

College of

**ENGINEERING**

*Fall 2006, CMPE-272*

*Under Guidance of Dr. Dan Harkey*

Project Report on

# Grid Computing

Maulik Thaker	<a href="mailto:maulikthaker@gmail.com">maulikthaker@gmail.com</a>	004676309
Kunal Vyas	<a href="mailto:Kunalvyas28@gmail.com">Kunalvyas28@gmail.com</a>	004700359
Vaibhav Sharma	<a href="mailto:Vaibhavsh83@yahoo.com">Vaibhavsh83@yahoo.com</a>	004689218
Vrushali Deo	<a href="mailto:vrushali_deo@yahoo.co.in">vrushali_deo@yahoo.co.in</a>	005256876
Vinaya Damle	<a href="mailto:vinayadamle@gmail.com">vinayadamle@gmail.com</a>	005239846

## Table Of Contents

<b>TABLE OF CONTENTS</b> -----	<b>2</b>
<b>1. ABSTRACT</b> -----	<b>3</b>
<b>2. MOTIVATION</b> -----	<b>3</b>
<b>3. INTRODUCTION</b> -----	<b>3</b>
3.1 BRIEF HISTORY-----	3
<b>4. GRID COMPUTING AS A TECHNOLOGY</b> -----	<b>4</b>
4.1 MAKING THE GRID WORK-----	5
4.1.1. <i>Resource Sharing</i> -----	5
4.1.2. <i>Resource Management</i> -----	6
4.1.3. <i>Quality of Service (distance no barrier!!)</i> -----	7
4.1.4. <i>Security</i> -----	8
4.1.6. <i>Standardization</i> -----	9
<b>5. LAYERED GRID ARCHITECTURE</b> -----	<b>10</b>
<b>6. CURRENT STATE GRID ARCHITECTURE</b> -----	<b>11</b>
<b>7. GRID TOOLS</b> -----	<b>12</b>
7.1 G-ECLIPSE-----	12
7.2 PYGLOBUS-----	14
7.3 MOAB GRID SUITE 4.5-----	14
<b>8. CURRENT GRID STANDARDS</b> -----	<b>15</b>
8.1 GLOBAL GRID FORUM-----	16
8.2 OASIS-----	16
8.3 DISTRIBUTED MANAGEMENT TASK FORCE-----	16
8.4 OPEN GRID SERVICES ARCHITECTURE (OSGA)-----	17
8.5 OPEN GRID SERVICES INFRASTRUCTURE (OSGI)-----	17
8.6 WEB SERVICES RESOURCE FRAMEWORK-----	18
<b>9. BENEFITS OF GRID COMPUTING</b> -----	<b>18</b>
9.1 BUSINESS BENEFITS-----	18
9.2 TECHNOLOGY BENEFITS-----	19
<b>10. CONCLUSION</b> -----	<b>19</b>
<b>11. REFERENCES</b> -----	<b>20</b>
11.1 BOOKS-----	20
11.2 WEB LINKS-----	20
11.3 PAPERS:-----	20

# 1. ABSTRACT

Many scientific problems today remain unsolved due to the lack of sufficient computing resources such as computing power and memory available. Needs are rapidly increasing with the ever increasing data generated in applications such as nuclear physics, astronomy, meteorology, biology and medicine to name a few. Grid computing is an emerging technology which aims at making unlimited computing resources available to the user by maximizing the use of existing resources. Grid computing enables the pooling of all the computers inter-connected by networks today to operate them in a single virtual environment. This gives the end user an illusion of a single computer with immense computational ability, unlimited data storage capacity and a wide variety of application software. In this paper, we propose to discuss the key concepts of Grid computing technology, the potential it promises, the proposed grid architecture, the emerging standards in order to ensure inter-operability, the various applications that the technology holds, the challenges involved in converting the vision of a computing grid into a reality and the past and future of Grid Computing technology.

# 2. MOTIVATION

Grid technology is one of the strongest and fastest growing commercial technologies in IT enterprise. It affects virtually each and every area of computing where cost effective infrastructure and computing is the limitation. Grid computing offers the exciting possibility of providing access to a massive amount of resources without actually owning them. Thus, individuals and organizations can now solve large problems with a limited infrastructure and capital expenditure and quickly adapt to changing business needs. This tremendous potential of a computing grid is causing a rapid adoption of grid technology in the commercial world today.

Government and scientific labs have been using grids now for several years but now many private companies have begun to deploy grids in businesses to provide a practical solution for the High performance computing demand.

*Just to mention – IBM are the pioneers of the Grid computing and a driving force in standardizing the grid deployments and architectures like Open Grid Services Architecture (OGSA), and its companion implementation standard, the Open Grid Services Infrastructure (OGSI). Intel has a \$9billion a year investment in research and development of deployable grid technologies. LIP in Portugal, a high energy physics research laboratory is involved in deploying Data Grids, Enabling Grids of EScience (EGEE) and many others from 2001. Hence it is in our interest to understand and familiarize ourselves with this technology before it archaic us. The stage could not be better for us than this, wherein we take a closer look to this powerful technology.*

# 3. INTRODUCTION

## 3.1 Brief History

Although Grid computing is a new concept and technology gaining momentum in the market, its conceptual roots point to the operating system Multics operating system (1965), which aimed at providing utility computing. Many other Grid-like projects had been going on during 1970's-80's, but the major breakthrough came in 1990's, when the meta-computing concept gained its momentum. In 1995, two meta-computing projects – FAFNER (Factory via Network-Enabled Recursion) and IWAY (Information Wide Area Year) were undertaken which have laid foundation for many key concepts used by Grid computing today. FAFNER aimed at factorizing very large

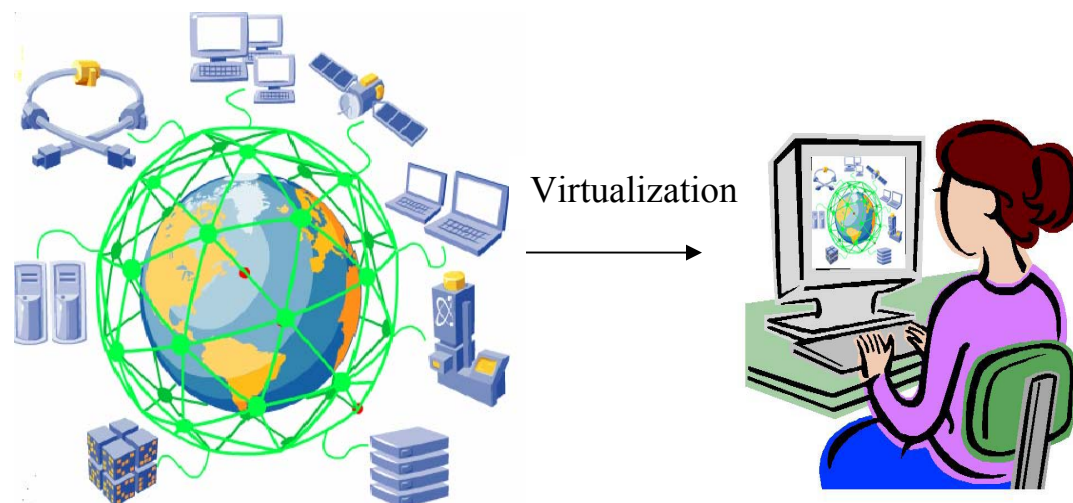
numbers, which is very relevant to the field of security while IWAY aimed at linking the super-computers using the existing network resources. IWAY techniques strongly influenced the Globus project which is now one the most active contributor to the Grid.

In 1997, in workshop at Argonne National Laboratory the official foundations of Grid computing were laid by a publication - "Building a computational Grid", which was followed in 1998 by the publication of "The Grid: Blueprint for a New Computing Infrastructure" (Grid Bible) by Ian Foster of Argonne National Laboratory and Carl Kesselman of the University of Southern California. Ian Foster previously involved in the I-WAY project, along with Kesselman, published a paper in 1997 called "Globus: a Metacomputing Infrastructure Toolkit", clearly linking the Globus Toolkit with meta-computing, which is at the heart of the Grid. Since then, a number of grid computing projects, its implementations and standards for future usability and compatibility are under construction.

## 4. Grid Computing as a Technology

Let us understand what grid technology is and what it should do so that it is worth inculcating into current trend. Currently we have World Wide Web service that enables us to share information and a single pc does all the computing. But Grid computing takes a step further. Computers and devices connected to grid, not only share the information but also the computing power and resources like databases, storage disks, software applications, CPUs, etc. Where a complex computing program would take hours to compute on a single powerful computer, Grid computing would finish the task in minutes by making several collaborative computing resources available dividing the task internally and getting the job done faster and more efficiently.

It models a virtual computer architecture that consists of many networked computers that distribute the computing tasks to perform them parallel. The larger tasks are broken down into smaller ones and allocated resources, using sharing proper policies, to execute them in parallel. The resource allocation policies are Service Level Agreements (SLAs) in which provision of certain level of service is agreed.



Grid computing has combination of different technologies, working on different platforms, jointly providing different solutions to the people. A grid involves virtualization – workload virtualization, information virtualization, system virtualization, storage virtualization, processing virtualization and many more to come. A grid can also be considered to be an evolution of clustering technology with

the ability to interconnect heterogeneous computing clusters. Thus one can understand that no single technology, platform or resource constitutes a grid, but it is the method by which all these different sets of resources are accessed and combined. Hence it is the process of using infinite remote resources using a single pc and in a way that the end user is least aware of it.

Grids can be classified into the following categories: -

*Based on scalability: -*

- Departmental Grid: To solve the problems for a specific group of people in an organization where resources are not shared. Such as Cluster Grid (term used by Sun Microsystems) and Infra Grid (term used by IBM)
- Enterprise Grid: Service to all users where resources are spread across enterprise. Such as Intra Grid (term used by IBM) and Campus Grid (term used by Sun Microsystems)
- Global Grid: Established to facilitate business or purchased in parts or as a whole from the service provider. Global grid (by Sun Microsystems) and Inter grid (by IBM)

*Based on functionality: -*

- Computational Grid: Provide access to the computational resources. Such as Desktop Grid, Server Grid, on-demand computing, collaborative, etc.
- Data/Storage Grid: Optimized for the data oriented operations such as data mining, sharing etc. Semantic Grid is one of the data grids.
- Utility Grid: Managed and maintained by the service provider as a commercial compute resource.
- Equipment Grid – used to share the data from equipment/sensors connected to the Grid. The computers connected to Grid can also control the equipment remotely.
- Scavenging Grid – used to harness the un-used CPU cycles on idle processors for solving complex problems using existing resources.

## **4.1 Making the Grid work**

We shall need some important concepts working in order to make the Grid computing working at any place. These areas should cover major aspects to which the developers need to work in enabling grid computing to work. Ideally coming up with all the key-points would be impossible considering the diversity of grid computing application. But we can jot down major techniques that would be employed in say 90% of the grid applications leaving 10% for the future researchers. These would be Resource sharing, Resource management, Quality of service, Security and Standardization.

### **4.1.1. Resource Sharing**

The main advantage of a computing grid is the de-centralization of control on resources. However, resources are owned by various organizations and people and hence are subject to different administrative, access control and security policies. Thus, the balancing task of providing the resources to the consumers while respecting the policy requirements of the resource owners is a major challenge.

Due to the involvement of a large number of parties, the resource sharing policies and user relationships implicated are much more complex as compared to the traditional distributed computing environment.

The sharing of resources is highly controlled and the terms are well defined – what can be shared, who can share, conditions under which sharing is enabled. Thus, the resulting

relationships consist of several components and each component has a hierarchical structure such as: -

Virtual Organizations → Organizational Unit → Projects → Users  
Grid → Site → Node  
Year → Month → Week → Day

These complicated set of sharing policies can be described using a sharing policy model (e.g.: fairshare policy tree model). In order to enable a large number of resources to be shared on the Grid, strong partnerships must be established among the various organizations involved. A collaborative effort towards building a grid of resources which can solve a problem of any level of complexity is needed.

#### *4.1.2. Resource Management*

Resource management refers to managing the pool of grid resources and optimally allocating them upon request from other entities on the grid. These resources may be storage capacity, CPU cycles, a piece of software, bandwidth, data or some equipment such as a sensor connected to the Grid. Mismanagement would really mean defeating the purpose of this technology. Consider a scenario where one allocates a Linux operating system as resource to service some windows-based application, which may put the application in doldrums. The challenges involved in resource management are owing to: -

- The grid resources are heterogeneous in nature
- The resources are distributed over a large number of geographical locations
- There is no centralized controlling authority for the resources
- The resources are dynamic in nature

Grid Resource Management involves the following key activities: -

**Resource Description** – Unlike in traditional distributed systems, Grid resources are extremely disparate, from the very functionality they offer to the software platforms being used. Hence, there must be means to describe the available resources in some formal, non-ambiguous manner. This is achieved using resource description languages such as RSL, ClassAds or using ontology based methods. However, today, no universal scheme for resource description exists and the emerging grid standards are striving towards formalizing this aspect.

**Resource Advertisement** – Advertisement of the resources by resource owners is necessary to advertise the attributes, cost and constraints of the resource along with the sharing policy to which it is subject to. Also, it is required to dynamically reflect the current state of the resource (e.g.: availability, percentage utilization). A directory service (e.g.: MDS) can be used to maintain this dynamic information with standardized protocols to access the directory information, registration of resources and notification of resource availability.

**Resource Discovery** – It involves the following steps: -

- Determining the set of resources which the user is authorized to access.
- Determining the requirements of the user for the particular task (e.g.: specific operating system, memory requirement, bandwidth requirement).
- Determining the set of authorized resources which meet the user's requirements

Various approaches exist for discovery of resources such as SD-RT (Shortest Distance-Routing Transferring) algorithm, request forwarding schemes (e.g.: best-neighbor, random, empirical), SOG based methods and UDDI (in the case of Web Services based Grid).

**Resource Selection** – The following steps are involved for selecting a single resource from the pool of filtered resources: -

1. Gather dynamic information about the resources.
2. Selecting a single resource that best matches the user criteria. Semantic web technologies such as rules, RDF and ontologies are used to improve the matchmaking.
3. SLA Negotiation – Service Level Agreement (SLA) is a contract between a resource provider and a user to provide some measurable output or a contract to perform a specified job. There are three types of SLAs: -
  - Resource SLA (RSLA) – commitment to provide a resource
  - Binding SLA (BSLA) – commitment to extend requirements after task submission or during task execution
  - Task SLA (TSLA) – commitment to perform a task as per specified requirements

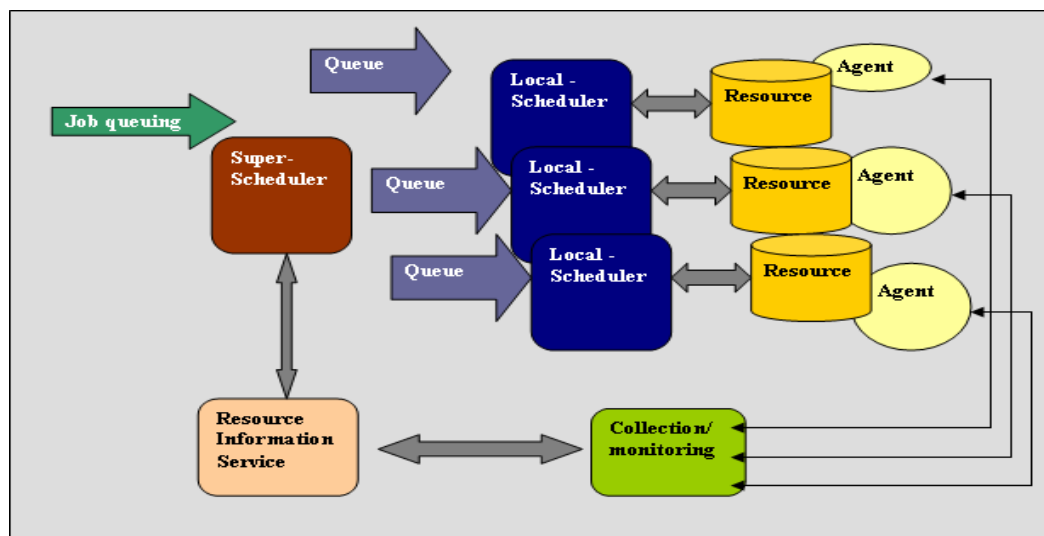
RSLA only promises resource availability, BSLA promises resource utilization and TSLA, in addition to these two, also promises performance of new tasks. The appropriate SLA for executing the grid service is negotiated.

**Resource Claim** - Claiming the resource means establishing a working relationship between the requester and the resource.

#### 4.1.3. Quality of Service (distance no barrier!!)

The amount of computation, Grid provides should not be an obstruction for quality of service the grid needs to provide to the end-user. The ancestor technologies of Grid computing failed to provide QoS along with resourceful computing. The developers of the Grid at middleware layer should take care of QoS features like response time, resource availability, job scheduling and many others.

**Job scheduling** - Job scheduling is one of most important factor on which QoS would depend and hence considering the area in which grid computing is to be deployed, appropriate strategies should be defined. We can although think of a generic architecture



which can be customized to make a perfect fit for the requirement. A generic scheduling architecture is as shown : -

The Super scheduler is also referred to as the *Grid scheduler*. The main difference between the grid scheduler and the local schedulers is that the grid scheduler does not own the resource but coordinates the execution of the job carried out by each of the local schedulers. It has to schedule the job by evaluating the available resources as some resources may not be able to allocate the required amount of memory or processors for the job or the queue of some resources is long enough to cause unacceptable execution delays. It may have to reschedule a job in case some local scheduler revokes the execution. The grid scheduler may consist of multiple layers to achieve this functionality with the local schedulers of each resource being at the lowest level of hierarchy. This architecture accommodates the heterogeneity of the resources since the higher level of schedulers are generic and transparent to the specific resource type.

#### 4.1.4. Security

The Grid comprises of a large number of resources owned by different parties which are utilized by a large number of users. Furthermore, a single user job might involve the interaction of resources owned across different parties which further complicate the security requirements. The major requirements for the security infrastructure of the Grid are: -

*Access Control* – This requirement determines the set of resources which a user with a particular permission level can access. Since the Grid is an aggregation of individual domains, it must be possible to control the access across administrative domains while respecting the local security policies in a fine-grained manner. Also, the resource owners must be able to revoke access to users in a timely manner.

*Uniform authentication infrastructure* – There must be a common method of authentication across administrative domains. Hence, a standard for encoding of user credentials must be adopted. Also, the Grid consists of multiple CAs and there must be a well-defined policy adopted by every CA for signing the security certificates and to certify the credentials across CAs. A non-repudiation framework must also be in place.

*Protection of user credentials and data* – User credentials such as private keys and passwords must be protected. Users may also require confidentiality of data stored on the Grid which might implies encrypted storage.

*Single sign-on* – The Grid provides a single-system illusion to the end-user and in order to create this the user must be required to provide his credentials only once. The middleware layer should then handle the task of using this information to log on to multiple servers while protecting the user credentials. Also, delegation of the user credentials and access rights to the server or an agent is essential whenever the user needs to access resource outside of his security domain.

*Secure group communication* – A computation can involve several processes running across domains. Also, the active processes can change dynamically as per the requirements of the computational task. Hence, it is essential to adopt a security policy which supports secure communication for dynamic groups.

*Group membership* – A user might reserve or request resources on behalf of multiple entities which can claim and use the resource. A user must be able to define his groups and resource reservation tickets must be transferable to entities belonging to the group.

*Trusted Computing* – Trust relationship, which is based on past experiences, might vary from “very trustworthy” to “very untrustworthy” and can vary dynamically with time. Users, service providers and Grid administrators must be able to specify trusted Grid domains or trust alliances which favor a subset of trusted hosts on the Grid.

*Task Administration* – Grid administrators need to monitor the grid for terminating unmanageable jobs. However, since the jobs span across multiple administrative domains, a co-coordinated effort is required in order to achieve this. Also, proper audits must be maintained to track the owners of these jobs and notify them.

*Auditing* – Grid administrators must be able to monitor the grid activities and log the activities for audit purposes. The logs must uniquely identify the job owner, time, participating resources and must be flawless. Also, stakeholders must be able to access the logs belonging only to them.

*Generic security policy* – The security policy should be generic and should not specify any implementation method in particular. The implementation should be possible with different security technologies.

#### *4.1.5. Communication Networks*

In order to build a grid, it is not sufficient to have powerful computers alone. The interconnecting networks must be fast enough to transfer huge amounts of data across the nodes with minimum propagation delay between the different processors involved in a task. The rapid improvement in the speed of communication networks has enabled a task to be efficiently executed on multiple processors that are physically apart and has thus created a paradigm shift from distributed computing to Grid computing. The penetration of optical fibers has led to the evolution of ultra high speed and low latency networks. Wireless connectivity to the grid will enable smaller devices such as mobiles, PDAs and sensors to be integrated into the grid.

Local Grids rely on LAN and SAN technologies. The key physical layer technologies used are Fast and Gigabit Ethernet and Fiber Channel (FC) and the network layer technologies are TCP/IP, VI, SCSI FCP, CIFS and NFS. Together with the powerful processors, these networking technologies have led to the emergence of high-performance clusters. These local grids are interconnected to form the Intra Grids using WAN technologies such as Frame Relay MultiLink Frame Service (MLF) and Multi Protocol Label Switching (MPLS). Since the grid consists of a variety of heterogeneous networks, a global grid necessitates some means of accessing the grid services transparently is required. Here, using the Internet and VPNs seems to offer the best choice.

#### *4.1.6. Standardization*

This aspect would be challenging, since it has been observed that not many organizations are hesitant in discussing their technology and implementations details. Since Grid computing involves sharing of heterogeneous resources and making them work together for providing a single solution to the complex problem, it becomes very much essential for standardization of the hardware and software access required at any layer. It is in the interest of all to pursue a common open standard that can be used by all for deploying Grids. Establishing and implementation of certain guidelines needs to be done that must be followed while developing any grid application. The critical question would be who would take this responsibility and will everybody agree to it?

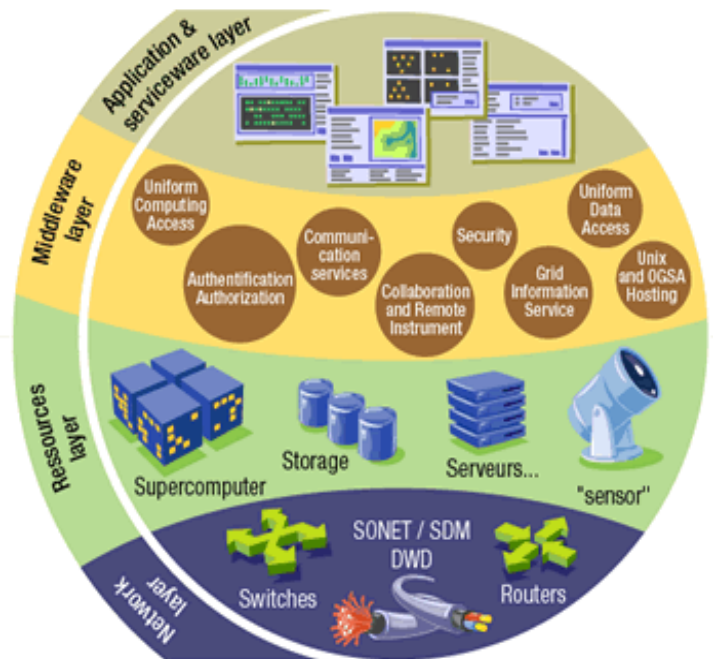
Grid specific standards are currently established by Global Grid Forum, a standards body that has more than 5000 individual researchers and practitioners working to make standards for grid computing. Global Grid Forum has been a significant force in standardization of Grid computing, with Open Grid Service Architecture (OSGA) being the most recent project – seen as the key reference for future grid developers. Many grid projects currently have adopted another standard tool – Globus Toolkit, which comprises of set of software tools to implement basic services and capabilities required to construct a computational grid. The open Grid Service Architecture will integrate the web services functionality into grid protocols. OGSAs has a set of technical specifications that defines the common framework that will allow business to build grids both across the enterprise and business partners. OGSAs comprises of grid services, web services and toolkit.

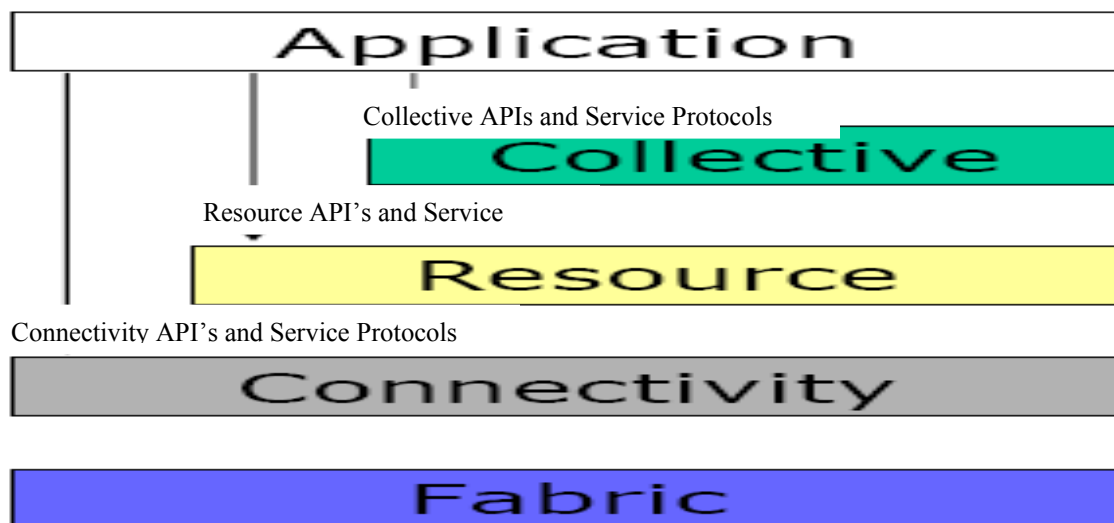
These concepts form the heart for grid computing technology. However, merely defining and understanding how each works is not enough. We shall need to define at what level the key area is functional and hence a need to come up with common vocabulary for defining these levels arises. This would call for developing a layered grid architecture, where each layer would service the above layer (taking analogy from OSI reference model). We need to define protocols and API's at each level for facilitating above services that make portable applications.

## 5. Layered Grid Architecture

Grid Architecture is composed of layers – the lower layers provide the hardware and network interfaces whereas the upper layers provide application services to the user. The bottommost layer is the **Network Layer** which connects the different resources on the Grid. It provides low-level hardware and network interfaces to the upper layers for communication. On top of this is the **Resource Layer** which consists of the actual resources that are shared on the Grid. Thus, the processors and clusters running different operating systems, databases and storage systems, sensors which can be directly connected to the Grid (through the network layer), all form the resource layer of the Grid.

The Network and the Resource Layer together form the physical infrastructure of the Grid and are commonly referred to as the **Fabric layer**. The various resources on the Grid are distributed across various geographical locations. Also, these resources are heterogeneous in nature. The next layer, which is the **Middleware Layer**, is the glue which connects these disparate resources together to create a single-system illusion for the user. The user applications form the topmost layer of the Grid – the **Application Layer**. This is the layer which is visible to the end user. Following figure displays the interaction amongst the layers while implemented in grid environment.





The applications needs to access the resources using specific APIs and protocols to access. The collective APIs and protocols are used to perform the aggregation of resources and the resource and connectivity protocols handle all "Grid specific" network transactions between the different resources on the Grid

## 6. Current State Grid Architecture

The fabric layer, considered the back-bone of grid computing provides with High performance networks in order for different resources on the grid to communicate. One can classify them on the basis of the coverage area (local, national or global) or on the basis of their data throughput (Mbps, Gbps, Tbps). The idea is to have nodes that possess high processing in terms of teraflops with terabytes of memory and thousands of terabytes of disk space, on the grid and make them work towards a complex problem. The HP 9000 SuperDome has main memory of 1TB with a clock cycle of 1 GHz and can process data at the rate of 512 Gflops/sec. The current wired hardware is almost ready to provide such demands but synchronizing them is a question; although one can leave this job on the middleware or the serviceware layer.

The middleware layer holds the key to success of the grid computing idea. The middleware layer can be further partitioned into: -

- Security Infrastructure Layer – It is concerned with authorization, authentication and providing secure connection by implementing message and transport level security.
- Core Grid Middleware Layer – The fabric is composed of heterogeneous, partitioned and distributed resources. This layer makes the upper layers transparent to these complexities. It also takes care of load balancing across the various resources.
- User-level Middleware Layer – It consists of resource managers such as schedulers and resource brokers which aggregate the resources to provide a single interface.

The basic aim for middleware is to automate the machine to machine transactions to interlace the computing resources and provide a uniform solution for the computational demands on the grid. The middleware layer can be termed as the “brain” of grid computing and is the key factor in making grid computing possible. The specific implementation of the middleware layer will depend on the type of grid it is being deployed but one can think of it as being run by intelligent programs called **Agents** that trigger by themselves to perform various critical operations. One can also think of it as **Brokers** bargaining the resources with the resource manager or provider on behalf of the end-users. A Broker shall be involved in basic machine negotiations, gaining access to resources

after security checks and perform computational activities requested by the user. On other hand, an intelligent agent, responsible for network management, might schedule itself to run periodically and optimize the network conditions and monitor quality of service provided. Thus, both agents and brokers, together facilitate the efficient use of resources on the grid.

The application layer consisting of user applications again is not easy task. All the current applications that are stand alone pc applications will have to be adapted to run on the grid environment. This means either modifying the existing applications or building them again from scratch depending on what is easier and affordable.

This layer can be further partitioned into: -

- Programming Environment and Tools Layer – which includes tools, portals, libraries, compilers, etc. needed to run the applications.
- Grid Applications Layer – This is comprised of the user applications themselves (financial, engineering, science, business).
- Service-ware – This includes software to perform maintenance activities such as account management (resource usage, who is providing the resources, who is consuming the resources) and billing.

In the current scenario, commercial organizations have started developing the tools and applications that run on the grid, although the ideal grid infrastructure may not be available but there exist commercial applications that run on the test-beds. T

The middleware and applications have to be gridified and this is easier said than done. Typical middleware software would be made up of thousands of software programs developed by hundreds of software engineers. The Sun Grid Engine – a production grid developed by Sun Microsystems, involves 500000 (half a million) lines of code developed by around a thousand of software experts. The European data grid project had some 300000 lines of code written by some 150 software engineers. Hence there are no prizes for guessing that developing grid applications and making them work is not only complex but very difficult. Efforts are being made to make the implementation easier and affordable at commercial level and a number of tools are being developed to achieve the same.

## **7. Grid Tools**

The complexity in deploying grids, has lead to production of specialized tools that can ease the development and the management process of grid applications. Let us now look at some commercially available tools that are currently or may in future affect the development of the applications.

### ***7.1 G-Eclipse***

G-eclipse addresses the top-layer of the grid architecture and to an extent the middleware layer applications. G-Eclipse is based on the Eclipse Framework and intends to ease the process of grid applications by classifying actors of the system into three broad categories – grid application users, grid operators and grid application developers.

Apart from this g-eclipse classifies the users into two groups. First, who may directly come into the UI contact of g-eclipse, thereby do not worry about the complexity of other core features. Other users might more be interested in the core features and use G-eclipse as

framework in a sense of API tool for developing their applications. It is a GUI-framework that can be used by the grid users depending on their requirement.

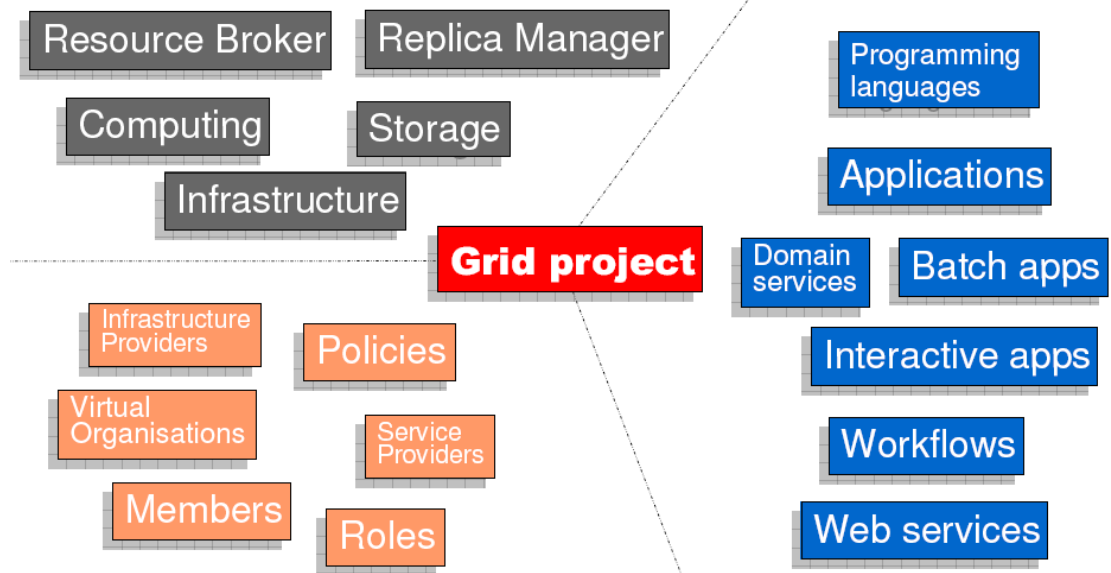


Figure – G-Eclipse tool as a whole – source [www.geclipse.org](http://www.geclipse.org) (g-eclipse summit)

The above figure displays various functions intended in geclipse grid project. It would provide services to three users - with blue colored being services utilized by the developers, green application user and the remaining by the grid operator.

Currently the g-eclipse would support G-lite middleware and related Virtual Organization management and security mechanisms, although it is intended to support variety of middlewares in future. Looking into the current architecture, g-eclipse has the Eclipse IDE as the current foundation layer on which other services are laid upon. Above this, Grid resource and management layer is laid which is mainly responsible for jobs management and resource allocation and consists of interfaces like – ContentChangeNotifier, ResourceManager, JobManager in a tree form using concept of specialization, making a resource tree. The security mechanism functions would have to be cleared prior to using these services. Authentication token is intended to be used as a security mechanism – which can be presented at every level where access control is active. The top layer would be Virtual organization management layer and the middleware support implementation layer. The User Interfaces provide access to virtual organization management and the middleware management. Geclipse gives UIs for accessing the framework and hence the grid resources in following formats – Views, Editors, Perspectives, Wizards and dialogs and Preference pages.

A *view* would give user the certain type of information and hence the possibility to work with contents in the view. Job view, authentication view, resource view, etc... are some of the view to be named.

Similarly an *editor* would allow the user to edit any textual or conceptual element in the grid. So a job editor would allow user to edit the job settings, executable input and output files etc...

A *perspective* is the collection of views and editors that are laid out in predefined way set by the user. This can be useful with different users customizing different perspectives for themselves, making it easy for them to use the tool.

*Wizards and dialogs* are standard forms of GUI tools and are no different in Geclipse. A user can use wizard and dialogs to create new jobs or a project wizard while starting a new project; instead of making sure the sequence of steps he followed every time, wizard would just remove that need for user to memorize.

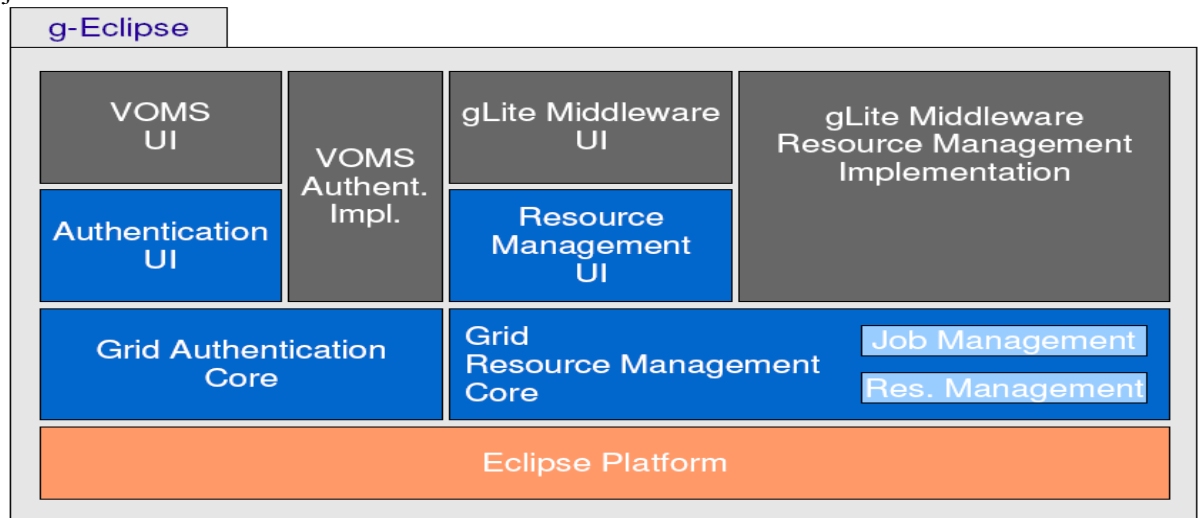


Figure Grid Architecture – source [www.geclipse.org](http://www.geclipse.org)

*Preference pages* are the global and initial settings for the g-eclipse framework and are generally loaded from the disk. But the user may want to customize them according his needs – for instance setting up a Virtual Organization(VO), host and port address might be needed that can be set into VO preference page.

The Eclipse community has large amount of tools and frameworks and G-eclipse would utilize these existing tools and integrate in G-eclipse and provide a powerful tool to the grid users that would reduce the complexity and the development life cycle of applications.

## 7.2 Pyglobus

Pyglobus is another tool in the market that aims to ease the development and application process of Grid computing. It currently provides a Python base high-level interface to the Grid services provided by the Globus Toolkit. This python based toolkit supports rapid application development process using modern object oriented software engineering techniques. Since Python is well suited for tying different technologies together, platform independent and has an excellent performance, it becomes an ideal choice as a language tool for providing high level interface to the grid toolkits.

The native Globus code is in C and Python has native extension modules that will cleanly interface with the C code. Also it would have wrapper functions that would take care of mapping between the two styles and remove the complexity of Grid programming by using abstraction, encapsulation and polymorphism techniques.

It would interface different services in form of API packages that would be used by the grid developers. Currently plan is to support basic services and hence packages like - Exceptions, Security, Resource Allocation, Secure I/O, Grid FTP, etc... are support. In further developments, it plans to support packages of Web services by providing a Web-kit servlet engine.

## 7.3 MOAB Grid Suite 4.5

MOAB is a grid workload management product that integrates job scheduling and management, monitoring and reporting of work load across resource clusters. It offers grid

optimized data staging, job submission and management and hence facilitates easy migration of existing distributed applications to the Grid. It can work with existing cluster security mechanisms and also seamlessly integrate with grid security mechanisms such as the Globus Toolkit. It provides fine-grained control over ownership-based access, priority and service levels to address the policy issues of the Grid and extensive control over resources and their workloads to maximize the grid utilization benefits.

The Moab Grid Suite incorporates the following: -

Moab Workload Manager for Grids – Used for policy-based workload management and scheduling

Moab Grid Manager – Used for grid and cluster level administration interface, monitoring and reporting

Moab Access Portal for Grids – Used for Web-based end user job submission and management

Some of the salient features of this solution are: -

- Intelligent job scheduling that matches job requests to the most suitable resources.
- Optimized data staging to minimize resource blocking
- Optimized scheduling decisions based on historical workload results
- Allows local optimization of grid workload
- Guarantee job prioritization with flexible grid policies that respect local policies and support grid service level agreements
- Resource reservation to ensure high resource availability
- Global view of all grid operations for grid self-diagnostics, planning, reporting and grid-wide and per cluster accounting
- Built-in usage controls, accounting and graphical usage reporting
- Establish resource ownership and enforce priority access with owner-based prioritization, preemption, and access guarantees

There are many tools coming up since it is a fact that grid applications are going to be need of the coming future and hence one shall need to develop grid applications very frequently and rapidly. By developing tools that can speed up the development process of grid applications, ease their management and make applications user friendly, grid computing task can be easily accomplished, at least at the software level, which is the key to its successful deployment. Also global standards have to be set so that when these tools and applications hit the market, there is no compatibility issue lingering and inter grid-application communications is possible.

## 8. Current Grid Standards

Grid computing potential has been realized by many industries and hence commercial involvement is increasing as the time goes off. This has impacted both grid architecture and its associated protocol standards. Grid developers need to conform to various standards to enable their applications adopted widely by the grid community. But infighting among the commercial organizations may not permit the continuation of the standards that are currently used and thereby hinder the growth of Grid computing. At present, **Global Grid Forum (GGF)** is the primary institution that is authorized to set the standards for the Grid. But it is not alone and is accompanied by many other organizations like OASIS, WWWC, Peer to Peer Working Group, Liberty Alliance, etc...

## **8.1 Global Grid Forum**

The GGF is involved developing standards and best practices for grid computing. It was formed in 1998 from the merger of the Grid Forum in North America, the Asia-Pacific Grid community, and the European Grid Forum (eGrid).

GGF creates four types of documents that provide information to the Grid community, similar thing to Internet Standards:

- *Informational*—a useful idea or set of ideas;
- *Experimental*—useful experiments;
- *Community practice*—common practices or processes that influence the community; and
- *Recommendations*—specifications, which are analogous to Internet standards-track documents.

Currently GGF has basic guidelines among seven areas—including, for example, architecture, data, and security—within which numerous working and research groups operate. Within the data area, standards under development include data access and integration services, Grid file systems, Grid FTP, grid storage, and data replication. Nearly 30 research groups and 5000 researchers investigate longer-term issues to develop specifications.

Joining a GGF working group involves simply subscribing to its e-mail list. The project members, meeting agendas, and work progress are all posted online.

## **8.2 OASIS**

OASIS is a not-for-profit organization that promotes industry standards for e-business, founded in 1993 as SGML Open and then changed its name in 1998 to reflect its expanded technical scope. This includes developing standards such as those related to the Extensible Markup Language (XML) and the universal description, discovery, and integration (UDDI) service. OASIS produces Web services standards that focus primarily on higher-level functionality such as security, authentication, registries, business process execution, and reliable messaging.

Participants in OASIS can be either unaffiliated individuals or member-company employees. At least three organizations must implement a standard before OASIS will approve it.

OASIS is involved actively in forming web service related standard in grid computing and active promoter of interfacing WSRF operations set with the grid. OASIS is member of W3C group.

## **8.3 Distributed Management Task Force**

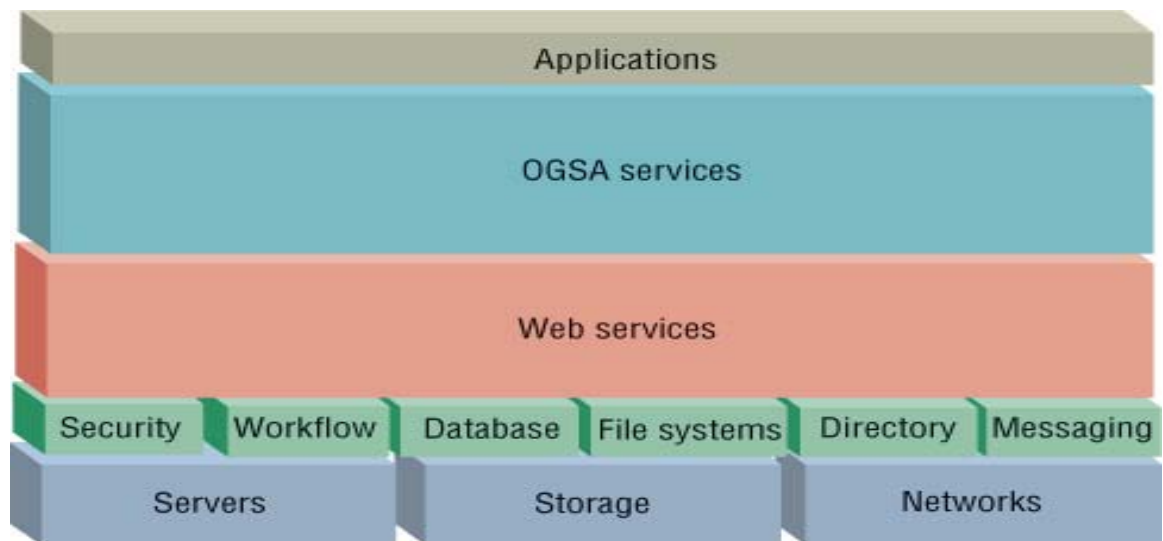
The DMTF founded in 1992, is industry-based standardization body to develop management standards and integration technologies for enterprise and Internet environments. Its technologies include the Common Information Model and Web-Based Enterprise Management. The DMTF formed an alliance with the GGF in 2003 for the purpose of building a unified approach to the provisioning, sharing, and management of Grid resources and technologies. It is actively spreading awareness of benefits of grid computing amongst the commercial organization and a strong promoter of grid deployment.

Let us have a brief idea of existing standards in the current situation that are practised followed by grid developers and operators.

## 8.4 Open Grid Services Architecture (OSGA)

The most important Grid standard to emerge recently is the Open Grid Services Architecture, which aims to define a common, standard, and open architecture for Grid-based applications. Developed by major efforts of GGF and IBM, OSGA describes architecture for a service-oriented grid computing environment for business and scientific use. OSGA is a distributed interaction and computing architecture based around services, assuring interoperability on heterogeneous systems so that different types of resources can communicate and share information. OSGA is equal to grid service plus web service and toolkit.

As figure shows, it aims to define all the fundamental services that an e-business or e-science application would use such as job and resource management, communications, and security, leaving various working groups within the GGF and other Grid-standards organizations to specify the services' interfaces, semantics, protocols, and other technical details.



Because the Grid is a dynamic environment in which service instances can come and go during task dispatching, resource configuration and provisioning, and system state changes, OSGA provides interfaces for lifecycle service management. It also supports state data associated with Grid services, an approach conceptually similar to traditional object-oriented programming environments. In addition, OSGA includes a callback operation in which clients can register interest in a service and receive notification of any change in that service.

## 8.5 Open Grid Services Infrastructure (OSGI)

Open Grid Services Infrastructure (OSGI) was published by the Global Grid Forum (GGF) as a proposed recommendation in June 2003. OSGI v1.0 defined a set of principles and extensions for using WSDL and XML Schema to enable stateful Web service. It was intended to provide an infrastructure layer for the Open Grid Services Architecture (OSGA), OSGI takes the statelessness issues (along with others) into account by essentially extending Web services to accommodate grid computing resources that are both transient and stateful. OSGI was based on the concept of *Grid services*, enhanced Web services that provided a standard set of mechanisms to manage state.

But OSGI is not that popularly accepted by the users. Many thought it was too large for one specification and was not a pure subset of Web services, as it required a modification to standard WSDL, called Grid WSDL. Finally, even though many other Web services systems have object-oriented implementations, some viewed OSGI as too object oriented. To support transient, potentially short-lived instances, OSGI used OO concepts such as statefulness and the factory pattern to create Grid service instances.

## **8.6 Web Services Resource Framework**

Hewlett-Packard, IBM, Fujitsu, and the Globus Alliance announced the WS-Resource Framework in January 2004. WSRF contains a set of specifications for expressing the relationship between stateful resources and Web services. The specifications define specific message exchange formats and related XML definitions. WSRF provides a set of operations that web services may implement to become stateful; web service clients communicate with *resource* services which allow data to be stored and retrieved.

Widespread dissatisfaction with OSGI led to a collaborative effort among architects from the Grid and Web services communities to define an alternative infrastructure based on unadulterated Web services specifications. After revising and updating the WSRF specifications based on industry feedback, a development team submitted the final results to two new OASIS technical committees, the WS-Resource Framework (WSRF) TC and the WS-Notification (WSN) TC.

The WSRF TC was formed to standardize four specifications:

- WS-ResourceLifetime—describes how to manage the lifetime of a resource and specifies Web services operations used to destroy a WS-Resource;
  - WS-ResourceProperties—defines how to query and modify WS-Resources described by XML Resource Property documents;
  - WS-ServiceGroup—describes how to represent and manage collections of Web services and/or WS-Resources; and
  - WS-BaseFaults—defines a base fault XML type for use when returning faults in a Web services message exchange.
- The WSN TC was created to standardize three other specifications defining Web services interfaces:
- WS-BaseNotification—handles asynchronous notification, including interfaces used by a notification producer or consumer;
  - WS-BrokeredNotification—handles asynchronous notification; and
  - WS-Topics—organizes and categorizes items of interest for subscription, known as topics.

## **9. Benefits of Grid Computing**

There are no two opinions that Grid computing has lot of potential and it is only in the coming time we shall be able to judge how cost effective this technology can prove. Although, at present grid computing is in its early stages, organizations have started implementing it and are reaping cost benefits from it.

### **9.1 Business benefits**

- Accelerate time to results and speed time to market
- Enable resource sharing across multiple departments or organizations
- Create a flexible and dynamic infrastructure

- Scale efficiently
- Increased productivity and collaboration
- Quickly adapt to changing business needs.
- Low initial capital investment required
- New market opportunity for computing resources as commodities

## ***9.2 Technology benefits***

- Consolidate workload management
- Provide capacity for high-demand applications
- Resilient, highly available infrastructure
- Balance workloads
- Enable recovery and failure

One of the success stories of grid applications, can be taken from, the very first grid application in banking sector – Deutsche Bank’s Security Custody Reporting System. The large scale financial grid application gave substantially lower cost and a very high return on investment. In 1996, the Bank needed to expand the reporting capabilities for its Global Custody business, which included account holdings and activity for over \$50 billion in securities. The older application was inefficient, without any User Interface and dos-based. Bank intended to replace this inefficient system with some reliable, scalable and Y2K compliant. The Base One International Corporation was contracted with this responsibility and it turned up with deploying a grid based solution. **The application was developed at ¼ of the cost and it resulted in annual savings of \$5 million, running in from last 5 years, 24 hours/day continuously without any break down or crash.** Deutsche Bank’s Securities Custody Reporting System became far more reliable and robust than their previous system thereby increasing the organization’s reputation in the market.

We are sure there would be many success stories piling up for grid computing with the potential that it promises.

## **10. Conclusion**

Grid computing is one of emerging technologies which promises tremendous potential. The various application areas such as astronomy, medicine, nuclear physics, etc. which stand short of computing resources will greatly benefit from Grid computing. Individual users will be able to develop applications without requiring huge capital investment. The technology will open up new market opportunities in the form of computing resources such as storage capacity, processing power, data, bandwidth, and applications, all being offered as commodities for potential consumers. Thus, a successful adoption of Grid technology will create a paradigm shift in the way in which we build our applications in the future.

We would also have a word of caution to the overly enthusiastic community. Grid computing is not a magic lamp that would take any application and run it a 1000 times faster without the need for buying any more machines or software. Also not every application is suitable or enabled for running on a grid and hence care must be taken while selecting the application area of grid.

The application developers will have to start thinking innovative and imagining how the applications can exploit the grid power. One will need to make new applications, keeping grid in mind and modify the existing codes for application, if the need of switching over arises. Though technology is in its early stages, but it has got good support of industry leaders like IBM, Intel, HP, Sun Microsystems and many others. There is no doubt about Grid computing being the need in the future and hence we hereby introduced reader to this emerging technology and we expect at this point, readers should have an introductory but broad impression of Grid computing.

## 11. References

### 11.1 Books

“The Grid Emerging Grid Standards” by Mark Baker of University of Portsmouth, Amy Apon of University of Arkansas, Clayton Ferner and Jeff Brown of University of North Carolina at Wilmington

“Grid computing: A practical guide to technology and applications” by Abbas Ahmar  
“Networking approach to grid computing” by Minoli Daniel

### 11.2 Web Links

Developing grid computing application part 1

<http://www-128.ibm.com/developerworks/grid/library/gr-grid1/index.html#N10081>

Developing grid computing application part2

<http://www-128.ibm.com/developerworks/grid/library/gr-grid2/index.html#N1008F>

Grid networking

[http://www.lightreading.com/document.asp?doc\\_id=33405&print=true](http://www.lightreading.com/document.asp?doc_id=33405&print=true)

New to grid computing

<http://www-128.ibm.com/developerworks/grid/newto/#intro>

Design an application for grid

<http://www-128.ibm.com/developerworks/grid/library/gr-design.html#N10048>

Introduction to grid computing

<http://support.sas.com/rnd/scalability/grid/index.html>

Miscellaneous

<http://gridcafe.web.cern.ch/gridcafe/>

<http://www-128.ibm.com/developerworks/grid/newto/>

<http://www.clusterresources.com/pages/products/moab-grid-suite.php>

<http://www.datasynapse.com/products/gridserver.asp>

<http://www.cs.wisc.edu/condor/>

<http://www.top500.org>

<http://www.cogkits.org/>

### 11.3 Papers: -

A Security Architecture for Computational Grids - Ian Foster, Carl Kesselman, Gene Tsudik, Steven Tueckel

Security Implications of Typical Grid Computing Usage Scenarios - Marty Humphrey