Web Services for High Performance Technical Computing

Sun ONE, N1, Open Source and Standards for
Data- and Numerically Intensive Grid Infrastructures

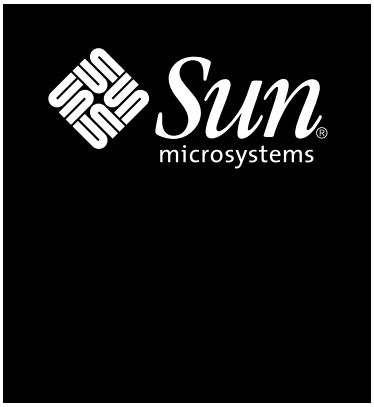


Table of Contents

Chapter 1

1 High Performance Technical Computing

1.1 Introduction

1.1.1 What is HPTC?

High Performance Technical Computing (HPTC) is a category of scientific and engineering applications that involve intensive resource requirements, in terms of numerical calculations, data manipulation, and network communications. Researchers use HPTC to bring the maximum amount of computing resources together to solve tough scientific problems.

Although HPTC technology is typically pioneered by the academic, scientific and engineering community, the targeted fields span both commercial enterprises and research institutions. Application domains for HPTC include complex industrial design (e.g. aerospace, automobiles, and power plants), complex modeling and simulation (e.g. astrophysics, meteorology, aerodynamics and fluid dynamics), life sciences (e.g. computational biology, genomics, pharmaceutical research, and medical imaging), as well as high energy physics, computational chemistry, and many other areas of research.

1.1.2 About This White Paper

This white paper describes the challenges of today's collaborative research environment, provides an overview of the key organizational and technical issues involved in HPTC, and shows how Sun Microsystems™ delivers a complete solution to these challenges. The paper is aimed at readers who want to implement technical computing applications, as well as service providers who build and manage infrastructure for HPTC.

This white paper is divided into the following main sections:

- Chapter 1, "High Performance Technical Computing", describes the problems that motivate organizations in their search for HPTC solutions. It explains the key concepts and issues concerning Services on Demand, research collaboration, and Grid computing.
- Chapter 2, "A Standards-Based Grid Architecture for Technical Computing", explains the principles of Grid computing in detail, and maps out each of the elements of a real-world architecture for HPTC.
- Chapter 3, "Sun Microsystems in Technical Computing", describes in detail the components of the Sun solution for Applications, Systems, and Management.
- Chapter 4, "Case Studies", shows how Sun's Services on Demand architecture is used in the real world to solve HPTC problems.

1.2 The Sun HPTC Vision: Services on Demand

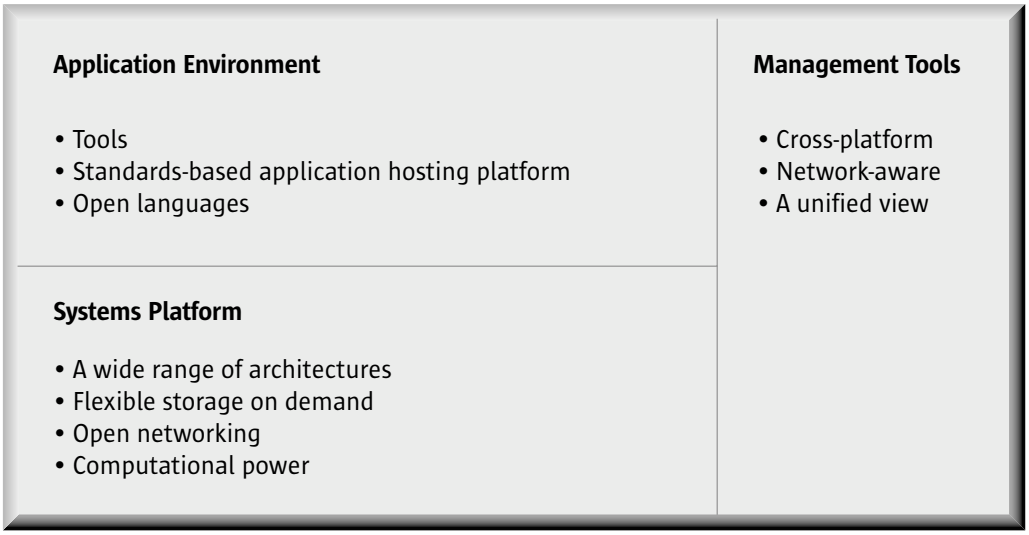
Sun's solution for collaborative HPTC is based on its overall vision for **Services on Demand:** Access to technical computing power anywhere, anytime, from any device, by any authorized user.

This vision enables technical research organizations to focus on how to provide results to technical users, rather than on the details of the physical infrastructure. Services on Demand provides more flexible access to computing power for a far wider population, while making this power much easier to manage and share. With this type of HPTC solution, IT can focus on buying computing, not computers, and on managing services, not servers.

To deliver HPTC power on demand, the technology must be based on a powerful and flexible infrastructure that provides **virtualization of resources**. This infrastructure unifies different applications, technologies, and systems under a single architecture with a consistent level of service:

- **Users** don't need to know any details about the underlying platform. They can access these virtualized resources without worrying about where they are located.
- **Developers** use standards-based tools to build applications on a secure, standard and feature-rich platform. These applications can then be deployed wherever they make sense to run on the network.
- **Systems Administrators** manage this diverse infrastructure for computation, storage and communications through a single, unified view.

As we shall see, this vision includes Grid computing technology, but the requirements go beyond the grid to encompass a solution that fits all the pieces together. Sun provides a complete range of solutions that responds to the essential challenges of making Services on Demand a reality.



1.3 The Collaborative Research Environment

More and more, researchers must collaborate across different groups and even institutions. This need for collaboration comes from a set of new areas where High Performance Technical Computing is one of the keys to successful research:

- **Industry-research partnerships.** These collaborations are becoming increasingly frequent, and can involve sharing of data, applications, and computing resources between organizations that often have different management policies and infrastructures. For example, new work in aerodynamics can involve joining together new research in dynamics modeling, from an academic institution, with new aircraft designs to be tested, from a commercial enterprise.
- **Multi-disciplinary research.** Innovation increasingly involves bringing together data and expertise from multiple groups operating in different areas. Example: Pharmaceuticals. New computational biology work is emerging to support the complete drug discovery, development, testing and approval cycles. This involves bringing together data and computations from genomic, post-genomic (e.g. Proteomics), biochemical, pharmacological, and protein modeling, then management of data and analysis for clinical trials.

- **Multi-site collaborations.** Much essential research is no longer possible without close collaboration between researchers and infrastructure hosted by different institutions. One team might bring the expertise and knowledge, another team might have the data needed for the research activity, and a third institution might provide critical computing resources. For example, at the Large Hadron Collider project at CERN[1], facilities will be accessed by around 6,000 researchers, spread around the world in many different institutions. 5-8 petabytes of data will be generated each year, and disk storage will be on the order of tens of petabytes. The computational power needed to run the applications and manage the data will be enormous. However, the majority of the computing systems will not be local to CERN but will be distributed across the globe. The large hadron collider (LHC) experiment is scheduled to start operating in 2006.

1.4 Divergent Requirements in HPTC

Different applications have different resource needs. HPTC solutions can satisfy real-world application demands by matching the right resources to each request. This section describes the complexities of these divergent requirements.

1.4.1 Making TeraFLOPS Meaningful

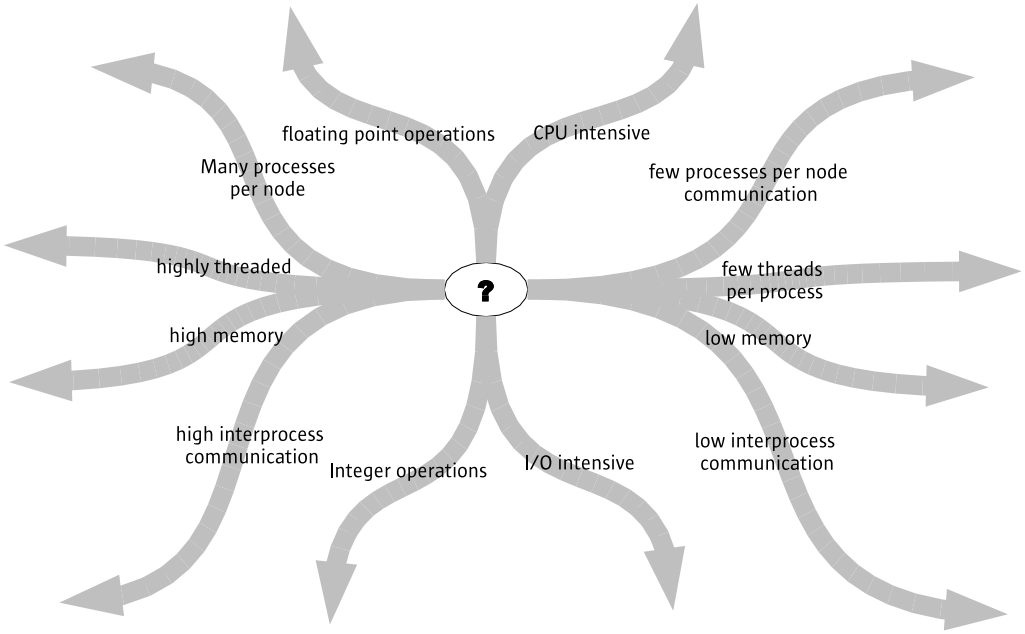
The top supercomputers today operate at speeds of up to tens of TeraFLOPS. However, the capability of such compute platforms is diverse. The ability to run different applications on top of these will vary from one computer to another. While instruction speed is an important metric, there are other factors that determine how useful the compute capability of a supercomputer is to a researcher. For instance, technical application designers must take into account the communication speed between nodes of a supercomputer, communication between processors, and the memory addressable by a single process. So for these real-world problems, high performance comes from matching the right resources to the application.

For example:

- A high energy physics researcher may need to submit a job that runs a monte carlo generation algorithm on a set of experimental results and then schedule analysis of the large result set. This could require a large single host, capable of ensuring both significant memory capacity available to local processes (for instance greater than 150 gigabytes), as well as very high-concurrency processing, where threads can access global data at high speed, and finally very high throughput of floating point operations. This type of application might also require terabytes of storage.
- Another researcher might wish to submit a job to do genome sequence matching using a BLAST-like algorithm to find local homologies within longer genomic sequences. In most cases, this requires many parallel processes with only minimal communication between them. Operations on character (integer) types predominate, and memory requirements per process are not that great, and so a commodity cluster configuration would be appropriate. There are, however, cases, where it makes sense to allocate large amounts of memory even to these types of jobs and reduce execution time by eliminating the time to load data from disk or cache into the CPU.
- Yet another researcher, in proteomics, may seek to schedule a protein folding application job that also requires highly parallel processing, but this time with a high degree of floating point operations and large amounts of local memory. This may run optimally on a cluster of large nodes.

1. http://info.web.cern.ch/info/Press/PressReleases/Releases2001/PR11.01ECERNopenlab.html

1.4.2 Matching the Platform to the Requirements



The research institution providing the service needs to accommodate the different requirements of the users, and perhaps the requirements of many thousands of users at once.

As the above diagram illustrates, a system that will meet the needs of all users will need to consist of many different capabilities. Applications in the HPTC space are sufficiently diverse that it will not be possible to cover all the requirements without a range of platform capabilities.

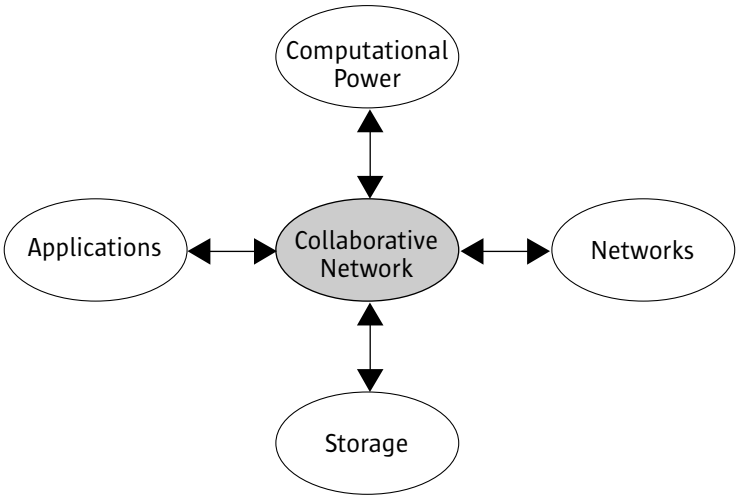
This range would include clusters that run Solaris™, Linux or other flavors of Unix operating systems. These types of clusters are made up of racks of commodity nodes (Intel, Sun SPARC®, Alpha, or others) that process a very high volume of session-oriented traffic. These systems function better when the tasks can run independently and do not require large amounts of memory, communications, data synchronization or data handling.

Other applications will only perform acceptably on platforms that provide increasingly aggregated CPU power in a single system image, while still being used in cluster configurations (mid-range SMP). These platforms are based on higher-speed interconnects that allow fast message passing between processes on separate nodes. With this interconnect, applications can communicate along a specially optimized communications bus when a task is finished or has produced a particular result set.

Still more powerful clusters of machines, with hundreds of gigabytes of memory available to each process, can accomplish the most demanding processing tasks. These solutions are based on ultra-fast interconnects between processors that lower-end systems cannot match.

1.4.3 Sharing Resources

HPTC requires large amounts of computing resources. As the complexity of research efforts grows, more and more problems require sharing compute resources among geographically dispersed groups of researchers. The question is just how to overcome the barriers to bringing all the compute resources together to solve problems.

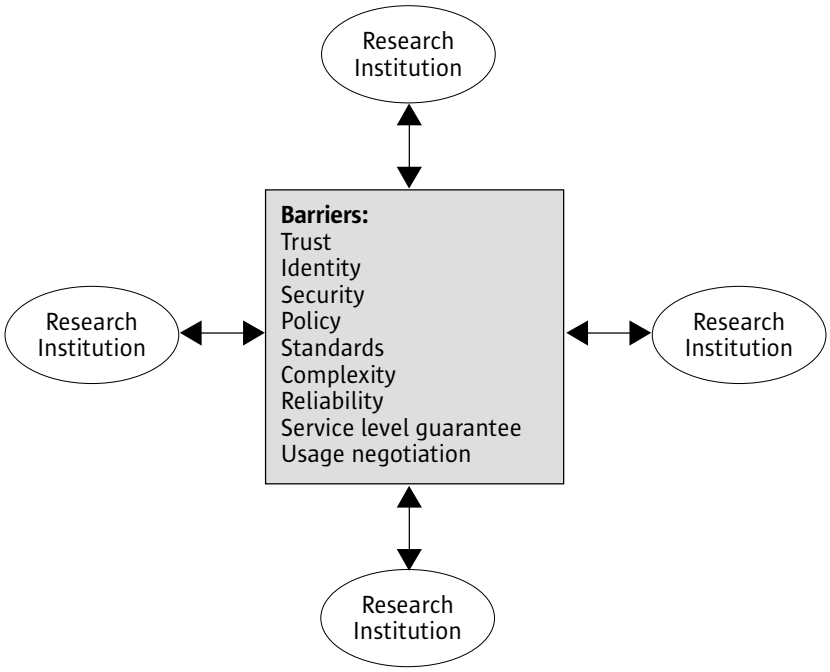


A complete solution to resource sharing must bring together five important HPTC components: computational power, data handling, network communications, application execution and data visualization. To ensure unencumbered collaboration between members of a distributed research community, many of the efforts underway in technical computing focus on how to get data to and from compute platforms, and how to ensure that the application pieces are in place at the right time (resource scheduling and job provisioning workflow).

1.5 Challenges to Collaboration

Sharing, collaboration and management of resources all require new approaches to HPTC. The Services on Demand approach enables organizations to combine application expertise, computational power and stored data in order to achieve advanced research goals with the optimal use of resources.

Despite the progress institutions have made in providing the tools needed to accomplish research activities, there are still some hurdles in place.



1.5.1 Establishing Trust

To collaborate, research institutions must operate within a network of trust. The kinds of collaboration activities that researchers demand are too fine-grained and frequent to depend on inefficient bilateral sharing agreements that must be negotiated whenever they want to access a resource away from their home institution. There is a need to implement automated trust mechanisms that can verify the status of a requesting entity and give access based on a system of trust.

Some areas of research are particularly sensitive to trust issues. For instance, some computational biology applications are partially funded by pharmaceutical companies who have a strong interest in keeping information confidential. On the other hand, some may be less concerned with confidentiality, but may place a high priority on data integrity, especially when their applications must manage large volumes of distributed data, which can be modified in unintended ways. Furthermore, medical institutions have a tremendous trove of useful information, but regulations (e.g. HIPAA) require high degrees of confidentiality and security before this information can be made available for research.

1.5.2 Sharing Identity and Authorization

The Grid vision requires both significant security mechanisms and simple, practical policy management. To achieve "virtualized" access to resources, where researchers only deal with service interfaces and not directly with underlying physical infrastructure, a federated identity model is potentially the best solution. This would allow users to authenticate with a service provider. The service provider may delegate parts of the service request to other providers and can use the identity of the user to authenticate into secondary providers.

1.5.3 Ensuring Privacy

Because bringing resources together across administrative domains is so important, the need to observe standard security measures is great. Firewalls are being implemented not just on system peripheries but also within administrative domains. This poses particular challenges to connectivity protocols. It also requires a greater alignment of access policy models between institutions. This enables protocol negotiation to successfully gain access for a user request without bypassing the intended security measures of the organization hosting the firewall.

Many existing security policies and implementations were designed for a single-site, single-system, single-vendor view of the world. Services on Demand and research collaboration both require a security architecture that is federated and network-centric. The network of trust, once established, can be used to help implement the reliable authentication, authorization and privacy required to support secure computing.

1.5.4 Defining Policy

Policy management affects more than just security. Grid computing is about managing access to resources, and policy management must focus on resource allocation issues that ensure the smooth operational management of computational resources. Policy, especially related to identity, can be used to prioritize requests, manage workflow across multiple requests, measure and charge for resource consumption and distributing results to different user communities according to established policy rules.

1.5.5 Negotiating usage

Currently, researchers apply for permission to use resources through complex administrative infrastructures, dealing with an application process that is not necessarily tailored for the domain of HPTC. This impedes the ability of researchers to form efficient and straightforward collaborations with colleagues at other institutions. There is a requirement for greater standardization on how a user can contract with a service-providing institution for access to the resources that the user needs for his or her research. This should be automated and managed as far as practical by the researchers themselves.

1.5.6 Service Level Guarantees

When researchers gain access to the resources they need, there must be some assurance of the level of service. Reliability of the resources and consistent provision of the same levels of performance are important criteria. The service level guarantees must be coordinated with the identity of the service requester.

1.5.7 Establishing Standards

Protocols for negotiating usage, aligning policy rules between institutions and implementing the underlying technical communication network all require adherence to technical standards. At the same time, each institution that provides services should be allowed the maximum flexibility to build their systems infrastructure with their choice of technology. Rather than a single technology platform across the network, the solution must be based on standards that support a loosely-coupled way of working between collaborating entities.

1.5.8 Managing Complex, Heterogeneous Platforms

Technical computing applications today span a number of different languages, tools, platform systems, and databases. This can create a management nightmare, and in a networked world it is not possible to pretend to solve the problem by decreeing a single environment for everyone. As research increasingly crosses domains and institutions, HPTC solutions must go beyond simply communicating or sharing data, to offer administrators a single, easily managed environment that can adapt to changes in resource demands and availability across diverse systems.

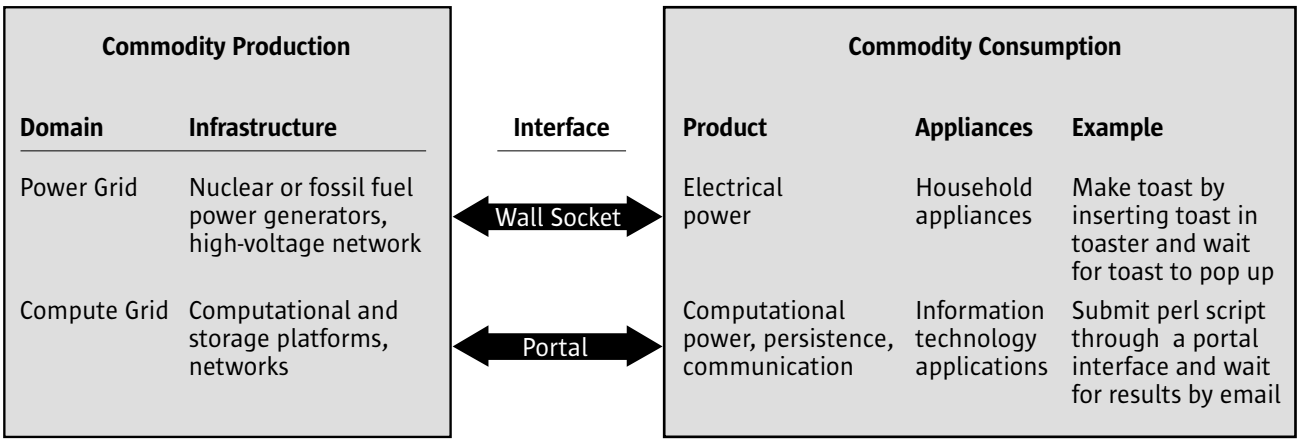
1.6 Grid Computing

Grid computing provides much of the technical infrastructure for Services on Demand. Grid computing promises a generalized access to computational services that makes it easier for users to use computational resources. It leads to better utilization of installed IT infrastructure, more flexible management of compute power across different platforms and different sites, and better availability of resources.

1.6.1 Commodity Access to Computing Power

Grid computing is a vision for breaking down barriers and ensuring uniform, consistent, highly available Services on Demand. The “Grid” borrows the notion of compute services as a utility from the way electricity is supplied through a power grid: it is a ubiquitous commodity. Commodity computing is the transformation of non-commodity capabilities —computation, persistence, communication—into commodity services. The latter is the basis for a profound change in the way that computational services are consumed.

The essence of a commodity product is supply ubiquity and roughly equal quality from one supplier to another. For consumers of the commodity product, the origins of the product and underlying production mechanisms become unimportant. The main differentiator becomes price, terms and conditions of delivery, etc. Delivering computational services as a commodity product (utility computing) should bring significant benefits to users of HPTC.



Supplying the commodity product requires advanced, non-commodity infrastructure, like large scale power generation facilities or nuclear power plants and high-voltage networks that are the infrastructure for electricity supply. Provided that a network exists, consumers have access to a simple-to-use product. Power grids, are, of course, one-dimensional since they provide a single, well-defined, homogeneous product, viz. electricity of a certain voltage and frequency.

The plan for a Grid of computational services has been around for years but recently has picked up considerable momentum. Many of the major software and hardware suppliers to the IT market have declared Grid strategies. While most of these are just plans, the benefits to researchers seem to be too great for it all to dissolve in a cloud of hype. Perhaps because of this, many of the ideas for the Grid come from research communities in the first place, for instance the Argonne National Research Laboratory and the UK eScience program among others. Computational grids differ from power grids in that the resource, computational "power", is quite heterogeneous in nature (integer vs floating point, memory and bandwidth limitations, programming languages, etc.).

1.6.2 The Open Grid Services Architecture (OGSA)

While Internet standards are the backbone of the Grid, numerous Grid-specialized protocols and architectural conventions need to be developed to enable a high degree of interoperability with the robustness required of a utility service.

The de-facto standard for delivering grid services is OGSA. This emerging standard is defined by the Global Grid Forum (GGF), which was formed in 2001 by the merger of Grid forums in Asia, Europe and the US. Its purpose is to actively encourage the exchange of information on computational Grids. It provides an open-standards process that is modeled on the IETF (Internet Engineering Task Force).

OGSA includes several specifications that are the basis for a service-oriented, on-demand utility computing vision.

1.6.3 Globus

Globus is a project dedicated to developing grid technologies. While the GGF is an emerging standards organization, Globus produces software implementations. The centerpiece of the Globus effort, the Globus Toolkit, is widely deployed among research institutions. It implements some of the functionality required to run a Grid. It includes security, remote job submission and control, resource directory services, and high performance secure data transfer (using an FTP-related protocol).

In January 2003, Globus published the first implementation of a new version, the Globus Toolkit 3.0 (GT3). GT3 is a complete redesign that is not fully compatible with previous versions, but this new version implements the most recent GGF OGSI standard (Grid Services Specification).

1.6.4 A Networked Marketplace for Grid Services

The most recent OGSA specifications allow grid services to be published on the network as Web Services. When combined with the type of infrastructure for networked applications and management that Sun provides, this standard can provide the basis for sharing computational resources across the network.

A major goal of grid development is to reach the point where grid services can be made available to other organizations on a commercial basis. This networked marketplace for computation, which requires commercial as well as technological development, would make grid services pervasive on the Internet. The momentum for this development has already begun with the drive towards collaborative solutions within the high performance computing community. We can expect that it will become more generalized once it has proven its success in the technical computing environment.

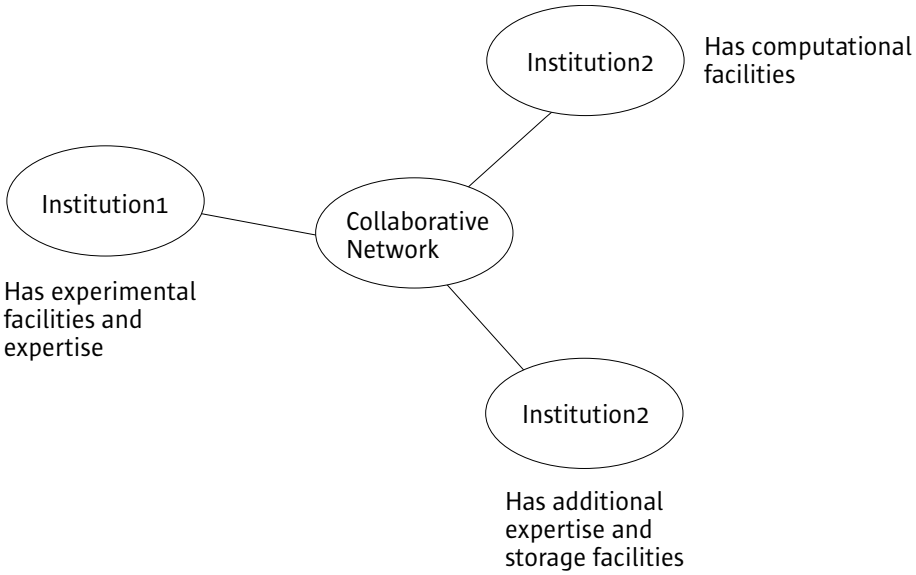
Chapter 2

2A Standards-Based Grid Architecture for Technical Computing

In this section, we outline an architecture for providing grid services based on widely recognized standards. The goal of the architecture is to overcome the barriers we indicated in the first section to build a robust and reliable network of computational services for technical computing.

2.1 Usage Scenario

Let's refer to the example in the illustration below. A researcher sends a request to a service to ask that a program (Fortran, C, C++, Java, etc) be run on a data set that is the result of an experiment.



The researcher's home institution has excellent experimental facilities (e.g. a high energy particle accelerator, biochemical laboratory, fluid dynamics lab, etc.). The results of the experimental activity is a large data set. The researcher builds a script or program that will perform analysis on the experimental data and return a set of data in turn. The researcher's own institution may not have the appropriate computational platform. Another cooperating institution does have the computational resources. In addition, the researcher is working in concert with colleagues at other institutions and the data, both experimental results and analysis result set, must be available to them.

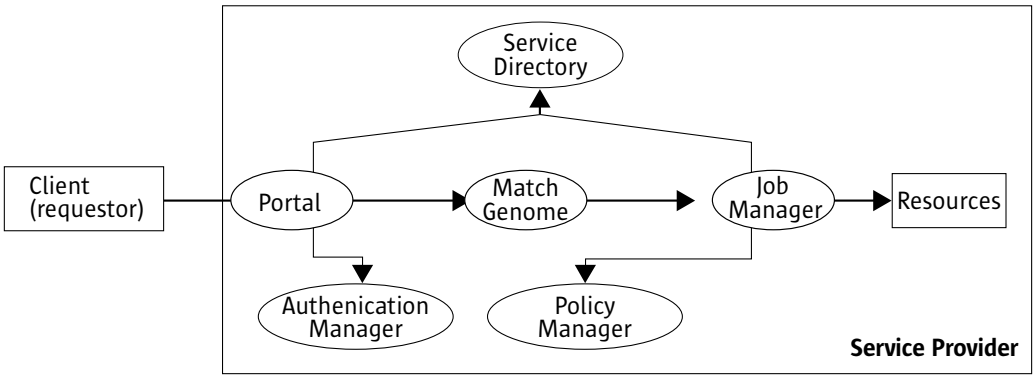
The requirements of this collaborative network will be the following:

- **Present information and application interfaces.** A portal mechanism for the service providers to communicate their services, what the usage policies are, look up their account information to see how much of their allocated usage has been consumed this month, etc.
- **Identify the requesting user of the resources**, maybe let the requester verify the service provider as well.
- **Secure the communication** to preserve data integrity and confidentiality of the data until the research is ready to be published.

- **Find the appropriate resources.** Ability to query institution's services for right resource fit ("Please run this script on a large multiprocessor machine with +1GHz CPU speed, with this data with at least 100GB of local memory, should take about eight hours")
- **Observe rules that govern resource usage.** Policies to tell the services whether they should allow access to everyone, to restricted researchers, to be able to make decisions if the resources are over-used or under-used in relation to the original request.
- **Track resource usage.** This could be helpful later to determine compliance with service levels, implementing policy, charging usage back to the requesting university.
- **Schedule the work** with the local machines in a way that is as isolated as possible from the user and front-end systems. An entity behind the scenes that manages the physical computational and data storage resources according to the policies of the services provider and the parameters of the requester.
- **Notify the user** and other constituents of the service about important events.

2.2 Basic Infrastructure

In the below diagram, we have a basic infrastructure—greatly simplified—showing how a client can get information about a service provider, request a service and allow the service provider to take care of the work according to specified policy constraints. The ovals in the diagram represent the complex of cooperating services that are required to meet our requirements as specified above:

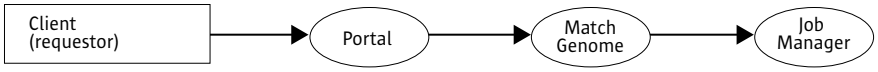


- **Service Directory**—A source of information on services in the system so that any service can find out about another one.
- **Portal**—The presentation layer to the clients, ie. end-users.
- **Match Genome**—This is an example, rather simplified, of an application service.
- **Job Manager**—This allocates compute jobs to specific resources and manages their execution.
- **Authentication service**—The Portal only lets certain individuals use the site and associates the login event with a user profile, i.e. an identity
- **Policy Manager**—This service is used by the Job Manager to ensure that resources are used according to a policy that corresponds to the identity of the authenticated user. An example would be ensuring that a user gets access to resources up to a particular time, or assures a particular level or resource access during job execution. Policies are critical for large Grids to ensure appropriate allocation of resources.
- **Resources**—These are the computational, storage and network communication resources that are ultimately being requested by the user.

A service is an entity capable of fulfilling a request. It is the combination of an *interface* and some *behavior* and users can communicate with it via some *protocol*. It is loosely coupled with those making use of it—so it can't be a programming construct, it must not require that its clients be running on the same product, technology, programming language, etc.

Furthermore, services need to talk to other services using some protocol that each understands. Though, we may be able to use a different protocol between services if we can find a way for each to know which protocol to use at the right time. The concept of a services here is much higher level and more general than an "object" from object-oriented languages, because it doesn't need to be implemented with a particular language or technology. It doesn't even need to be implemented in an object-oriented language. Users of the services don't need to be the ones creating or administering the service.

We could further abstract the provision of resources to the end user so they are not directly involved with knowing about the underlying hardware resources. We could add a further service to our diagram to see an interesting distinction that illustrates this point:



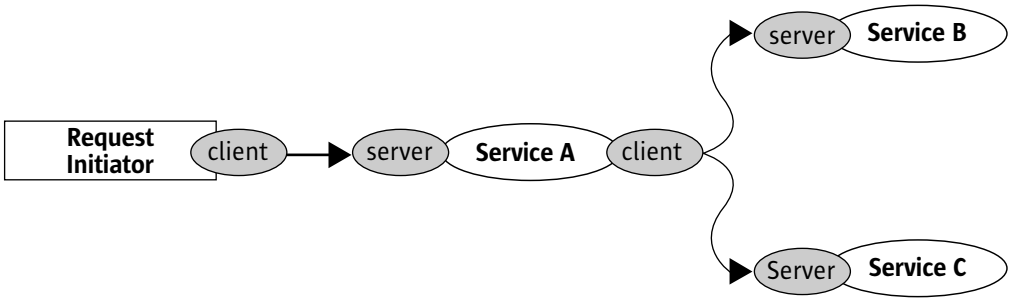
In the illustration above, let's suppose that "Match Genome" is some sort of genome matching service that takes a file containing gene sequence information as a parameter and returns a result set. Notice there is a difference between Portal and Job Manager on one hand and the Match Genome on the other hand. The first two services might be designated as infrastructural services, because they do not represent a helpful application in themselves (from the researcher's point of view) as Match Genome does.

One temptation in designing our system would be to separate out the so-called infrastructural services and allow them to communicate by a protocol only known between them and not through the public protocol that connects a user with a specific application service. However, there are some compelling reasons why we don't want to create two classes of service:

- **Sharing**—If the infrastructural services use the same interface conventions as the application services, we can more easily share them, since the application services use a standard way of exposing their interfaces already (i.e. users external to the service provider are able to connect to them).
- **Flexibility**—There may be cases where it is difficult to decide if a particular service is infrastructural. For instance, the Directory service could be accessed directly by an external user (external to the service provider, i.e. in a different administrative domain).
- **Complexity reduction**—It will be much easier to maintain a single set of protocols.

So this lets us consider almost everything a service. Services themselves therefore are clients to other services. For instance, our Portal service will use the Authentication Service (act as a client to it) to establish the credentials of that user.

This helps sharpen our needs somewhat and provides a much more generalized model of services that we can build on in subsequent sections.



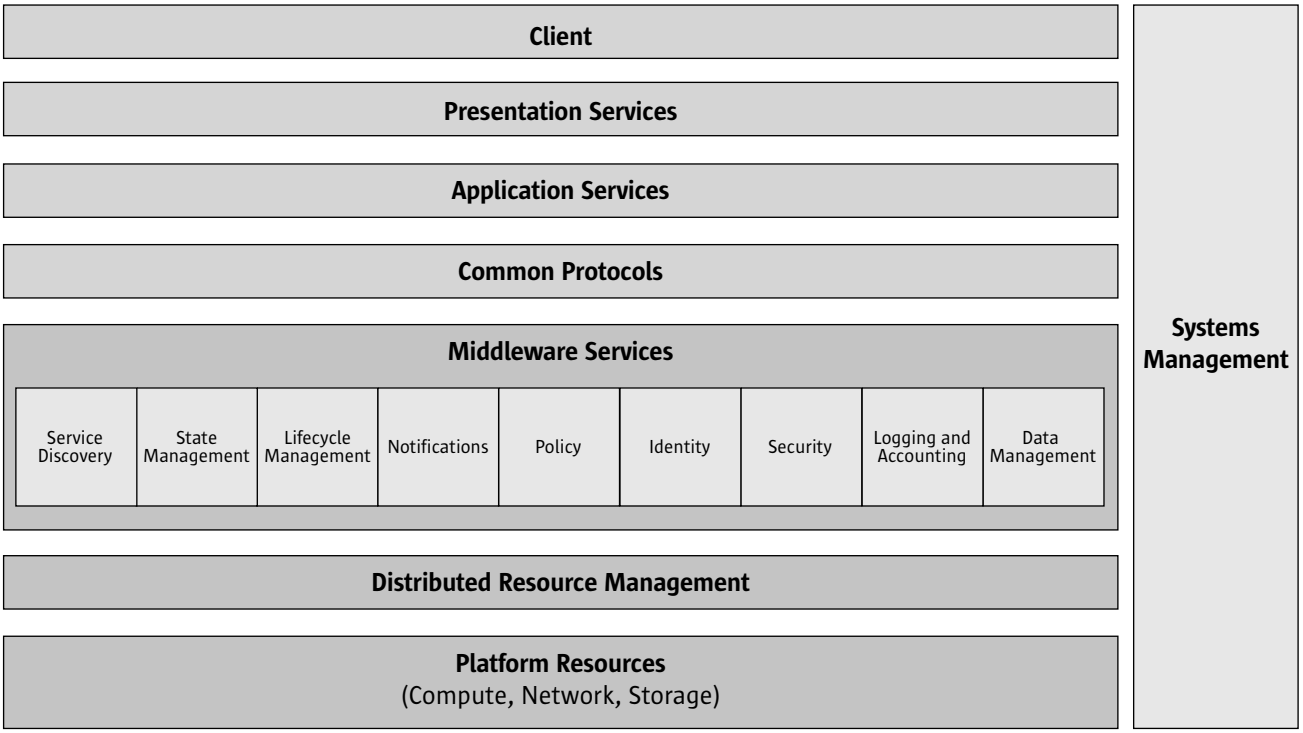
The next thing to do is decide on a protocol that can be used between all the services in a system. We might not want to come right down to a particular protocol, like, say, CORBA-IIOP or HTTP GET or even some non-IP based protocol, because different concrete protocols might be needed for given situations and end-user requirements, or even just to get through a firewall, even if we don't want to use a lot of them. For instance, the protocol between the "Request Initiator" (the entity first asking for a service to be completed) might use HTTP to send a request to Service A and Service A might talk RMI-IIOP to Service B. So how does a service find out what protocol another service uses to communicate?

The solution is to have a protocol for figuring out what the protocol is. We'll see that this is the solution proposed by the OGSA. We can now think in terms of the original Grid vision mentioned in the first section where there is a global network of inter-operating services working to fulfill very complex tasks for researchers around the world.

2.3 Architecture for a Network of Services

In our previous model we skipped a lot of the services that would be required in a real Grid deployment, in fact we have presented rather simplified views of the services we did include. The above figure illustrates a few more aspects of a real system that need to be taken into account.

This architecture applies to a distributed network of interoperating service providers. So, for instance, policy must be applied from one research institution to another. Identity must be shared amongst many institutions who host resources that are meant to be shared with recognized users.



One can see that we have most of the basic services at the top of the diagram:

- **Client**—A way to access the service provider, a web browser, a Java™ application or some other client-based technology. This may be a service within the local domain or outside it.
- **Presentation Services**—This might be a Web site that provides information about the available services and applications of the hosting sites, but is also the infrastructure that is on the front-line between the services provided and the end users generally. It is also the interface between administrative domains, for instance, between two research institutions.
- **Application Services**—The services that constitute the applications that can be accessed through the portal that perform some useful benefit to the research community.
- **Middleware Services**—Services that are needed by most application services.
- **Resource Manager**—Executes the job across a domain of system resources, implementing sensible policies while doing so.
- **Platform Resources**—The resources, computational, network, and storage.
- **Common Protocols**—Standard ways to communicate are the starting point for any attempt at building a collaborative network. Client applications need to be able to establish sessions with servers. So we need to agree on what communication protocols to use above the network connection. We also need to agree on a minimum set of behaviors between client and server to ensure that two parties in the network can
  - Get a list of available services
  - Establish a connection with a service
  - Query a service about its capabilities
  - Query a service about its state
  - Use the service
  - Deal with exceptional situations like errors

We need to do these things no matter what services the client requires. This set of behaviors also has to be able to deal with firewalls, security, policy rules, etc.

2.3.1 Connecting to a Service (SOAP)

Simple Object Access Protocol (SOAP) is a communication protocol based on XML. It is widely supported and can provide a good foundation for a standard communication protocol in a network of collaborating Grid services. It provides access through some firewalls, it can be transported over HTTP and SMTP. It provides a means for clients and services built on different technology platforms to communicate using XML as the basic standard. There is an abundance of tools for building clients and servers with SOAP. While many related standards are still being developed that are necessary, it provides a good starting point for a standards-based exchange of data. We should note however there may be times when other invocation protocols may be useful.

2.3.2 Finding out about a Single Service (WSDL)

Another standard that complements SOAP exceedingly well is Web Services Description Language (WSDL). SOAP is the communication and invocation protocol that lets a client establish connections and invoke the capabilities of the server. WSDL lets the client find out:

- What operations are available on a server
- What kind of messages the server expects to receive and send
- What is the exact transport protocol to use for these operation requests

As one can see from the last point here, WSDL will let us choose our transport protocol dynamically. This means we can use the most appropriate protocol as necessary.

2.3.3 Choosing from among many Services (UDDI)

WSDL is an XML document. With it, a client has what it needs to establish a connection with a service. To find the WSDL document, we need to introduce one more piece, the service directory. A standard for a directory of services is Universal Description, Discovery and Integration (UDDI). We can also refer to this as service advertisement. A service provider uses UDDI to advertise the services that are available. A client uses this to find the service that is appropriate for it to use. Essentially, a service advertises a WSDL document in a UDDI directory. The client queries the UDDI directory and gets a WSDL document back.

2.3.4 Adding Reliability, Robustness and other Grid Qualities (OGSA)

The foregoing standards are critical to developing a common way to communicate, and they provide the basis for a network of interoperating services. However, they are not sufficient for a Grid of services. The SOAP/WSDL/UDDI complex does not address issues about how a service behaves. It does not tell us how we initially find a service. It does not give us a standard way to query a service about what capabilities it offers in relation to the application value that it provides. Furthermore how can we introduce mechanisms to ensure a consistent and robust level of service?

The Open Grid Services Architecture (OGSA) specification provides many of the answers. It supports the Web services protocols as underlying standards for interoperability and builds on these to provide a basis for a real network of Grid services.

SOAP does not say anything about managing the state of a service. Nor does it specify the behavior a client can expect about the lifecycle of a service. Services will have to communicate with each other about important events. For instance, a registry of services may wish to be notified if one of the listed services is not available anymore, or a handle resolver (handles that identify services) may need to find out if a service instance has been destroyed.

Some of the things specified by the Grid Service Specification:

- **Identify service instances.** A way to identify service instances independently of their technical implementation is needed. The environment that hosts the service may create and destroy objects as needed to accomplish implementation-dependent tasks or to comply with the underlying platform constraints. For instance, the hardware the service is running on may go down before the service has completed fulfilling the request. It is part of the robustness and reliability requirement of the Grid that the identity of a service instance (i.e. the association of a request from a client with the attempt by the service to satisfy this request) be guaranteed to persist independently of the transient events affecting the underlying hosting platform.
- **State management and lifecycle framework.** The service must provide the client with a framework for understanding the data it receives from the service. For instance, how long is the data valid? When will it start to be valid? Until when can the client expect a service to be available?
- **Bootstrap protocol.** Before a client can start working with a service, it needs to find the directory of services, get the service description, ask for an instance of the service, etc. It will be helpful to establish a protocol for getting a reference to the first service we'll use in this case: the directory service. A bootstrapping protocol will help out here. This lets us optimize the connection to services the client has already begun to use in a previous session or if the client is reconnecting after a transient network partitioning event that broke the client connection.
- **Notifications.** Clients will need to be made aware of important events, like when a service is created, destroyed. When has a service request been fulfilled? A framework for notifying participants in the network of Grid services is necessary.

The OGSA also provides standard descriptions of interfaces for several of the infrastructural services we mentioned above:

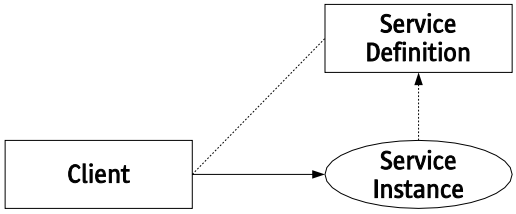
- **Grid Service**—The standard interface from which all services are defined. This simple interface description is the critical point of departure for all services in our network of services.
- **Factory**—A standard service type (a kind of Grid Service) that handles creation of service instances.
- **Registry**—The base interface for the registry of services, where any client finds out about what services are available.
- **Reference Resolver**—The base interface for the service that enables a client to resolve an identifier for a service to the reference that permits the physical protocol binding.

An important feature of the Grid Service specification is the ability to handle data in a standard way. The OGSA provides data naming and handling based on XML data types. This allows a service to be queried about what kind of data it provides to the client.

2.4 Service Hosting

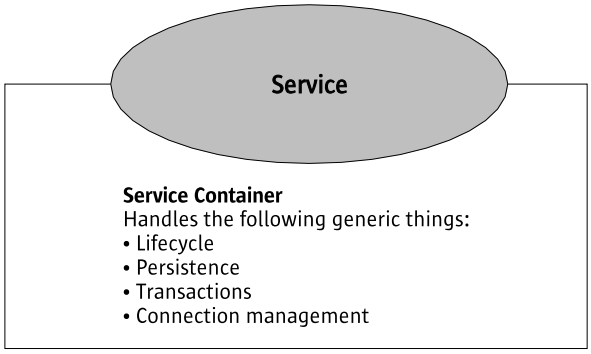
We have a rough idea of the kinds of functionality that we need for our vision of Services on Demand. Implementing and deploying services in a concrete infrastructure requires a range of considerations that are not specific to a Grid computing infrastructure, but are required by any service-oriented architecture.

When services are requested by a user (could be another service), we need to manage the data that is associated with the service instance. This constitutes the state of a service and along with some kind of identifier constitutes the identity of the service instance.



It will be helpful to have a small model in mind while discussing this: Services have a definition that is independent from any instance created to serve a requesting client. The client uses the definition to understand the interface syntax of the service. Typically, there will be a different instance of each service for each requesting client, though this does not always have to be the case. In any case, a service instance will need its own unique identifier, so there can be many service instances for one definition. Ideally, this identifier should be globally unique even across administrative domains, since we are aiming to put together a network of services, a Grid. That will make the location of services unimportant from the user's point of view. Service instances need to be created and destroyed according to a set of rules that ensures the hosting environment is not stretching available resources. It also needs to ensure that the client can reliably depend on the existence and accessibility of a service according to rules the client knows about.

Services in a Grid infrastructure need to be accessible by remote clients, which could be end-user operated graphical user interfaces or a service under another administrative domain. Remotely accessible service architectures have been around for a long time now. We'll refer to the software environment that hosts these as service hosting containers. An example is a Web container. This might be a container for deploying and hosting a static HTML page, a Java servlet or a Soap service. Another way of hosting services is through an application server, like a J2EE™ compliant application server that uses the Enterprise JavaBeans™ standard. The advantage of full-featured, standards-based service containers is that they take care of a number of functions that are bound to be present no matter what functionality a service finally provides to the user.



In the above diagram, a service that performs some application-specific task implements only what is necessary to accomplish the application goal, while the service container carries out the tasks that are the same from one application to another.

- **Lifecycle**—Service instances will typically be created when a user requests the service and destroyed when they are finished. The service container can manage this to a large extent by providing factory interfaces that manage lifecycle. The container can also implement optimizations like pooling of instances. Scalability can be achieved by swapping instances out of memory that are still needed but are in a dormant state. They can then be brought back into non-persistent memory when the connection with the client has a request on it.
- **State management**—Services will have some behavior that operates over a domain of data, the state of the service. It will be useful if the service container can help manage this data by perhaps providing some optimizations in the case of services where the data is read-only or the data is rarely directly changed by the service itself, for instance, if it comes directly from a database. As mentioned with lifecycle management, if a service has its state reconstituted from a dormant state, it would be nice if the service container would be responsible for making sure the state is the same as when it was taken out of memory.
- **Persistence**—The container could manage the persistence of the service, for instance, if the data came from a database, it would be rather convenient to have the service container worry about where the database is, what kind of protocol it is using, when and how the data needs to be read from the database. This is particularly the case if the service container can allow the service developer to declaratively define the data properties of the service.
- **Transactions**—Transactions may be complex for a service requiring a consistent handling of data to be supported by sophisticated transaction management functionality. For instance, ensuring atomic transactions with more than one data source.
- **Connection**—The connection with the client usually needs more sophisticated handling than a simple TCP/IP connection. The service container can take care of numerous issues that will contribute to the stability and robustness of the system. Connections that have been interrupted can be re-established without the client needing to perform complex logic to figure out if the service is still accessible. Connections can be pooled when a single client is accessing several services in the same hosting environment. Optimizations can be implemented when the client and server are co-located on the same host.
- **Events**—Clients and services will need to be kept up-to-date on important events in the system that could be relevant to the correct functioning of the system, make it easier for the users to understand what is happening or keep track of activity for auditing purposes.

2.5 Presentation and Service Delivery

Providers of Grid services need to present their services to the outside world in two ways: 1) information about services for interaction with other services in the network and 2) information about services provided to human users.

Ultimately, human end-users, researchers, want to access system resources through applications that further their research goals, i.e. bring as much computational power as possible to bear on the problems they are trying to solve. They will need to access services through an interface that presents them with a range of application choices.

The portal is the interface for a set of services that is user-facing. It should include functionality for the following:

- **Personalization**—Services should be presented to the user increasingly in a way that is suited to his or her needs as a researcher. Personalizing the presentation of application services will make it easier for the user to see the information that is relevant to him or her. So, a genomics researcher will not be interested in an application for complex sub-atomic particle collision event simulation and a high energy physics researcher does not need to see applications for modeling the propagation of plankton bloom in the ocean. But a researcher may want to be updated on available applications from disciplines that are adjacent to his or her own field. Therefore, the ability to customize the view of available services will be a personal choice of the individual researcher.
- **Aggregation**—The applications that are available to researchers will come from many sources and a portal serves to bring these all together under a single, configurable view.
- **Interact with applications**—Initiate jobs, get notified that jobs are finished.
- **Locating services**—Users will need to search across a potentially broad virtual space for the resources they need to accomplish their goals. A portal can provide more or less extensive searching functionality.

In addition, the portal will be the starting point from which identity, policy and security issues are negotiated with the user. Users need to be able to see what status they have, and what usage rights are accorded to them by a service provider. In the case that they are granted a certain amount of compute time on a computational facility, for instance, they need to see how much time is left.

2.6 Distributed Job Scheduling

When a user submits an application request, i.e. lets the system know that he or she would like to have a job done by the system, the Grid service must schedule the incoming request with a manager that is local to the resources that will be performing the job. The current standard interface for accomplishing this among distributed resources is the Globus implementation in combination with DRMAA compliant DRM systems.

2.6.1 Globus Toolkit

The Globus Toolkit is used to schedule jobs across domains or within an administrative domain. It is not used, however, to schedule jobs to an individual node within a cluster or to an SMP system. For this purpose, a DRM (Distributed Resource Management) system is utilized. It has several services that enable users to discover resources, inquire about capabilities and submit a job with a specification for how the job is to be performed and the profile of the resources to use.

Globus can be divided in four parts:

- **Connection and Security**—The Grid Security Infrastructure provides the security protocol that authenticates requesting clients and the server. It uses the Generic Security Service API (GSS-API), based on X.509 public key encryption standards and secure sockets layer. Security properties provided are authentication, privacy, and data integrity.
- **Information services**—The Monitoring and Directory Services (MDS) is composed of components that query and index the capabilities of a domain of computational resources.

- **Resource management**—The Globus Resource Allocation Manager (GRAM) provides a secure external entry point into the local Distributed Resource Manager (see below).
- **Data Management**—The data management module provides data transfer services using an extended FTP protocol to ensure secure and reliable transfer.

Globus is often used in conjunction with a full-featured Distributed Resource Management System (DRMS).

2.7 Distributed Resource Management

Distributed resource management allows job management to be extended to an entire domain of resources that may contain heterogeneous platforms and operating systems. This allows an organization to increase resource usage efficiency by scheduling across many machines and doing so according to rules that enable more optimal usage of those resources. Quality of service can be enhanced by automatically tracking nodes that are down or performing inefficiently. A DRM system has the capability to virtualize a domain of resources so that system administrators can operate over the capabilities of a network and not just on the more complex physical topology.

DRM systems provide batch processing capabilities, dynamic allocation of resources, fault tolerance and failover. Managers can examine job status and remotely suspend and resume jobs.

DRM systems have an important job to play in implementing policy constraints on a domain. Policy constraints are critical to providing quality of service guarantees and make sure that the owners of a given resource can stipulate the usage conditions under which they share with a larger domain.

In a heterogeneous Grid environment, it is critical that DRM systems managing different domains comply to standardized specifications. Hence, the DRMAA working group of the Global Grid Forum (www.drmaa.org) seeks to "develop an API specification for the submission and control of jobs to one or more Distributed Resource Management (DRM) systems. The scope of this specification is all the high level functionality which is necessary for an application to consign a job to a DRM system including common operations on jobs like termination or suspension. The objective is to facilitate the direct interfacing of applications to today's DRM systems by application's builders, portal builders, and Independent Software Vendors (ISVs)."

Not only will DRMAA allow commercial software vendors e.g. in the EDA arena to enable their software to utilize DRM capabilities without tying into a specific DRM system, it will also allow to schedule jobs across multiple domains, each manage by different DRM systems like Condor, PBS, LSF or Sun Grid Engine.

2.8 Identity, Authorization, Privacy

Many other services will rely on identification of users to be able, for instance, to associate the user with a set of rules that govern resource usage, allow or forbid access to the system, or allow the user to be logged in automatically wherever necessary and where policy permits (single sign-on). Identity is therefore a critical aspect to many other functions within a Grid infrastructure. Especially in a system that will cross many administrative domain boundaries, a way needs to be found to ensure that identity of users and organizations is efficiently managed.

A system of trust needs to be instituted with the global grid of services. The identity and security infrastructure will depend on the network of trust that exists among the world-wide participants of the Grid.

Security mechanisms need to be built on top of the trust and identity networks to provide basic services such as authentication, authorization, privacy, and single sign-on. The usage of services needs to be logged to enable subsequent audits of activity and backtracking to determine problems. Modules need to exist for managing user, policy and individual service information (for instance, configure the access control to functionality within a single service).

- **Authentication** — A user must be associated with an identity that is known to the system. Subsequently, the user can be authorized to access resources.
- **Privacy** — It must be possible to provide confidentiality during communication between a user and the service. The user must also be able to expect data to be accessed only by him or her as configured by the user. Careful thought needs to go into the need to share data among multiple users without compromising privacy.
- **Authorization** — After being authorized to interact with a system, a user may be able to access some resources but is rarely allowed to access all resources. Provision must be made for specifying to which resources a user has access.
- **Single sign-on (SSO)** — In a collaborating network of services, it is critical that a user's identity be propagated automatically to whichever services are involved in satisfying an authorized request as far as this is consistent with policy.
- **Policy** — Policy must be enacted in relation to the identity of a user.

Tools need to be provided to handle the Identity infrastructure, including development tools and management tools. This functionality may need to be exported through a portal for interactive use by the users or administrators or programmatically through standard protocols for services to share identity.

2.8.1 Distributed Identity Management

In a loose network of services as embodied by the Grid, identity services cannot be strongly centralized. There needs to be a way to share identity amongst distinct spheres of administration. There may be a set of primary information managed by the home institution of a researcher, but another institution may need to manage information related to that identity that is only pertinent to the access of resources that the home institution does not possess. This is especially the case for resource access policy management where service-providing institutions need to associate a set of rules that constrain resource based on user identity.

- **Directory services** — LDAP (Light-Weight Directory Access Protocol) is the standard for directory services. It is a protocol for communicating with a directory of resources including human agents.
- **Management interfaces** — Interfaces that allow user interfaces to be built for user self-management as well as administrative management must be present.
- **Protocols** — Distributed login protocols have their own requirements. Implementations for doing decentralized login will be required.
- **Standards** — One standard that is gaining increasing acceptance is SAML (Security Assertion Markup Language) for communication between authentication services.
- **Heterogeneity** — The ability to deploy on heterogeneous platforms is a critical success factor for an identity management infrastructure.
- **Policy management** — This component provides interfaces to create, delete, and modify policy rules that define the service and access privileges for users. For example, the protection of a Web resource can be based on a user's identity as well as on conditions such as "can be accessed only from 9AM to 5PM, Monday through Friday."

2.9 Logging and Accounting

Usage events need to be tracked to understand what a user did with a system, how much resources they consumed (e.g. amount of CPU time). The non-repudiation function will rely on logging. The accounting function is necessary to ensure that usage is correctly recognized within the framework of service level agreements.

2.10 Data Management

Data needs to be handled in as uniform a way as possible by the whole system to reduce complexity and enable some of the other functions to work properly.

A platform with the above characteristics could be built with numerous technologies and protocols. In fact, many institutions have infrastructure and offer services that provide much of this functionality. The problem is that they are largely isolated because of the proprietary behavior and protocols they use. Even using a similar transport protocol, say HTTP or FTP, would not allow diverse organizations to share things as complex as a Grid infrastructure. Therefore, part of the standard for Grid services needs to include standard behavior of services, i.e. some semantics. Naturally there can be no standardization of all behavior since we need to be able to build application-specific services (bioinformaticists won't be interested in a service that helps us study turbulent compressible magnetoconvection in star sunspots, for instance). However, the behavior common to all services can be standardized as we have attempted to show above.

2.11 Tools

Tooling will be required to efficiently produce and manage applications:

- **Development** — Development of the Grid infrastructure and specialized technical applications demands development environments, compilers in Java, C/C++, Fortran and other languages.
- **Integration** — Even new applications and services will likely take advantage of existing systems. There is a need to provide tools to make it easy to integrate existing backend systems.
- **Deployment** — There must be a way to deploy new applications and allow participants in the Grid to become aware of new services.

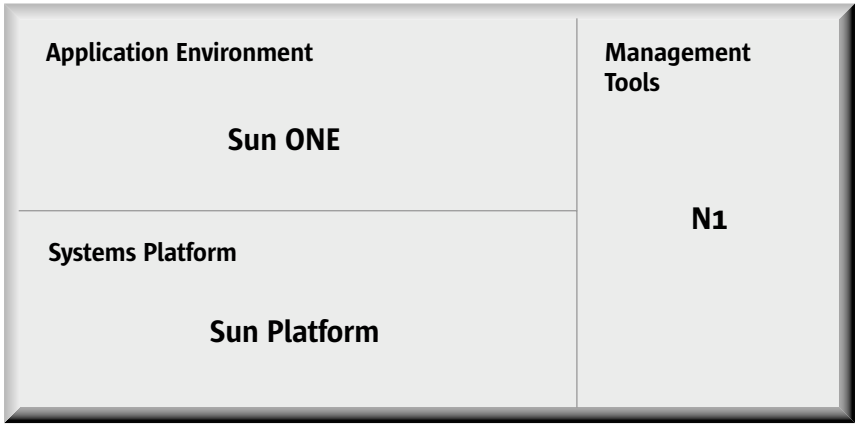
In the following we'll give an overview of some of the components that Sun Microsystems provides to satisfy the requirements we've just outlined in a standards-based way.

Chapter 3

3 Sun Microsystems in Technical Computing

Sun Microsystems provides many of the required building blocks identified in the last section. While Sun provides a full product range for building a Grid infrastructure, it is the adherence to standards and open protocols that will ensure a successful Grid endeavor. It is as important as ever to build against a plan of interoperability. The Global Grid will be built on open standards. Sun provides three categories of components that are built on open standards:

- **Application Environment**—The set of software needed to deliver user functionality.
- **Systems Platform**—The set of system resources that are exposed to the user through applications, primarily networks, compute and storage platforms.
- **Management Tools**—The ability to manage complex, heterogeneous environments from large SMP to highly granular Blade systems running Solaris or Linux is a key factor in enabling the successful operation of a Grid.



3.1 Application Stack

Sun ONE contains the tools, middleware and hosting software for building a Grid infrastructure. The core of Sun ONE is the J2EE software platform. J2EE itself has at its heart, Java, a language and a platform standard. Java is the world-wide standard for building the broadest array of systems. While HPTC applications will be built using a variety of languages (see below), Java provides the common glue for tying everything together across platforms and across administrative domains. The J2EE platform provides almost all the needed middleware services for a utility-based, services on-demand infrastructure. The sections below will go into some detail about a very few aspects of Sun ONE that are particularly relevant to a utility computing model:

- **Web Services**—This standards-based technology is heavily supported in Sun ONE both on the platform level and in Sun ONE development tools.
- **Sun ONE Grid Engine/Enterprise Edition**—Sun's widely deployed Grid software.
- **Sun ONE Identity Server**—Standards based tools and services for implementing identity management across domains.
- **Sun ONE Portal**—Presentation services for services on-demand.

3.1.1 Web services Toolkit

The development process has been evolving over the last few years. As computing speed has increased, so has the ability to build larger and more complex systems. With this complexity has come the high cost of development. In some areas the tools have needed to become intelligent to assist in the development process. Some of the most noticeable changes have been in the performance of today's compilers. These compilers are doing the jobs of the skilled developers in the past, but the tools are not perfect. Most of these tools can optimize code for a specific platform, but today's applications are made up of assembling many different resources into an application. It is not about the single machine anymore.

This change in the way we look at computing is a by-product of the improvement in networking. The challenge in the enterprise now is how to develop and deploy within a distributed framework. When you include campus, enterprise, and global grids then the need to track all elements of an application becomes critical. This is where conventional tools fail. The traditional tool can track all elements of an application because the application runs within the same machine as the development tool.

In the computing architectures of the future, tools will need to work with an authentication framework in place, and be able to distribute monitoring agents to all the host machines that may be running a section of code. This can be a daunting challenge, one that can only be solved in conjunction with the Operating Environment. If you confine your vision to a single Operating System, then you are limited to that environment. In the case of Java with it's virtual machines, you are now distributing the Operating Environment to many different Operating Systems. Now with the agents written for Java, your tools can distribute agents throughout the environment.

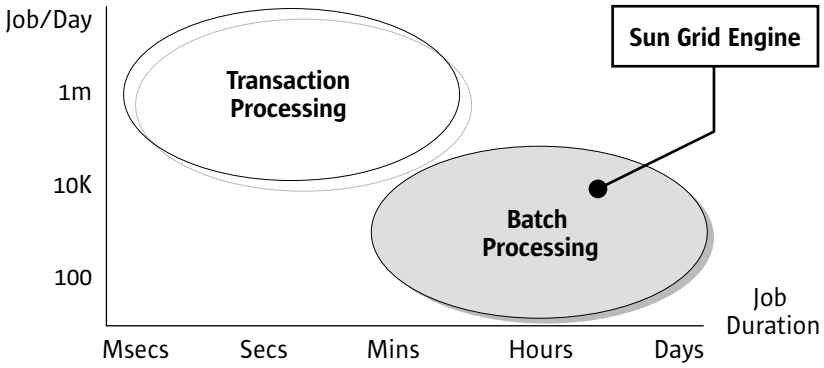
One point to understand when looking at Web Services and evolution of traditional application development. Traditional approaches make certain assumptions about security and data protection. They rely on the Operating Systems to protect the data. When you look at a Web Services architecture the Operating Environment must take on this responsibility, which means security needs to be blended into each message or data element which is moving between two points. Now this security is dependent on the ideas and points raised in section 2.9. In section 3.1.3 we will be talking about specifics of how to manage that security and identity framework. A properly secured Web Services environment will tag all data (XML) with the proper security tags. This where SAML steps in to help.

All the compilers at Sun have evolved into a common look and feel and is distributed through the Sun ONE Studio 7 (formerly Forte™ Developer). Some key elements that should be noted are: NetBeans™, fpp, Interval Arithmetic, J2EE, XML. The Sun ONE Studio 7 (Forte' for Java) enables you to create, assemble, and deploy enterprise applications as Web Services. It includes powerful new wizards that rapidly generate XML-based Web Services from Java technology components. While most of this is oriented towards J2EE, it does not preclude a mix of programming languages.

When you assemble tools to complete the task, you can leverage the best tool to do the work. That is to say that you can build the high performance code in Fortran and wrap the data movement in XML and SAML and use Java to preprocess and prep the data for the Fortran program to execute on the data. This is where the Sun ONE Studio 7 excels. The tight Integration of the compilers and the Operating Environment will assist the developer in deploying the application. This integration environment includes the Sun ONE Web Server (Enterprise Edition), and the Sun ONE Application Server. The deployment to a Web or application server is made possible with IDE "plug-ins." For further information, please review the many white papers available from Sun that discusses these topics.

3.1.2 SGE/EE

Sun Grid Engine is a DRMAA compliant DRM system. It is the essential component in the software stack enabling a cluster grid and an important component within distributed Grid infrastructures.



Compute intensive simulations are in production in industries. There are many examples:

- Electronic Design Automation (EDA) for simulation, verification and regression testing for software development and hardware design
- Life Sciences for genetic sequencing and proteomics
- Financial services to perform risk analysis
- Scientific research
- Automotive and aeronautical (MCAE)
- Oil and gas
- Digital Content Creation for movie and TV industry

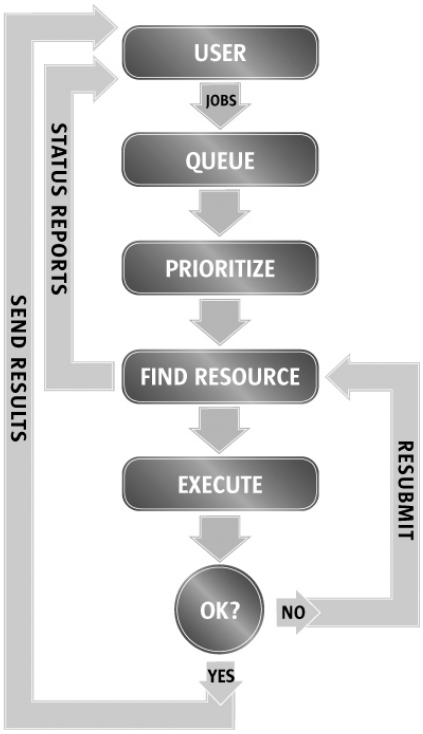
Applications in these industries are compute intensive. The figure above shows the difference between commercial transactions normally handled in a data center and grid computing jobs.

The volume of commercial transactions is as high as millions per day. They may last milliseconds or seconds. By contrast, typical grid jobs are thousands per day, and they last from a few minutes to many weeks. In commercial transactions, the system administration needs to figure out what resources to allocate to each job. Grid computing jobs specify the resources needed to get processed .

Future releases of Sun Grid Engine Enterprise Edition will integrate with N1™ platform software like N1 Provisioning Server. This will open the door to a single console for both commercial data center. Grid computing is now part of the N1 initiative in the Sun Software Division.

How Sun ONE Grid Engine standard works

Sun Grid Engine provides the user with the means to find a suitable host for the user's job without an exact knowledge of the cluster's equipment and its utilization policies. All the user has to do is to specify the job requirements and let Sun Grid Engine manage the task of finding a suitable and lightly loaded host.



The diagram shows the logical flow of jobs within a Sun Grid Engine cluster grid. The software constantly looks for slots at each node in the cluster grid with enough resources to run the job.

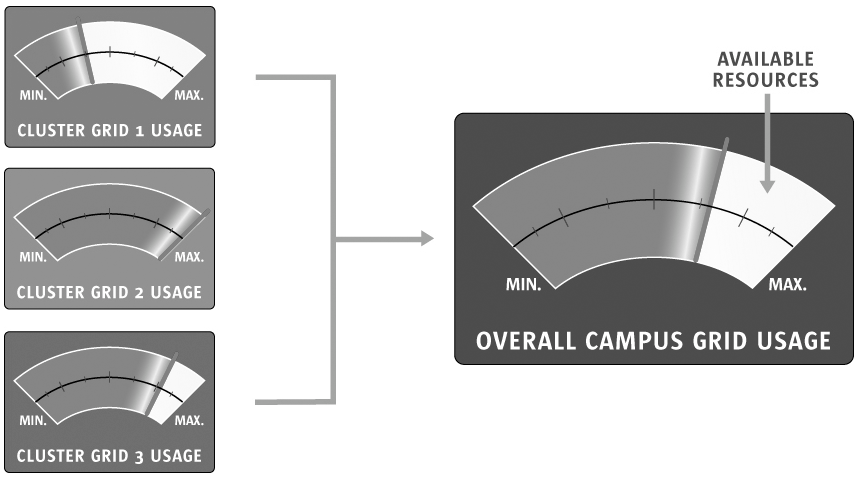
The utilization increases from 10-20% before Sun Grid Engine to up to 98% after Sun Grid Engine enters the production stage. The productivity jump is significant. A job that runs for 10 days at 10% utilization in a network without any grid software will run in one day in a cluster grid, leaving nine more days free to do other things.

Sun Grid Engine can be downloaded for free from
http://wwws.sun.com/software/gridware/sge.html

The documentation is available on line at:
http://www.sun.com/products-n-solutions/hardware/docs/Software/Sun_Grid_Engine/index.html?

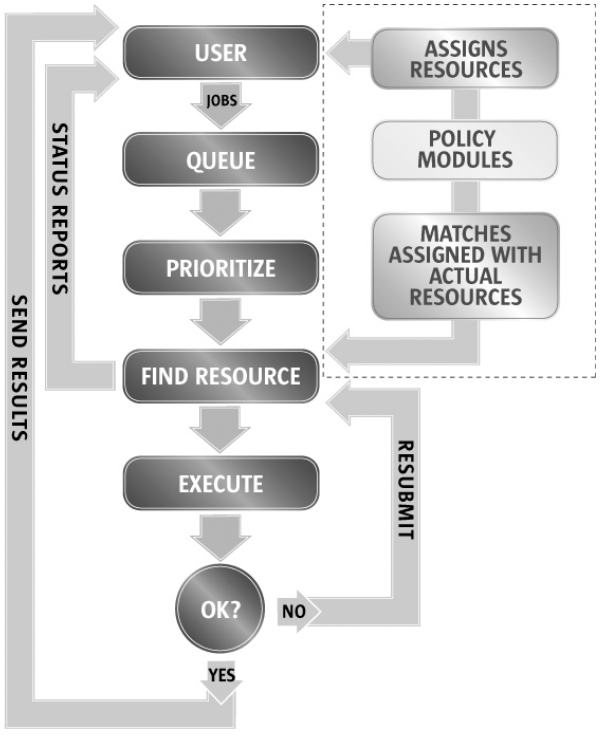
Platforms supported are Solaris 9, 8, 7, and 2.6 Operating Environments for SPARC, and Sun Linux and Linux (compatible to kernel 2.4) on x86.

How Sun Grid Engine Enterprise Edition Works



Let's assume we have three cluster grids used separately in three departments.

Department 2 reports lack of compute resources and requests more hardware. Department 3 orders more resources, wisely thinking that they do not want to wait until the last moment. These costly procurements can be easily avoided by consolidating all three clusters in one single Enterprise Cluster and still have resources for a while in the future.



The answer to the question “Why is grid cluster consolidation not done today?” is not a technical one. It is more of a psychological issue. Many users fear resources won’t be available when one needs them. The existence of policies, easy to implement and easy to understand, help address this concern. The product will not accepted without paying attention to human behavior.

Sun Grid Engine, Enterprise Edition has all the functionality of Sun Grid Engine, but in addition, it has a policy module as shown in the diagram:

Users, departments and projects are each assigned a percentage of the total compute resources available in the Enterprise Grid created by consolidating all departmental cluster grids. The software will maintain the assignment over a day, a week, a month or any other unit of time over which the assignments are budgeted.

How Enterprise Edition policies work

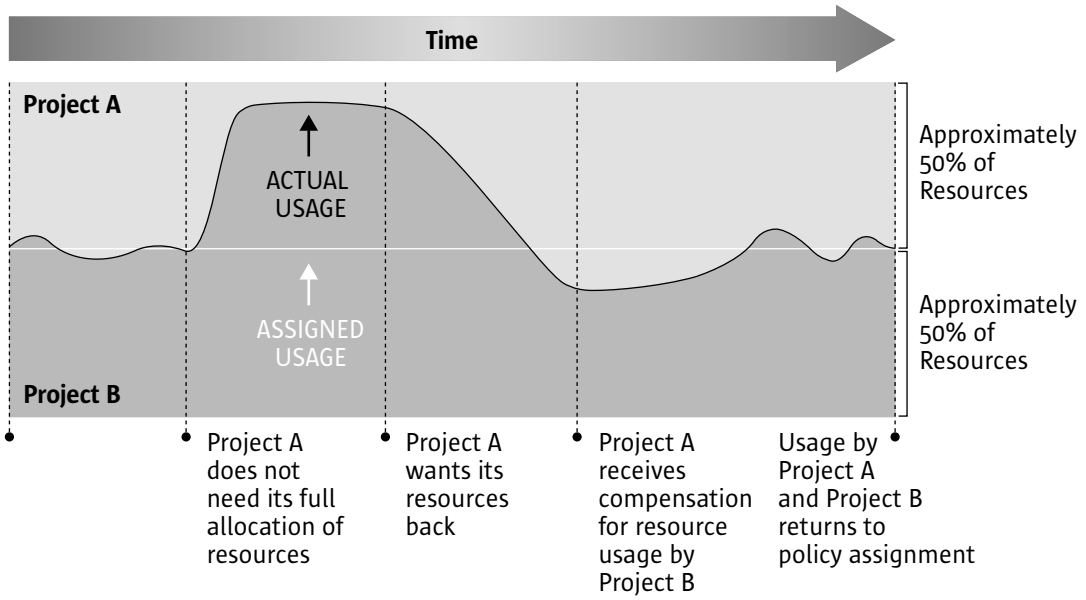
There are four types of policies in Sun Grid Engine Enterprise Edition.

- Share tree policies
- Functional policies
- Deadline policies
- Manual override

To illustrate how policies work, we select a very simple example of just two projects in an Enterprise Edition grid. In reality, we can have hundreds of projects simultaneously submitted to the grid.

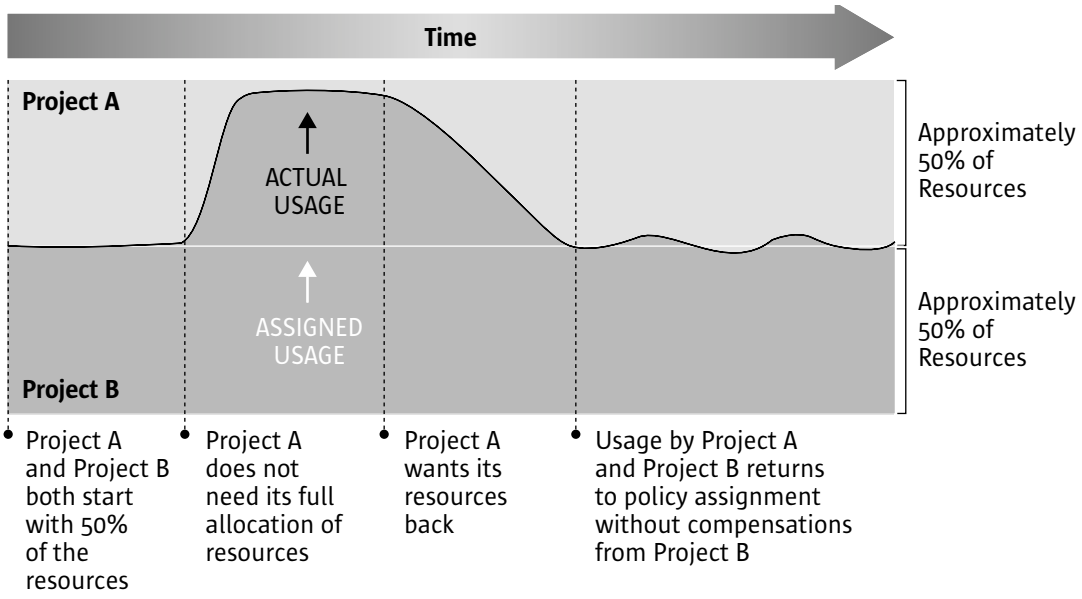
Each policy is illustrated in the diagrams below:

1. Share Tree Policy

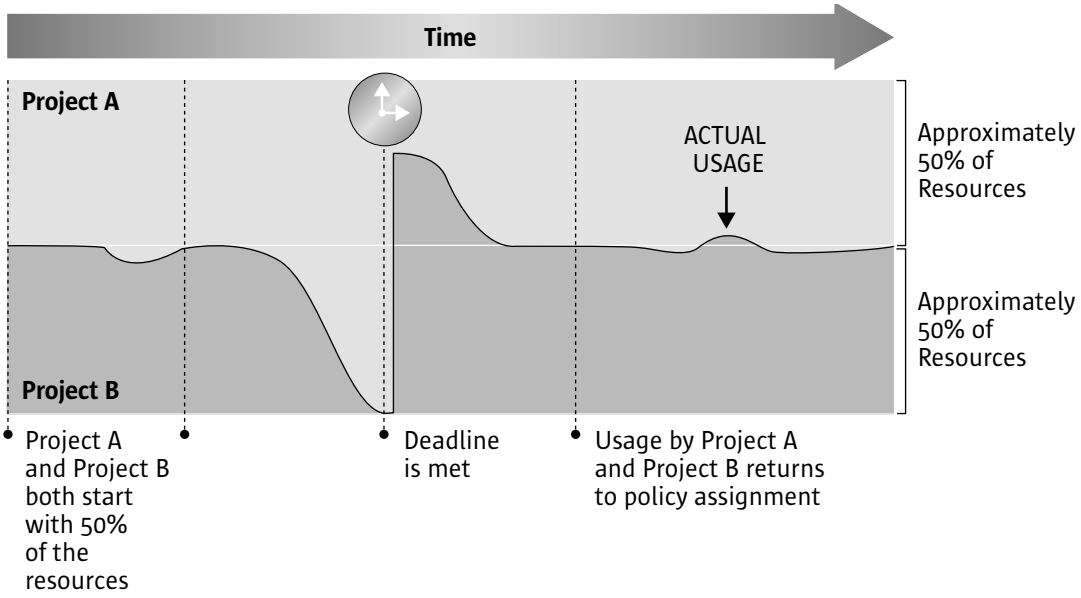


In a *share tree* policy, compute resources used above the assigned amount should be returned later. As we see in the diagram, Projects A and B are assigned 50% each. Project A is not active over a period of time. When it returns, Project A wants its resources back. Project B compensates, but not with an equal amount. Resources borrowed three weeks ago, are paid pack perhaps at 90% discount. Resources borrowed a day ago, may have only a 5% discount. The discount levels can be selected via parameters in the set up screens.

A functional policy is a particular case of a share tree policy, where the discounts applied to resources borrowed are 100%. Therefore, there is no awareness of the past usage. While in system, each Project A and B have 50% of the resources. If only Project B is active, will take 100% of the resources, and return to the assigned 50% when Project A is active again.



3. Deadline Policy



A deadline policy assures resources are available for a job to run before a certain deadline. Automatically, Enterprise Edition will increase the resources available to the job as shown in the diagram.

The resource assignment AFTER the deadline job executes depends upon the background policy. If the background policy is share tree, Project A will compensate Project B for the additional resources consumed to run the deadline job. If the background policy is functional, each project assumes the assigned compute usage values and there is no compensation.

A manual policy simply overrides any other policy.

User satisfaction in Enterprise Grids

		User A	User B
Entitlement	What I was told I would receive	50%	50%
Accumulated Allocation	What I actually received	47%*	52%*

The table above summarizes the outcome of using Enterprise Edition: each user or project received resources almost identical to the resources assigned and promised. All expectations are met. Economies of scale at enterprise level expand benefits from a single department to the entire enterprise.

Sun Grid Engine Enterprise Edition is a price listed product available on CD only. The documentation is on line at:
http://www.sun.com/products-n-solutions/
hardware/docs/Software/S1GEEE/index.html

Users of Sun Grid Engine can upgrade to Sun Grid Engine Enterprise Edition without any changes in the script.
To request a 30 day trial, use the following URL:
http://wwws.sun.com/software/gridware/sgeee_get.html

Sun ONE Grid Engine Information Stack

- **Sun Grid Strategy**
  - http://www.sun.com/executives/realitycheck/reality-081502.html
  - http://wwws.sun.com/software/grid/SunGridComputingBrief.pdf
  - http://wwws.sun.com/software/grid/Grid-brochure.pdf
- **Sun ONE Grid Engine**
  - http://wwws.sun.com/software/gridware/sge.html
  - http://www.sun.com/products-nsolutions/hardware/docs/Software/Sun_Grid_Engine/index.html
- **Sun ONE Grid Engine Web Training Course**
  - http://suned.sun.com/US/catalog/courses/WE-1600-90.html
- **Sun Cluster Grid, Blueprints**
  - Part 1: http://www.sun.com/solutions/blueprints/0802/816-7444-10.pdf
  - Part 2: http://www.sun.com/solutions/blueprints/0902/816-7765-10.pdf
- **Grid Engine Open Source project**
  - http://gridengine.sunsource.net/

3.1.3 Sun ONE Identity Server

Identity management is the most crucial aspect of an authentication framework. The ability to define and deploy an architecture that meets the needs of a dynamic community is paramount to the success of any distributed system or grid. Sun is a member of the "Liberty Alliance." Project Liberty is a framework that is being designed to allow for cross-agency (corporations, banking, universities, governments, or anyone) federated authentication and authorization. The ability to authenticate and authorize are the two most critical elements in processing the validity of any transaction. You can define transaction as being any data request, or host access within a Web Services infrastructure.

The Sun ONE Identity Server is the first step in the development and deployment of a Federated (i.e. Liberty-enabled) architecture. It is designed to store and manage identity profiles, access privileges and application and network resource information.

You can consolidate and aggregate identity information from disparate sources as student administrative, human resource applications, network operating systems, messaging systems, or telephony systems, to build a single unified view stored within the directory. With the proper access control entries, and access control lists, the Sun ONE Identity Server provides strong enterprise-wide security for authentication of employees, customers, students, faculty, and partners. Included within the server is the ability to issue, renew, suspend, revoke, and manage X509-based digital certificates. All built on a standards-based, distributed directory that offers a common storage space for the data.

The server provides fine-grained access control to Web and non-Web based resources for secure delivery of data, currently supporting the Liberty Alliance federated identity and SAML Web Services security standards. See http://www.projectliberty.org

3.1.4 Sun Portal Server and Sun Technical Compute Portal

Sun ONE Portal Server and Sun Grid Engine Portal

The Sun Grid Engine Portal (GEP) is an integrated solution for high performance and technical computing (HPTC) environments that offers a new class of Web services to organizations. Based on the Sun ONE Portal Server and the Sun Grid Engine resource management software, the Sun GEP solution gives organizations a single, unified interface to technical applications. This advanced solution brings the operating advantages of portal technology to HPTC environments by combining portal capabilities, resource management software, and easy to use tools. As a result, the Sun GEP solution enables organizations to centralize application management and compute resources, as well as provide highly secure communication and applications access to a larger and more diverse set of users—without increasing time, cost, risk or effort.

The GEP software integration package is available to the open source community and can be downloaded from http://gridengine.sunsource.net/project/gridengine/gep/GEP_Intro.html

Sun ONE Portal Server

The Sun ONE Portal Server is the industry's first fully integrated, secure portal and identity management solution. Building upon the identity management capabilities of the Sun ONE Identity Server, the Sun ONE Portal Server enables organizations to protect information assets while building secure relationships and communities with and among customers, partners, suppliers, and employees.

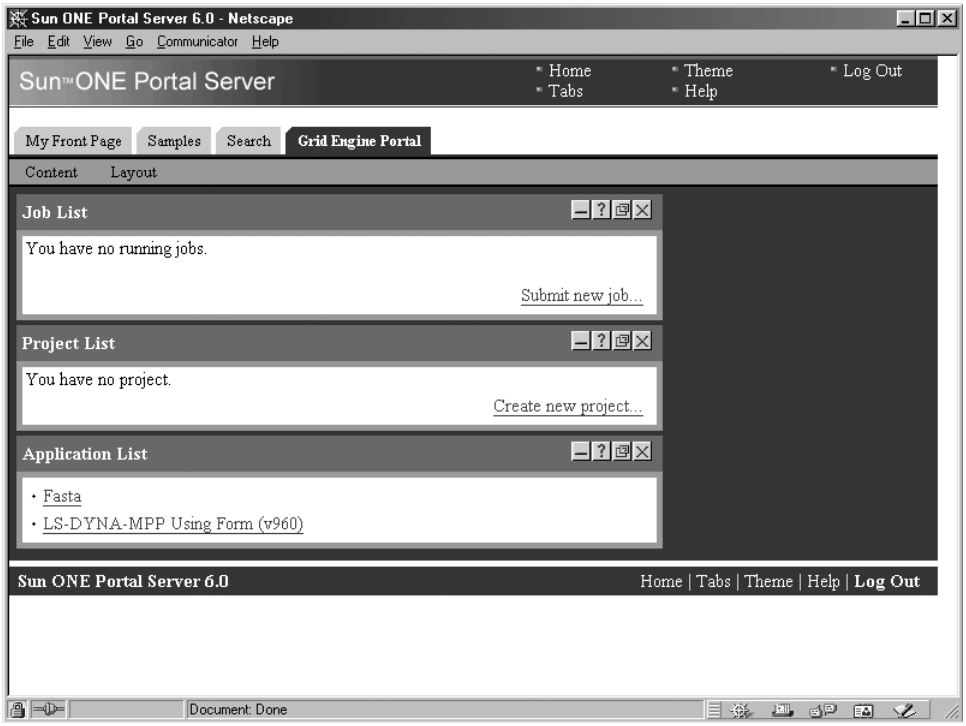
A community-based server application, the Sun ONE Portal Server aggregates key content, applications, and services that are personalized based on user role or identity, user preferences, and system determined relevancy. Providing an Internet/intranet services deployment platform, the Sun ONE Portal Server enables organizations to easily deploy technically demanding, secure portals and give users access to computing resources through a familiar Web interface.

Several key features contribute to the Sun ONE Portal Server's ability to create and manage productive, secure, and cost-effective HPTC portals:

- Integrated portal services and identity management, giving organizations the ability to manage and administer users and policies, and to provision services.
- Single sign-on, giving organizations the flexibility to continue to add services, including Web Services, while making access more convenient for users.
- Enhanced Security, providing authentication and authorization services and enforcing fine-grained policy rules.
- Secure search and delivery, providing documents, files, and resources based on the user's role, ensuring it only delivers results for documents that the user has permission to access.
- Delegated administration, enabling portal administration to be delegated to other users or lines of business within or even outside the organization.
- Single administrative console, providing a central location for administering and managing the portal, search, identity, and directory services, thereby reducing administrative costs.

Sun Grid Engine Portal—Making HPTC Resources Accessible

The Sun ONE Portal allows information, applications, and services to be displayed on a single browser page. The Sun GEP Integration Tools are a series of channels—appearing as menus or windows on the browser desktop. See the example below.



A sample page demonstrating channels that give users dynamic updates on current running jobs as well as access to historical project data, perhaps from previous application runs.

A powerful tool for securely accessing aggregated computing resources over the Internet or intranet, the GEP offers tremendous benefits to organizations running HPTC applications:

- **Easy access, anytime, anywhere:** Jobs can be submitted locally or remotely, and HPTC resources can be made available to users without duplicating resource investments.
- **Improved Resource Management:** The Sun Grid Engine software takes advantage of idle compute resources network-wide and matches them to individual job requirements. The result is more processing power available per user that results in increased productivity and greater return on investments.
- **Superior ease-of-use:** The Sun GEP makes job submission and monitoring easy. With the Sun GEP, users can manage job submissions by filling out simple Web based forms—no complex scripts or UNIX® skills are necessary. Users are presented with a form that prompts for all required inputs. Furthermore, users can check job status dynamically, receive e-mail notification when jobs are complete, download and visualize output files remotely, and share results with ease.
- **Easy access to legacy applications:** Administrators can quickly add legacy applications to the portal for sharing and remote use—often within minutes. No application modification is required for batch applications that use a command line interface. In essence, a legacy application can be transformed into an Internet- or intranet- available application simply by plugging it into the Sun GEP.
- **Integratable Solutions:** The Sun GEP solution enables organizations to create solutions based on open standards and technologies, ensuring operability across heterogeneous platforms, systems, and environments.
- **Built on proven products:** Unlike Grid Computing portals that are engineered specifically for niche markets, the Sun GEP is built on robust, high performance, commercially available Sun products. As a result, the Sun GEP provides a solution that can scale horizontally and vertically to provide maximum compute power and optimum data flow to any organization running HPTC applications.

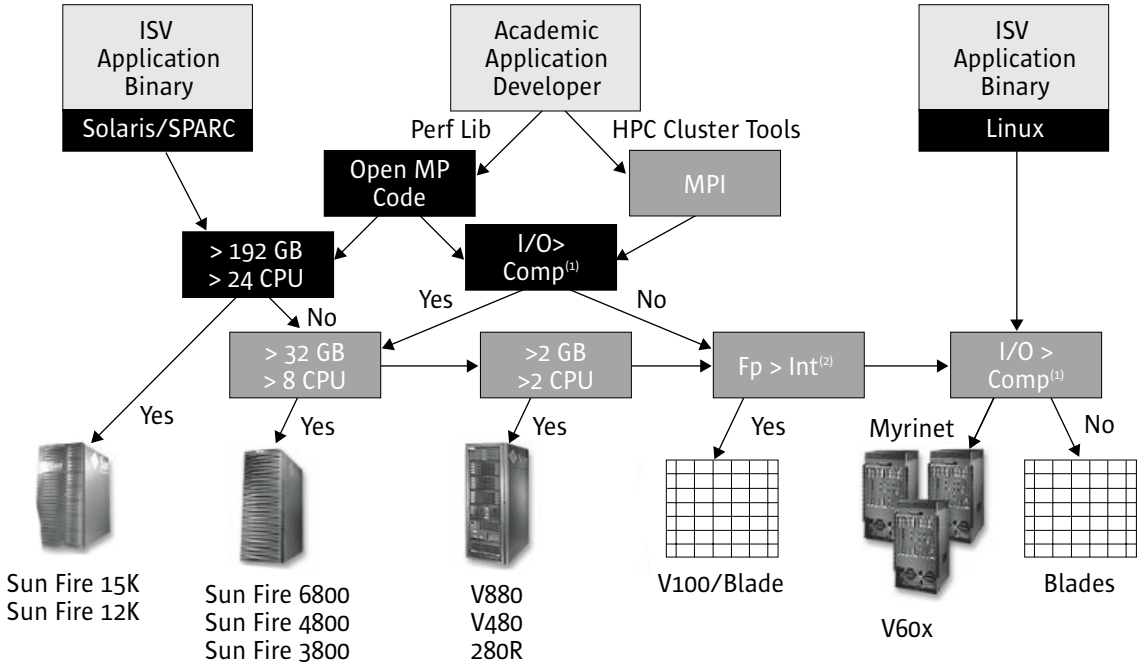
3.2 Platform Systems

Along side the systems vision that will drive innovation and connect collaborating researchers throughout the world, Sun delivers highly scalable systems from the commodity cluster range to supercomputer class systems.

3.2.1 From large SMP to Blades

In order to achieve the highest possible level of efficiency, the decision on which server infrastructure to execute a code should be based on application requirements, not on management or access criteria. The server platform offering the best performance, availability and total cost of ownership (TCO) for a given application or application mix at a given price point is the right server. Technologically, the main questions are about the requirements for linear addressable memory and the communication profile of an application. If 32-bit addressing is sufficient, IA32 most likely will be the right platform—otherwise SPARC-based Sun systems offer a very versatile spectrum of 64-bit systems.

The other important decision criteria is the ratio of I/O operations compared with pure compute time. Many applications like large databases, fluid dynamics simulations or genome-to-genome comparisons can benefit dramatically if most of the data for computation can be kept in main memory and thus waiting time for input/output is reduced. These types of application perform well on Sun's spectrum of SMP servers. If the time spent on communication between threads is insignificant or non-existent (as in image rendering or text search operations), a cluster of small nodes can provide a very cost effective environment. The good news is that you can trust Sun to help you architect the environment that fits your needs—from the largest SMP system on the market to the building blocks for Beowulf clusters—Sun offers a spectrum unique in the industry. The entire 64-bit space is covered by binary compatible SPARC servers, and for 32-bit applications Sun offers the LX50 and IA32 based Blade servers.



(1) Time spent for I/O compared to compute (2) Floating point stronger than integer

- For applications that do not require low latency and high bandwidth interconnects for their communication, the number of CPUs available to a given process is essential. The more CPUs that can be deployed on a given amount of space, the better. For these types of workloads, Sun has developed ultra dense blade servers. Blade servers achieve a density of 160 CPU's per standard 19" inch rack and are available both in 64-bit and 32-bit (IA32) versions.

  More information at: http://www.sun.com/servers/entry/blade/index.html

- Blade Servers come with built in Gigabit Ethernet capability. For some applications, namely MPI workloads (ref 3.2.2), the communication latency involved with Gigabit Ethernet slows down the execution performance. In these cases, Cluster interconnects like Myrinet are required (ref. 3.2.5) to build more tightly coupled clusters. For 32-bit applications, Sun offers IA32 platforms (http://www.sun.com/servers/entry/v60x).

  SPARC based 64-bit Servers (http://www.sun.com/servers/entry/v120) offer similar options for 64-bit loads. Both can be equipped with PCI compatible external interconnects. In addition to network expandability, 1RU servers offer more RAM capacity than Blade Servers, though they don't achieve the same density.

- The fastest communication between threads can be accomplished by shared memory systems with fast, low latency interconnects like the Sun Fire Plane in Sun SMP servers (ref. 3.2.4). Entry level SMP systems or clusters thereof in combination with scalable programming models (ref. 3.2.2.) allow the construction of very versatile, yet cost effective compute environments. Servers like the Sun Fire V880 (http://www.sun.com/servers/entry/880/index.html)

offer already 64 GB of memory to a single process with a minimum communication delay. Clustered ith PCI-based interconnects, large MPI jobs can utilize the very fast engine and scale up to 64 nodes (with 64 GB of memory each). This allows the deployment of a wide variety of applications, especially in environments with mixed workloads.

- Midrange and Midframe Servers (http://www.sun.com/servers/midrange) offer more of the same principles compared to smaller SMP systems. More in terms of compute capacity and memory per node—the SF6800 allows up to 24 UltraSPARC CPUs and 192 GB of memory. More in terms of interconnect—the Sun Fire 6800 offers the option to build clusters using the new and exciting ultra low latency, ultra high bandwidth interconnect Sun Fire Link (ref. 3.2.4).
- Sun's highend servers (http://www.sun.com/servers/highend) are industry leading in their class. The top of the line Sun Fire 15k server can operate up to 106 CPUs with up to 576 GB of memory in a single systems image. This is sufficient to load very large datasets into memory and compute them very efficiently. Only a minimum of time is lost for data access or communication. In order to improve the locality of data in relation to the CPU requesting the data for an operation, Solaris 9 offers a feature called MPO (ref. 3.2.3). Like the SF6800, the SF12k and SF15k can be clustered to provide a large memory environment for compute intensive applications in Physics, Chemistry, Engineering (such as the Cambridge High Performance Computing Facility in Cambridge, England) or for very large Data Analysis (as is found at Tokyo University).

3.2.2 Scalable Programming Models and Development Tools

Two primary high performance computing programming models are supported in the Sun high performance and technical computing environment:

- single-process model
- multiprocess model

The single-process model includes all types of multithreaded applications. These may be automatically parallelized by Sun's high performance compilers using parallelization directives (e.g., OpenMP) or explicitly parallelized with user-inserted Solaris or POSIX threads. Multithreading (MT) is a software technique that breaks program code into segments that can be executed in parallel on multiple processors, for overall faster application performance. Multithreaded applications enhance productivity by decreasing the time it takes to perform one job. Developers can use Sun ONE Studio compilers to assign multiple tasks in an application to independent threads of execution, with Solaris Operating Environment automatically assigning each thread to an available processor.

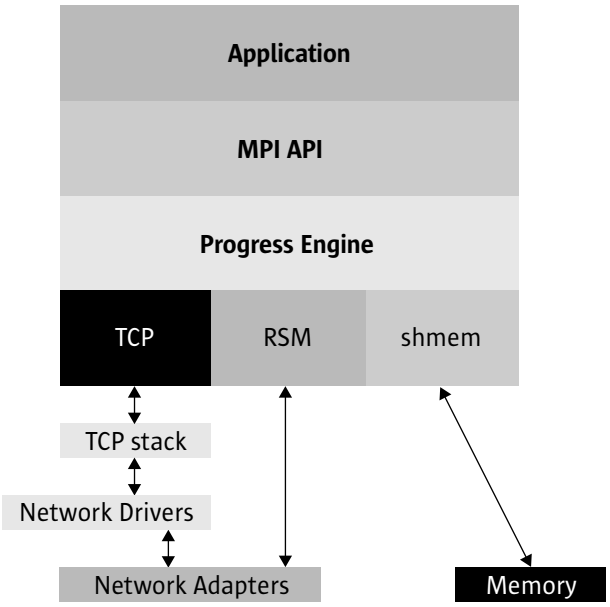
Multiprocessing is the execution of a program (or multiple programs) simultaneously on multiple processors. Multiprocess functionality must be built into the hardware, and supported by the operating environment. The multiprocess model supports the Message Passing Interface (MPI) industry standard for parallel applications that run both on single SMPs and on clusters of SMPs or thin nodes.

A third, hybrid model is also supported: the mixing of threads and MPI parallelism to create applications that use MPI for communication between cooperating processes and threads within each process. Such codes may make most efficient use of the capabilities of individual SMP nodes in the high performance cluster environment. The hybrid parallel programming paradigm can only be considered in a thread-safe framework.

Sun HPC ClusterTools™ Software is a complete software development environment for analyzing, debugging, and performance monitoring of parallel MPI applications on the Sun system solutions based on the UltraSPARC processor architecture under the Solaris Operating Environment. Sun HPC ClusterTools software includes Sun's native implementation of the MPI standard, Prism parallel debugger, Sun Cluster Runtime Environment, and Sun Scalable Scientific Subroutine Library. Sun MPI is fully thread-safe with locking pushed as low as possible within the implementation to allow for concurrency: multiple user threads may be active within the library at any given time. In addition, both 32- and 64-bit versions of the library are included with Sun HPC ClusterTools and Sun MPI programs may be debugged and tuned with the Prism parallel development environment. Sun MPI, which currently supports applications spanning up to 256 compute nodes and consuming up to 2048 processes, includes extensive optimizations for running both within and between SMP nodes.

The architecture of the Sun MPI library is shown the following figure. The lowest level of the library, called the Protocol Module Layer, includes support for several communication mechanisms. To accommodate multiprocess communication across a wide variety of networks and interfaces, Sun HPC ClusterTools software has extracted the communications code into individual Loadable Protocol Modules that can be called by the main Sun MPI library. Each protocol module enables multiprocess communications across a network through a different communications protocol.

Sun HPC ClusterTools software supports communication between processes across TCP networks, shared memory interfaces (SHM) in an SMP system, and remote shared memory (RSM) over Sun Fire Link interconnect. Shared memory interfaces provides very low latency and high bandwidth availability within the SMP systems. The TCP protocol module allows Sun MPI jobs to run across any available TCP-capable interconnects. While this allows the use of numerous commodity interconnects offering a wide variety of interconnect bandwidths, it does not address the issue of low-latency communication that is of interest for many classes of high performance computing applications. RSM protocol provides direct and low-latency memory-to-memory communication between cluster nodes over RSM-capable interconnects. Applications demanding a low-latency communication framework will benefit from the direct remote memory access capability in a high performance cluster over RSM capable interconnects.



Myricom, Inc. has been developing a dynamically Loadable Protocol Module under the Sun Community Source License program for their latest Myrinet 2000 interconnect. Because these modules are loaded at run time by the standard Sun MPI library, users of such third-party modules will not need access to Sun HPC ClusterTools Software source code to make use of this capability.

When communication between two processes in a Sun MPI job is initiated, the library chooses the most efficient transport pathway between those two processes. For example, two processes co-located on a single SMP node will communicate via shared memory segments under control of the shared memory protocol module, whereas two processes on different nodes will use the protocol module corresponding to the best available network connection between those nodes. These connection decisions are made automatically on a pair wise basis at run time.
Protocol modules offer several advantages:

- New communications protocols can be developed and supported dynamically with the existing Sun HPC ClusterTools installations.
- No recompilation or re-linking is necessary for existing Sun MPI parallel applications to use the new and additional dynamic Loadable Protocol Modules.

3.2.3 Memory Placement Option

The Memory Placement Optimization (MPO) feature allows the Solaris Operating Environment to recognize the memory locality effects present in the Sun Fire servers, and intelligently place memory pages and processes throughout the SMP server in order to improve performance.

The Memory Placement Optimization feature in the Solaris 9 Operating Environment can improve application performance significantly, and users should keep that in mind when considering their adoption plans for Sun's latest operating environment. The MPO feature was designed especially to explore the benefits of the Sun Fire server architecture, and demonstrates how tight integration of operating environment and server architecture can create high performance and computational efficiency. The achievable performance improvements from MPO utilization are heavily dependent on workload characteristics, and on specific server configurations. High Performance Technical Computing workloads should in general present larger benefits than Business Computing workloads. In the same fashion, larger Sun Fire 12K and 15K server configurations should benefit more from MPO than Sun Fire 3800-6800 servers. As Sun introduces new technologies into its microprocessor and server lines, technology improvements like MPO should help users reach the highest performance levels on their applications.

The main benefits of using Memory Placement Optimization are lower memory latencies and increased bandwidths for memory-intensive applications, resulting in higher performance. The improvements that can be observed are highly dependent on the application's memory access patterns, and although MPO policies have been developed using a representative set of real workloads, the specific results of MPO introduction may vary significantly. Every effort was made to make sure that, even in the worst case, the default MPO settings would not decrease the performance of an application.

High Performance Technical Computing (HPTC) workloads have a heavy emphasis on scientific, floating-point-type calculations, and are very memory intensive in general. The tests have demonstrated that, in HPTC workloads, the effects of MPO are larger, and can vary more, than in the case of business workloads. The performance differences are naturally highly dependent on the workload.

3.2.4 Sun Fireplane System Interconnect

The UltraSPARC III-processor-based Sun Fire server family uses common technology and components to create efficient shared memory processing (SMP) systems at different sizes and capacities. The basic building blocks for these servers are system devices like processors, memory units and I/O controllers. The system devices are connected through the Sun Fireplane system interconnect, which has a crossbar switch design based on a point-to-point protocol. As the number of system devices increases, so does the sophistication and capacity of the Sun Fireplane interconnect implementation. The Sun Fireplane-based servers can be divided into four categories by the number of interconnect levels required:

1. **Small server**—Four system devices (two processors, one memory unit, and one I/O controller) require only one level of interconnect.
2. **Workgroup server**—18 system devices (eight processors, eight memory units, and two I/O controllers) require two levels of interconnect.
3. **Mid-size server**—56 system devices (24 processors, 24 memory units, and eight I/O controllers) require three levels of interconnect.
4. **Large server**—180 system devices (72 processors, 72 memory units, and 36 I/O controllers, or 106 processors, 72 memory units and 2 I/O controllers) require four levels of interconnect.

The Sun Fire servers are constructed by placing system devices into physical components, like boards or assemblies. Processors and memory devices share a single board, called a Uniboard in the case of the Sun Fire 3800-6800 (mid-size) servers and Sun Fire 12K and 15K (large) servers. A Uniboard contains up to 4 UltraSPARC III processors, and up to 32 GB of memory distributed into 16 logical banks. A unique feature of the UltraSPARC III processor and Sun Fireplane interconnect design is that the memory control units are located inside the processors. Each processor controls 4 logical banks.

The Sun Fire 15K, with its fast UltraSPARC-III processors, Fireplane interconnect, Solaris multithreaded operating environment, a simple shared-memory SMP programming model, robust software development and administrative tools, and RAS features combine to deliver new levels of usable performance for high performance computing environments. The exceptionally high bandwidth of the Fireplane interconnect design results in highly scalable systems, while preserving the low latency expected for SMP based systems.

3.2.5 Sun Fire Link

For some applications, it is critical to make a large memory space available in order to solve problems in an efficient way. Dr. Ron Horgan from Cambridge University mentions, for example, protein misfolding disorders which requires simulation on an atomic level. The fibril structure calculation for 1000 atoms requires 400 GB of memory as the required memory scales as square of number of atoms. There are numerous similar demanding applications in fluid dynamics or high energy physics.

While 64-bit processing allows addressing of large memory spaces, the performance of the application is heavily impacted by the physical layout and access to memory. The more closely aligned the memory access, the better the application performance. Sun focuses on large SMP systems, which can be clustered to provide the best possible overall application performance to memory bound applications.

The Sun Fire Link product is Sun's highest performing cluster interconnect. It is available on the Sun Fire 6800 and Sun Fire 12K/15K servers and is supported by Sun Cluster and Sun HPC ClusterTools. Sun Fire Link therefore serves both our commercial cluster market and our high performance technical computing market. The high data rate and low latency of Sun Fire Link should improve the performance of cluster applications.

Sun Fire Link can also be used as a high performance pipe between two Sun Fire servers for file transfers using TCP/IP.

Clustering of servers in general requires three components

- A cluster-aware operating system, such as the Solaris Operating System.
- Clustering software as the interface between Solaris and our customers' applications. Customers have a choice here depending on the nature of the application: Sun Cluster or HPC ClusterTools.
- An interconnect between the servers for heartbeat messages, data and application coordination information.

Sun offers a choice of cluster interconnects at various performance levels and price points as shown in the following table. The goal is to match the performance of the interconnect to the requirements of the customer's application.

Type	Usage	Max. # nodes	Data rate per link (HW)	Latency (software) Microseconds	Price
Sun Fire Link	Sun Cluster & HPC	8	1200 MBps	<4	$$$
SCI	Sun Cluster	4	200 MBps	<10	$$
Myrinet (3rd party)	HPC	16	140 MBps	16	$
Gigabit Ethernet	Sun Cluster & HPC	Large	100 MBps	100	$

3.3 Management

3.3.1 N1

Overview of N1

N1 is Sun's vision, architecture, and products for the next-generation data center. N1 aggregates widely distributed computing resources (servers, storage, software and networking) and enables them to operate as a single, powerful entity. By automating away the complexity associated with managing technology, business and IT managers will be able to dramatically optimize the utilization, efficiency and agility of their data centers.

N1 is designed to reduce management complexity and cost, increase data center resource utilization, improve infrastructure responsiveness and agility, and ensure investment protection:

- Makes data centers work like systems
- Unifies heterogeneous resources into "pools"
- Up-levels operation to business service
- Enables policy-driven services and utility computing

With N1, IT management and administration functions begin to operate in a shared real-time mode. In the past, an application had a dedicated server and set of storage assigned to it. Because applications aren't designed to share resources, nor can they predict what the user load will be at different times, each instance of an application needed to have its own excess capacity to handle peak usage loads.

N1 allows data center resource capacity to be shared by any number of services. So if one service requires additional capacity, while at the same time another requires less, N1 handles it. It's automatic.

N1 also enables administrators to manage a much larger number of systems because they manage the service, not the underlying resources. With N1, system administrators are now free to focus on maintaining service-level quality and implementing new, competitive features. All of this adds to the bottom line via increased business agility, reduced complexity, streamlined management, increased resources, and lower costs.

N1 Architecture

The N1 system architecture redefines the meaning of a system by adopting a perspective that is broader than what has previously defined a computing system. The traditional view is a collection of compute, memory, network, and storage resources in a single cabinet, all working together to support a single application. Even today, IT architects recognize that the system is not just what we know as a single computer, but the entire set of components that work together to deliver a network service. This perspective encompasses the multiple computers, storage systems, and network hardware that are used to deliver services with the scalability, availability, and security that businesses require. While the definition of a system is changing, we have been missing the tools to operate and manage it as a single entity.

N1 takes a broader definition that "The Network is the Computer™," viewing the entire set of computing, storage, and networking components as a pool of resources that can be operated and managed as one. It integrates and abstracts all of the resources needed to provide high-level services that are closely aligned with a company's business objectives.

In the early days of computing, scheduling programs to run on specific CPUs with pre-assigned resource allocation was a manual task performed by system operators. Today, modern operating environments use the abstractions of process scheduling, virtual memory, file systems, and other elements to provide easy-to-use program interfaces by virtualizing the resources and automating many of the tasks that were once manual. This is an example of using abstraction to simplify the interaction between systems and the people that run them.

Raising the Level of Abstraction

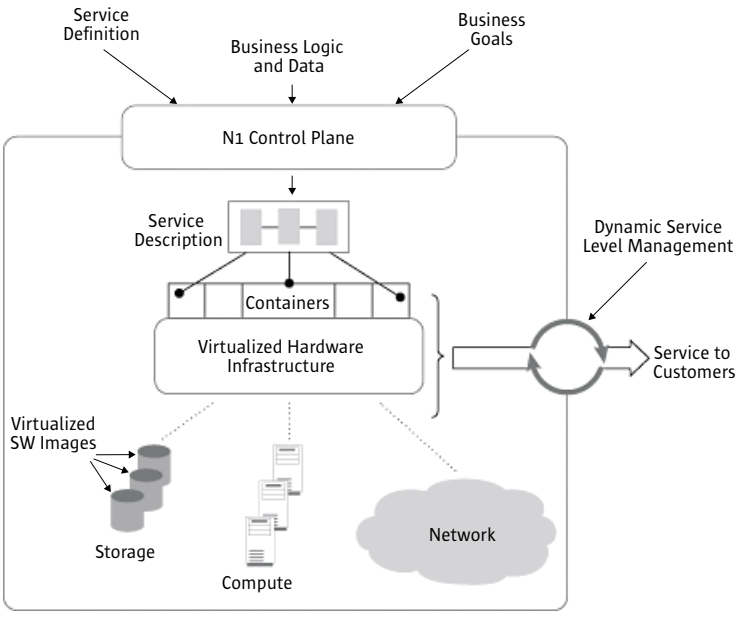
Over the years, computing infrastructures have become even more complex. The N1 fabric operating environment creates a higher level of abstraction that enables administrators to manage not just programs, but the business services that are deployed across a network. The fabric operating environment takes service-level objectives and business goals and assists IT architects in designing a service description that can meet service-level objectives. The service description can be very detailed, specifying all of the required service containers and network topology. Or the details can be left undefined, allowing N1 to make its own recommendation. In the future, the translation of service requirements to resource allocation will be automated, facilitating an even more rapid response to business needs.

The fabric operating environment requires IT architects to provide the following information:

- A service that fulfills a business need and has service-level objectives expressed in business terms
- Business logic and data that are written in various programming paradigms, including the Sun Open Net Environment (Sun ONE), Microsoft.NET, and POSIX
- Business goals that specify the relative priority and importance of the various services

With the business goals carefully defined, the fabric operating environment can use several key abstractions to implement a solution, including:

- **Service descriptions**—Each service is typically implemented using a multitier, possibly multisite infrastructure that roughly corresponds to the application network maps that IT architects produce today.
- **Containers**—Business logic is hosted in containers such as an application server or an operating environment, and are hosted on hardware containers such as servers, dynamic system domains, and server appliances. The fabric operating environment exerts fine-grained resource control over each container so that the multiple containers running on a single platform will all meet their service-level objectives.
- **Virtualized hardware infrastructure**—This abstraction specifies the pool of computing resources connected through networks, including computing platforms, dynamic system domains, network infrastructure elements, and storage systems.
- **Virtualized software images**—These take the form of a software image repository that includes the business logic and container code. The images can be off-the-shelf or customized software that supports business objectives.
- **Control plane**—This is the mechanism that translates business objectives into N1 objects, and is a central function of the fabric operating environment. See figure below.



N1 Roadmap

Sun is delivering N1 through a three-phase roadmap:

- **Phase 1: Virtualization**
  The process of turning data center resources into a pool that can be tapped by a number of business services.
- **Phase 2: Provisioning**
  Automation of software and hardware installation and configuration for new business service development, for example, mapping services on to the pools of resources created during Virtualization phase.
- **Phase 3: Policy-based Automation**
  The ability to dynamically manage the allocation of IT resources based on pre-defined business policy, e.g. allows you to write a policy statement such as "the billing service gets priority the last week of the month" and have that rolled out through both the provisioning and the virtualization engines.

The first N1 product is the N1 Provisioning Server 3.0 Blades Edition:
The N1 Provisioning Server 3.0 Blades Edition is the first blade virtualization solution in the industry—it can reprovision blades from one server farm to another, add new blades, shelves, and even racks into the data center environment, and bring new blades online to replace failed blades from its unique graphical user interface. Solaris, Linux and Windows operating environments are all supported by Sun N1 Provisioning Server technology.

Key Highlights

- Automates the configuration and deployment of servers, firewalls, load balancers, and network resources
- Virtualizes resources to manage the blade server platform as a system
- Delivers multitenancy N1 data center capabilities to the blade server platform
- Offers an entry point to N1 that can grow to manage the entire data center
- Provides an internal event database for integration with billing, inventory management, and service-level agreement (SLA) enforcement
- Designed to enable the deployment of an entire server farm in under an hour

Where to get more information

N1 in general—http://www.sun.com/n1

Provisioning Server—http://www.sun.com

3.3.2 Sun High Performance Management Tools

While high performance and technical computing has become more similar to mainstream computing, it also shares some of the same systems management challenges. HPTC systems are expected to deploy easily, to maintain certain levels of service, and to be managed cost-effectively by means of automation.

Sun's system management strategy coordinates Sun's technology, marketing, support, and ISV partnering efforts to produce a comprehensive, integrated set of products available for enterprise network and system management. Sun's strategy is based on providing two levels of functionality to meet the management needs of all Sun customers, whether they have small or very large installations.

In the single server and workgroup space, the focus is to provide ease of installation and management of a small number of systems "out of the box". Some of the system management tools are already co-packaged with the Solaris Operating Environment. Tools like Sun Management Center help the system administrator to accomplish day-to-day server operations via an easy to use graphical user interface.

The complexity, size, or mission-critical nature of networked systems creates additional requirements for system management tools. System management grows into enterprise-wide management—organizations must manage all of their computing and communications resources. Management of these resources is vital to carry out everyday business activities and stay competitive.

Systems management technology requires continuous innovation. To this end, Sun Microsystems, a leader in designing products for increased applications and data availability, has created the Sun Management Center (MC) systems management framework.

Sun MC software provides a single point of management for the enterprise Sun systems, which have long been the preference for both technical and commercial computing users, capitalize on Sun's strengths in reliability, availability, and serviceability (RAS), as well as performance and scalability. Sun MC software gives administrators one of the most comprehensive tools for managing Sun systems, as well as a platform that integrates easily with other enterprise management frameworks.

Sun Management Center 3.0 software is Sun's most advanced systems management tool. Its three-tiered framework offers a single point of management for all Sun systems, the Solaris Operating Environment, applications, and services for data center and highly distributed computing environments. With Sun MC software, IT organizations can efficiently manage and arbitrate between users, applications, and resources.

Sun MC software is an advanced system management tool, designed to support Sun systems. It provides a platform for the enterprise's administrative and management operations to help ensure all systems and the services they provide are highly available. A powerful tool for managing the enterprise network, Sun MC software enables system administrators to perform remote system management, monitor performance, and isolate hardware and software faults for hundreds of Sun systems, all through an easy-to-use Web interface.

To extend the capabilities of the Sun MC product, Sun also offers the Sun MC Developer Environment, which allows new management modules to be easily created and applications to be easily integrated with the Sun MC console.

The features of Sun Management Center 3.0 focus on helping IT professionals improve service levels and decrease costs by using scalable, adaptable, tightly integrated management tools:

- Web-based interface provides access to management information using secure SSL communication.
- Enhanced, proactive event and alarm management, and tight integration with Sun Remote Services, provide early notification of potential service problems.
- Enterprise-ready functionality, such as grouping and group operations, command-line interface utilities, and module configuration propagation, enable IT staff to manage above the level of the individual systems.

Sun Management Center provides a broad range of functionality for comprehensive systems management:

Reliability, availability, and serviceability (RAS) features:

- Proactive and automated management of complex and common administrative tasks reduces the likelihood of costly errors and helps assure availability.
- Dynamic reconfiguration and multipathing enhance higher system availability.
- Predictive failure analysis enables administrators to predict potential memory and disk hardware failures on a statistical basis, enhancing decision making and increasing availability.
- Graphical user interface (GUI) uses Java technology to provide heterogeneous GUI support, a common look and feel for all Sun Management Center applications, and the flexibility to manage the enterprise from any platform, increasing administrator efficiency.

System administration features:

- Automated management of common administrative tasks, easing the burden of everyday administration
- Configuration management controls, enabling administrators to remotely monitor or perform dynamic reconfiguration tasks, reducing reboots and the amount of costly downtime
- Health monitoring, a sophisticated set of heuristics that incorporates a large body of administrative knowledge, includes an intelligent rules-based health monitor that correlates metrics and gives suggested steps for resolution, simplifying administration.
- Log-file scanning enables administrators to search and parse logs, and register for a particular status, the foundation of application health monitoring.
- Physical views, displaying photo-realistic images of hardware components and relevant information such as network interface types, disk types, processor module speeds, etc. Components with associated events are highlighted, allowing administrators unfamiliar with a particular Sun platform to quickly determine which components need to be replaced.
- Logical viewer, showing a tree hierarchy of the managed host or domain, including all hardware and operating system components. So if an event is associated with a particular component, the logical viewer can identify its exact location within the hierarchy.
- Graph views, displaying CPU, memory, disk, and network performance metrics
- Per-process display that extracts specific information on process resource usage and behavior, allowing an administrator to monitor the load an application imposes on the system.
- Application monitoring, allowing administrators to check the health and performance of application processes, to examine and parse log files for recurring problems and particular status messages, and to monitor application files and directory statistics
- Real-time performance analysis enables administrators to isolate potential and existing bottlenecks.
- Event and alarm management, using complex rules and alarm limits to automate diagnosis, provide ongoing status information, and immediately notify administrators of significant events via the console, email, pagers, etc.
- Domain-aware agents, allowing dynamic system domains on high-end Sun servers to be monitored independently.

Integration features

- Single event model enables information to be easily shared with multiple consoles or users.
- Standard interfaces and protocols enable integration with third-party management tools, including Tivoli TME 10 TEC 3.6, HP OpenView ITOps, BMC Patrol, and Computer Associates Unicenter TNG, providing a complete enterprise management solution.
- Full SNMP (versions 1, 2,c and V2usec) and RMI connectivity enable information to be shared with other enterprise management tools.

Scalability and customization features

- Common management platform is scalable from a single system to thousands of server and desktop systems.
- Configuration flexibility enables Sun Management Center technology to fit the needs of the environment, making it easy to add customized rules, scripts, actions, and more.
- Extensible agent architecture enables administrators to add functionality and management features with ease.
- Developer environment enables organizations to plan, design, develop, and integrate Sun Management Center modules as well as third-party applications, tools, and customized solutions based on the Sun Management Center framework.
- Rules writing documentation enables rules to be created and customized for the Sun Management Center environment.

Security features

- Implements enterprise-wide security measures such as authentication, data integrity, and access control lists for data management and active functions.

Ease-of-use features

- Single point of management enables effective use of administrative resources, Dynamic agent modules enable functionality to be added or removed asynchronously and dynamically.
- Domain-aware agents independently monitor dynamic system domains on the Sun Enterprise™ 10000 server and its associated resources.
- Multiple system support enables administrators to remotely monitor and manage all systems running the Solaris Operating Environment.
- Logical element grouping of Sun systems
- Topology views, showing a hierarchical topological map of the objects being managed, quickly familiarizing administrators with the Sun elements in the environment. A “discovery” process automatically builds the topology view, which can be divided into several administrative domains for distributed management.

3.3.3 Managing Heterogeneous Environments

Sun’s strategy includes the core set of management applications and continued partnering with leading ISVs. Key ISVs such as Computer Associates and Tivoli, combined with Sun Professional Services and the leading systems integrators, make it possible to deliver the functionality required to meet even the most demanding needs. This strategy allows customers to have the functionality of traditional operations and data center management applications merged with UNIX network and distributed resource management applications.

One area in which Sun’s innovative technology is changing the nature of system management is through Java based solutions. The industry is moving fast to redo the management models that leverage and capitalize on the web, and Sun ONE Studio is the first and best way to develop them.

Sun ONE Studio addresses the problem that has frustrated network managers for years—application integration. Sun ONE Studio tools allow the creation of seamless, integrated enterprise management applications with consistent behavior and look-and-feel everywhere they run. Sun ONE Studio is a development environment that enables both software vendors and internal IT applications developers to create applications that will run across different hardware platforms and operating systems. All that is required is for Java to be able to run on the customer’s operating system of choice—Solaris, Linux, Windows, MacOS, or any others that have Java support baked in. The Java approach is endorsed by more than 40 vendors, including Bay Networks, Cisco, Computer Associates, Lotus, Platinum, Tivoli and many others.

Chapter 4

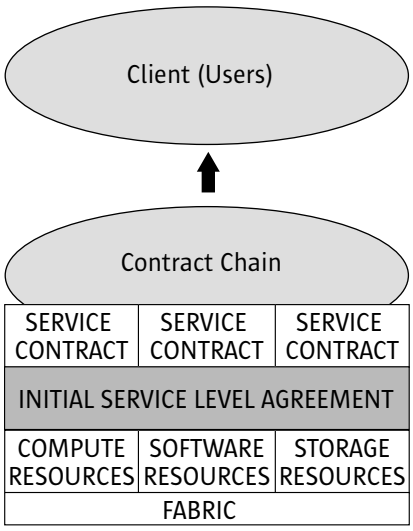
4 Case Studies

4.1 The London e-Science Centre at Imperial College London

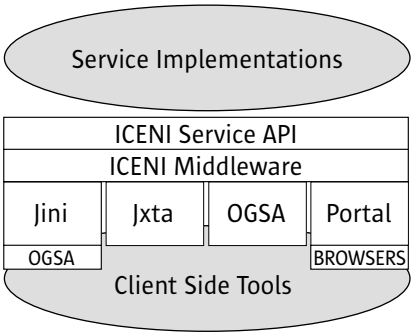
The London e-Science Centre (LeSC) at Imperial College London is a Sun Centre of Excellence in e-Science and one of eight regional e-science centres formed under the UK e-Science Core Programme funded by the Engineering and Physical Sciences Research Council and the Department of Trade and Industry. The Centre is involved in the development of Grid middleware through its own work with ICENI—the Imperial College e-Science Networked Infrastructure—and they are also deploying Grid infrastructures over their computing resources as part of the UK e-Science Grid.

4.1.1 ICENI—Imperial College e-Science Networked Infrastructure

ICENI is an integrated next-generation grid middleware that is being developed by LeSC through our involvement in several applied e-science projects. The focus is the capture and effective exploitation of meta-data to meet the needs of the applied e-scientist. Critical information is the capability of a resource and how it is exposed as a service for use by a community, and the behavior of an application defined by the workflow within and between its composed component services. This information may be coupled with additional requirements, such as minimum execution time, to inform the effective distribution of an application across the available resources. The meta-data is reused at multiple levels within the middleware.



The service oriented architectures within ICENI are built upon an implementation neutral interface. This allows the meta-data relating to the resource—its interface and behavior—to be reused across different service oriented architectures. A 'contractual' abstraction is used to describe who can use which methods on the service interface and at what time, potentially allowing users to sub-contract resources by delegation to third parties trusted by the user but not necessarily by the service provider.



Currently, ICENI services implemented with Jini™ are exposed through a gateway into other infrastructures such as the Open Grid Services Architecture (OGSA). Currently implementations of the ICENI discovery and service interfaces are being prototyped with Jxta and 'pure' implementations of ICENI services with OGSA. By developing our own service abstraction we are able to implement our service interface and meta-data structures across multiple infrastructures to provide a transport neutral service oriented architecture.

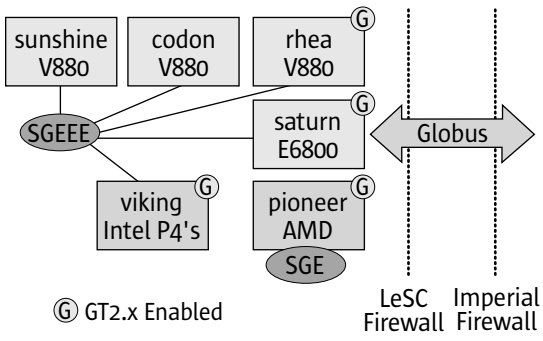
ICENI is being used to explore and define the meta-data needed by grid schedulers from the service infrastructure, the application and the user. This meta-data relates to service behavior and interface, the availability of a service to an individual or a community, the workflow within and between services in the application or the user's job, and whether the user is concerned with minimum execution time, or if the service is being charged for cost.

4.1.2 Imperial College Parallel Computing Centre

The Imperial College Parallel Computing Centre (ICPC) supports the London e-Science Centre in its software development and applied e-science projects by providing access to high performance computing resources. These resources are managed on behalf of specific research groups and the wider College community and currently comprise a Sun Enterprise 6800, several Sun FireV880's and large Linux clusters based around the Intel Xeon and AMD processors.

Scheduling across these diverse compute resources is managed through a single instance of Sun Grid Engine Enterprise Edition (currently 5.3p3) running on the E6800. This maintains a fair-share usage policy across the users on the Intel Xeon Linux cluster and the other Solaris systems. A separate Sun Grid Engine Standard Edition manages the AMD cluster.

The LeSC and ICPC are exploring several different mechanisms to interact with the SGEEE installation. The first of these is through the Globus Toolkit (v2.2.x) which is the currently the defacto standard Grid middleware within the UK e-Science programme. LeSC has developed and is providing to the worldwide Grid community an integration of SGE and SGEEE into GT2. The GT2 submission infrastructure is targeted at a broad range of scheduling infrastructures. An OGSI-compliant Grid Service that will allow the full capability of SGE and SGEEE to be available from remote programmatic clients is also being developed. Where remote programmatic interaction with SGE and SGEEE is inappropriate, the development of a generic e-Science Portal at Imperial College (EPIC) to host a range of services, including job submission through the Grid Engine Portal, is being considered.



4.1.3 Managing a Service Infrastructure

ICENI is being used to expose the local grid fabric to applied e-science collaborators as services to enable their own research. These applications range from high energy physics, bioinformatics, material modeling and climate prediction. By virtualising the resources as services, one is able to interact with these services through a standard interface. Extending the resource and service abstractions to include storage and software resources and allows a user to compose these abstractions to define the workflow between these services.

Future grid infrastructures are likely to consist of federated service oriented architectures comprising services of many different types with very diverse access policies. Different groups will have different views of this service infrastructure. A user will see the services that they have access to while a system manager will see the services over which they have administrative rights. As part of the ICENI middleware, an OGSA service browser has been developed that can be used to interact and visualize services within such an architecture.

Virtualising the resources enables the mechanisms needed to build an open economic infrastructure to enable the trading of Grid Services. A project funded by the UK e-Science Core programme—A Market for Computational Services—being led by LeSC, is now underway to prototype and develop such an infrastructure within OGSA.

Further information can be found at http://www.lesc.ic.ac.uk

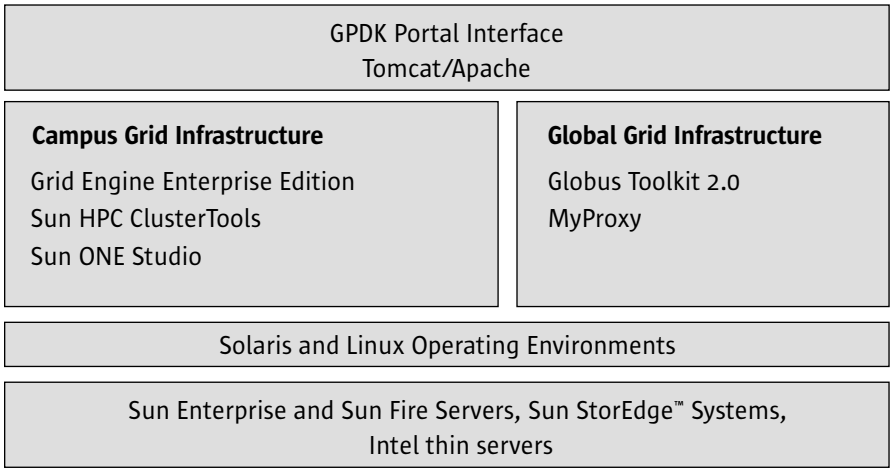
4.2 White Rose

The White Rose Grid (WRG), based in Yorkshire, UK is a virtual organization comprising of three Universities: The Universities of Leeds, York and Sheffield. There are four significant compute resources (Cluster Grids) each named after a white rose. Two cluster grids are sited at Leeds (Maxima and Snowdon) and one each at York (Pascali) and Sheffield (Titania).

The White Rose Grid is heterogeneous in terms of underlying hardware and operating platform. Whilst Maxima, Pascali and Titania are built from a combination of large symmetric memory Sun servers and storage/backup, Snowdon comprises a Linux/Intel based compute cluster interconnected with Myricom Myrinet.

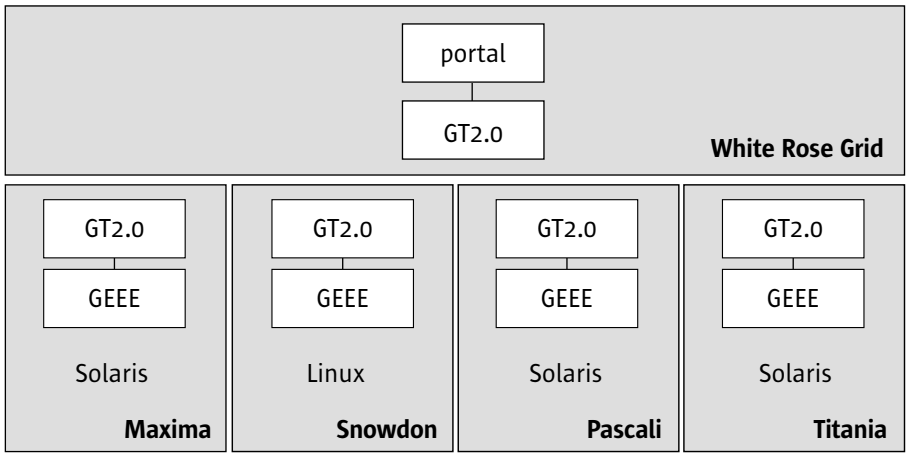
The software architecture can be viewed as four independent Cluster Grids interconnected through global grid middleware and accessible, optionally, through a portal interface. All the grid middleware implemented at White Rose is available in open source form.

The WRG software stack is composed largely of open source software. To provide a stable HPC platform for local users at each site, Grid Engine Enterprise Edition[3], HPC ClusterTools[5] and Sun ONE studio provide DRM, MPI support and compile/debug capabilities respectively.



Users at each Campus use the Grid Engine interface (command line or GUI) to access their local resource. White Rose Grid users have the option of accessing the facility via the portal. The Portal interface to the White Rose Grid has been created using the Grid Portal Development Kit3 (GPDK) running on Apache Tomcat. GPDK has been updated to work with Globus Toolkit 2.0 and also modified to integrate with various e-science applications.

Each of the four WRG cluster grids has an installation of Grid Engine Enterprise Edition. Globus Toolkit 2.0 provides the means to securely access each of the Cluster Grids through the portal.



Grid Engine Enterprise Edition

Grid Engine Enterprise Edition is installed at each of the four nodes, Maxima, Snowdon, Titania, and Pascali. The command line and GUI of Enterprise Edition is the main access point to each node for local users. The Enterprise Edition version of Grid Engine provides policy driven resource management at the node level. There are four policy types which may be implemented:

- **Share Tree Policy:** Enterprise Edition keeps track of how much usage users/projects have already received. At each scheduling interval, the Scheduler adjusts all jobs' share of resources to ensure that users/groups and projects get very close to their allocated share of the system over the accumulation period.
- **Functional Policy:** Functional scheduling, sometimes called priority scheduling, is a non-feedback scheme (i.e. No account taken of past usage) for determining a job's importance by its association with the submitting user/project/department.
- **Deadline Policy:** Deadline scheduling ensures that a job is completed by a certain time by starting it soon enough and giving it enough resources to finish on time.
- **Override policy:** Override scheduling allows the Enterprise Edition operator to dynamically adjust the relative importance of an individual job or of all the jobs associated with a user/department/project.

At White Rose, the Share Tree policy is used to manage the resource share allocation at each node. Users across the three Universities are of two types: (a) local users—those users who have access only to the local facility (b) WRG users—users who are allowed access to any node in the WRG. Each WRG node administrator has allocated 25% of their node's compute resource for WRG users. The remaining 75% share can be allocated as required across the local academic groups and departments. The WRG administrators also agree upon the half-life associated with SGEEE so that past usage of the resources is taken into account consistently across the WRG.

Globus

As depicted in the Illustration on page 46, Each WRG Cluster Grid hosts a Globus Gatekeeper. The default job manager for each of these gatekeepers is set to Grid Engine using the existing scripts in the GT2.0 distribution. In order that the Globus job manager is able to submit jobs to the local DRM, it is simply necessary to ensure that the Globus gatekeeper server is registered as a submit host at the local Grid Engine master node. The Globus grid-security file referenced by the gatekeeper servers includes the names of all WRG users. New users' grid identities must be distributed across the grid in order for them to be successfully authenticated. In addition, at each site all WRG users are added to the userset associated with the WRG share of the Enterprise Edition controlled resource. This ensures that the sum usage by WRG users at any cluster grid does not exceed 25%.

Portal interface

The portal technology used at White Rose has been implemented using the Grid Portal Development Kit. GPDK has been designed as a web interface to Globus. GPDK uses Java Server Pages (JSP™) and JavaBeans™ and runs in Apache Tomcat, the open source web application server developed by Sun Microsystems. GPDK takes full advantage of the Java implementation of the Globus CoG toolkit.

GPDK JavaBeans are responsible for the functionality of the portal and can be grouped into the five categories; Security, User Profiles, Job Submission, File Transfer, and Information Services. For security, GPDK integrates with MyProxy. MyProxy enables the Portal server to interact with the MyProxy server to obtain delegated credentials in order to authenticate on the user's behalf.

Some development work has been done in order to port the publicly available GPDK to GT2.0. Specifically:

- GPDK was modified to work with the updated MDS in GT2.0
- Information Providers were written to enable Grid Engine Queue information to be passed to the MDS. Grid users can query MDS to establish the state of the DRMs at each Cluster Grid.

As with many current Portal projects, the WRG uses the MyProxy Toolkit as the basis for security. The Illustration on page 46 shows that prior to interacting with the WRG, a user must first securely pass a delegated credential to the portal server so that the portal can act upon that user's behalf subsequently. The MyProxy Toolkit enables this.

The event sequence up to job submission is as follows:

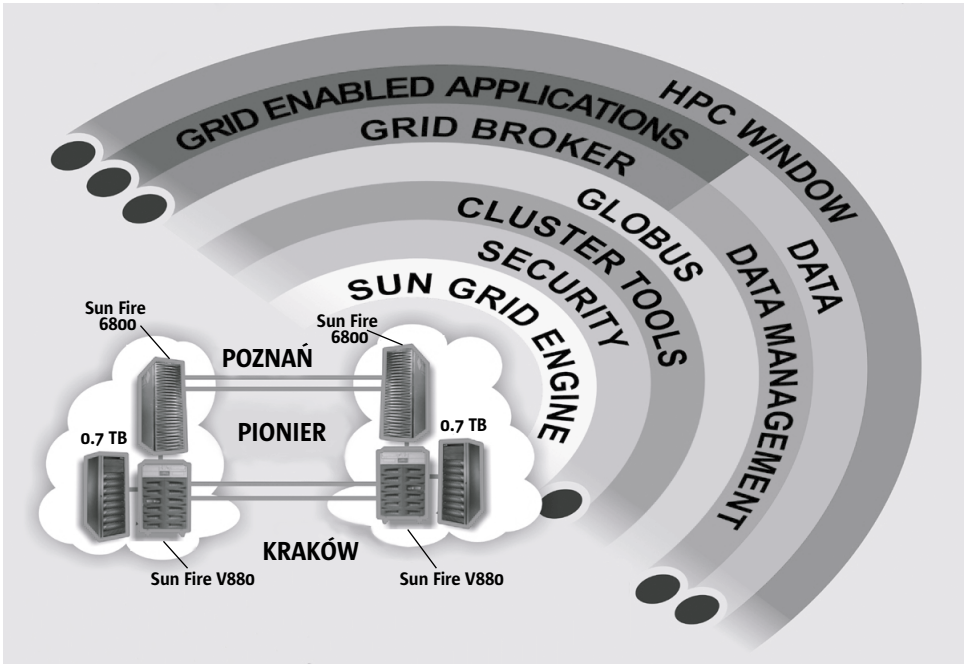
1. When the user initially logs on, the MyProxy Toolkit is invoked so that the portal server can securely access a proxy credential for that user.
2. The users can view the available resources and their dynamic properties via the portal. The Globus MDS pillar provides the GIIS, LDAP based hierarchical database which must be queried by the portal server.
3. Once the user has determined the preferred resource, the job can be submitted. The job information is passed down to the selected Cluster grid where the local Globus gatekeeper authenticates the users and passes the job information to Grid Engine Enterprise Edition.

4.3 Progress: Polish Research on Grid Environment on Sun Servers

The Poznan supercomputer project PROGRESS (Polish Research on Grid Environment for SUN Servers) aims at building an access environment to computational services performed by cluster of SUN systems. It involves two academic sites in Poland; Cracow and Poznan. The project was founded by the State Committee for Scientific Research. Project partners are: Poznan Supercomputing and Networking Center; Academic Supercomputing Center of University of Mining and Metallurgy, Cracow; Technical University Lodz and Sun Microsystems Poland.

Currently, there are two cluster grids accessible through the PROGRESS portal: two Sun Fire 6800 (24 CPUs) in Cyfronet Cracow and two Sun Fires 6800 (24 CPUs) connected using Sun Fire Link sited at Krakow. The distance between the locations is about 400 km. Both locations also use Sun Fire V880 and Sun Storedge 3910 as the hardware supporting the Distributed Data Management System discussed below. At the development stage only large Sun SMP machines are used, but the architecture of developed system allows the existing computing resources to be augmented by hardware from other vendors.

As in the White Rose implementation, Globus Toolkit 2.0 (2.2?) has been implemented to provide the middleware functionality. Also, as in the White Rose case, Sun Grid Engine Enterprise Edition is installed to control each of the Cluster Grids. In this case, however, the portal interface has been built using Web Services elements based upon J2EE.



Overview of PROGRESS project

The main task of this project is to give unified access to distributed computing recourses for the Polish scientific community. Other aims are:

- the development of novel tools supporting grid-portal architecture (Grid service broker, security, migrating desktop, portal access to grid)
- the development and integration of data management and visualization modules
- Enabling the grid-portal environment for other advanced applications (PIONIER program)

System Modules

The PROGRESS architecture can be described in terms of its constituent modules. As well as using the Globus Toolkit, each of these modules provides a major piece of functionality for the PROGRESS grid. The four main modules are:

- Portal Environment
- Grid Service Provider
- Grid Infrastructure
- Data Management System

Portal Environment

The main module of the Progress Portal Environment is the GSP (Grid Service Provider). It is a new layer introduced into the Grid Portal architecture by the PROGRESS research team. The GSP provides users with three main services: a job submission service (JS), an application management service (AM) and a provider management service (PM).

- The JS is responsible for managing the creation of user jobs, their submission to the grid and monitoring of their execution.
- The AM provides functions for storing information about applications available for running in the grid. One of its main features is the possibility of assisting application developers in adding new applications to the application factory.
- Finally, the PM allows the GSP administrator to keep up-to-date information on the services available within the provider.

Grid Service Provider (GSP)

The PROGRESS GSP services are accessible through two client interfaces: WP (Web Portal) and MD (Migrating Desktop). The Web Portal, which is deployed on the Sun ONE Portal Server 7.0, performs three functions:

- grid job management: creating, building, submitting, monitoring execution and analyzing results,
- application management,
- provider management.

A screenshot depict a typical user's view of the portal interface is shown in the illustration below.

PROGRESS Portal

The MD, which is a separate Java client application, provides user interface for grid job management and DMS file system management. Both user interfaces are installed on a Sun Fire 280R machine, which serves as the PROGRESS system front-end.

Additionally PROGRESS Portal gives access to services like: news services, calendar server, and messaging server, deployed on Sun ONE calendar and messaging server.

Grid Infrastructure

The Grid Resource Broker (GRB) is developed in PSNC and enables execution of PROGRESS grid jobs in a grid clusters. Clusters are managed by Sun Grid Engine Enterprise Edition software with Globus deployed upon it. The GRB provides two interfaces: a CORBA interface and a web services one. Grid job definitions are passed to the GRB in the form of an XRSL document and the GRB informs the JS about events connected with the execution of a job (start, failure or success).

Sun ONE Grid Engine Enterprise Edition is implemented at both sites as the local distributed resource manager. Grid Engine Enterprise Edition provides policy driven resource management at the node level. The Share Tree policy is used to manage the resource share allocation at each node.

Two types of users can have access to resources:

- local users, accessing compute resources through Grid Engine GUI
- portal users, accessing nodes using the PROGRESS portal or Migrating Desktop.

In the event of extensions to the PROGRESS grid, Grid Engine would also be used. Each Cluster Grid also hosts Globus Gatekeeper.

Data Management System (DMS)

PROGRESS grid jobs use the DMS to store the input and output files. The DMS provides a web services based data broker, which handles all requests. The DMS is equipped with three data containers the file system, the database system and the tape storage system. A data file is referenced within the DMS with a universal object identifier, which allows for obtaining information on the location of the file.

Users can download, or upload file using one of three possible protocols: FTP, GASS or GridFTP.

4.4 The University of Houston Campus Grid

Several major research activities at the University of Houston require access to considerable computational power and, in some cases, large amounts of storage. To accommodate many of these needs locally, a *campus-wide grid* was created that combines this facility with departmental clusters using Grid Engine for job submission. Future plans include collaborating with regional and national partners to form a wide area grid.

A variety of on-going scientific research projects at the University of Houston (UH) require access to significant computational resources. Researchers in Chemistry, Geophysics, Mechanical Engineering, Computer Science and Mathematics are among those who routinely make use of parallel and clustered systems across campus. One of the most demanding collaborative research efforts involves a team of scientists who are working on the numerical simulation and modeling of atmospheric pollution with a special focus on subtropical Gulf Coast regions such as Houston-Galveston-Brazoria. Their work includes developing and deploying a parallel version of a community Air Quality Model code that will forecast the impact on atmospheric pollution of various strategies for the reduction of volatile organic compounds and nitric oxide. A photochemical air quality model consists of a set of coupled partial differential equations, one for each chemical species. The input to these equations is the complete local weather data and concentrations of chemical precursor molecules, ideally from real-time monitoring. The execution scenario for this

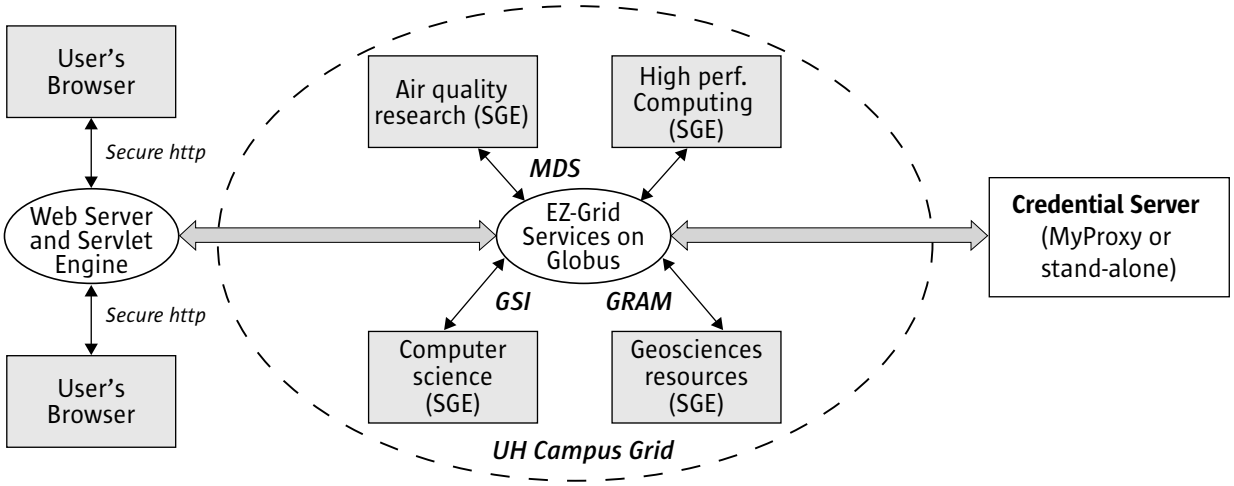
project therefore involves the execution of a limited area weather model—indeed, multiple limited area models are executed on increasingly smaller, but more closely meshed, domains—in conjunction with the chemical model. It also relies on global weather data that is automatically retrieved each day. The grid system is required to start the weather model once this data is locally available and, once the weather code has reached a certain phase, initiate execution of the chemical code. Since these are run at separate grid sites, the files must be automatically transferred.

Geophysicists at UH are engaged in research, development and evaluation of seismic processing and imaging algorithms, which generally involve the handling of very large data sets. Geophysicists use the systems described below to develop methods for *high resolution imaging of seismic data* in order to better identify and quantify hydrocarbon reserves, and aggressively disseminate results to the oil and gas industry. Industry partners provide real-world data and the opportunity to verify results via testing in a working oil field. Work includes developing a 3-D prestack wave equation depth migration algorithm: 3-D prestack imaging is the most compute intensive application currently run within the supermajor oil and geophysical service companies, consuming the vast majority of CPU cycles. The workload generated by these scientists includes both low-priority long running jobs and short, high-priority imaging algorithms. High-speed access to our storage system is critical for their execution.

UH has both large and small computational clusters connected through optical fiber across the campus. The hardware available in the High Performance Computing Center (HPCC) includes a cluster of Sun Fire 6800 and 880 platforms connected via Myrinet. They are available to a broad cross-section of faculty and are deployed for both research and teaching. This facility is heavily utilized, with a high average queue time for submitted jobs. We therefore exploit the availability of other resources on campus to alleviate this problem by operating a number of different systems including those at the HPCC in a campus-wide grid

The campus grid is divided into several administrative domains corresponding to the owner of the hardware, each of which may contain multiple clusters with a shared file system. In our environment all basic grid services, such as security and authentication, resource management, static resource information, and data management, are provided by the Globus toolkit [5][6][7], whose features may be directly employed by accredited users to submit jobs to the various clusters. An independent certification authority is managed by HPCC. Sun Grid Engine (SGE) [14] serves as the local resource manager within domains and is thus the software that interfaces with the Globus resource manager component. However, it is a daunting task for many application scientists to deal with a grid infrastructure; we have therefore developed a portal interface to make it easy for them to interact with grid services. It can be used to obtain current information on resources, move files between file systems and to track their individual account usage and permissions, as well as to start jobs. The locally developed EZGrid system [2], using Globus as middleware, provides an interface to authenticate users, provide information on the system and its status, and to schedule and submit jobs to resources within the individual domains via Grid Engine. The development of EZGrid was facilitated by the Globus CoG Kits [10], which provide libraries that enable application developers to include Globus' middleware tools in high-level applications in languages such as Java and Perl. The portal server has been implemented with Java servlets and can be run on any web server that supports Java servlets.

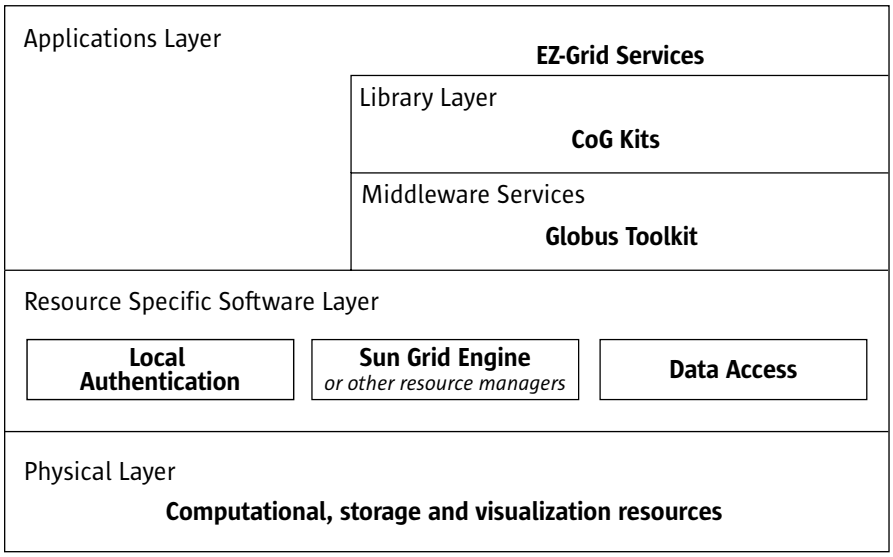
Users can access grid services and resources from a web browser via the EZGrid portal. The grid's credential server provides the repository for the user credentials (X509 certificate and key pairs and the proxies). It is a secure stand-alone machine that holds the encrypted user keys. It could be replaced by a MyProxy server [11] to act as an online repository for user proxies. However, this adds an upper limit to the mobility of the users due to the limited lifetime of the Globus proxies delegated. The stand-alone credential server model was adopted to allow users to access grid services with unlimited mobility through a browser even if they have no access to their grid identities. Appropriate mapping of the portal accounts to proxies allow users to perform single sign-on and access grid resources. The portal supports the export of encrypted keys from the user to the credential server using secure http sessions. In some scenarios, the credential server can also be used to generate the user credentials, thus ensuring enhanced security.

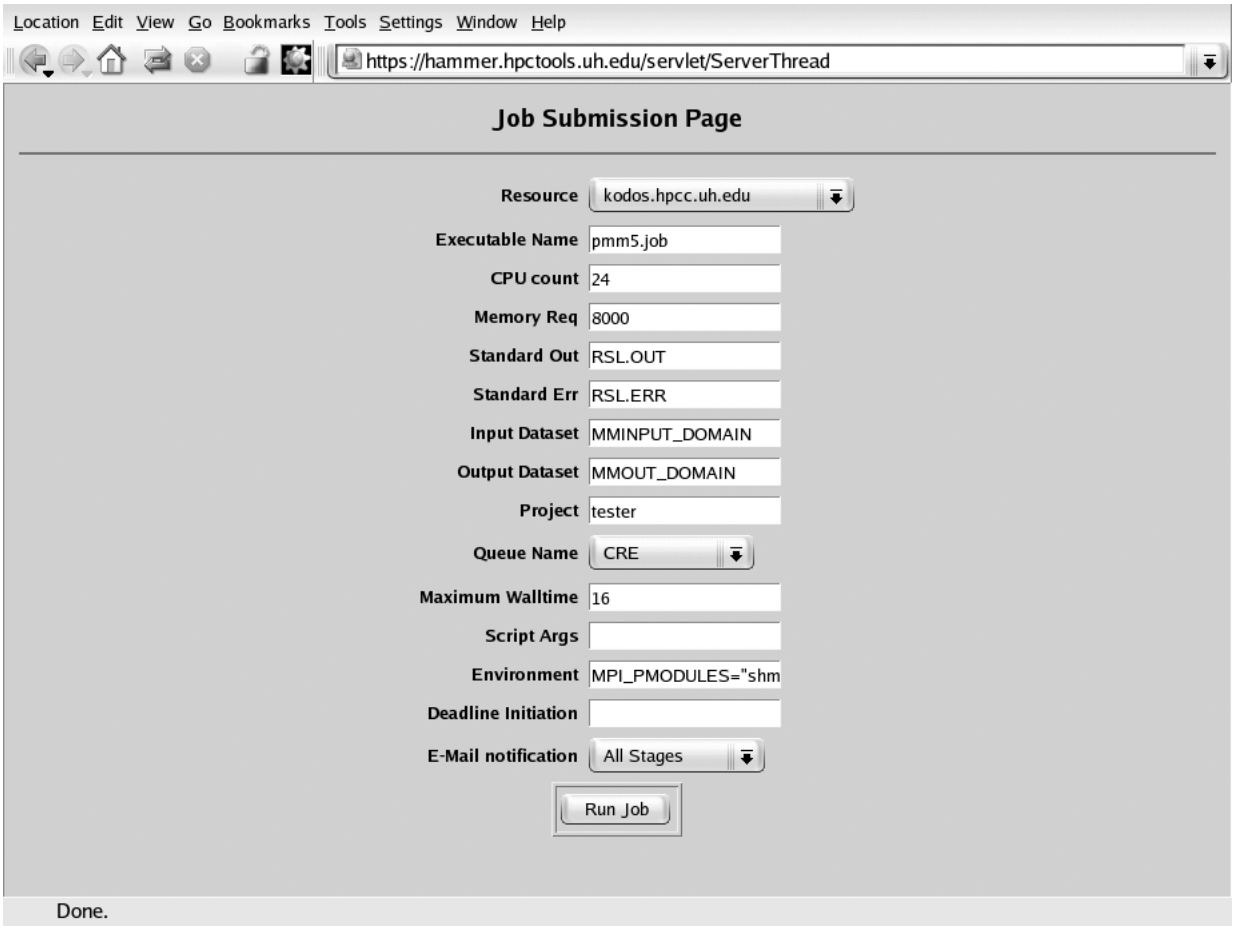


EZ-Grid provides the following major grid services for its users:

- **Single sign-on:** Globus proxy (temporary identity credential) creation using Grid Security Infrastructure [8] [1] and X509 certificates. This allows the user to seamlessly establish his or her identity across all campus grid resources.
- **Resource information:** Viewable status information on grid resources, both static and dynamic attributes such as operating systems, CPU loads and queue information. Static information is obtained primarily from Globus Information services such as MDS [3] and dynamic scheduler information and queue details are retrieved from SGE. Users can thus check the status of their jobs, load on the resources and queue availability. Additional information provided includes application profiles (metadata about applications) and job execution histories and so forth.
- **Job specification and submission:** a GUI that enables the user to enter job specifications such as the compute resource, I/O and queue requirements. Automated translation of these requirements into Resource specification language (RSL) [13] and subsequent job submission to Globus Resource Allocation Managers (GRAM) [4] are supported by the portal. Scripts have been implemented to enable job hand-off to SGE via Globus services. Further, automated translation of some job requirements into SGE parameters is supported.
- **Precise usage control:** Policy-based authorization and accounting services [12] to examine and evaluate usage policies of the resource providers. Such a model is critical when sharing resources in a heterogeneous environment like the campus grid.
- **Job management:** Storage and retrieval of relevant application profile information, history of job executions and related information. Application profiles are metadata that can be composed to characterize the applications.
- **Data handling:** Users can transparently authenticate with and browse remote file systems of the grid resources. Data can be securely transferred between grid resources using the GSI-enabled data transport services.

The following diagram shows how EZ-Grid interacts with other middleware tools and resource management systems.





Conclusions

Public domain software such as Globus and Sun Grid Engine may be used to construct a grid environment such as the campus grid we have described. However, simplified access to grid services is essential for many computational scientists and can have a positive impact on the overall acceptance of this approach to resource utilization. A variety of projects, including HotPage [16], Gateway and UNICORE [15], provide a portal interface that enables the user to access information provided by Globus or the underlying grid services. Toolkits such as GridPort [9] and GPDK [17] have been developed to simplify the construction of portals that exploit features of Globus. While the system is quite similar to these latter efforts, it provides more extensive functionality, in particular by exploiting information that can be easily retrieved from Sun Grid Engine also.

Custom portals for the main grid users are now being built, including those involved in air quality modeling at UH, and in the process, the modularization of the code is improving. There is much scope for improvement and expansion in our environment. Plans are underway to include additional clusters into the grid, and to extend the user base. It has already been used for classroom assignments. New releases of Globus and endeavor are being tracked to be as up-to-date as possible. Although the needs of larger systems have been taken into account, the EZGrid software will need some revision and functional extension once systems are linked into larger grids with additional organizations.

[1] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," International Journal of Supercomputer Applications, Summer 1997.

[2] I. Foster and C. Kesselman, "The GRID: Blueprint for a new Computing Infrastructure," Morgan Kauffman Publishers, 1999.

[3] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of Supercomputer Applications, 15(3), 2001.

[4] Sun Grid Engine, Sun Microsystems, http://www.sun.com/software/gridware

[5] B. M. Chapman, B. Sundaram, K. Thyagaraja, "EZGrid system: A Resource broker for Grids," http://www.cs.uh.edu/~ezgrid

[6] G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, "CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids," ACM 2000 Java Grande Conference, 2000.

[7] J. Novotny, S. Tuecke, V. Welch, "An Online Credential Repository for the Grid: MyProxy," Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

[8] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecture for Computational Grids," ACM Conference on Computers and Security, 1998, 83-91.

[9] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch, "A National-Scale Authentication Infrastructure," IEEE Computer, 2000.

[10] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, "Grid Information Services

for Distributed Resource Sharing," 2001.

[11] Resource Specification Language, RSL, http://www.globus.org/gram/rsl_spec1.html

[12] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, "A Resource Management Architecture for Metacomputing Systems," Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[13] B. Sundaram, B. M. Chapman, "Policy Engine: A Framework for Authorization, Accounting Policy Specification and Evaluation in Grids," 2nd International Conference on Grid Computing, Nov 2001.

[14] J. Boisseau, S. Mock, M. Thomas, "Development of Web Toolkits for Computational Science Portals: The NPACI HotPage", 9th IEEE Symposium on High Performance Distributed Computing, 2000

[15] Uniform Interface to Computing resource, UNICORE, http://www.unicore.de

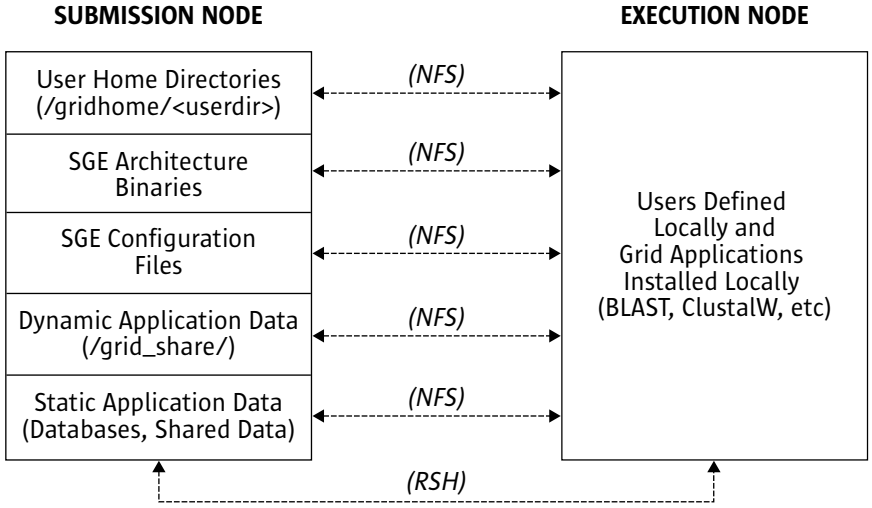
[16] GridPort, https://gridport.npaci.edu/

[17] J. Novotny, "The Grid Portal Development Kit", Concurrency—Practice and Experience, 2001

4.5 Canadian Bioinformatics Resource

The National Research Council of Canada's Canadian Bioinformatics Resource (NRC-CBR) is a distributed network of collaborating institutes, universities and individuals across Canada dedicated to the provision of bioinformatics services to Canadian researchers. NRC-CBR is also a Sun Center of Excellence in Distributed BioInformatics. Some highlights of NRC-CBR include: the largest installation of the Sequence Retrieval System(SRS) in North America; official European Molecular Biology Network (EMBnet) Node for Canada; founding and active member of the Asian Pacific Bioinformatics Network (APBioNet); North American node for ExPASy proteomics server. The excellent high bandwidth and world leadership of Canadian networks, namely CANARIE Inc.'s CA*net4, is leveraged to integrate data storage and stand-alone applications at our member sites with centrally maintained databases and dedicated hardware to provide users with an environment that evolves with their needs because they drive it, and scales smoothly as membership grows. NRC-CBR is conceived as a collaboration of peers, each contributing the unique expertise of their organization under a common umbrella, thus web services are distributed across member organizations, datasets and applications developed at member sites are distributed through NRC-CBR, and hardware resources are donated by members for the common benefit, all underwritten by the funding of core services by the National Research Council of Canada.

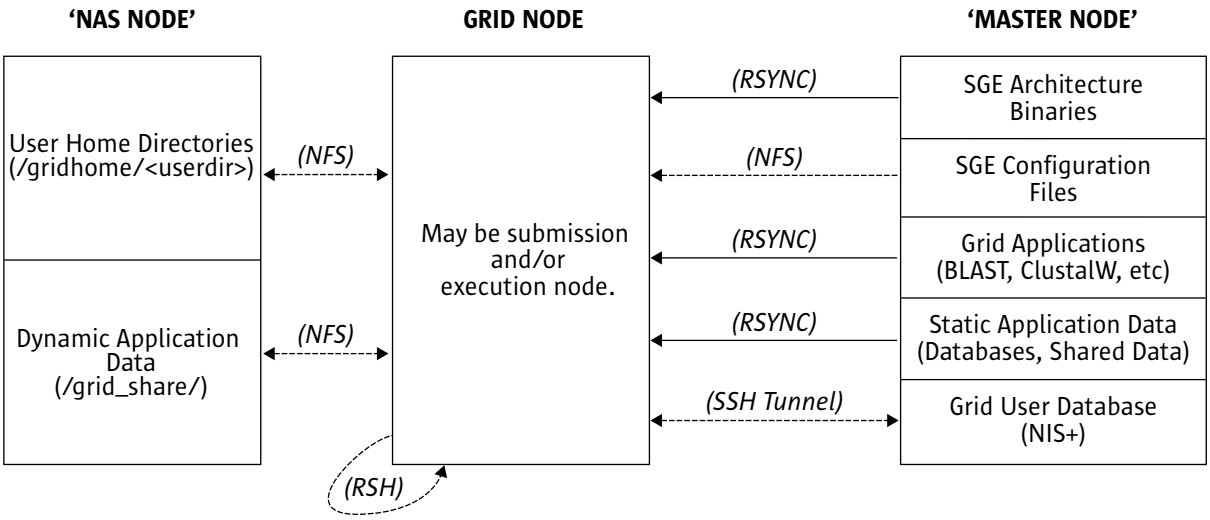
The provision of service may be the mission, but NRC-CBR has a record of commitment to research to further that mission. They are grid-enabling their servers across the country, to provide a closer integration of member sites. In its bid to develop a Bioinformatics Grid for Canada, NRC-CBR is employing Cactus, GridEngine and GLOBUS in collaboration with CANARIE, C3.ca and other NRC institutes. These collaborative efforts have formed the basis of an MOU between NRC, CANARIE and C3.ca to form Grid Canada. Grid Canada, through NRC, will collaboratively interconnect the HPCE sites of more than 30 universities and NRC, including NRC-CBR's HPCE. Cactus developers will collaborate with NRC-CBR to develop bioinformatics and biology thorns for integration with Cactus. Perl and Java will be integrated into the Cactus core as bioinformatics applications are typically written with these languages. Integration of Cactus, Grid Engine and GLOBUS components will be necessary to facilitate diverse functionality in applications using distributed multi-vender platforms.



In correlation with the grid developments, NRC-CBR is developing secure portal access to its services to effectively present users with a single sign-on secure environment where the underlying national grid infrastructure and distributed NRC-CBR resources are transparent to the user community. Biologists are only concerned with using the tools and accessing the data, not the complexities of the underlying grid or informatics architecture.

While new experimental approaches permit whole-organism investigations of proteomics, metabolomics, gene regulation and expression, they also raise significant technical challenges in the handling and analysis of the increased data volume. The Canadian Bioinformatics Resource (CBR) provides biologists across Canada with access to bioinformatics applications and databases, large-volume data storage, basic tutorials and help desk support. CBR provides this service to scientists at the National Research Council of Canada as well as to academic and not-for-profit users associated with Canadian universities, hospitals and government departments. Any infrastructure serving this community must provide a low-cost, secure and intuitive environment integrating a wide range of applications and databases. From an administrative perspective, the solution must also be scaleable to accommodate increasing usage and data flow, and also promote economies of scale with regard to systems administration and user support. Over the last year or so it has become clear that many of these requirements are served by emerging grid technologies.

CBR is in the early stages of upgrading existing centralized infrastructure, forming a bioinformatics grid to integrate geographically distributed computational and data resources. An initial step toward this goal has been the formation of a CBR 'WAN cluster grid', linking member sites using the Sun Grid Engine Enterprise Edition software. As the figure on page 55 shows, the initial configuration did not establish a clear division between data, administrative and execution elements of the grid architecture, and NFS mounting was used as the sole means of making data and binaries available on grid nodes, with network latency slowing overall performance. In the absence of NIS+ (due to conflicting local and grid user definitions) or a shared repository of application binaries, grid administration became a significant and increasing overhead. In order to ensure that all results were returned to the same location, the grid submission command qsub was wrapped in a script to set the output destination to the NFS-mounted user's home directory.



A range of bioinformatics applications are being made to interface with the CBR grid for eventual deployment; these applications fit into two categories, the first containing heavily-used small utilities, which benefit from the scheduling features of a grid to spread load and parallelize workflow. As an example, most of the utilities in the European Molecular Biology Open Software Suite (EMBOSS) fit well into this category. In the second category are applications which can be parallelized but which are tolerant of the concurrency issues inherent in the loosely coupled distributed environment of a grid. Examples include any database search, such as the Sequence Retrieval System, SRS (provided the databases to be searched can be subdivided and maintained on many nodes), advanced pattern matching searches such as BLAST or Patternmatcher, or multiple sequence alignment where the algorithm is based on many pairwise alignments, such as ClustalW. In conjunction with these developments, CBR plans to implement a revised grid architecture which more closely fits the requirements of low maintenance costs and a heterogeneous environment (see figure above). The most obvious change is that the principal elements of grid function have been separated; administration and configuration are largely restricted to the 'Master node', which performs no other grid function. The rsync remote filesystem synchronization process will be used to ensure all grid nodes securely share a common configuration and appropriate OS and architecture-dependent application binaries. Although CBR makes extensive use of the CA*net4 high-speed Canadian research network stretching 5,500 miles from Halifax to Vancouver, not all member sites have high-bandwidth connections, and performance can be improved by adopting a mixed strategy of rsync for transfers of large or static datasets, and NFS (tunneling through Secure Shell for security) where datasets are small or dynamic. Experience has also shown us that grid execution nodes frequently serve other non-grid functions at member sites and therefore we will initially provide NIS+ user authentication to supplement locally defined users, and are investigating LDAP as a means to more securely provide this functionality. The new grid architecture also includes a 'NAS node' which (like the 'Master node') is neither a submission nor execution node, serving instead as a centralized point of Network Attached Storage for home directories and user datasets NFS mounted onto grid submission and execution nodes, making user management, password administration, permissions and quotas easier to administer as both grid and user base grow.

CBR's investigations into grid architecture and implementation have revealed that although the technology is not yet mature, existing applications and protocols can be used to create a usable and maintainable grid environment with an acceptable level of customization. In the future, CBR will be investigating better data access methods for distributed databases (virtual data views and database federation) and interfacing the CBR grid with the cluster grid of the High Performance Virtual Computing Laboratory, a consortium of four universities in eastern Ontario, Canada. This linkage will be made using tools from the Globus public-domain grid toolkit to provide single sign-on portal access to all CBR resources, many of which will be configured as grid services. In this way, CBR hopes to provide enhanced bioinformatics services which more efficiently and intelligently use the computing resources distributed across Canada.

Chapter 5

5 Summary

The term "*Grid Computing*" has many different definitions. Many of them are simplistic as they do not provide a description on how to solve basic problems like authentication, accounting, and web services in the context of an existing infrastructure. Grids cannot be built as an island, they have to use existing environments—which offers the opportunity to leverage numerous technologies simultaneously. At the same time, recognizing this diversity, there must be a way to deal with the resulting complexity. One of the main benefits of the Grid is the ability to connect heterogeneous environments to a common resource pool. This requires, however, open standards. Not only do standards enable the smooth interworking of system components, they also allow access to a widely available skills base, which is key to building software infrastructures that deliver compute and data capabilities to end-users.

Sun's vision is founded on open standards and using the most widely available tools and techniques. The Sun ONE application platform has all the components and tools needed to build the most sophisticated services infrastructure.

The Services on Demand vision would not be possible without flexible platform provisioning, capable of providing highly scalable systems from the commodity cluster range to supercomputer class systems, from Linux to Solaris based systems. This is a matter of getting the right resources for the job.

The diversity that characterizes these platforms requires not just powerful management tools but also a powerful focus on the overall architecture of management simplicity. Sun's strategic N1 vision architecture and products provides the basis for reducing complexity, saving on costs and increasing data center utilization.

Collaborative work in technical research depends on bringing together massive amounts of computing resources across administrative domains. Offering these resources as easily accessible services will provide significant benefits to solving scientific problems that depend on collaboration. Sun provides the tools today to build the Services On Demand infrastructure that will make Grid computing a reality.

Appendix A: Sun Services

Sun Professional Services℠ and Sun's integrator partners address client needs by offering a set of Sun ONE Core Services, based upon the SunTone℠ Architectural Methodology, that provide a quick, cost-effective, and efficient means for delivering solutions to challenging business problems.

These services offerings include:

- **Sun ONE Rapid Needs Assessment.** A 2-4 week project, led by a senior Consultant/Project Manager, to analyze the key business and technology requirements, deadlines, risks, current situation and critical success factors, in order to confirm feasibility in principle and build a concrete plan for a way forward to achieving the goals.
- **Sun ONE Architecture Assessment Service:** A 4-6 week project, led by architects and/or consultants, involving an assessment of an organization's technical architecture, with the focus on identifying best practices as well as opportunities for enhancement to support evolution toward a Sun ONE Services on Demand architecture.
- **Sun ONE Workshop:** A 3-day onsite workshop, facilitated by architects and consultants, focused on assisting the client in the definition of a strategic plan for moving to a Sun ONE Services on Demand architecture.
- **Sun ONE Inception Service:** A 6-8 week project, led by a Sun Professional Services Project Manager and a senior architect, focused on definition of an Architecture Framework that addresses the business, functional, and service-level requirements of an organization's Sun ONE strategic plan. Additionally, an Elaboration Plan, including a Prototype Plan, is developed as a part of this service.
- **Sun ONE Elaboration Service:** An 8-12 week project, composed of multiple iterations and led by Sun Professional Services project manager and senior architect(s), focused on refinement and prototyping of the Architecture Framework developed in the Inception Phase, as well as creation of a comprehensive Construction Plan.
- **Sun ONE Construction Service:** A 4-8 month project, depending upon scope defined in the Construction Plan, involving Sun Professional Services architects acting as mentors and in quality assurance roles to ensure the integrity of the Sun ONE architecture as it is constructed by Sun architects, engineers, and SI and ISV partners.
- **Sun ONE Transition Service:** A 4-8 week project, led by Sun Professional Services architects and engineers, focused on knowledge and skills transfer to enable client staff or consultants on the ongoing maintenance and support of the delivered system(s).

Professional Services engagement managers are adept at understanding client-specific challenges and crafting a service offering to meet each client's unique need

Appendix B: Glossary

DRMAA — Distributed Resource Management Application API

DRMS — Distributed resource management system

GARA — Globus Architecture for Reservation and Allocation

GASS — Global Access to Secondary Storage

GRAM — Globus Resource Allocation Manager

Grid Service — A service the conforms to the Grid Service Specification of the OGSI

GSH — Grid Service Handle

GSI — Grid Security Infrastructure

GSR — Grid Service Reference

MDS — Monitoring and Discovery Service

OGSA — Open Grid Services Architecture

OGSI — Open Grid Services Interface

SOAP — Simple Object Access Protocol

WSDL — Web Services Definition Language

WSIL — Web Service Inspection Language

XML — eXtensible Markup Language

Sun ONE — Sun Open Networking Environment

J2EE — Java 2 Platform, Enterprise Edition

Appendix C: Contributors

Editor-in-Chief: Heinz J. Schwarz

Principal author (Chapters 1 and 2): Paul Wolf, Ripeco

Chief reviewers: Shahin Khan, Marc Hamilton, Peter Overton

Barbara Chapman and Babu Sunderam, University Houston, Chapter 3.4.3

Jason Burgess and Terry Dalton, Canada NRC-CBR, Chapter 3.4.4

David Wolland, Ripeco

Steven Newhouse, Imperial College

Cezary Mazurek and Krzysztof Kurowsky from Poznan Supercomputing and Networking Center, Chapter

Authors: Miha Ahronowitz, James Coomer, Charu Chaubal, Dan Fraser, John C. Fowler, Derek Gardiner, Wolfgang Gentzsch, Ferhat Hatay, Brian Hammond, Radoslaw Rafinski, Stefan Unger, Heinz J. Schwarz from Sun Microsystems

Appendix D: References and Resources

Web resources and references

Globus Toolkit

http://www.globus.org

Global Grid Forum

http://www.ggf.org

Simple Object Access Protocol (SOAP) 1.1

http://www.w3.org/TR/SOAP

Universal Description, Discovery and Integration (UDDI):

http://www.uddi.org

Web Service Definition Language

http://www.w3.org/TR/wsdl

OGSA Architecture

http://www.globus.org/research/papers/ogsa.pdf

The Anatomy of the Grid: Enabling Scalable Virtual Organizations

http://www.globus.org/research/papers/anatomy.pdf

Computational Grids., I. Foster, C. Kesselman. Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman, 1999.

http://www.globus.org/research/papers/chapter2.pdf

Open Grid Services Infrastructure (OGSI)

http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-26_2003-03-13.pdf

The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration

http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf

Sun Resources

Computational Biology SIG

http://www.sun.com/edu/hpc/compbio-sig

International Informatics Infrastructure Consortium (i3c)

http://www.i3c.org

Java

http://java.sun.com

Products (hardware/software)

http://www.sun.com/products

Sun Collaborative Visualization Solutions

http://www.sun.com/desktop/visualize

Sun Global Education and Research

http://www.sun.com/edu

Sun Edu Computational Biology

http://www.sun.com/edu/hpc

Sun ONE

http://www.sun.com/software/sunone

Sun open source

http://www.sunsource.net

Sun Grid Engine

http://wwws.sun.com/software/gridware

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 800 786-7638 or +1 512 434-1577 **Web** sun.com

We make the net work.

Sun Worldwide Sales Offices: Argentina: +5411-4317-5600, Australia: +61-2-9844-5000, Austria: +43-1-60563-0, Belgium: +32-2-704-8000, Brazil: +55-11-5187-2100, Canada: +905-477-6745, Chile: +56-2-3724500, Colombia: +571-629-2323 Commonwealth of Independent States: +7-502-935-8411, Czech Republic: +420-2-3300-9311, Denmark: +45 4556 5000, Egypt +202-570-9442, Estonia: +372-6-308-900, Finland: +358-9-525-561, France: +33-134-03-00-00, Germany: +49-89-46008-0 Greece: +30-1-618-8111, Hungary: +36-1-489-8900, Iceland: +354-563-3010, India–Bangalore: +91-80-2298989/2295454; New Delhi: +91-11-6106000; Mumbai : +91-22-697-8111, Ireland: +353-1-8055-666, Israel: +972-9-9710500 Italy: +39-02-641511, Japan: +81-3-5717-5000, Kazakhstan: +7-3272-466774, Korea: +822-2193-5114, Latvia: +371-750-3700, Lithuania: +370-729-8468, Luxembourg: +352-49 11 33 1, Malaysia: +603-21161888, Mexico: +52-5-258-6100 The Netherlands: +00-31-33-45-15-000, New Zealand–Auckland: +64-9-976-6800; Wellington: +64-4-462-0780, Norway: +47 23 36 96 00, People's Republic of China–Beijing: +86-10-6803-5588; Chengdu: +86-28-619-9333 Guangzhou: +86-20-8755-5900; Shanghai: +86-21-6466-1228; Hong Kong: +852-2202-6688, Poland: +48-22-8747800, Portugal: +351-21-4134000, Russia: +7-502-935-8411, Saudi Arabia: +9661 273 4567, Singapore: +65-6438-1888 Slovak Republic: +421-2-4342-94-85, South Africa: +27 11 256-6300, Spain: +34-91-596-9900, Sweden: +46-8-631-10-00, Switzerland–German: 41-1-908-90-00; French: 41-22-999-0444, Taiwan: +886-2-8732-9933, Thailand: +662-344-6888 Turkey: +90-212-335-22-00, United Arab Emirates: +9714-3366333, United Kingdom: +44-1-276-20444, United States: +1-800-555-9SUN OR +1-650-960-1300, Venezuela: +58-2-905-3800, Or Online at sun.com/store

Printed in USA 6/03 FE2021-0/xK