https://doi.org/10.48047/AFJBS.6.12.2024.6410-6423

African Journal of Biological Sciences

Journal homepage: http://www.afjbs.com

Research Paper Open Access

Implementing Grid Computing in Using the Fractal Technology

Dr. Ashish Avasthi, Dr. Manish Kumar, Dr. Shikha Khullar
Professor, Department of Computer Science
Poornima University,Jaipur, Rajasthan
Professor, Department of Computer Science
Poornima University,Jaipur, Rajasthan
Associate Professor, Department of Computer Science
Poornima University,Jaipur, Rajasthan

Article History

Volume 6, Issue 12,July 2024

Received: 02 June 2023

Accepted: 01 July 2024

Published:19 July 2024

doi: 10.48047/AFJBS.6.12.2024.6410-6423

Abstract

Grid computing can mean different things to different individuals. The grand vision is often presented as an analogy to power grids where users (or electrical appliances) get access to electricity through wall sockets with no care or consideration for where or how the electricity is actually generated. In this view of grid computing, computing becomes pervasive and individual users gain access to computing resources like processors, storage, data, applications etc. as needed with little knowledge of where those resources are located or what the underlying technologies. Grid computing makes research projects possible that formerly were unfeasible due to the physical location of vital resources.

In this paper we work on the concept of the fractal grid architecture which can be used implement and help in reducing the cost for estimating the error and also help in studying the complete grid without actually expanding it. As the grid is self-symmetrical it can also be enlarged very easily.

Keywords: Grid, pervasive, fractal, Symmetrical

1. Introduction

Grid Computing resembles the electric grid which runs across the country. The growth of the Internet, along with the availability of powerful computers and high-speed networks as low-cost commodity components, is changing the way scientists and an engineer do Computing, and is also changing how society in general manages information and information services. The use of these technology increases the sharing of the information and the resources over the different areas. These new technologies have enabled the clustering of a wide variety of geographically distributed resources, such as

Supercomputers, storage systems, data sources, instruments, and special devices and services, which can then be used as a unified resource. Furthermore, they have enabled seamless access to and interaction among these distributed resources, services, applications, and data. The new paradigm that has evolved is popularly termed as "Grid" computing. Grid computing and the utilization of the global Grid infrastructure have presented significant challenges at all levels including conceptual and implementation models, application formulation and development, programming systems, infrastructures and services, resource management, networking and security, and have led to the development of a global research community.

The Grid vision has been described as a world in which computational power (resources, services, data) is as readily available as electrical power and other utilities, in which computational services make this power available to users with differing levels of expertise in diverse areas, and in which these services can interact to perform specified tasks efficiently and securely with minimal human intervention. Driven by revolutions in science and business and fueled by exponential advances in computing, communication, and storage technologies, Grid computing is rapidly emerging as the dominant paradigm for wide area distributed computing.

Its goal is to provide a service-oriented infrastructure that leverages standardized protocols and services to enable pervasive access to, and coordinated sharing of geographically distributed hardware, software, and information resources. The Grid community and the Global Grid Forum1 are investing considerable effort in developing and deploying standard protocols and services that enable seamless and secure discovery, access to, and interactions among resources, services, and applications. This potential for seamless aggregation, integration, and interactions has also made it possible for scientists and engineers to conceive a new generation of applications that enable realistic investigation of complex scientific and engineering problems. This current vision of Grid computing certainly did not happen overnight. In what follows, we trace the evolution of Grid computing from its roots in parallel and distributed computing to its current state and emerging trends and visions.

1.1 **The Origins of the Grid**

While the concept of a "computing utility" providing "continuous operation analogous to power and telephone" can be traced back to the 1960s and the Multics Project [4], the origins of the current Grid revolution can be traced to the late 1980's and early 1990's and the tremendous amounts of research being done on parallel programming and distributed systems. Parallel computers in a variety of architectures had become commercially available, and networking hardware and software were becoming more widely deployed. To effectively program these new parallel machines, a long list of parallel programming languages and tools were being developed and evaluated [14]. This list included Linda, Concurrent Prolog, BSP, Occam, Programming Composition Notion, Fortran-D, Compositional C++, pC++, Mentat, Nexus, lightweight threads, and the Parallel Virtual Machine, to name just a few. To developers and practitioners using these new tools, it soon became obvious that computer networks would allow groups

of machines to be used together by one parallel code. NOWs (Network of Workstations) were in regular use for parallel computation. Besides just homogeneous sets of machines, it was also possible to use heterogeneous sets of machines. Indeed, networks had already given rise to the notion of distributed computing. Using whatever programming means available, work was being done on fundamental concepts such as algorithms for consensus, synchronization, and distributed termination detection. Systems such as the Distributed Computing Environment (DCE) were built to facilitate the use of groups of machines albeit in relatively static, well-defined, closed configurations2. Similarly, the **C**ommon **O**bject **R**equest **B**roker **A**rchitecture (CORBA) managed distributed systems by providing an object-oriented, client-side API that could access other objects through an Object Request Broker (ORB)3. Since different codes could be run on different machines, yet still be considered a part of the same application, it was possible to achieve a distributed end-to-end system capability, such as data ingest, processing, and visualization/post processing. This was sometimes called metacomputing [3]. Even then, the analogy between this style of computing and the electrical power grid was clear [16]: "The Metacomputer is similar to an electricity grid. When you turn on your light, you don't care where the power comes from; you just want the light to come on. The same is true for computer users. They want their job to run on the best possible machine and they really don't care how that gets done." [S. Wallach, 1992] Of course, the development of programming languages and tools that attempted to transparently harness "arbitrary" sets of machines served to highlight a host of issues and challenges. First, there was no real way to discover what machines were available. Furthermore, these new programming systems were focusing on new and novel syntax for expressing and managing the semantics of parallel and distributed computation. Once up and running, an application code had no idea of the state of its execution environment or what process or network performance it was getting, unless it did passive self-monitoring or deployed its own monitoring infrastructure. When a process, machine, or network connection failed, it was up to the user to diagnose what happened. (When you are using an experimental compiler and printf() ceases to work, you know you are in big trouble.) Needless to say, fault tolerance was virtually non-existent. 1995 was a watershed year. Under the leadership of the National Center for Supercomputing Applications, Argonne National Lab, the San Diego Supercomputing Center, and Sandia National Lab, the I-WAY (International Wide-Area Year) was hosted at Supercomputing `954 in San Diego [5]. The I-WAY sought to demonstrate the potential of distributed, virtual supercomputing by hosting over sixty applications on a national testbed [11, 12]. This testbed was cobbled together in a matter of months across many different institutions and included relatively primitive tools for the scheduling of machines for different applications and for security for their access [7]. The trials and tribulations of such an arduous demonstration paid-off since it crystallized for a much broader segment of the scientific community, what was possible and what needed to be done [15]. In early 1996, the Globus Project officially got under

way after being proposed to ARPA in November 1994. The process and communication middleware system called Nexus [9] was originally built by Argonne National Laboratory to essentially be a compiler target and provide remote service requests across heterogeneous machines for application codes written in a higher-level language. The goal of the Globus project [1] was to build a global Nexus that would provide support for resource discovery, resource composition, data access, authentication, authorization, etc.

The first Globus applications were demonstrated at Supercomputing `974. Globus was by no means alone in this arena. During this same time period, the Legion project [10] was generalizing the concepts developed for Mentat into the notion of a "global operating system". The Condor project [6] was already harvesting cycles from the growing number of desktop machines that a typical institution was now deploying. The UNICORE project (Uniform Interface to Computing Resources) [2] was started in Germany in 1997.

Backing up in time to 1995, Smarr and Catlett at the National Center for supercomputing Applications (NCSA) had constructed a cluster of SGI Power Challenge parallel computers that they called the Power Challenge Array. They envisioned a distributed metacomputer of these machines at partner sites around the country that they intended to call the SGI Power Grid. When the NSF supercomputing centers were recompeted in 1996, researchers at the consortium of NCSA, University of Illinois at Urbana-Champaign, Rice University, Indiana University, Argonne National Laboratory, and University of Illinois at Chicago decided to expand on the Power Grid concept. In their proposal to the NSF, they stated: "(Our vision) is the integration of many computational, visualization and information resources into a coherent infrastructure… We refer to the integrated resources as the `Power Grid' or simply the Grid".

2. Current Goals and Vision

This evolution of the Grid has led to the current vision of Grid computing – a vision of uniform and controlled access to computing resources, seamless global aggregation of resources enabling seamless composition of services, and leading to autonomic self-managing behaviors. This vision applies to all manner and scale of computing resources from personal digital assistants (PDAs), to enterprise Grids, to an open-ended, global-scale Grid environment, i.e., the Grid. Seamless Aggregation of Resources and Services: Early Grid computing efforts focused on aggregating geographically distributed resources spanning multiple administrative domains, and this remains to be a key goal. Aggregation included both the aggregation of capacity (e.g., clustering of individual systems to increase computational power or storage capacity) as well as the aggregation of capability (e.g., combining a specialized instrument with a large storage system and a computing cluster). Key capabilities to enable such aggregation included protocols and mechanisms to secure discovery, access to and aggregation of resources for the realization of virtual organizations, and the development of applications that can exploit such an aggregated execution environment. These goals have motivated the generalization of all Grid resources into services. A key driver for this generalization was the emergence of

Web Services as a dominant technology in the e-commerce domain. This formulation of Grid computing built on the concept of a Grid service as the fundamental abstraction, allowing Grids and Grid applications to consist of dynamically composed services. A Ubiquitous Service - Oriented Architecture : While Grid computing historically grew from the desire to do distributed, virtual supercomputing, the capabilities needed to accomplish this are actually quite fundamental with a far-reaching, broader impact. The ability to do resource and data discovery along with resource scheduling and management in a secure, scalable, open-ended environment based on well-known and widely adopted services enables a wide variety of application domains and styles of computation. These fundamental capabilities enable not only distributed supercomputing, but also internet computing, web computing, cycle harvesting, peer-to-peer computing, etc. In short, we could refer to this as a ubiquitous service-oriented architecture -- machines large and small (from wireless PDAs to big iron supercomputers) and services that they provide could be dynamically combined in a spectrum of virtual organizations according to the needs and requirements of the participants involved. Such an architecture should also be language/programming model agnostic and facilitate interoperability. Hence, rather than imposing a particular programming model, it should enable the integration of a wide variety of programming
models. For example, rather than imposing an object model, as CORBA does, it should allow a CORBA-based object-oriented system to be composed with another non-OO system, etc. By providing a common set of interoperable Grid-level services, it should be easier to produce an interoperable set of application-level services. Autonomic Behaviors: The inherent scale, heterogeneity, dynamism, and non-determinism of Grids and Grid applications have resulted in complexities that are quickly breaking current paradigms, making both the infrastructure and the applications brittle and insecure. Clearly, there is a need for a fundamental change in how Grids and Grid applications are developed and managed. This is leading researchers to consider alternative paradigms that are based on the strategies used by systems in nature to deal with complexity, dynamism, heterogeneity, and uncertainty. This emerging vision aims at realizing computing systems and applications capable of configuring, managing, interacting, optimizing, securing, and healing themselves with minimum human intervention, and has lead to a number of recent research initiatives such as Autonomic Grids, Cognitive Grids, and Semantic Grids.

While Grids have come a very long way from the efforts of several labs trying to address thorny, fundamental issues in distributed computing by building research prototypes, Grids still have a very long way to go before they are a practical, widely deployed reality. At the current time, several basic Grid tools are stabilizing and many Grid projects, including some very well funded international projects, are deploying sizeable Grids.

However, one can argue that Grids will not be a practical reality until (1) there is a core set of Grid services, with (2) sufficient reliability, that are (3) widely deployed enough to be useable. This is the current challenge for making

Grids a reality. The issue is how to make this happen.

2.1 Expanding the Scale and Scope of Deployment

A number of very large Grid projects are currently underway. Examples include the EU DataGrid project, the NSF TeraGrid project, and the Japanese NaReGI project. Many other smaller projects are currently underway, too, involving just a few institutions in a specific application domain. There are also a number of Grid-like commercial products for cycle harvesting, distributed scheduling, etc. In all these cases, however, the deployment and use of the Grid tools involved is not as easy as one would like. At this point, serious Grid deployment and use requires a group of knowledgeable, dedicated people. Hence, tools must be simpler for reliable deployment and use by non-specialists. Tools should also be configurable to the intended scope of deployment. Most Grid tools have been designed to be open ended to support the concept of an open-ended "Grid" while, in fact, they have been used in an enterprise-scale deployment. Currently we are using the two types of architectural approaches for the purpose of setting the Grid in the real sense. The first type of the architecture is the P2P Grid architecture. Peer to Peer system offers an alternative to such traditional client-server systems for a number of application domains. In P2P system, every node of the system acts as both client and server and provides part of the overall recourses/information available from the system. In pure p2p system no central coordination or central database exists and no peer has a global view of the system. The participating peers are autonomous and self-organize the system's structure. Peer will not always available but still all existing data and services should be accessible which should addressed by sufficient high replication. Premier goal of design P2P system is to support global search functionality without using central directories. Now with the p2p Grid system there are some common myths these are that when we talk about P2P system, mostly people think about illegal music sharing files over internet.

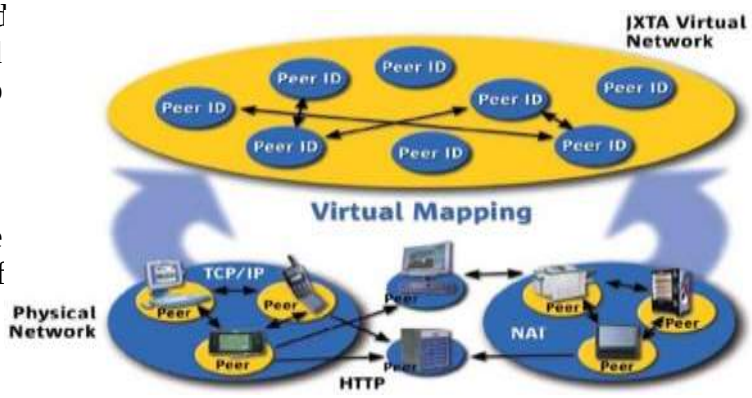

Figure 1. Showing the arrangement of the peer to peer (p2p) Grid arrangement

However, originally in 1959, P2P was developed for the defense and space projects. Today, there are many applications are available than music sharing, and in this paper we will talk about how P2P is useful in business, research, commerce, science, and many other fields. This is means that there is no such laws that is followed by the grid connection or architecture in the p2p arrangement but this not true. However, there are some advantages of the Grid Computing Peer to Peer system is provide a large source of host computer and omit the geographical recourses,

which works same as Super computer works. Now, large and time consuming projects can be done in short time period with economical benefits.

3. Types of Grid Architecture

There are two fundamental approaches to achieve this exist:

- **Unstructured**: the data is distributed randomly over the peers and constrained broadcasting mechanism is being used. Example is Gnutella.
- **Structured**: A distribute, scalable indexing structure is built up to route search requests. Examples are Free Net and P-Grid.

In system following the first approach peer can manage their data completely independently, the approach is fully decentralized, the types of search predict are not limited, and no update dependencies exist. However, these advantages are paid with high search costs in terms of messages or additional delay.

The second approach is clearly superior in terms of search efficiency, but the need to establish a distributed index usually requires some form of central coordination or global knowledge. They exhibit the advantages of the structured approach but try to maintain the beneficial characteristics of unstructured networks, such as complete decentralization and support for sub-networks.

P-Grid shares some common properties with Chord [8] and CAN [13] but requires much less tight coupling since it does not exploit global knowledge as these systems do and thus is more related to Free Net. Since we believe that only fully decentralize system will be able to fully exploit advantages in terms of scalability and self-organization we focus our interest on second class.

Now as there are advantages there are some disadvantages are also attached with it. The most common limitation with the peer to peer grid is that because of structured network, grid computing is more costly and limiting the resources. The p2p grid also slows when it come in the term of connectivity within the domain or the given range. Apart from the p2p grid another type of grid which are used in the present grid computing is the symmetrical or the iterative or self replicating grid architecture. One of the such type of grid architecture is Based One's work on distributed computing evolved over more than ten years of developing tools for building commercial business applications. It should therefore come as no surprise that Base One uses terminology like "batch jobs", dating back to early business systems. The concept of a batch job, however archaic it may seem, still has an important place in modern data processing: it is the abstraction of a logical unit of work, apart from its embodiment in a particular program running on a particular machine. As it turns out, this conventional representation of business applications in terms of discrete job steps is another characteristic that facilitates adaptation into a multi-processing environment. The basic of such type of grid architecture revolves around the model of a "virtual supercomputer", comprised of loosely-coupled "batch job servers", which asynchronously perform tasks that are specified and coordinated through *database-driven* control structures. The model is *virtual*, because it doesn't entail the actual addition of a physically separate machine. Rather, it uses the available processing power and

resources of ordinary PCs and database servers, which may already exist and continue to work in their previous roles. The result is a form of supercomputer, because it presents itself as a single, unified computational resource that can be scaled to virtually unlimited capacity and processing power.

One of the key features that distinguish Base One's architecture from other grid computing models, as well as conventional mainframes and "cluster" supercomputers, is the central role of a database in the Virtual Supercomputer. The significance of this database-driven design is that it greatly simplifies synchronizing the work of multiple processors, while providing a highly scalable solution to the problem of large-scale multi-processing.

4. Suggested Work

In this paper we emphasis and study the difference in the style of computing which can occur after the induction of the grid computing. As we all know the grid computing is the developed concept which had given the space for the cloud computing most commonly utilized by the major international players of the software market like ORACLE, Microsoft, Amazon etc. But if study the use and the concept of the grid computing in context with Indian scenario then we came to know that as information technology has became the backbone of the industry no matter it is the IT industry or the Non IT industry.

As it is acceptable by the person or the owner of the IT industry to pay large amount in setting the IT infrastructure for the proper functioning of the industry. But when it comes for the industries where the industry is not a IT industry then the expense on the IT is not very much acceptable. The use of the Grid is basically important for such type of industry. As we know that the amount of the computing varies from time to time. In the grid computing the client industry can ask the grid solution provider to provide the computational power when and where required. This help in reducing the investments on the IT infrastructure by the non IT based company. Now with use of Grid Technology it only have to order the amount of computational power which it required for the fast and proper functioning. It does not require to set the high IT infrastructure and it can also not required to maintain the computers and the other devices. When ever it want to increase the computational power it only ask on demand. The money which is saved can be utilized on the other important work. For example the India is considered one of the largest suppliers of the sugar and as we all now sugar industries only work for the six month and rest of the year they remain closed. So for such type of industries the grid computing is very important. From an enterprise perspective, the on-demand nature of grid computing helps to support the performance and capacity aspects of service-level objectives. The self-service nature of grid computing allows organizations to create elastic environments that expand and contract based on the workload and target performance parameters. And the pay-by-use nature of grid computing may take the form of equipment leases that guarantee a minimum level of service from a grid provider.

Virtualization is a key feature of this model. IT organizations have understood for years that virtualization allows them to quickly and easily create copies of

existing environments —sometimes involving multiple virtual machines — to support test, development, and staging activities. The cost of these environments is minimal because they can coexist on the same servers as production environments because they use few resources. Likewise, new applications can be developed and deployed in new virtual machines on existing servers, opened up for use on the Internet, and scaled if the application is successful in the marketplace. This lightweight deployment model has already led to a "Darwinistic" approach to business development where beta versions of software are made public and the market decides which applications deserve to be scaled and developed further or quietly retired.

Grid computing extends this trend through automation. Instead of negotiating with an IT organization for resources on which to deploy an application, a grid is a self-service proposition where a credit card can purchase compute cycles, and a Web interface or API is used to create virtual machines and establish network relationships between them. Instead of requiring a long-term contract for services with an IT organization or a service provider, clouds work on a pay-by-use, or pay-by-the-sip model where an application may exist to run a job for a few minutes or hours, or it may exist to provide services to customers on a long-term basis. Computed grids are built as if applications are temporary, and billing is based on resource consumption: CPU hours used, volumes of data moved, or gigabytes of data stored. The ability to use and pay for only the resources used shifts the risk of how much infrastructure to purchase from the organization developing the application to the grid provider. It also shifts the responsibility for architectural decisions from architect towards the grid developer. As presently many of the organization are using the ERP (Enterprise Resource Planning) to increase the production and manufacturing of the goods use of the grid is the demand of the time. Now with use of the Grid computing the industries are joint over the different geographical areas. The grid computing can also be used to send the information in the quick time across the various branches of the industry where the grid computing has be done. Use of the grid technology the industries does not require to built the custom software for the different industries of the same type. In this context a single software is developed and keeping in mind the required demand each industry is provided its login from where it can use the software. There are the various architecture in which the grid computing can be divided.

4.1 Business Architecture

A grid computing offers unprecedented control in allocating resources dynamically to meet the changing needs of a business. This is only effective when the businesses service level objectives have been clearly articulated and guide the cloud's enterprise management layer. Application performance metrics and SLAs must be carefully documented and monitored for an effective grid deployment. To maximize the distributed capabilities afforded by grids, business processes should identify areas where asynchronous or parallel processes can be used.

Key Business Architectural Principles

• Business Alignment, Cost Optimization

• Compliance with Laws and Regulations

• Business Agility

• Minimize Cost

4.2 Application Architecture

Application services should abstract resource allocation and avoid the tight binding of its resources to invokers of the service. Dependencies on static references to infrastructure (for example, storage, servers, network resources), as well as tightly coupled interfaces to dedicated systems, should be avoided.

To take advantage of the grid's scalability capabilities, applications should take advantage of distributed application design and utilize multi-threading wherever possible. Applications should leverage distributed locking, GUID generation, and integration layers to provide the greatest flexibility in deploying on grid architecture.

Key Application Architectural Principles

• Technology Independence, Adherence to Standards

• Common Development Methodology

• Loosely coupled Interfaces.

4.3 Information Architecture

The Grid computing offers the potential to utilize information anywhere in the grid architecture. This increases the complexity associated with meeting legal and regulatory requirements for sensitive information. Employing an Information Asset Management system provides the necessary controls to ensure sensitive information is protected and meets compliance requirements. This is essential when considering public or hybrid grid as information may leave the confines of the data center, which may violate certain legal and regulatory requirements for some organizations.

Key Information Architectural Principles

• Implement Information Lifecycle Management

• Regulatory and Legal Compliance

• Enforce Data Privacy.

4.4 Technology Architecture

Implementing Service Oriented Architectures (SOA) provides the most effective means of leveraging the capabilities of grid computing. SOAs distributed nature, service encapsulation; defined service level objectives, virtualized interfaces, and adherence to open standards align with Grid's architectural requirements.
Grids are highly distributed nature requires a more robust security management and infrastructure. Implementing federated identity hubs and defined communication zones are typically required for grid architecture deployments to control access across multiple cloud nodes--especially when those nodes exist outside the organization.

Grid infrastructures simplify the deployment of grid application servers which offer improved scalability and disaster recovery.

Key Technology Architectural Principles

• Control Technical Diversity

• Adherence to Standards

• Scale Capacity and availability to satisfy Business Objectives

• Virtualized dependencies to hardware and software

• Unified Security Infrastructure.

5. Limitation

Security management is major obstacle to overcome in the route of commercializing GRID infrastructures. The traditional GRID infrastructure, such as the GRID Security Infrastructure (GSI) from Globes, using the X.509 certificates as its authentication mechanism, depends on interfaces at the protocol level to provide the security infrastructure. However, this approach has concentrated on authentication and does not cover all aspects of security management. In particular there is little support for authorization management, the specification and enforcement of security policies, the treatment of cases where the (agents managing the) collaborating resources have no prior knowledge of each other (or their certifying authorities).

In the paper "Building Trust on the GRID - Trust Issues Underpinning Scalable Virtual Organizations", one way to identified the need to supplement this infrastructure by raising the level of the trust within GRID architecture. This builds upon the established literature in trust analysis, which provides a framework for analyzing how trust should be transmitted between agents in distributed systems. Especially when dealing with how to propagate trust between agents with little or no prior knowledge of each other. The basis of this infrastructure is the explicit declaration and publication of trust policies by participating resources on the GRID using an appropriate policy specification exchange language. Agents wishing to utilize resources would be able to present their credentials, policies and requirements to the participating resources. It is also an automated process would verify the credentials, possibly referring to trusted third parties, to establish identity, deduce authorization based upon supplier and consumer policies and to authorize (or revoke) access under the specified the conditions of use. To support this, one would need to add at least the following:

- Resource brokerage services to facilitate resource discovery and allocation in compliance with a contractual realization of QoS requirements.
- A means of publishing, negotiating and exchanging policy statements.
- Appropriate trust support services, such networks of trust authorities, and an infrastructure allowing for the dynamic formation of certification chains.
- A trust management framework able to cope with the complexity and uncertainty underpinning most interactions in open dynamic systems such as the GRID architectures; this will need to draw a distinction between perceived and actual security, relate trust to enterprise objectives and weigh it against transaction risk.

A policy-driven security management system, which is able to support the dynamic formation of collectives of entities which are required to share resources to achieve certain goals.

Security requirements are fundamental to the grid design. The basic security components are comprised of mechanisms for authentication, authorization, and confidentiality of communication between grid computers. Without this functionality, the integrity

and confidentiality of the data processed within the grid would be at risk. To properly secure your grid environment, there are many different tools and technologies available. This chapter examines some of those technologies.

In order to better understand grid security, it is best to start with some basic grid

security requirements and security fundamentals. Grid security builds on well-known security standards

6. Conclusions

In many ways Peer to Peer grid technology has proven that it is the best technology to work over the internet on commerce, businesses, educations, science, researches, and many other projects by eliminating the geographical and economical limitations of the resources. By using this technology now, we can finish our projects in short time and without depending on one main server or super computers. Still, we need to concentrate on security and privacy issues and data losses through the internet connections. We can develop a great Peer to Peer Grid or the iterative grid computing by paying attention to find solutions for its issues. Another very important issue regarding the unifying the grid computing and the business is the security issue. The strong security policy should be implemented very strongly.

Grid offers real alternatives to IT departments for improved flexibility and lower cost. Markets are developing for the delivery of software applications, platforms, and infrastructure as a service to IT departments over the "Grid Architecture". These services are readily accessible on a pay-per-use basis and offer great alternatives to businesses that need the flexibility to rent infrastructure on a temporary basis or to reduce capital costs. Architects in larger enterprises find that it may still be more cost effective to provide the desired services in-house in the form of "private grids" to minimize cost and maximize compatibility with internal standards and regulations. If so, there are several options for future-state systems and technical architectures that architects should consider to find the right trade-off between cost and flexibility. Using an architectural framework will help architects evaluate these trade-offs within the context of the business architecture and design a system that accomplishes the business goal.

7. Future Scope

The Grid Technology is now converted and expanded with invent of the Cloud Computing. This will help the Information Technology to cross all the barriers coming in between the growth of the industries. Now the information is easily available across the industry and they don't require the infrastructure and the computational power can be increased on the demand basis. The user does not need to bother about the place or for the infrastructure when it decides to increase the computational power. This help the user to invest the money at the base area of the industry and will always backed up with the It infrastructure. The main concern for the grid computing in India is that in the country we does not have a good frequency or the bandwidth of the internet across the country with lack of expertise and the people are pre dominantly prefer to use the old technology as they think that the older technology is more accurate, fast and

also secure. This myth of the people is proving to becoming the large hindrance in the popularization of the grid computing. On the grid computing another concept of customizing the software required for the operation in the industry can be centrally available. As the data is stored in the distributed form the possibility of the data loss is also reduced to the large level. So we advocate to the user to use the grid computing if they want the proper and correct utilization of their available resources over the time. This is of the grid computing will increased their productivity by decreasing the lead time and the response time for the any RFC. This also help them by providing the more flexibility in their product by the use of the techniques like make to stock, make to order, configure to order and engineer to order. The use of the grid computing will also help in the technique of convergence which is very useful for the industry.

8. References

[1] "The Globus Alliance", http://www.globus.org.

[2] "Unicore Forum", http://www.unicore.org.

[3] C. Catlett and L. Smarr, "Metacomputing", Communication of the ACM, 36(6), 1992, 44-52.

[4] F. J. Corbat and V. A. Vyssotsky, "Introduction and overview of the Multics system", Proc. AFIPS 1965 FJCC, 27(1), 1965, 185-196.

[5] T. DeFanti, I. Foster, M. Papka, R. Stevens and T. Kuhfuss, "Overview of the I-WAY: Wide Area Visual Supercomputing", International Journal of Supercomputing Applications and High Performance Computing, 10(2/3), 1996, 123-131.

[6] D. H. J. Epema, M. Livny, R. v. Dantzig, X. Evers and J.Pruyne, "A Worldwide Flock of Condors: Load Sharing among Workstation Clusters", Journal on Future Generations of Computer Systems, 12, 1996, 53-65.

[7] I. Foster, J. Geisler, W. Nickless, W. Smith and S.Tuecke, "Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment," in Proceedings of 5th IEEE Symposium on High Performance Distributed Computing, IEEE Computer Society Press 1996, 562-571.

[8] I. Foster and C. Kesselman, eds., The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, 1998.

[9] I. Foster, C. Kesselman and S. Tuecke, "The Nexus Task-Parallel Runtime System," in Proceedings of First International Workshop on Parallel Processing, 1994, 457-462.

[10] A. S. Grimshaw and W. A. Wulf, "The Legion Vision of a Worldwide Virtual Computer", Communications of the ACM, 40(1), 1997, 39 - 45.

[11] H. Korab and M. Brown, eds., Virtual Environments and Distributed Computing at SC`95: GII Testbed and HPC Challenge Applications on the I-WAY, ACM/IEEE, 1995.

[12] C. Lee, C. Kesselman and S. Schwab, "Near-real-time Satellite Image Processing: Metacomputing in CC++", IEEE Computer Graphics and Applications, 16(4), 1996, 79-84.

[13] I. Platform Computing, "The Politics of Grid: Organizational Politics as a Barrier to Implementing Grid Computing", 2004, http://www.platform.com/adoption/politics.

[14] D. Skillicorn and D. Talia, "Models and Languages for Parallel Computation", ACM Computing

Surveys, 30(2), 1998, 123-169.
[15] R. Stevens, P. Woodward, T. DeFanti and C. Catlett, "From the I-WAY to the National Technology Grid", Communication of the ACM, 40(11), 1997, 51-60.
[16] S. Wallach, Information Week, 1992.
[17] J. Joseph, M. Ernest, and C. Fellenstein, *Evolution of grid computing architecture and grid adoption models*, IBM Systems Journal Vol 43, No 4, 2004.
[18] M. Baker, A. Apon, C. Ferner, and J. Brown, *Emerging Grid Standards*, page. 43-50, IEEE Computer, April 2005.

9. Website References

1. http://www/106.ibm.com/developerworks/library/gr-heritage
2. http://www.bitd.clrc.ac.uk/Activity/CORAS;Presentation =PAPER;
3. http://www.bitd.clrc.ac.uk/Activity/iTRUST;Presentation=PAPER;
4. http://www.bitd.clrc.ac.uk/Activity/TRUSTCOM;Section=4761;Presentation=PAPER;
5. http://www.erathore.com/mainsite/projects/lightfootnetworks/lisim.pdf
6. http://grids.ucs.indiana.edu/ptliupages/publication/p2pgridbook.pdf
7. http://www.jeffyjeffy.com/code/p2p_is_changing_the_internet/p2p_is_changing_the_internet.html
8. http://www.jimpinto.com/writings/grid.html
9. http://www.speech.sir.com/academics/gibbs/students/busn6080-f1-2001-wmc1/gray/gray.htm
10. http://www.public.asu.edu/~mujtaba/Articles%20and%20Papers/khambattim_communities.pdf
11. http://www.pervasive.iu.edu/researchptl.html
12. www.gridcomputing.com
13. http://grid.ihep.su/indexGrC.html
14. http://www.gridcomputingplanet.com/news/article.php/3281_1465041
15. http://citeseer.ist.psu.edu/574558.html