rights Management. e-Science demands particular functionality with respect to management of the digital content, including for example copy protection and intellectual property management.
- Curation. Much of the web infrastructure focuses on the machinery for delivery of information rather than the creation and management of content. Grid infrastructure designers need to address metadata support from the outset.

To address some of these issues we can look to work in other communities. For example, the multimedia industry also demands support for digital rights management. MPEG-21 aims to define 'a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities' [MPEG21], addressing the multimedia content delivery chain. Its elements include declaration, identification, content handling, intellectual property management and protection. Authoring is another major concern, especially collaborative authoring. The Web-based Distributed Authoring and Versioning (WebDAV) activity [WebDAV] is chartered "to define the HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable, while supporting user needs".

In summary, although the Web provides an effective layer for information transport, it does not provide a comprehensive information infrastructure for e-Science.

4.2.2 Expressing Content and Meta-content

The Web has become an infrastructure for distributed applications, where information is exchanged between programs rather than being presented for a human reader. Such information exchange is facilitated by the XML (Extensible Markup Language) family of recommendations from W3C. XML is designed to mark up documents and has no fixed tag vocabulary; the tags are defined for each application using a Document Type Definition (DTD) or an XML Schema. A well-formed XML document is a labelled tree. Note that the DTD or Schema addresses syntactic conventions and does not address semantics. XML Schema are themselves valid XML expressions. Many new 'formats' are expressed in XML, such as SMIL (the synchronised multimedia integration language).

RDF (Resource Description Framework) is a standard way of expressing metadata, specifically resources on the Web, though in fact it can be used to represent structured data in general. It is based on 'triples' where each triple expresses the fact that an object O has attribute A with value V, written A(O,V). An object can also be a value, enabling triples to be 'chained', and in fact, any RDF statement can itself be an object or attribute – this is called reification and permits nesting. RDF Schema are to RDF what XML Schema are to XML: they permit definition of a vocabulary. Essentially RDF schema provides a basic type system for RDF such as Class, subClassOf and subPropertyOf. RDF Schema are themselves valid RDF expressions.

XML and RDF (with XML and RDF schema) enable the standard expression of content and metacontent. Additionally a set of tools has emerged to work with these formats, for example parsers, and there is increasing support by other tools. Together this provides the infrastructure for the information aspects of the third generation Grid.

W3C ran a "Metadata Activity", which addressed technologies including RDF, and this has been succeeded by the Semantic Web Activity. The activity statement [Semweb] describes the Semantic Web as follows:

"The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications. The Web can reach its full potential if it becomes a place where data can be shared and processed by automated tools as well as by people."

This vision is familiar – it shares much with the Grid vision. The Scientific American paper [BernersLee01] provides motivation, with a scenario that uses agents. In a nutshell, the Semantic Web is intended to do for knowledge representation what the Web did for hypertext. We discuss this relationship further in Section 5.

The DARPA Agent Markup Language Program (DAML) [Hendler00], which began in 2000, brings Semantic Web technologies to bear on agent communication (as discussed in the previous section). DAML extends XML and RDF with ontologies, a powerful way of describing objects and their relationships. The Ontology Interchange Language (OIL) has been brought together with DAML to form DAML+OIL. W3C has created a Web Ontology Working Group which is focusing on the development of a language based on DAML+OIL.

4.3 Live Information Systems

The third generation also emphasises distributed collaboration. One of the collaborative aspects builds on the idea of a 'collaboratory', defined in a 1993 US NSF study [Cerf93] as a "centre without walls, in which the nation's researchers can perform their research without regard to geographical location - interacting with colleagues, accessing instrumentation, sharing data and computational resource, and accessing information in digital libraries." This view accommodates 'information appliances' in the laboratory setting, which might, for example include electronic logbooks and other portable devices.

4.3.1 Collaboration

As an information dissemination mechanism, the Web might have involved many users as 'sinks' of information published from major servers. However, in practice, part of the web phenomenon has been widespread publishing by the users. This has had a powerful effect in creating online communities. However, the paradigm of interaction is essentially 'publishing things at each other', and is reinforced by email and newsgroups, which also supports asynchronous collaboration.

Despite this, however, the underlying internet infrastructure is entirely capable of supporting live (real-time) information services and synchronous collaboration. For example:

- Live data from experimental equipment,
- Live video feeds ('webcams') via unicast or multicast (e.g. MBONE),
- Video conferencing (e.g. H.323, coupled with T.120 to applications, SIP),
- Internet Relay Chat

- Instant messaging systems
- MUDs,
- Chat rooms,
- Collaborative Virtual Environments.

All of these have a role in supporting e-Science, directly supporting people, behind the scenes between processes in the infrastructure, or both. In particular, they support the extension of e-Science to new communities that transcend current organisational and geographical boundaries.

Although the histories of these technologies predate the Web, they can interoperate with the Web and build on the web infrastructure technologies through adoption of appropriate standards. For example, messages can be expressed in XML and URLs are routinely exchanged. In particular the web's metadata infrastructure has a role: data from experimental equipment can be expressed according to an ontology, enabling it to be processed by programs in the same way as static data such as library catalogues.

The application of computer systems to augment the capability of humans working in groups has a long history, with origins in the work of Doug Englebart [Englebart62]. In this context, however, the emphasis is on facilitating distributed collaboration, and we wish to embrace the increasingly 'smart' workplaces of the e-Scientist including meeting rooms and laboratories. Amongst the considerable volume of work in the 'smart space' area we note in particular the Smart Rooms work by Pentland [Pentland96] and Coen's work on the Intelligent Room [Coen98]. This research area falls under the "Advanced Collaborative Environments" Working group of the Global Grid Forum (ACE Grid), which addresses both collaboration environments and ubiquitous computing.

4.3.2 Access Grid

The Access Grid [AccessGrid] is a collection of resources that support human collaboration across the Grid, including large-scale distributed meetings and training. The resources include multimedia display and interaction, notably through room-based videoconferencing (group-to-group), and interfaces to grid middleware and visualisation environments. Access Grid nodes are dedicated facilities that explicitly contain the high quality audio and video technology necessary to provide an effective user experience.

Current Access grid infrastructure is based on IP multicast. The ISDN-based videoconferencing world (based on H.320) has evolved alongside this, and the shift now is to products supporting LAN-based video conferencing (H.323). The T.120 protocol is used for multicast data transfer, such as remote camera control and application sharing. Meanwhile the IETF has developed Session Initiation Protocol (SIP), which is a signalling protocol for establishing real-time calls and conferences over the Internet. This resembles HTTP and uses Session Description Protocol (SDP) for media description.

During a meeting, there is live exchange of information, and this brings the information layer aspects to the fore. For example, events in one space can be communicated to other spaces to facilitate the meeting. At the simplest level, this might be slide transitions or remote camera control. These provide metadata, which is generated automatically by software and devices,

and can be used to enrich the conference and stored for later use. New forms of information may need to be exchanged to handle the large scale of meetings, such as distributed polling and voting.

Another source of live information is the notes taken by members of the meeting, or the annotations that they make on existing documents. Again, these can be shared and stored to enrich the meeting. A feature of current collaboration technologies is that sub-discussions can be created easily and without intruding – these also provide enriched content.

In videoconferences, the live video and audio feeds provide presence for remote participants – especially in the typical access grid installation with three displays each with multiple views. It is also possible for remote participants to establish other forms of presence, such as the use of avatars in a collaborative virtual environment, and there may be awareness of remote participants in the physical meeting space.

The combination of Semantic Web technologies with live information flows is highly relevant to grid computing and is an emerging area of activity [Page01]. Metadata streams may be generated by people, by equipment or by programs – e.g. annotation, device settings, data processed in real-time. Live metadata in combination with multimedia streams (such as multicast video) raises quality of service (QoS) demands on the network and raises questions about whether the metadata should be embedded (in which respect, the multimedia metadata standards are relevant). A scenario in which knowledge technologies are being applied to enhance collaboration is described in [Buck02].

5. Summary and Discussion

In this paper, we have identified the first three generations of the Grid:

- First generation systems involved proprietary solutions for sharing high performance computing resources;
- Second generation systems introduced middleware to cope with scale and heterogeneity, with a focus on large scale computational power and large volumes of data;
- Third generation systems are adopting a service-oriented approach, adopt a more holistic view of the e-Science infrastructure, are metadata-enabled and may exhibit autonomic features.

The evolution of the Grid has been a continuous process and these generations are not rigidly defined – they are perhaps best distinguished by philosophies rather than technologies. We suggest the book [Foster98] marks the transition from first to second generation, and the 'anatomy' [Foster01] and 'physiology' [Foster02] papers mark the transition from a second to third generation philosophy.

We have seen that in the third generation of the grid, the early Semantic Web technologies provide the infrastructure for grid applications. In this section, we explore further the relationship between the Web and the Grid in order suggest the future evolution.

5.1 Comparing the Web and the Grid

The state of play of the Grid today is reminiscent of the Web some years ago: there is limited deployment, largely driven by enthusiasts within the scientific community, with emerging standards and a degree of commercial uptake. The same might also be said of the Semantic Web. Meanwhile, the Web has seen a shift from machine-to human communications (HTML) to machine-to-machine (XML), and this is precisely the infrastructure needed for the Grid. Related to this, the Web Services paradigm appears to provide an appropriate infrastructure for the Grid, though already Grid requirements are extending this model.

It is appealing to infer from these similarities that Grid deployment will follow the same exponential model as Web growth. However, a typical grid application might involve large numbers of processes interacting in a coordinated fashion, while a typical Web transaction today still only involves a small number of hosts (e.g. server, cache, browser). Achieving the desired behaviour from a large scale distributed system involves technical challenges that the Web itself has not had to address, though Web services take us into a similar world.

The Web provides an infrastructure for the Grid. Conversely, we can ask what the Grid offers to the Web. As a Web application, it raises certain challenges which motivate evolution of web technologies – such as the enhancements to Web Services in OGSA, which may well transcend grid applications. It also provides a high performance infrastructure for various aspects of Web applications; for example in search, data mining, translation and multimedia information retrieval.

5.2 The Semantic Grid

The visions of the grid and the semantic web have much in common but can perhaps be distinguished by a difference of emphasis: the Grid is traditionally focused on computation, while the ambitions of the Semantic Web take it towards inference, proof and trust. The Grid we are now building in this third generation is heading towards what we term the Semantic Grid: as the Semantic Web is to the Web, so the Semantic grid is to the Grid. This is depicted in Figure 2.[1]

[1] The term was used by Erick Von Schweber in GGF2 and a comprehensive report on the Semantic Grid was written by the present authors for the UK e-Science Programme in July 2001 [DeRoure01]. This particular representation of the Semantic Grid is due to Norman Paton of the University of Manchester, UK.

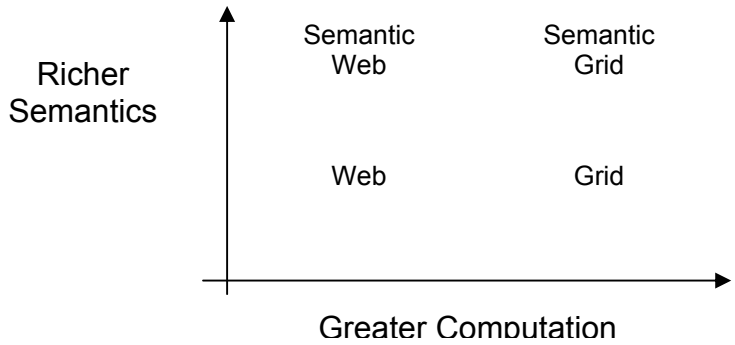

Figure 2: The Semantic Grid

The Semantic Grid is achievable now in a simple but very powerful form – it is metadata-enabled and ontologically principled. Third generation Grid is addressing the way that *information* is represented, stored, accessed, shared and maintained - information is understood as data equipped with meaning. We anticipate that the next generation will be concerned with the way that *knowledge* is acquired, used, retrieved, published and maintained to assist e-Scientists to achieve their particular goals and objectives - knowledge is understood as information applied to achieve a goal, solve a problem or enact a decision. The Semantic Grid involves all three conceptual layers of the Grid: knowledge, information and computation/data. These complementary layers will ultimately provide rich, seamless and pervasive access to globally distributed heterogeneous resources.

5.3 Research Issues

The general view of the Grid is that of a three-layered system made up of computation/data, information and knowledge layers. Even though the computation/data layer of the Grid is the layer which is perhaps the most mature in terms of the time, experience and where most software is available and directly useable, it still lacks many essential aspects that will allow the provision of seamless, pervasive and secures use of system resources. A certain number of these aspects are now being addressed as the information and knowledge layers of the Grid evolve. The following generic areas are seen as ones that require further work:

- Information Services – the mechanisms that are used to hold information about the resources in a grid need to provide extendable, fast, reliable, secure, and scalable services.
- Resource Information – all manner of grid information will be necessary to enable the Grid to work correctly. This information will range from security data through to application requirements and from resource naming data through to user profiles. It is vital that all this information can be understood, interpreted and used, by all the services that require it.
- Resource discovery – given a resource's unique name or characteristics there need to

be mechanisms to locate the resource within the globally distributed system. Services are resources. Some resources may persist, some may be transitory, and some may be created on demand.

- Synchronisation and coordination – how to orchestrate a complex sequence of computations over a variety of resources, given the inherent properties of both loosely- and tightly-coupled distributed systems. This may involve process description, and require an event-based infrastructure. It involves scheduling at various levels, including meta-scheduling and workflow.
- Fault tolerance and dependability – environments need to cope with the failure of software and hardware components, as well as access issues – in general, accommodating the exception-handling that is necessary in such a dynamic, multi-user, multi-organisation system.
- Security – authentication, authorisation, assurance, and accounting mechanisms need to be set in place, and these need to function in the context of increasing scale and automation. For example, a user may delegate privileges to processes acting on their behalf, which may in turn need to propagate some privileges further.
- Concurrency and consistency – the need to maintain an appropriate level of data consistency in the concurrent, heterogeneous environment. Weaker consistency may be sufficient for some applications.
- Performance – the need to be able to cope with non-local access to resources, through caching and duplication. Moving the code (or service) to the data (perhaps with scripts or mobile agents) is attractive and brings a set of challenges.
- Heterogeneity – the need to work with a multitude of hardware, software and information resources, and to do so across multiple organisations with different administrative structures.
- Scalability – systems need to be able to scale up the number and size of services and applications, without scaling up the need for manual intervention. This requires automation, and ideally self-organisation.

At the information layer, although many of the technologies are available today (even if only in a limited form), a number of the topics still require further research. These include:

- Issues relating to e-Science content types. Caching when new content is being produced. How will the Web infrastructure respond to the different access patterns resulting from automated access to information sources? Issues in the curation of e-Science content.
- Digital rights management in the e-Science context (as compared with multimedia and e-commerce, for example).
- Provenance. Is provenance stored to facilitate reuse of information, repeat of experiments, or to provide evidence that certain information existed at a certain time?
- Creation and management of metadata, and provision of tools for metadata support.
- Service descriptions and tools for working with them. How best does one describe a service-based architecture?
- Workflow description and enaction, and tools for working with descriptions.
- Adaptation and personalisation. With the system 'metadata-enabled' throughout, how much knowledge can be acquired and how can it be used?
- Collaborative infrastructures for the larger community, including interaction between scientists, with e-Science content and visualisations, and linking smart laboratories

and other spaces.
- Use of metadata in collaborative events, especially live metadata; establishing metadata schema to support collaboration in meetings and in laboratories.
- Capture and presentation of information using new forms of device; e.g. for scientists working in the field.
- Representation of information about the underlying grid fabric, as required by applications; e.g. for resource scheduling and monitoring.

The Semantic Grid will be a place where data is equipped with a rich context and turned into information. This information will then be shared and processed by virtual organisations in order to achieve specific aims and objectives. Such actionable information constitutes knowledge. Hence the knowledge layer is key to the next stage in the evolution of the Grid, to a fully fledged Semantic Grid. The research agenda to create the Semantic Grid is the subject of the companion paper 'The Semantic Grid: A Future e-Science Infrastructure'.

References

[Abramson00] D. Abramson, J. Giddy, and L. Kotler, 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.

[Abramson01] D. Abramson, P. Roe, L. Kotler, and D. Mather, ActiveSheets: Super-Computing with Spreadsheets. 2001 High Performance Computing Symposium (HPC'01), Advanced Simulation Technologies Conference, April 2001.

[Abramson00a] D. Abramson, J. Giddy, and L. Kotler, 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.

[AccessGrid] Access grid, http://www.accessgrid.org

[Akarsu98] E. Akarsu, G.C. Fox, W. Furmanski, and T. Haupt, "WebFlow: High-Level Programming Environment and Visual Authoring Toolkit for High Performance Distributed Computing", SC98: High Performance Networking and Computing, Orlando, Florida, 1998.

[Allen01] Gabrielle Allen, Thomas Dramlitsch, Ian Foster, Nick Karonis, Matei Ripeanu, Ed Seidel, Brian Toonen. Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus. Winning Paper for Gordon Bell Prize (Special Category), Supercomputing 2001 (Revised version: August 2001)

[Almond99] J. Almond and D. Snelling, UNICORE: uniform access to supercomputing as an element of electronic commerce, Future Generation Computer Systems, 15(1999) 539-548, NH-Elsevier.

[Armstrong99] R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski. Toward a common component architecture for high performance scientific computing. In Proceedings of the 8th High Performance Distributed Computing (HPDC'99), 1999.

[Autonomic] IBM Autonomic Computing, http://www.research.ibm.com/autonomic/

[Avaki] Avaki, http://www.avaki.com/

[Baker96] M.A. Baker, G.C. Fox, and H.W. Yau, A Review of Cluster Management Software, NHSE Review, Volume 1, No. 1, May 1996. http://www.nhse.org/NHSEreview/CMS/

[BernersLee01] Berners-Lee,T., Hendler,J. and Lassila ,O. "The Semantic Web", Scientific American, May 2001.

[Bolosky00] W. J. Bolosky, J. R. Douceur, D. Ely, M. Theimer "Feasibility of a Serverless

Distributed File System Deployed on an Existing Set of Desktop PCs", Proc. international conference on measurement and modeling of computer systems (SIGMETRICS 2000), pp. 34-43. ACM Press, 2000.
[Buck02] Buckingham Shum, S., De Roure, D., Eisenstadt, M., Shadbolt, N. and Tate, A. (2002) CoAKTinG: Collaborative Advanced Knowledge Technologies in the Grid. Proceedings of the Second Workshop on Advanced Collaborative Environments, Eleventh IEEE Int. Symposium on High Performance Distributed Computing (HPDC-11), July 24-26, 2002, Edinburgh, Scotland.
[Buyya00a] R. Buyya, D. Abramson, and J. Giddy, 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.
[Buyya00b] R. Buyya, J. Giddy, D. Abramson, An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications, The Second Workshop on Active Middleware Services (AMS 2000), In conjunction with HPDC 2001, August 1, 2000, Pittsburgh, USA (Kluwer Academic Press).
[Buyya01] R. Buyya, The Virtual Laboratory Project: Molecular Modeling for Drug Design on Grid, IEEE Distributed Systems Online, Vol. 2, No. 5, 2001. http://www.buyya.com/vlab/
[Catlett92] C. Catlett and L. Smarr, "Metacomputing," Communications of the ACM, June 1992, pp. 44-52.
[Cerf93] V.G. Cerf et al., "National Collaboratories: Applying Information Technologies for Scientific Research", National Academy Press: Washington, D.C., 1993.
[Clark01a] D. Clark, "Face-to-Face with Peer-to-Peer Networking", Computer, Vol. 34, No. 1, January 2001, pp. 18-21
[Clarke01b], Clarke, I., Sandberg, O., Wiley, B. and Hong, T.W., "Freenet: A Distributed Anonymous Information Storage and Retrieval System", in ICSI Workshop on Design Issues in Anonymity and Unobservability, 2001.
[Coen98] M.A.Coen, "A Prototype Intelligent Environment," Cooperative Buildings – Integrating Information, Organisation and Architecture, N. Streitz, S.Konomi and H-J.Burkhardt (eds), LNCS, Springer-Verlag, 1998.
[Condor] Condor, http://www.cs.wisc.edu/condor/
[DAML02] DAML Services Coalition (Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry R. Payne and Katia Sycara), "DAML-S: Web Service Description for the Semantic Web", in The First International Semantic Web Conference (ISWC), June, 2002, pp 348-363.
[DATAGRID] The DataGrid project, http://eu-datagrid.web.cern.ch/
[DeRoure02] De Roure, D., Jennings, N. R. and Shadbolt, N. R., "The Semantic Grid: A Future e-Science Infrastructure", in this volume, 2002.
[Druschel01] P. Druschel and A. Rowstron, "PAST: A large-scale, persistent peer-to-peer storage utility", HotOS VIII, Schoss Elmau, Germany, May 2001.
[Englebart62] D. Englebart "Augmenting Human Intellect: A Conceptual Framework" AFOSR-3233, Oct. 1962. http://sloan.stanford.edu/mousesite/EngelbartPapers/B5_F18_ConceptFrameworkInd.html
[Entropia] Entropia, http://entropia.com/
[EuroGrid] EuroGrid, http://www.eurogrid.org/
[FAFNER] FAFNER, http://www.npac.syr.edu/factoring.html
[Fensel] D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF

[FIPA] The Foundation for Physical Agents http://www.fipa.org/
[Foster96] I. Foster, C. Kesselman, S. Tuecke, The Nexus Approach to Integrating Multithreading and Communication, J. Parallel and Distributed Computing, 37:70-82, 1996.
[Foster97a] I. Foster, J. Geisler, W. Nickless, W. Smith, S. Tuecke "Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment" in Proc. 5th IEEE Symposium on High Performance Distributed Computing. pp. 562-571, 1997.
[Foster97b] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", International Journal of Supercomputer Applications, 11(2): 115-128, 1997.
[Foster98] Ian Foster and Carl Kesselman (eds), "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, July 1998. ISBN 1-55860-475-8.
[Foster01] I. Foster, C. Kesselman, S. Tuecke "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications and High Performance Computing, 2001.
[Foster02] Foster, I., Kesselman, C., Nick, J. and Tuecke, S., The Physiology of the Grid: Open Grid Services Architecture for Distributed Systems Integration, presented at GGF4, Feb. 2002 http://www.globus.og/research/papers/ogsa.pdf
[Fox02a] G.C. Fox, From Computational Science to Internetics: Integration of Science with Computer Science, in Ronald F. Boisvert and Elias Houstis (eds.), Computational Science, Mathematics and Software, Purdue University Press, West Lafayette, Indiana, ISBN 1-55753-250-8. Pages 217-236, July 2002.
[Fox02b] Geoffrey Fox, Dennis Gannon, Sung-Hoon Ko, Sangmi Lee, Shrideep Pallickara, Marlon Pierce, Xiaohong Qiu, Xi Rao, Ahmet Uyar, Minjun Wang, Wenjun Wu. Peer-to-Peer Grids. Book chapter, to be published.
[Fu00] Kevin Fu, M. Frans Kaashoek, and David Mazières "Fast and secure distributed read-only file system" in the Proceedings of the 4th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2000), San Diego, California, October 2000.
[GGF] Global Grid Forum – http://www.gridforum.org/
[Gnutella] Gnutella, http://www.gnutella.co.uk/
[GridPort] SDSC GridPort Toolkit, http://gridport.npaci.edu/
[Grimshaw97] Grimshaw A., Wulf W. et al., "The Legion Vision of a Worldwide Virtual Computer". Communications of the ACM, vol. 40(1), January 1997.
[Haber91] Haber, S and Stornetta, WS, "How to time-stamp a digital document", Journal of Cryptography, Vol 3 No 2 pp 99-111, 1991.
[Haupt99] T. Haupt, E. Akarsu, G. Fox and W Furmanski, "Web Based Metacomputing", Special Issue on Metacomputing, Future Generation Computer Systems, North Holland 1999.
[Haupt02] T. Haupt, P. Bangalore, G. Henley, "Mississippi Computational Web Portal". Accepted for publication in *Concurrency and Computation: Practice and Experience*.
[Hendler00] J. Hendler and D. McGuinness, "The DARPA Agent Markup Language," IEEE Intelligent Systems, vol. 15, no. 6, Nov./Dec. 2000, pp. 72–73.
[Hoschek00] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, Data Management in an International Data Grid Project, Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing (Grid'2000), Bangalore, India, 17-20 Dec. 2000, Springer-Verlag Press, Germany.
[HotPage] HotPage, https://hotpage.npaci.edu/
[IPG] NASA Information Power Grid, http://www.ipg.nasa.gov/
[JavaGrande] Java Grande, http://www.javagrande.org/
[Jennings01a] N. R. Jennings (2001) "An agent-based approach for building complex software systems" Comms. of the ACM, 44 (4) 35-41.

[Jennings01b] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, M. Wooldridge Automated Negotiation: Prospects, Methods and Challenges. Journal of Group Decision and Negotiation, 10, 2, 199-215, 2001.
[Jini] JINI, http://www.jini.org
[Jones96] J. P. Jones, "NAS Requirements Checklist for Job Queuing/Scheduling Software," NAS Technical Report NAS-96-003 April 1996, http://www.nas.nasa.gov/Research/Reports/Techreports/1996/nas-96-003-abstract.html
[JXTA] JXTA, http://www.jxta.org/
[Kubiatowicz00] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. "OceanStore: An Architecture for Global-Scale Persistent Storage", in Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000.
[MISS] http://www.erc.msstate.edu/~haupt/DMEFS/welcome.html
[MPEG21] ISO/IEC JTC1/SC29/WG11 Coding of Moving Picture and Audio, MPEG-21 Overview, document N4041.
[Napster] Napster, http://www.napster.com/
[NET] Distributed.Net, http://www.distributed.net/
[NLANR] NLANR Grid Portal Development Kit, http://dast.nlanr.net/Features/GridPortal/
[OMG] OMG, http://www.omg.org
[Page01] Kevin R. Page, Don Cruickshank and David De Roure. It's About Time: Link Streams as Continuous Metadata. Proc The Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01) p.93-102. 2001.
[Parabon], Parabon, http://www.parabon.com/
[PBS] Portable Batch System, http://www.openpbs.org/
[Pentland96] Pentland "Smart Rooms" Scientific American 274, No. 4, pp 68-76, April 1996.
[Pierce02] M.E. Pierce, C.Youn and G.C. Fox, The Gateway Computational Web Portal. Accepted for publication in *Concurrency and Computation: Practice and Experience*.
[PLATFORM] Platform Computing, http://www.platform.com
[Rajasekar01] Arcot K. Rajasekar and Reagan W. Moore, Data and Metadata Collections for Scientific Applications, European High Performance Computing conference, Amsterdam, Holland, June 26, 2001.
[Rivest77] Rivest, R.L., Shamir, a. and Adelman, L. "On Digital Signatures and Public Key Cryptosystems," MIT Laboratory for Computer Science Technical Memorandum 82, April 1977.
[Semweb] W3C Semantic Web Activity Statement - http://www.w3.org/2001/sw/Activity
[SETI] SETI@Home, http://setiathome.ssl.berkeley.edu/
[SGE] Sun Grid Engine, http://www.sun.com/software/gridware/
[UDDI] http://www.uddi.org/
[WebDAV] Web-based Distributed Authoring and Versioning, http://www.Webdav.org/
[Wooldridge95] M. Wooldridge and N.R. Jennings, Intelligent Agents: Theory and Practice. Knowledge Engineering Review Volume 10 No 2, June 1995.
[WSDL] Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001 http://www.w3.org/TR/wsdl
[WSFL] Web Services Flow Language (WSFL) Version 1.0, http://www-4.ibm.com/software/solutions/Webservices/pdf/WSFL.pdf
[WSMF] Web Services Modelling Framework, http://www.cs.vu.nl/~dieter/wese/
[XLANG] Web Services for Business Process Design,

http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
[Zhou93] S. Zhou, X. Zheng, J. Wang, and P. Delisle, "Utopia: A load sharing facility for large heterogeneous distributed computer systems." Software Practice and Experience, December 1993.
[Zhuang01] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz and John Kubiatowicz. “Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination”, Proceedings of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001), June 2001.