Design and Prototype Development of Banking Grid

A Dissertation submitted to the University of Hyderabad in partial fulfillment of the Degree of

MASTER OF TECHNOLOGY

in

Information Technology

by

SURESH KUMAR V

Department of Computer and Information Sciences

School of Mathematics, Computer and Information Sciences

University of Hyderabad
(P.O.) Central University, Gachibowli
Hyderabad – 500 046
Andhra Pradesh
India

May, 2011

CERTIFICATE

This is to certify that the dissertation entitled "**Design and Prototype Development of Banking Grid**" submitted by **Suresh Kumar V** bearing Reg. No **09MCMB37** in partial fulfillment of the requirements for the award of Master of Technology in Information Technology is a bonafide work carried out by him at IDRBT under my supervision and guidance.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. V.N. Sastry
Associate Professor, IDRBT.
Supervisor

Head of the Department

Dean of the School

DECLARATION

I, **Suresh Kumar V** hereby declare that this Dissertation entitled "**Design and Prototype Development of Banking Grid** ", submitted by me under the guidance and supervision of **Dr. V.N. Sastry, Associate Professor, IDRBT**, is a bonafide work. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date: Name: **Suresh Kumar V**

Signature of the Student

Regd. No. **09MCMB37**

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. V. N. Sastry, Associate Professor, IDRBT, Hyderabad, who generously supported and guided me throughout my project, IDRBT for providing me with the infrastructure and technical support that I needed for this project. The project would not have been possible without his assistance.

I also thank Mr. B. Sambamurthy, Director, IDRBT, Prof. T. Amaranath, Dean, School of MCIS, Prof. C.Raghavendra Rao, Head of the Department (DCIS), University of Hyderabad for extending their cooperation.

The guidance of all the faculty members of IDRBT and the Department of Computer and Information Sciences, University of Hyderabad has been precious and timely. I wish to thank them for being very patient, understanding and helpful.

I would also like to thank my friends for their valuable suggestions and helpful discussions.

Suresh Kumar V,
University of Hyderabad & IDRBT,
Hyderabad.
E-mail ID: suresh.votla@gmail.com

Abstract

Grid Computing is a technology that enables seamless sharing of a heterogeneous set of resources for data intensive computing tasks based on Open Grid Services Architecture (OGSA) standards. The technology of Grid Computing enables virtualization of distributed infrastructure resources. A Grid is a group of computers, servers and storage virtualized as one large computing system across the enterprise. Grids unleash the latent power unused at any point in time, and accelerate computing intensive processes. Grid Computing uses software to divide and distribute pieces of a program to as many as several thousand computers. It frees up CPU cycles from those underutilized servers and desktop resources, making them available for usage by other networked applications.

As the business of banking grows increasingly complex, specialized customer needs are being met by specialized applications. In every bank, the portfolio of applications and technologies is expanding at a phenomenal rate. These applications are most often deployed on server machines within the bank's environment. While these servers are loaded to their peak, there is a need for computing resources and better utilization of existing resources. The bank's IT environment consists of a large number of desktop workstations at the branches. With newer desktop machines surging with higher processing power, but used to perform relatively lighter tasks, a large portion of this processing capacity is also underutilized. Thus, normally at any given point in time, a bank's IT environment includes significant redundant or idle processing capacity.

The project focuses on the study of application of grid computing and helps banks better utilize computing power latent in the bank's IT infrastructure, to meet advanced processing needs. Globus toolkit and Condor are open source middleware frameworks, for building distributed systems. This project analyses the integration of the above mentioned technologies and frameworks according to their advantages and apply them to develop a Banking Grid. We also developed a grid portal to access the resources on the grid.

TABLE OF CONTENTS

List of Figures

CHAPTER 1
INTRODUCTION

1.1 Grid Computing:

The concept of coupling geographically distributed resources for solving large-scale problems is becoming increasingly popular; forming what is popularly called grid computing. Grid computing is becoming more and more important in the current days as a possible solution for the increasing needs of resources in terms of computing power and storage capacity for the current research challenges in the world.

This new paradigm offers very amazing views for the ways in which the new generation information networks will be established and managed as well as the new highest performances ever seen. However, its difficulties are also higher than before because this new perspective needs very abstract and complex tasks to perform high-end, general and extensible results. This new environment is mainly based on the wide distribution of services. Services that provide high-end features to a global infrastructure and delivered through different partners and different sets of non-homogeneous resources.

In recent years, the amount of data being stored, accessed and analyzed has increased tremendously. The advancements in hardware and software have further facilitated the storage and access of this enormous data. In addition, the datasets are also stored in a distributed manner and are shared by many people across several different domains. Grid technologies and infrastructure are developed to support the sharing and coordinated use of diverse resources in dynamic, distributed virtual organizations (VOs) [1]. A virtual organization is termed as a set of individuals and resources across multiple organizations (for example, a collaboration formed by two independent physical organizations). These individuals and resources can be dynamic in nature. There are a number of projects worldwide [25], which are actively exploring the development of various grid computing system components, services, and applications. They include Globus [3], Condor [10],NetSolve [23], Nimrod/G [27], AppLes [30], and JaWS [31].

According to Ian Foster, the essence of grid computing can be captured in a simple checklist [24], according to which a grid is a system that:

- ***Coordinates resources that are not subject to centralized control***

A grid integrates and coordinates resources and users that live within different control domains, for example, the user's desktop vs. central computing; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings

- ***using standard, open, general-purpose protocols and interfaces***

A grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access. These protocols and interfaces need to be standard and open so that system is able to execute generic applications.

- ***to deliver nontrivial quality of service***

A grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating - for example - to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater than that of the sum of its parts.

Grids are becoming mission-critical components in research and industry, offering sophisticated solutions in leveraging large scale computing and storage resources. Grid resources are usually shared among multiple organizations in an opportunistic manner. However, an opportunistic or "best effort" quality-of-service scheme may be inadequate in situations where a large number of resources need to be allocated and applications which rely on static, stable execution environments.

1.1.1 Elements of Grid Computing

Grid has emerged recently as an integration infrastructure for the sharing and coordinated use of diverse resources in dynamic, distributed virtual organizations (VOs). It provides a distributed system middleware that allows different communities to access and share data, networks, and other resources in a controlled and secure manner.

Scientific and Business communities are increasingly collaborating, and this is giving rise to the need for more sophisticated technologies for data and resource sharing.

The key components of grid computing include the following:

1. Resource management: a grid must be aware of what resources are available for different tasks
2. Security management: the grid needs to take care that only authorized user can access and use the available resources
3. Data management: data must be transported, cleansed and processed.
4. Services management: users and applications must be able to query the grid in an effective and efficient manner

More specifically, grid computing environment can be viewed as a computing setup constituted by a number of logical hierarchical layers. Figure 1 represents these layers. They include grid fabric resources, grid security infrastructure, core grid middleware, user level middleware and resource aggregators, grid programming environment and tools and grid applications.

The major constituents of a grid computing system [26] can be identified into various categories from different perspectives as follows:

- Functional view
- Physical view
- Service view

Basic constituents of a grid from a *functional* view are decided depending on the grid design and its expected use. Some of the functional constituents of a grid are

1. Security (in the form of grid security infrastructure)
2. Resource Broker
3. Scheduler
4. Data Management
5. Job and resource management
6. Resources

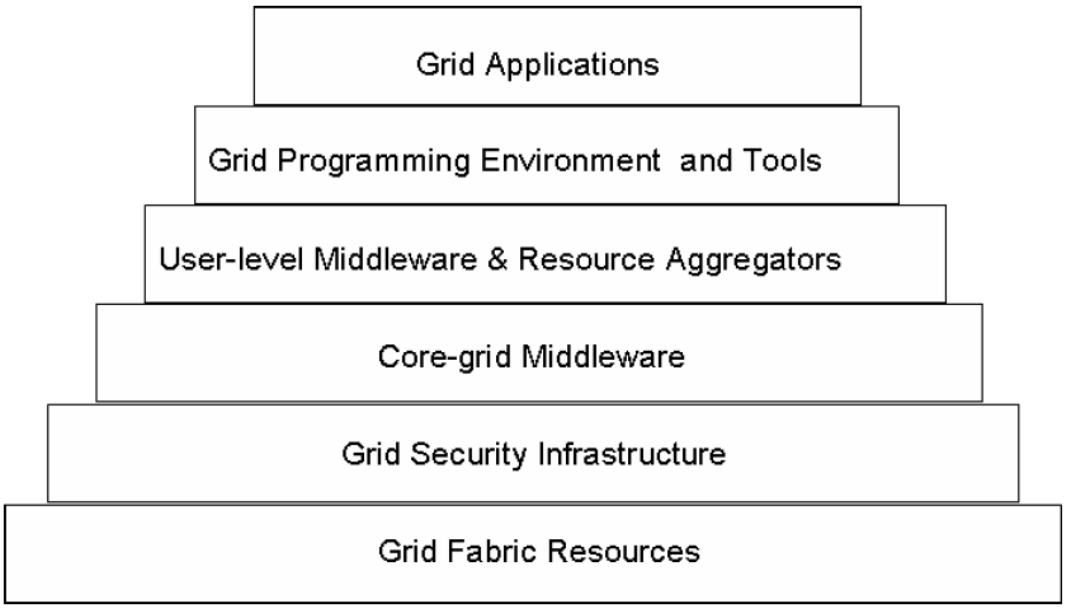


Figure 1.1: Grid Computing Layers

A resource is an entity that is to be shared [27]; this includes computers, storage, data and software. A resource need not be a physical entity. Normally, grid portal acts as a user interaction mechanism which is application specific and can take many forms. A user-security functional block usually exists in the grid environment and is a key requirement for grid computing. In a grid environment, there is a need for mechanisms to provide authentication, authorization, data confidentiality, data integrity and availability, particularly from a user's point of view. In the case of inter-domain grids, there is also a requirement to support security across organizational boundaries. This makes a centrally managed security system impractical. The grid security infrastructure (GSI) provides a "single sign-on", run-anywhere authentication service with support for local control over access rights and mapping from global to local identities.

Different types of resources:

- *Hardware resource* - A hardware resource represents a particular host on the Grid. Hardware resources contain service resources and software resources.
- *Service resource* - A service resource is a resource that represents a "service" that is accessible over a network. All service resources have at least one "port"

associated with them, through which they can be invoked by clients, and a "protocol" for communicating with the service.

- *Software resource* - A software resource is a resource that represents"software". At minimum, a software resource has a "path" on a given hardware resource.

1.1.2 Grid Architecture

Grid systems and applications aim to integrate, virtualize, and manage resources and services within distributed, heterogeneous, dynamic "virtual organizations". The realization of this goal requires the disintegration of the numerous barriers that normally separate different computing systems within and across organizations, so that computers, application services, data, and other resources can be accessed as and when required, regardless of physical location. Key to the realization of this Grid vision is standardization, so that the diverse components that make up a modern computing environment can be discovered, accessed, allocated, monitored, accounted for, billed for, etc., and in general managed as a single virtual system—even when provided by different vendors or operated by different organizations. Standardization is critical if we are to create interoperable, portable, and reusable components and systems; it can also contribute to the development of secure, robust, and scalable Grid systems by facilitating the use of good practices.

At a very high level, Grid Architecture can be best represented in component layers. This layered architecture is by no means a rigid dictation of the components, but it is extensible and follows the open architectural framework.

It is important to denote the analogy with current Internet model shown in figure 2, where Internet Protocol is in the core, over it is a broad range of different protocols for different purposes and under it are the set of protocols that enable access to different physical resources. As this model has brought Internet expansion successfully, the same approach should get the Grid success.

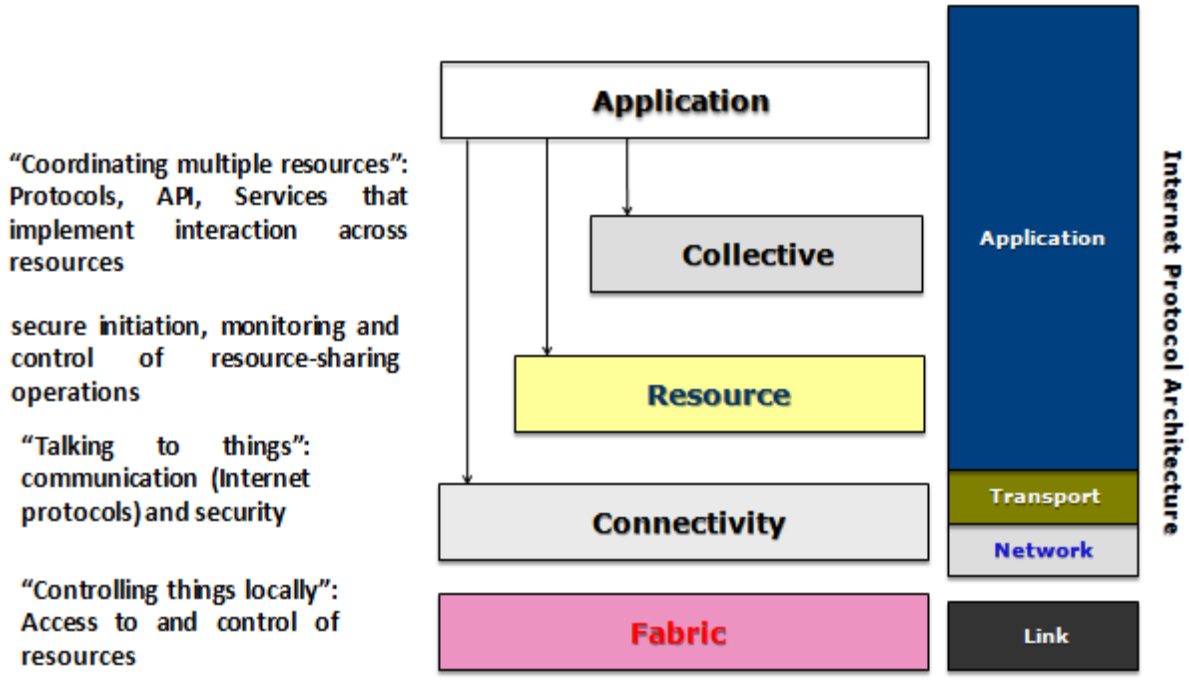


Figure 1.2: Grid Architecture **Vs** IP Architecture

Application layer

Application layer is the last level in the Grid protocols stack and comprises of all the end-to-end user applications which interact with collective layer (also with lower layers) in order to provide all desired features. Applications are based on high level languages and frameworks that implement the mentioned services within every layer. So that is the mean for achieving interoperability and portability at user level.

Collective layer

The collective layer is built upon the resource layer to extend its individual capabilities and provide global aware interactions across collection of resources hosted in different locations. Again, like in the TCP/IP stack equivalent (transport level), this layer can be considered the first end-to-end layer. As a result of this design, provisioning properties mentioned before are supplied by this layer.

Some of these services are the following ones:

1 Directory services: discovering and enquiring of VOs properties or status.
2 Co-allocation, scheduling and brokering services: to request and establish distributed allocation of jobs.
3 Monitoring and diagnostic services: for supporting failure tolerance and improve delivered QoS.
4 Data replication services: for maximizing performance data transfers in term of latency, reliability and cost.
5 Grid-enabled programming services: for enabling high abstract programming languages to use global Grid services seamlessly.
6 Software discovery services: to discover best software implementations in term of platform requirements and user requirements.
7 Authorization services: acting as the policy definition point inside each VO (implementing RBAC security model).
8 Collaborative services: for coordinated exchange of information between large user communities.

Resource layer:

Resource layer protocols enable secure initiation, monitoring and control of operations over individual resources. Like in the analogy with IP stack, this layer establishes the constraints upon which upper layer protocols will interact with resources. So, this level performs its actions entirely over individual resources. This distinction is the key oppositeness to collective layer in terms of, for example, atomic issues (like the reliability property provided only by TCP in TCP/IP model).

Two classes of protocols can be considered:

1 *Information protocols:* for enquiring about structure details or status like load or policy issues.
2 *Management protocols:* for accessing and processes creation, monitoring and control. The important aspect is this layer serves as policy application point (so called policy enforcing point) in order to ensure consistency with the policy definition under which the resource is shared.

Connectivity layer

The connectivity layer establishes the communication and authentication protocols which enable data exchange among different fabric resources. It is built upon fabric layer in order to provide the abstraction to resource layer which enables seamless access to fabric resources.

Communication requirements comprise transport, routing and naming capabilities. In most of the cases, this will be mapped to equivalent protocols in IP stack, but general use of resources, specially networks, involves the development of some mechanisms for adding extensibility and abstraction to that approach.

Regarding authentication requirements, this layer should provide the next features:

1 Single sign on (SSO): ability to get authenticated only once (just entering into Grid services).
2 Delegation: ability to endow a program to access services under user's behalf. This one together with SSO enable the integral security model required in Grid environments.
3 Integration with local security: seamless integration with local security solutions already deployed into some organization (such us Kerberos, Active Directory, Java Enterprise System, etc.)
4 User-based trust relationships: mechanisms for centralizing usage granting when services are provided among different providers. This avoids interactions between them as a prerequisite of use.

Fabric layer

The fabric layer is the lowest one in the Grid protocols stack mainly responsible to provide access to the capabilities offered by resources. In this case, resources comprise of physical ones (computational, storage, network, etc.) or logical sets of these ones (clusters, distributed file systems, etc.).

Of the two main characteristics of Grid, this layer mainly provides the virtualization property for higher level protocols so it must deal with the details of implementations of every resource. Thus, there is a subtle dependency between sharing operations that can

happen at higher levels and the functions implemented at fabric one. Features provided by resources (i.e. the fabric layer) should be prepared for enquiry and management mechanisms. The first ones allow browsing the resources structure and eventually knowing their status and the second ones use of their capabilities under some controlled way to provide quality of service over demand.

Finally, some capabilities that should be provided at this layer are:

1 Computational resources: monitor and control of processes, allocation, reservation, status, characteristics browsing, etc.
2 Storage resources: global and performance transferring, capabilities management or enquiry mechanisms.
3 Network resources: control over resources for enabling QoS, enquiry mechanisms, etc.
4 Code repositories: software configuration management capabilities for source and object code.

1.1.3 Main characteristics of Grids

The main characteristics of a grid computing environment can be listed as follows:

1 **Large scale:** A grid must be able to deal with a number of resources ranging from just a few to millions.
2 **Geographical distribution:** Grid resources may be spread geographically.
3 **Heterogeneity:** A grid hosts both software and hardware resources that can be ranging from data, files, software components or programs to sensors, scientific instruments, display devices, personal digital organizers, computers, super-computers and networks.
4 **Resource sharing and coordination:** Resources in a grid belong to different organizations that allow other organizations (i.e. users) to access them. The resources must be coordinated in order to provide aggregated computing capabilities.
5 **Multiple administrations:** Each organization may establish different security and administrative policies under which resources can be accessed and used.

6 **Accessibility attributes:** Transparency, dependability, consistency, and pervasiveness are attributes typical to grid resource access. A grid should be seen as a single virtual computing environment and must assure the delivery of services under established Quality of Service requirements. A grid must be built with standard services, protocols and interfaces thus hiding the heterogeneity of the resources while allowing its scalability. A grid must grant access to available resources by adapting to dynamic environments where resource failure is common.

1.2 Types of Grid Computing

The grids are classified into different types depending on the architecture and type of resources being shared.

Present-day grids encompass the following types:

1. Computational grids, in which machines will set aside resources to "number crunch" data or provide coverage for other intensive workloads.
2. Scavenging grids, commonly used to find and harvest machine cycles from idle servers and desktop computers for use in resource-intensive tasks (scavenging is usually implemented in a way that is unobtrusive to the owner/user of the processor)
3. Data grids, which provide a unified interface for all data repositories in an organization, and through which data can be queried, managed and secured.
4. Service grids, which provides different types of services to share in the grid.

There are two ways of defining computing resources which can process large amounts of data very quickly: high-throughput, or high-performance. High-performance is simple to define as a machine that performs calculations extremely quickly. High-throughput however is a more complex one. High-throughput computing (HTC) is a computer science term to describe the use of many computing resources to accomplish a computational task. High-performance computing uses supercomputers and computer clusters to solve advanced computation problems

High-throughput computing means the ability to process large amounts of data in short amounts of time. Unlike high-performance, which relates to the speed at which individual components operate, high-throughput can be achieved even on relatively slow machines by connecting many of them together so that they can work in parallel. The key to HTC is to efficiently harness the use of all available resources. As computers became smaller, faster, and cheaper, users moved away from centralized mainframes and purchased personal desktop workstations and PCs. Many organizations are using PCs for individual employees. But these systems are not utilized to their full extent. In many cases the utilization of these resources is below 30%. To tackle this problem of wasting computing resources, many researchers have developed different systems to efficiently harness the use of these resources. Some of the popular systems in High throughput Computing are Condor, BIONIC and GridGain.

1.3 Motivation & Project Objectives:

In the last couple of years many banks have been having quite a tough time due to economical crisis. One of the ways they have been able to get through these difficult times is through more intelligent use of technology, to analyze and predict the risks, and taking appropriate decisions. In order to perform the required risk modeling in this new world order, it is estimated the banks will need something like 10x the computing power they were using before the crisis. So many banks are now looking seriously into making more use of cloud computing and grid computing services.

As the business of banking grows increasingly complex, specialized customer needs are being met by specialized applications. In every bank, the portfolio of applications and technologies is expanding at a phenomenal rate. These applications are most often deployed on server machines within the bank's environment. While these servers are loaded to their peak, there is a need for computing resources and better utilization of existing resources. The bank's IT environment consists of a large number of desktop workstations at the branches. With newer desktop machines surging with higher processing power, but used to perform relatively lighter tasks, a large portion of this processing capacity is also underutilized. Thus, normally at any given point in time, a

bank's IT environment includes significant redundant or idle processing capacity. Banks can make use of these resources to meet their increasing demand for IT resources. For instance, Monte Carlo simulations are easily parallelized and distributed to every compute node participating in the grid. The advantage is instead of executing these simulations on one server and grinding away for, they now can be broken apart and distributed across hundreds or even thousands of compute nodes, resulting in dramatic reductions in the amount of time taken to complete. The result is that organizations using this technology are gaining a competitive edge, and optimal utilization of resources.

The project objectives are,

- To study the applicability of grid computing in banks
- To develop banking grid architecture to harness the underutilized resources in the banks
- To develop a grid access portal based on J2EE and JSP frameworks, which offer users, access to various grid services in a dynamic grid environment.

1.5 Literature Survey

In this section, we present a detailed literature survey in grid computing models and architectures. The aim of this survey work is to present current research trends from the point of view of the grid computing models, application of grid computing in banking domain.

Ian Foster and Carl Kesselman gave the initial definition of grid in their book in 1998, Entitled "Grid: Blueprint for a New Computing Infrastructure". The definition is computational grid is a reliable, consistent, ubiquitous, inexpensive hardware and software infrastructure, used for high-end computing as in [28]. Grid wants to make the Local Area Network, Metropolitan Area Network and even Internet into a huge supercomputer, achieve the overall share of knowledge resources, storage resources, computing resources, information resources and expert resources. Same as internet, the concept of grid computing comes from the needs to achieve sharing resources among different domains. The main feature of the grid is sharing of resources as in [27].

"A Grid-enabled CPU Scavenging Architecture and a Case Study of its Use in the Greek School Network", work done by Fotis Georgatos, Vasileios Gkamas, Aristeidis Ilias, Giannis Kouretis, Emmanouel Varvarigos [2] presented a CPU scavenging architecture suitable for desktop resources for harnessing the idle resources at PC Laboratory resources of the Greek School Network.

"From Dedicated Grid to Volunteer Grid: Large Scale Execution of a Bioinformatics Application", work done by Viktors Bertis, Raphaël Bolze, Frederic Desprez, Kevin Reed [5] discussed about large scale execution of Bio-informatics applications on desktop computing grid.

"Architecture and performance of an enterprise desktop Grid system", work done by Chien A [6] discussed about the exploitation of idle cycles on pervasive desktop PC systems, offers the opportunity to increase the available computing power to an organisation.

"SETI@home: An experiment in public-resource computing", work done by Anderson, D.P [20] discussed a model for sharing of public resources by Non-profit organisations for executing highly compte intensive experiments.

"Finacle Connect: Grid Computing for Banks", A report written by Deepak N. Hoshing, Ravi Venkataratna, [21] analyzed various factors and addressed the wastage of computing resources in banks. Also they have suggested some models to effectively utilize the resources in banks.

"Distributed Portfolio and Investment Risk Analysis on Global Grids", work done by R. Moreno-Vozmediano, K. Nadiminti, S. Venugopal, A. B. Alonso-Conde, H. Gibbins, and R. Buyya [11] discussed about distributed computing methods for Monte Carlo simulations, which can run in parallel on distributed resources.

"The GENIUS Grid Portal", work done by R. Barbera, A. Falzone [12], "Grid Application Development on the Basis of Web Portal Technology" work done by Gabor and Peter [15], describes the concepts behind the development of J2EE-framework and describes how it can be used for developing web based portals.

1.5 Contributions

We summarize the main contributions of the work below:

Banking Grid Architecture: We have proposed architecture to efficiently coordinate and share underutilized resources of banks. We have proposed two models, both Intra-bank and Inter-bank grid. It helps banks better utilize computing power latent in the bank's IT infrastructure.

Banking Grid Portal Using J2EE: We have developed a banking grid portal through which users can access grid resources. the functionalities include remote job submission, monitoring, getting results and querying the resource status etc.

1.6 Organization

Organization of thesis is as follows:

Chapter 1: Introduction to Grid Computing, discussed about Grid Computing and its architecture. Also discussed about different types of grids. Followed by motivation and project objectives.

Chapter 2: Design of Banking Grid- Intra Bank, in this chapter we have proposed architecture of banking grid and discussed about various components of the system.

Chapter 3: Design of Banking Grid- Inter Bank, in this chapter we have proposed architecture of banking grid for multiple banks and discussed about various components of the system along with issues in implementation.

Chapter 4: Development of Banking Grid Portal, in this chapter we have discussed about the banking grid portal and its design and implementation details.

Chapter 5: Conclusion followed by references and appendix

1.7 Conclusion

This chapter provides an introduction to grid computing, an emerging technology of enormous promise. We discuss the elements which constitute the grid. We also discuss in great detail about grid architecture. Next, we discuss on various models of Grid computing and their applicability in different areas. Next, we discuss the literature survey and analyzed some of the contributions in the area of grid computing and banking. The motivation, project objectives and the contribution of the thesis are presented along with the organization of the thesis.

CHAPTER 2
DESIGN OF BANKING GRID
-Intra Bank

2.1 Introduction:

For last 10 years, the largest computing systems in the world have been based on ‘‘distributed computing’’ the assembly of large numbers of PCs over the Internet. These ‘‘grid’’ systems sustain multiple teraflops continuously by aggregating hundreds of thousands to millions of machines and demonstrate the utility of such resources for solving a surprisingly wide range of large-scale computational problems in data mining, molecular interaction, financial modeling, etc. These systems have come to be called ‘‘distributed computing’’ systems and leverage the unused capacity of high-performance desktop PCs (up to 3:2 GHz machines with multi-gigaop capabilities), high-speed local-area networks (100 Mbps to 1 Gbps switched), large main memories (256 MB to 2 GB configurations), and large disks (60–250 GB disks). Such distributed computing systems leverage the installed hardware capability (and work well even with much lower performance PCs) and thus can achieve a cost per unit superior to the cheapest hardware alternatives by as much as a factor of 5 or 10. As a result, distributed computing systems are now gaining increased attention and adoption within enterprises to solve their largest computing problems and attack new problems of unprecedented scale.

Inter-Bank Network Architecture:

The network of banks is spanned into wide area, consisting of several number of branches. The central system is the heart of the bank network, which consists of critical systems like CBS, databases, Application servers. Each branch has teller machines (mostly desktop PCs), used to do transactions of customers. Every branch is connected to central system through different channels; like leased lines, Internet using VPN, VSAT etc. Also the connections are redundant; to ensure the connectivity when some disasters happen. The model of network is presented in the below figure

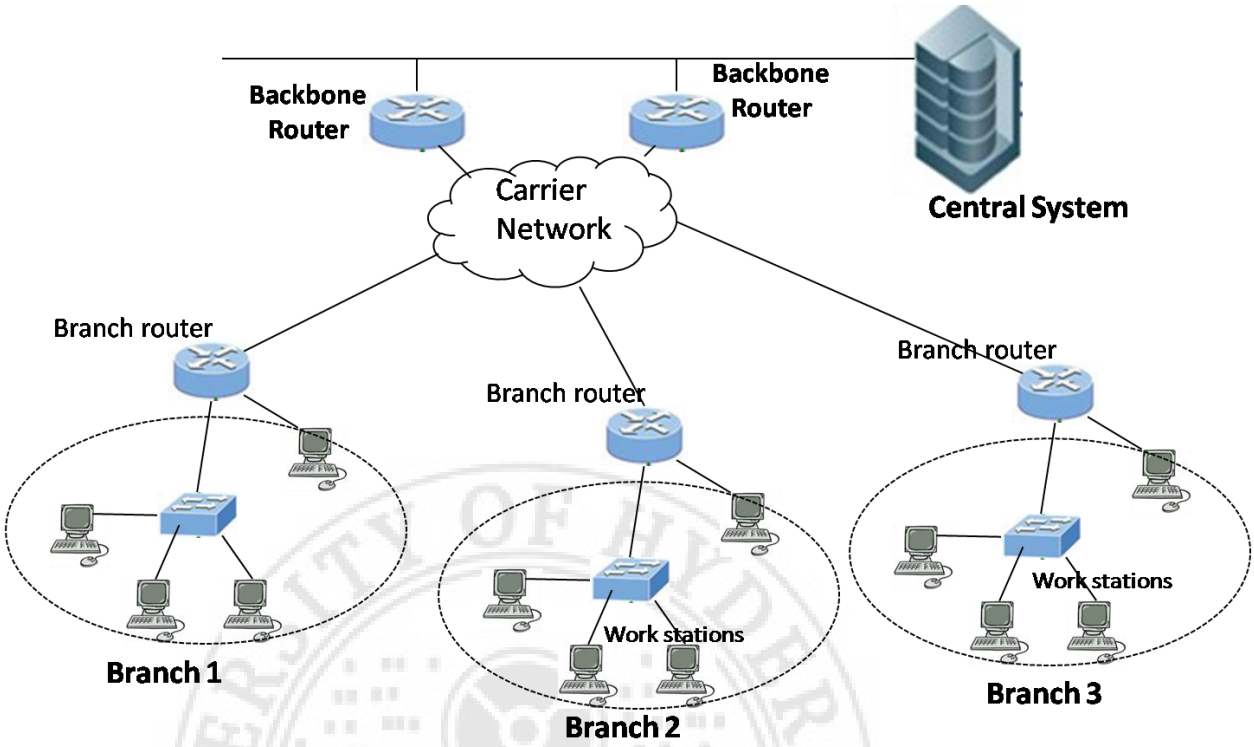


Figure 2.1: Bank internal Network

In this Chapter, we present a banking grid architecture/system to leverage the existing resources in a bank. This architecture also proposes how resources are managed, monitored and allocated in grid. The grid middleware offers services that help in coupling a grid user and (remote) resources through a resource broker or grid enabled application. It offers core services such as remote process management, co-allocation of resources, storage access, transfer of files, information (directory), security and authentication. Globus Toolkit (GT4) with Condor scheduler which contains Globus Authorization Framework is used as a grid middleware for authentication, authorization of users and resources. Job allocation, monitoring, scheduling, execution tasks are managed by Condor system also discussed.

2.2 Requirements of Intra Banking Grid

The new technologies like grid for desktop PC, distributed computing to be widely accepted within the organization, the systems must achieve high levels of efficiency,

robustness, security, scalability, manageability, standardization, and openness/ease of application integration.

Efficiency: The system must harvest unused cycles efficiently, collecting virtually all of the resources available. The Proposed system gathers over 95% of the desktop cycles unused by desktop user applications.

Robustness: The system must complete computational jobs with minimal failures, masking underlying resource and network failures. In addition, the system must provide predictable performance to end-users despite the unpredictable nature of the underlying resources.

Scalability: Grids consist of large and dynamic users. So, the system mechanism must be able to deal with large number of users. Traditional Service Oriented Architecture (SOA) should be used for building grid applications. The system must scale to the use of large numbers of computing resources. Because large numbers of PCs are deployed in many enterprises, scaling to 1000s, 10,000s, and even 100,000s are relevant capabilities. However, systems must scale both upward and downward performing well with reasonable effort at a variety of system scales.

Manageability: Any system involving thousands to hundreds of thousands of entities must provide management and administration tools. Typical rules of thumb, such as requiring even one administrator for every 200 systems, would be unacceptable. We believe distributed computing systems must achieve manageability that requires no incremental human effort as clients are added to the system. A crucial part of manageability is client cleanliness: it is crucial that the computing resources state is identical after running an application as it was before running the application.

Openness/ease of application integration: Fundamentally, the distributed computing system is a platform on which to run applications. The number, variety, and utility of the applications supported by the system directly affect its utility. Distributed computing systems must support applications developed with all kinds of models, with many distinct needs and with minimal effort.

Standardization: In order to achieve interoperability the implementation and design should be built on latest technologies, standards which are widely accepted.

Maximum Resource Usage: Every Resource in grid environment should be utilized at maximum based on its availability. Resource idle time should be negligible. Tasks should be assigned continuously to those resources which have high configuration along with performance.

Dynamic allocation of Resources: Complex systems require simultaneous execution of code on very high numbers of CPUs. Those resources must be negotiated in advance and guaranteed to be available when the task's time slot arrives. This implies the need for a sophisticated distributed negotiation protocol which is supported by advance reservation mechanisms.

Concurrent execution of jobs: Based on the status, performance, configuration of computational resources jobs should be submitted and executed concurrently on various nodes in grid environment.

Fault tolerance: In grids, nodes and network failures will inevitably occur. However, to assure that an entire application will not be aborted after a single failure, distributed checkpoints and restart protocols must be used to stop and migrate the whole application or part of it.

2.3 Proposed Intra-Bank Grid Architecture

A layered view of banking grid architecture in a bank is given in figure 2.2. The system architecture is composed of four separate layers. At the bottom is the Physical Node Management layer that provides basic communication and naming, security, resource management, and application control. On top of this layer is the Resource Scheduling layer that provides resource matching, scheduling, and fault tolerance. Job Management layer provides management facilities for handling large numbers of computations and files. The User Interface layer abstracts the complexity of the below layers from the user. The actions performed by the user, will be passed to the lower layers. We briefly describe

these layers, and then describe the advantages of this approach in achieving a robust, scalable distributed computing resource.

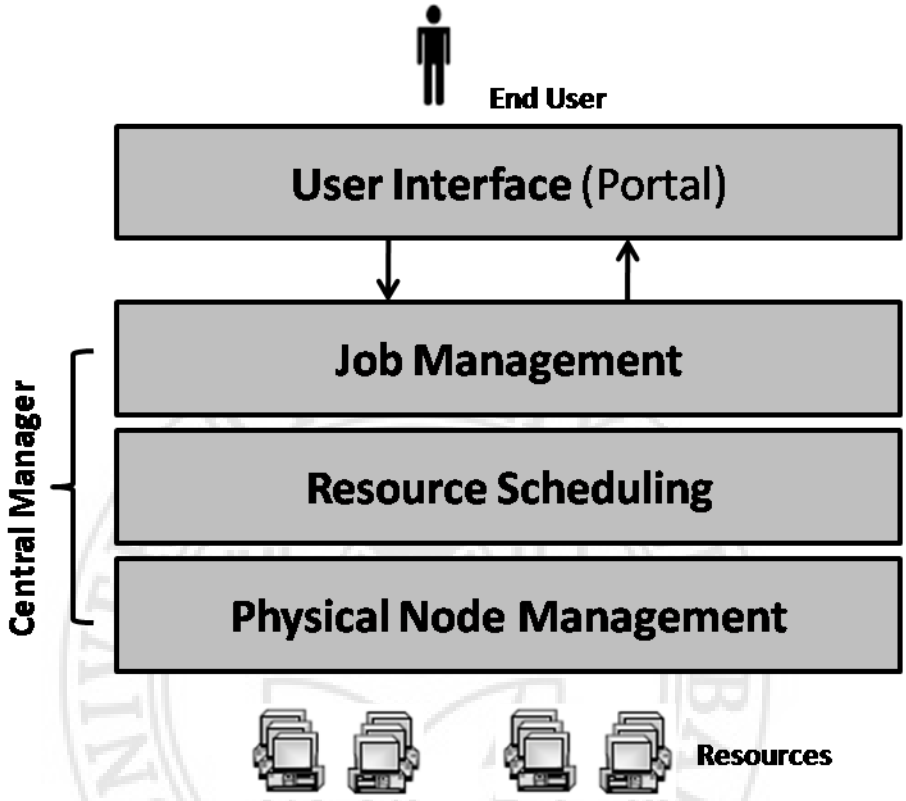


Figure 2.2: Layered view of Architecture

Physical Node Management: The distributed computing environment presents numerous unique challenges to providing a reliable computing capability. Individual client machines are under the control of the desktop user or IT manager. The Physical Node Management layer manages these and other low-level reliability issues. The resource management services capture a wealth of static and dynamic information about each physical node (e.g. physical memory, CPU speed, disk size, available space, client version, data cached, etc.), reporting it to the centralized node manager.

Resource Scheduling: The distributed computing system consists of resources with a wide variety of configurations and capabilities. The Resource Scheduling layer accepts units of computation from the user or job management system, matches them to appropriate client resources, and schedules them for execution. Despite the resource

conditioning provided by the Physical Node Management layer, the resources may still be unreliable (indeed the application may be unreliable in its execution). Therefore the Resource Scheduling layer must adapt to changes in resource status and availability and to failure rates that are considerably higher than in traditional cluster environments.

Job Management: A distributed computing application often involves large amounts of computation submitted as a single large job. This job is then broken down into a large number of individual subjobs each of which is submitted to the grid for execution. The Job Management layer of the system is responsible for decomposing the single job into the many subjobs, managing the overall progress of the job and aggregating the results of the subjobs. This layer allows users to submit a single logical job (for example, a Monte Carlo simulation, a parameter sweep application, or a database search algorithm) and receive as output a single logical output. The details of the decomposition, execution and aggregation are handled automatically.

User Interface: This layer abstracts the complexity of the below layers from the users, and provides a graphical user interface to the user. The actions performed by the user will be transformed to system understandable calls and passed to the lower layers. User can view the status of jobs, resources and output of the jobs.

The complete architecture and important components of the system are shown in figure 2.3. The central manager and web servers are located at the bank center. The web server provides and manages access to the system. The central manager consists of Scheduler, Monitoring and Discovery service, Grid Authorization mechanisms and Manager Process. The Manager process controls all the process in the system. The GridAPI is used to integrate the portal with the grid.

The client program will be loaded on each node in the grid. The goal of Client is to harvest unused computing resources by running jobs unobtrusively on the machine (Execute node). All the Execute nodes are controlled by the central manager. On each node two processes called LJS (Local job scheduler) and Grid resource information services are running. The GRIS is responsible for updating the node status to the central manager. The local job scheduler will take care of executing the job received from the

central manager, sending back the results and check pointing the jobs. The Desktop Client provides security for the client machine by mediating job access to the file system, registry, and graphical user interface. This prevents the job's processes from doing harm to the machine, and ensures that after job execution, the machine's state is the same as it was before running the job.

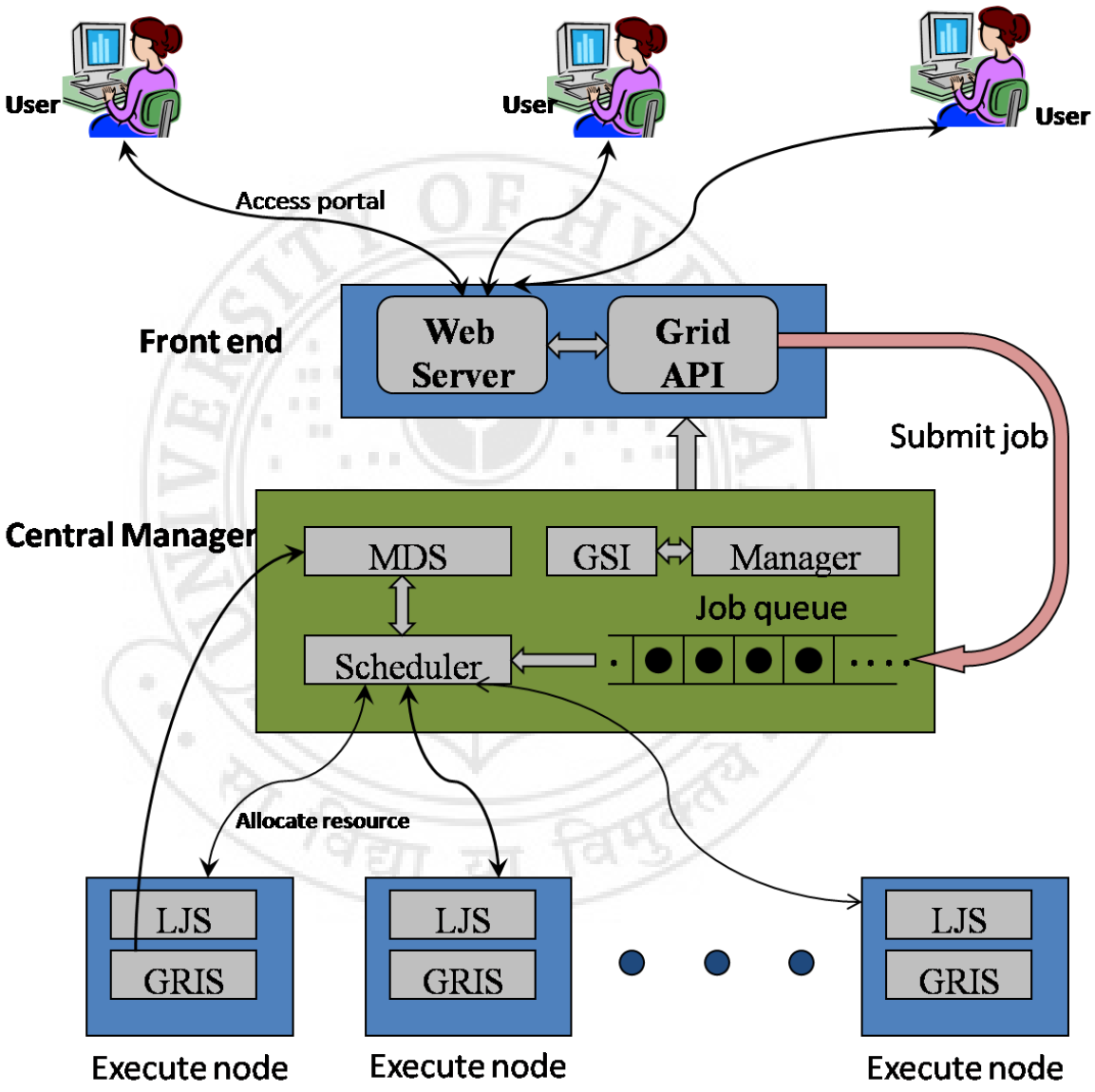


Figure 2.3: Intra-Bank Grid Architecture

2.4 Overview of Components:

Central Manager: The central manager is the heart of the grid, as everything is managed by the central manager. The Job scheduler, Authorization service, Monitoring and discovery services are hosted at central manager. Central manager is responsible for the allocation of resources based on the requirement and availability of resources, monitoring the execution of jobs on allocated resources, check pointing of job state, transferring executables, input/output corresponding to the job to the execution node, and aggregating the results from sub jobs etc.

There can be only one central manager for the pool. The machine is the collector of information, and the negotiator between resources and resource requests. These two halves of the central manager's responsibility are performed by separate daemons, so it would be possible to have different machines providing those two services. This machine plays a very important part in the Condor pool and should be reliable. The central manager will ideally have a good network connection to all the machines in the pool, since they all send updates over the network to the central manager. All queries go to the central manager. The central manager will be located at banks network center, where all important CBS and other data storage systems are maintained.

Web Server: It allows many computers to access specific services simultaneously. A web server can be referred to as either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed from remote locations in a network. The primary function of a web server is to deliver web pages on the request to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, files etc. The web server hosts the services, and serves the requests received from the clients. A client, commonly a web browser, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. To host the grid access portal, we used the Apache tomcat web server.

MDS: The Monitoring and Discovery of resources service is responsible for resource discovery, resource monitoring in the grid. All the resources are managed locally with in a banking domain. The status of a node (active/inactive), status of CPUs in a node,

network connections, and other monitoring statistics are collected by the monitoring system. All the nodes in the grid periodically send their status messages to the central manager, where the collector daemon service collects the information received from nodes. An information repository is maintained to query the resource information by other processes. Whenever a job request comes, the scheduler queries the MDS and allocates the available resource to execute the job.

GRIS: Grid Resource Information Service is the local resource information provider running on the client nodes. GRIS registers its information with the Central manager periodically.

Job Scheduler: The Job Scheduler is responsible for scheduling jobs to clients and their robust execution. As jobs are submitted to the system, the scheduler places them in a job queue. The job scheduler is also responsible for providing fault tolerance, insuring progress towards completion, and identifying faulty jobs. A job is rescheduled to run on a different client if the client becomes disconnected for too long or fails to return a result within the expected amount of time. If the job fails to complete after a given number of tries the job is marked as faulty and returned to the job manager. The job manager can then choose to re-submit the job or notify the user.

Job Queue: The job scheduler manages the job queues. As and when the user submits the job to the system, they are placed in the job queue. Each job will be processed, and allocate resources for execution. If no resource is available the job will be kept in waiting state until the scheduler finds some free resource in the system.

LJS: Local Job Scheduler is responsible for execution of job and sending back the results after the job has been completed. It is managed by the Job scheduler in the central manager.

Grid FTP: GridFTP is an extension of the standard File Transfer Protocol (FTP) for use with Grid computing. The aim of GridFTP is to provide a more reliable and high performance file transfer for Grid computing applications. GridFTP achieves much greater use of bandwidth by allowing multiple simultaneous TCP streams. Files can be transferred in pieces simultaneously from multiple sources.

Execute Node: In collaborative applications, participants agree on certain level of secure communication based on communication policy specifications.

Grid Authorization System: Currently, the most common solution for mutual authentication and authorization of grid users and services is the Grid Security Infrastructure (GSI). The GSI provides fundamental security services needed to support grids. It is based on Public Key Infrastructure. The users and hosts authenticate to the system using X.509 certificates.

Grid Portal: A grid portal may be constructed as a Web page interface to provide easy access to grid applications. The Web user interface provides user authentication, job submission, job monitoring, and results of the job. It abstracts the complexity of the grid from the user.

2.5 Conclusion

In this chapter, Intra-Bank Grid architecture is proposed which enables banks to harness the power of underutilized computing resources. We analyzed different aspects of the system. We have discussed environment from resource sharing to job scheduling and also discussed about various components.

CHAPTER 3
DESIGN OF BANKING GRID
-Inter Bank

3.1 Introduction:

In the previous chapter we have discussed about the Intra-Bank Grid architecture. It enables a bank to coordinate and make use of available resources efficiently. In this chapter we are proposing interbank grid architecture. This enables multiple banks to share their resources securely and efficiently. The architecture proposed in the previous chapter for a single bank can be expanded to the multiple banks with an additional layer to interconnect all the central managers of the banks. The central switch provides connectivity between the central managers.

As an intra-grid topology is extended to an inter-grid topology, the complexity increases. For example, the non-functional and operational requirements such as security, information services, reliability, and performance become more complex. In intra bank grid, resources of the same bank are shared. And all the resources are in the banks internal network. So there is not so much concern about security and management of grid. But in the case multiple banks collaborating to share their resources, security and management policies plays important role. In this chapter we have discussed about some new roles like central authority which is responsible for the functioning of the grid.

Inter-Bank Network Architecture:

In the previous chapter we have seen the intra bank network architecture. It consists of a central node and Execute nodes connected in the bank internal network. To share resources from multiple banks, the central nodes should be connected through a central switch. And also we need to enable the flocking of jobs to/from other pools in the grid. Flocking is nothing but sending jobs to and from other pools when the resources in the local pool are busy. The inter-bank network architecture is shown in the following figure 3.1

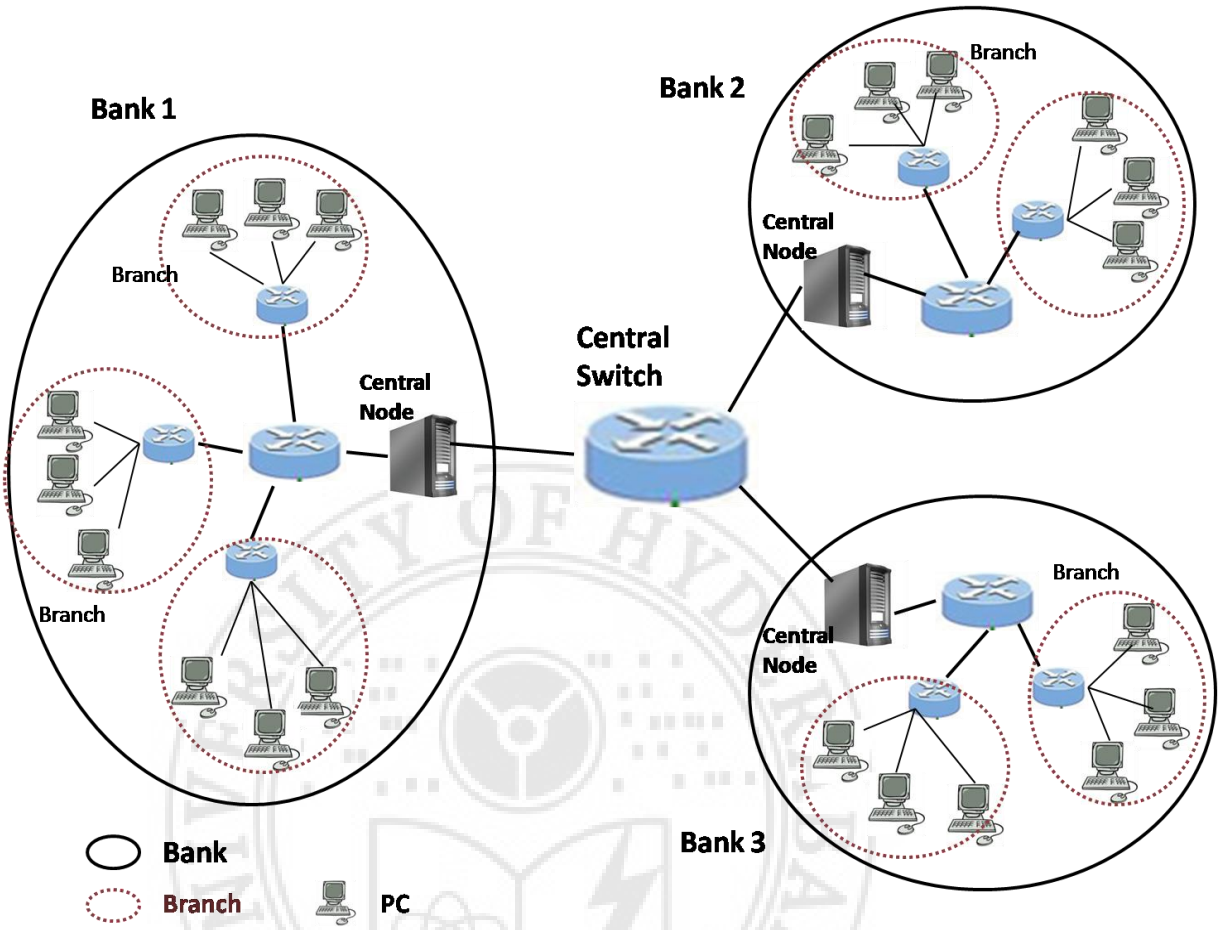


Figure 3.1: Inter Bank Network

As shown in the figure all the central nodes are connected to each other through central switch. The grid middleware offers services that help in coupling resources from multiple administrative domains. Along with the services provided in intra grid, it also provides VO(Virtual organization) services such as remote process management, co-allocation of resources, storage access, transfer of files, information (directory), security and authentication in inter-organizational grids. Globus Toolkit (GT4) with Condor scheduler which contains Globus Authorization Framework is used as a grid middleware for authentication, authorization of users and resources. Job allocation, monitoring, scheduling, execution tasks are managed by Condor system is already discussed.

Another model for inter-bank grid:

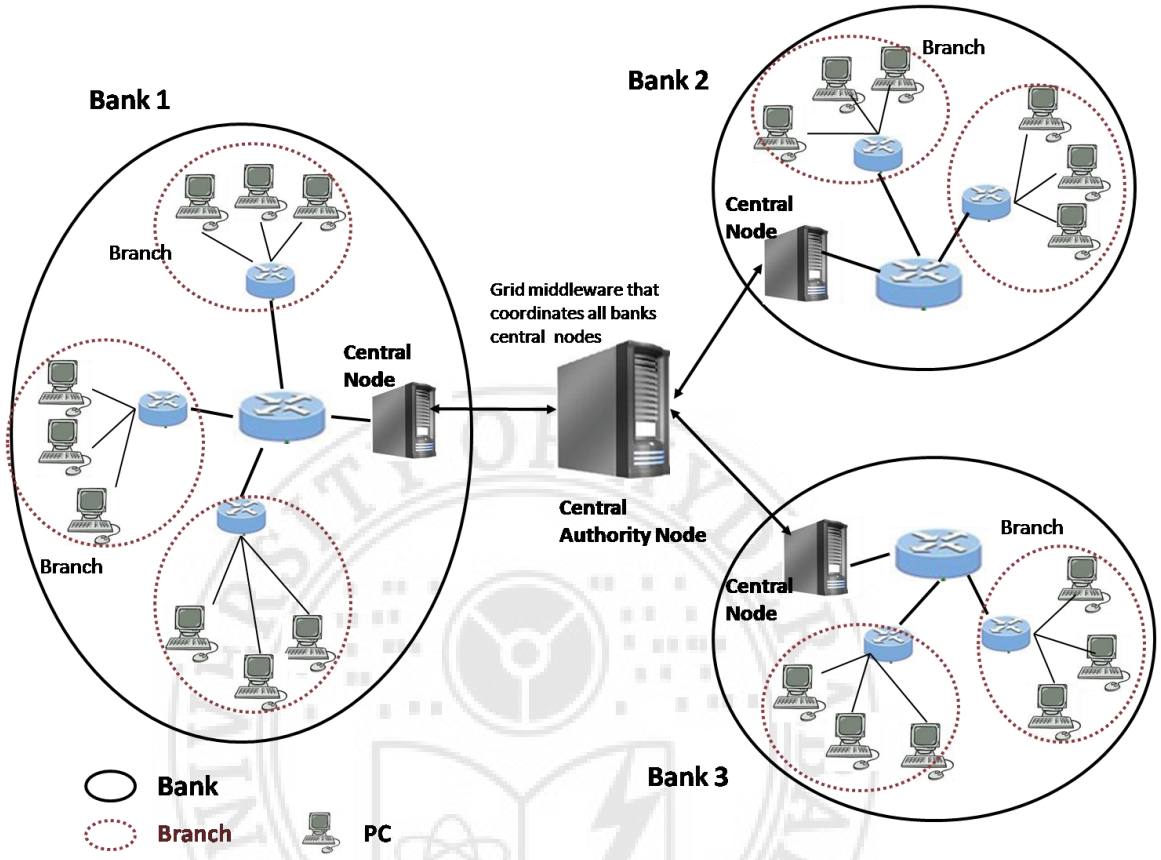


Figure 3.2: Inter Bank-2 Network

As shown in the figure all the central nodes are connected to a central authority node. It coordinates central nodes of all the banks in the grid. It will monitor the status of all nodes and controls access among the central nodes. It will also provide needed grid middleware services to the central nodes of the banks.

Advantages: the main advantage is this will ensure interoperability between the participating banks. It will act as a mediator between the central nodes. The management of the grid will become less complex. The bank central nodes will query the authority node to get the resource status and forwards the jobs to other bank nodes when the resources in the local pool are busy.

Because of the hierarchical flow of data, the central authority node should be capable of handling large amounts of data.

3.2 Requirements of Inter Banking Grid

The inter-bank grid has more requirements along with intra-bank requirements

Central Authority: There must be a central authority to manage all the activities in the inter-bank grid. It should be responsible for enforcing policies, providing connectivity between the banks, resolving any disputes between the participating parties. It can be formed by the members from all the banks in the grid.

Efficiency: The system must harvest unused cycles efficiently, collecting virtually all of the resources available. The efficiency of the system shouldn't degrade when resources from multiple entities are collaborated.

Robustness: The system must complete computational jobs with minimal failures, masking underlying resource and network failures when the resources are from multiple domains each may have different platform. In addition, the system must provide predictable performance to end-users despite the unpredictable nature of the underlying resources.

Scalability: Grids consist of large and dynamic users. So, the system mechanism must be able to deal with large number of users. Traditional SOA architecture should be used for building grid applications. The system must scale to the use of large numbers of computing resources. Because large numbers of PCs are deployed in many enterprises, scaling to 1000s, 10,000s, and even 100,000s are relevant capabilities.

Manageability: Managing resources from multiple administrative domains is complex. Each organization has its own policy in using and accessing resources. The system must ensure the enforcement of policies when accessing resources. Any system involving thousands to hundreds of thousands of entities must provide management and administration tools. Each organization has its administration methods. A crucial part of manageability is client cleanliness: it is crucial that the computing resources state is identical after running an application as it was before running the application.

Standardization: In order to achieve interoperability the implementation and design should be built on latest technologies, standards which are widely accepted.

Maximum Resource Usage: Every Resource in grid environment should be utilized at maximum based on its availability. When the resources in the local pool are busy, the idle resources in other pools should be utilized efficiently. To achieve this resource management and job scheduling services should interact with each other.

3.3 Proposed Inter-Bank Grid Architecture

Figure 3.2 below shows a layered view of architecture of inter-bank grid. The system architecture is similar to the intra-bank architecture. To deal with resource management from multiple VO's (virtual organizations) a layer called VO management is included. At the bottom is the Physical Node Management layer that provides basic communication and naming, security, resource management, and application control. On top of this layer is the Resource Scheduling layer that provides resource matching, scheduling, and fault tolerance. Job Management layer provides management facilities for handling large numbers of computations and files. The User Interface layer abstracts the complexity of the below layers from the user. The actions performed by the user, will be passed to the lower layers. We briefly describe these layers, and then describe the advantages of this approach in achieving a robust, scalable distributed computing resource.

The VO management layer is responsible for controlling the access to resources in other domains. It is possible to configure what organizations should allow to use resources in local pool. It provides access to the information services of other pools, like the status of the resources in that pool. It allows local job scheduler to forward jobs to the remote job scheduler, when resources in local pool are not available.

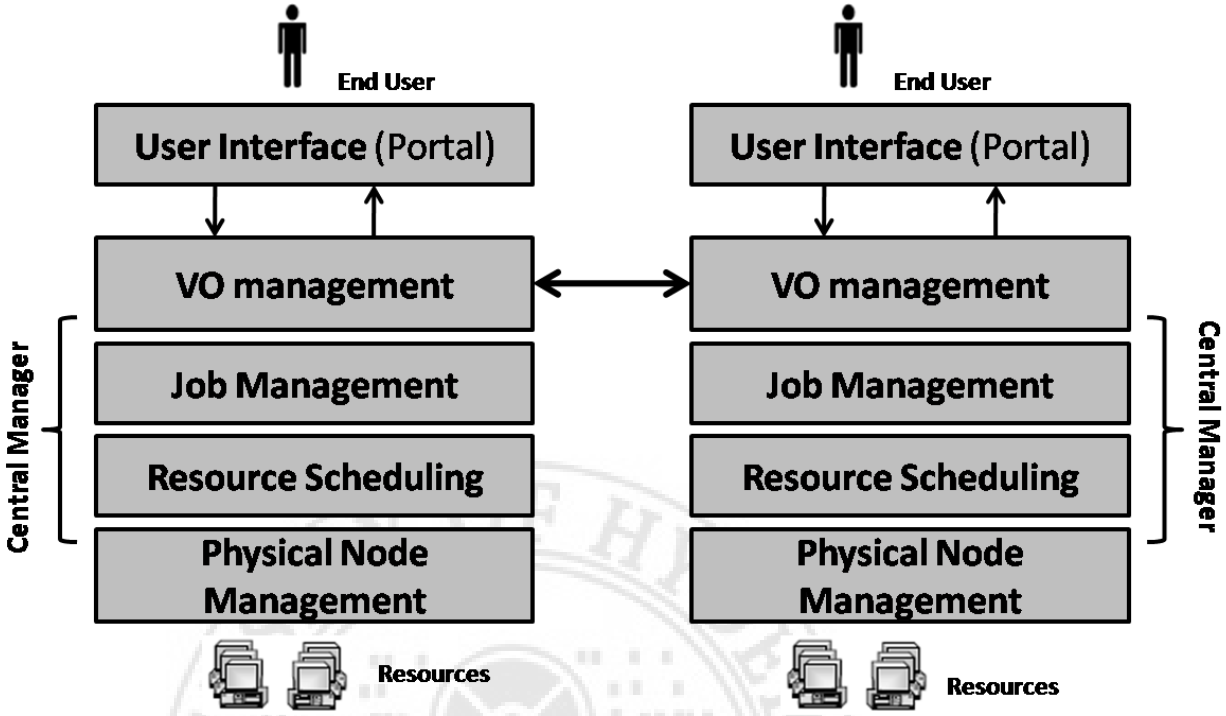


Figure 3.3: Layered view of Inter Bank Grid Architecture

VO Management: This layer abstracts the complexity of sharing resources from multiple domains. It provides transparency to lower layers, in accessing the resources of other pools. This layer makes it transparent to the job scheduler that the jobs received from other job managers are put in job queues, and allocate resources for execution. The remaining layers works just like in intra-bank grid.

Physical Node Management: The distributed computing environment presents numerous unique challenges to providing a reliable computing capability. Individual client machines are under the control of the desktop user or IT manager. The Physical Node Management layer manages these and other low-level reliability issues. The resource management services capture a wealth of static and dynamic information about each physical node (e.g. physical memory, CPU speed, disk size, available space, client version, data cached, etc.), reporting it to the centralized node manager.

Resource Scheduling: The distributed computing system consists of resources with a wide variety of configurations and capabilities. The Resource Scheduling layer accepts units of computation from the user or job management system, matches them to

appropriate client resources, and schedules them for execution. Despite the resource conditioning provided by the Physical Node Management layer, the resources may still be unreliable (indeed the application may be unreliable in its execution). Therefore the Resource Scheduling layer must adapt to changes in resource status and availability and to failure rates that are considerably higher than in traditional cluster environments.

Job Management: A distributed computing application often involves large amounts of computation submitted as a single large job. This job is then broken down into a large number of individual subjobs each of which is submitted to the grid for execution. The Job Management layer of the system is responsible for decomposing the single job into the many subjobs, managing the overall progress of the job and aggregating the results of the subjobs. This layer allows users to submit a single logical job (for example, a Monte Carlo simulation, a parameter sweep application, or a database search algorithm) and receive as output a single logical output. The details of the decomposition, execution and aggregation are handled automatically.

User Interface: This layer abstracts the complexity of the below layers from the users, and provides a graphical user interface to the user. The actions performed by the user will be transformed to system understandable calls and passed to the lower layers. User can view the status of jobs, resources and output of the jobs. User can see the resource status from other pools in the grid.

The complete architecture and important components of the system are shown in figure 3.3. The central managers and web servers are located at each of their bank centers. The web server hosts the grid portal, which provides and manages access to resources in grid. Every bank has its own grid portal. The Grid authorization framework provides security services.

The MDS (Monitoring and Discovery service) is responsible for updating the node status to the central manager. And local MDS can query MDS of other pools to get the status information. In the figure the components of inter-bank grid (two banks) has shown

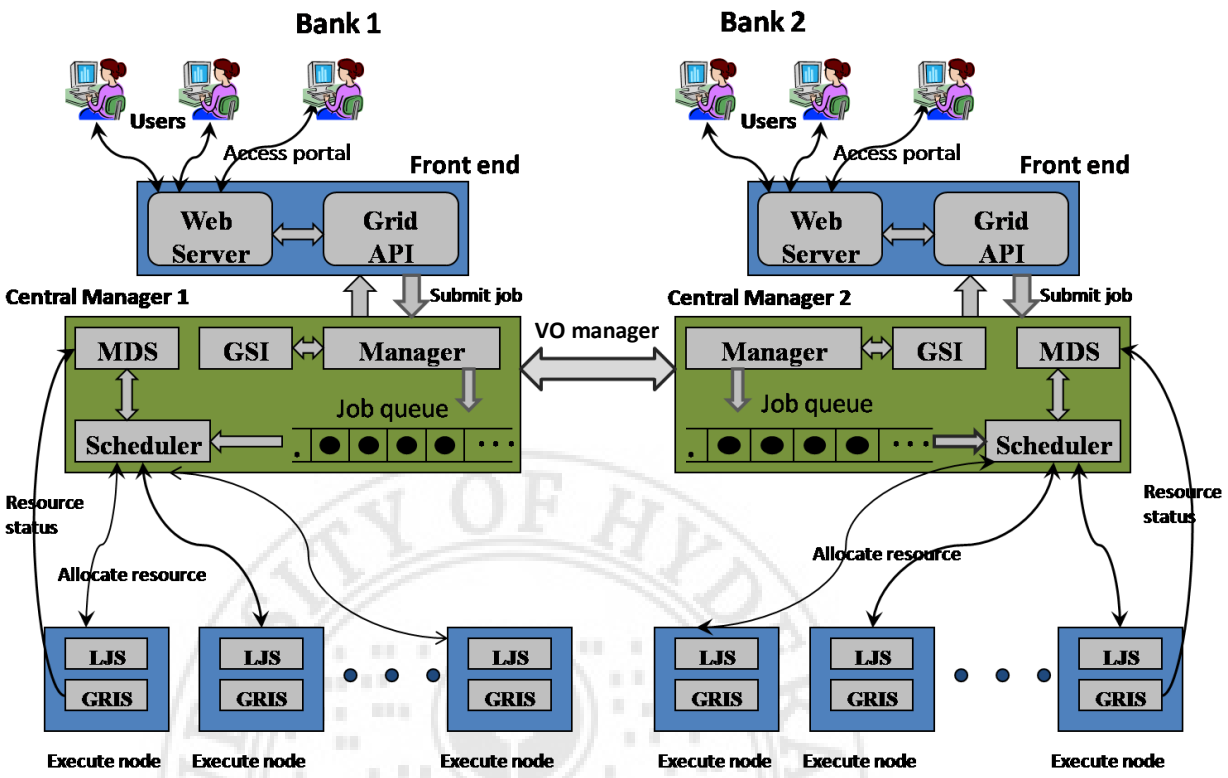


Figure 3.4: Inter-Bank Grid Architecture

3.4 Overview of Components:

We have already discussed about each of the component in the previous chapter.

VO Manager: The VO manager is the heart of the inter-bank grid, as all the central managers are communicated through this service. It provides transparency to the Job scheduler, Authorization service, Monitoring and discovery services hosted at each central manager. The VO manager maintains a list of other pools, which are allowed to use resources of local pool. It only accept request and give access to authenticated central managers.

Grid Authorization System: Grid Security Infrastructure (GSI) provides fundamental security services needed to support grids. It is based on Public Key Infrastructure. The users and hosts authenticate to the system using X.509 certificates. Along with basic security features, the digital certificates can be used in real time to strengthen the security

in inter-bank grid. The central authority can act as a certifying authority to issue certificates to the participating entities.

3.5 Various issues in implementation

To implement the proposed inter-bank architecture in real time, there are many issues that need to be addressed.

Security: Banks give first preference to data security, because it involves large number of customers. When sharing resources with other banks, care should be taken. The system should be designed in such a way that the jobs running on the remote nodes must be restricted access to local file system and confined to its environment only. The sandboxing technologies should be used, in which the above requirements are met.

Resource utilization: The quantity and type of resources that each bank has is different from each other. Some banks are very small compared to other banks. So some banks may not wish to share their resources with smaller banks, which have less number of resources to contribute. All these factors play a major role in implementation of grid.

3.5 Conclusion

In this chapter, we have proposed Inter-Bank Grid architecture which enables banks to share their resources with other banks securely and efficiently. We analyzed different aspects of the system. We have discussed environment from resource sharing to job scheduling and also discussed about various components. Also we have discussed various issues in implementation and role of central authority to manage the grid.

CHAPTER 4
DEVELOPMENT OF BANKING GRID PORTAL

4.1 Introduction:

A Grid portal is a user's point of access to a Grid system. A Grid portal is a web server that provides an interface to Grid services, allowing users to submit jobs, transfer files, and query Grid information services from a standard web browser. Portal hides low-level grid access mechanisms by high-level graphical interfaces, making even non grid expert users capable of executing distributed applications on multi-institutional computing infrastructures.

Need for the Portal: All the services provided by the Grid middleware are currently exposed to the users through rather complex Command Line Interfaces (CLI). There are tens of different commands with many options and rigid sequences and the description of jobs has even a dedicated language (the Job Description Language or JDL) to be learned. All this has the negative effect of discouraging many potential users from learning how to profit from the Grid advantage. So to make it easier for normal users, encourage them to use grid services, portals have been developing. The proposed web based application portal would serve as an intermediary between the user and the grid.

The primary requirements for a Grid portal system from a user' perspective involves access to Grid services. These include

Security services: Users will log onto a portal using a web browser and they will authenticate by means of a user-id and password. While better technologies exist for authentication, this one is what user want and trust. More secure systems, such as smart cards are possible and choosing the access method depends on the level of security need to be provided. Once the user is authenticated to the portal server, it is the job of the portal server to act as the user's proxy in most grid interactions.

Remote Job Management: The ability to submit jobs to the Grid for execution and monitoring is a classic service provided by portals. User can monitor the current status of a job. User can terminate the jobs in middle and see present the status of the job. Users can submit many jobs to the grid. And many users can access the portal simultaneously.

Access to Information Services: The portal also provides information services to the user. The resource status like total available resources, allocated resources and number free resources in the grid, the jobs running on the grid, total jobs, jobs which are waiting in queue etc., User can monitor the current status of a job. The portal also shows the time taken to complete the job.

Application Interfaces: The key to portals is being able to hide Grid details behind useful application interfaces. The user needs to be able to launch, configure and control remote applications in the same way he or she uses desktop applications. The portal must provide a way to easily integrate new application interfaces and expose them to the user and hide the details of Grid middleware.

4.2 Design of Banking Grid Portal

The objective of developing a portal is

- To provide an easy-to-use interface to the applications on the grid.
- To set focus on the applications and hide the specifics of grid computing from the user to the extent possible.

As shown in the figure 4.1, the conceptual model of portal, The portal fills the usability gap between the user and the grid resource. The portal provides a simple interface to the user, where he/she can submit the job they wish to execute. The job is then written to a job description file and sent to the grid scheduler. The scheduler decides where and when the job is executed, depending on resource availability. While the job is running, the user can see status updates about the job. Once it is finished, the user can retrieve the results through the portal.

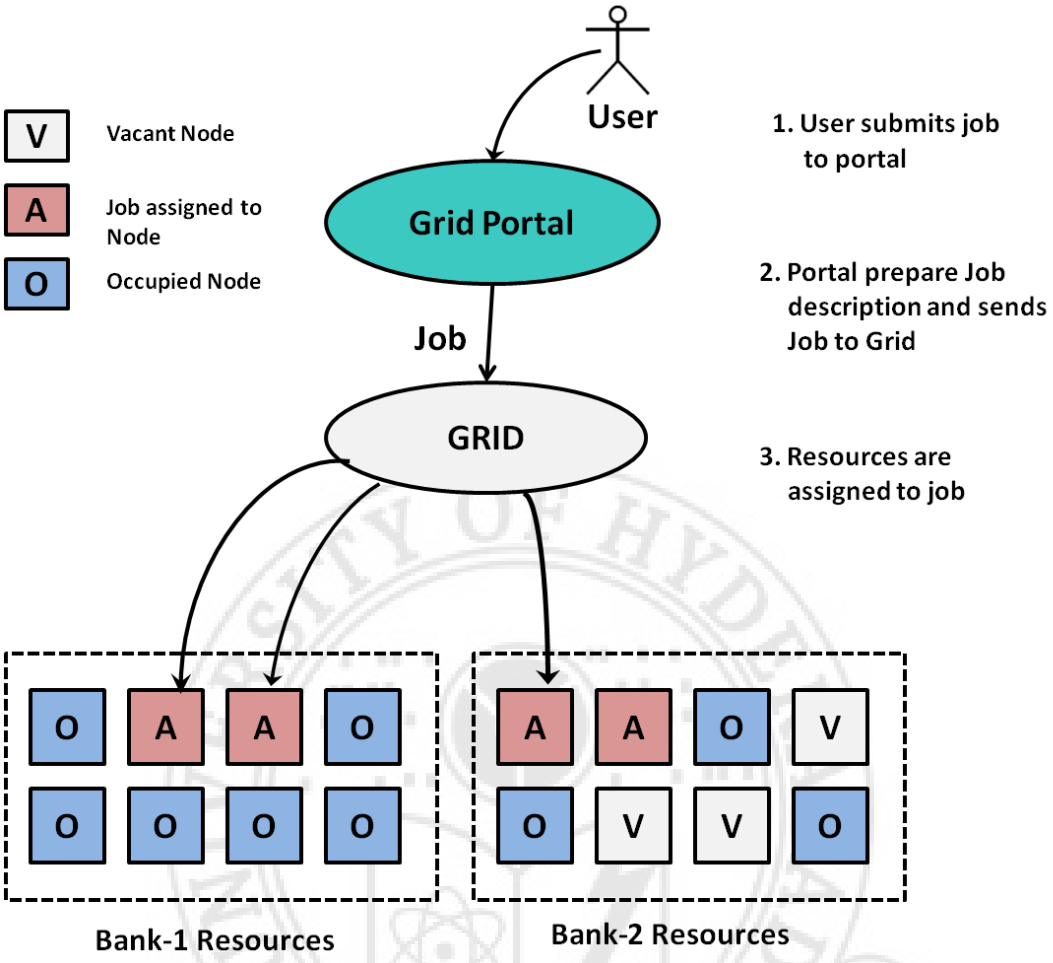


Figure 4.1: Conceptual model of Grid Portal

4.2.1 Components of the Portal

The portal can be seen as three-tier architecture. It consists of Web browser or client, Web server and Grid middleware services layer. The web server is the interface between the User and Grid Middleware services. GridAPI provides methods to communicate with the grid services. All these components hide the complexity from the user and provide a simple interface to access grid services.

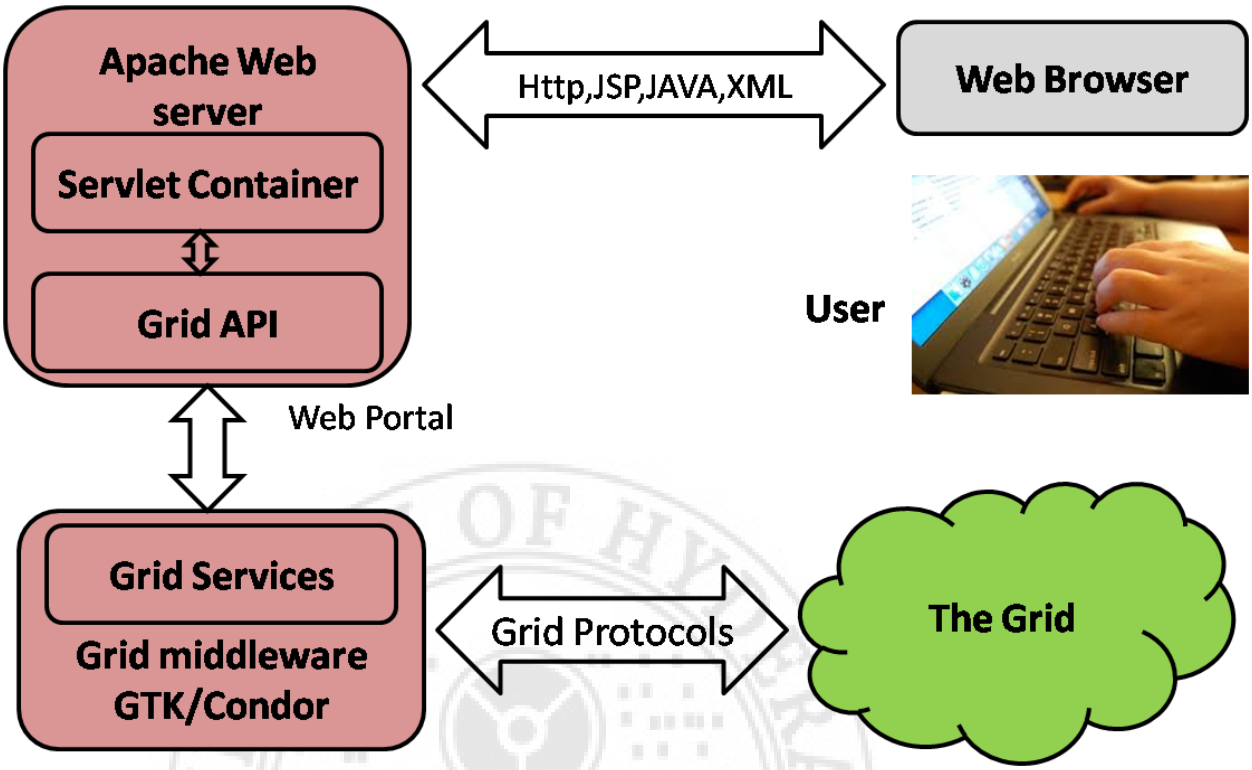


Figure 4.2: Components of the portal

4.2.2 Overview of Components

The build the Grid portal the following components are needed

Web browser: A web browser is a software application for retrieving, presenting, and traversing information resources on the web server. The primary purpose of a web browser is to bring information resources to the user. The browser uses HTTP protocol to communicate with the web server. The user requests a page or resource on the server. The server returns the requested page using the same protocol. We have tested the portal on the Mozilla and Internet Explorer browsers.

Web server: A web server can be referred to as either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed through the Internet. The primary function of a web server is to deliver web pages on the request to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, files and media files. A client, commonly a web browser, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. We used Apache Tomcat 6.0 Web server to host the portal.

Grid API: An application programming interface (API) is a particular set of rules and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers. To convert the user calls received from the portal into the grid service calls, we are using GridAPI provided by Globus and Condor systems. This acts as glue between the front end layer and complex back-end grid services.

Grid services: Making use of GridAPI we have developed some grid services can accessed by the user through the portal. The services are job submission to the grid, monitoring the resources of grid, querying the status of jobs and getting the results of the jobs. These services make users working with the grid easier as the jobs are running in their workstations.

4.2.3 Control flow Diagrams

- **Complete portal and interaction between each component**

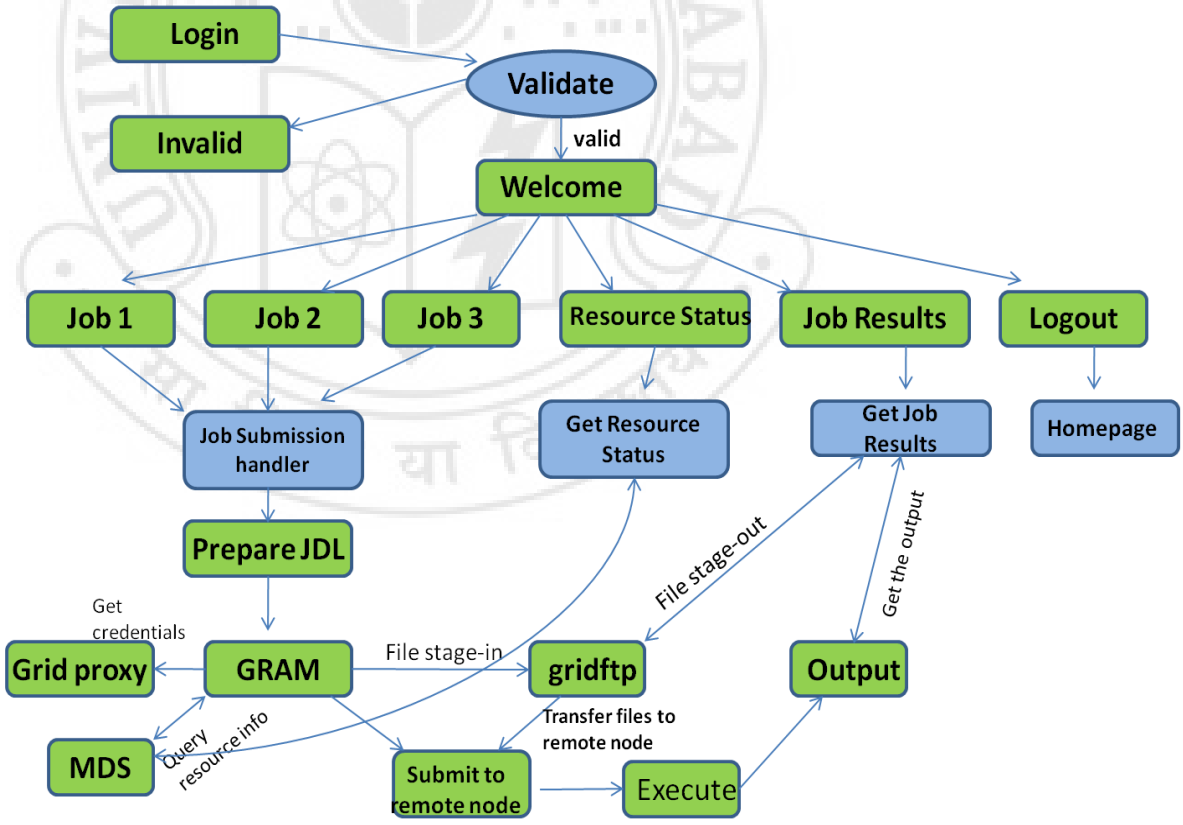


Figure 4.3: Complete portal flow diagram

- **User authentication phase**

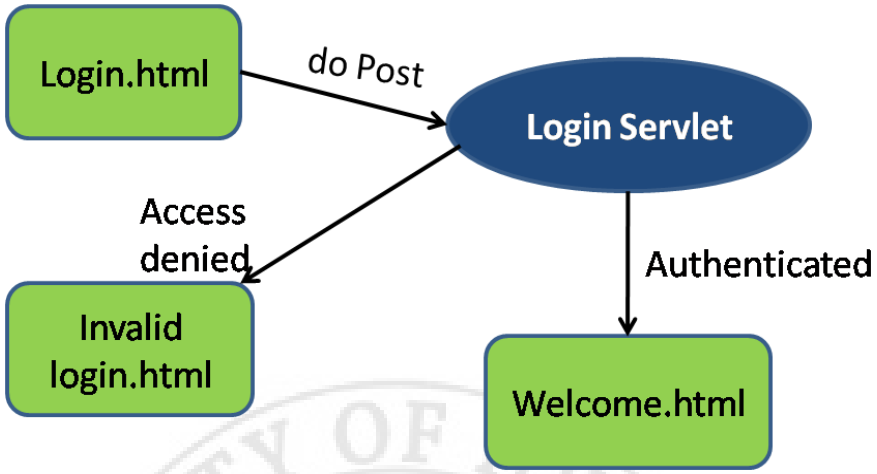


Figure 4.4: Grid portal login flow

- **Grid application submit phase**

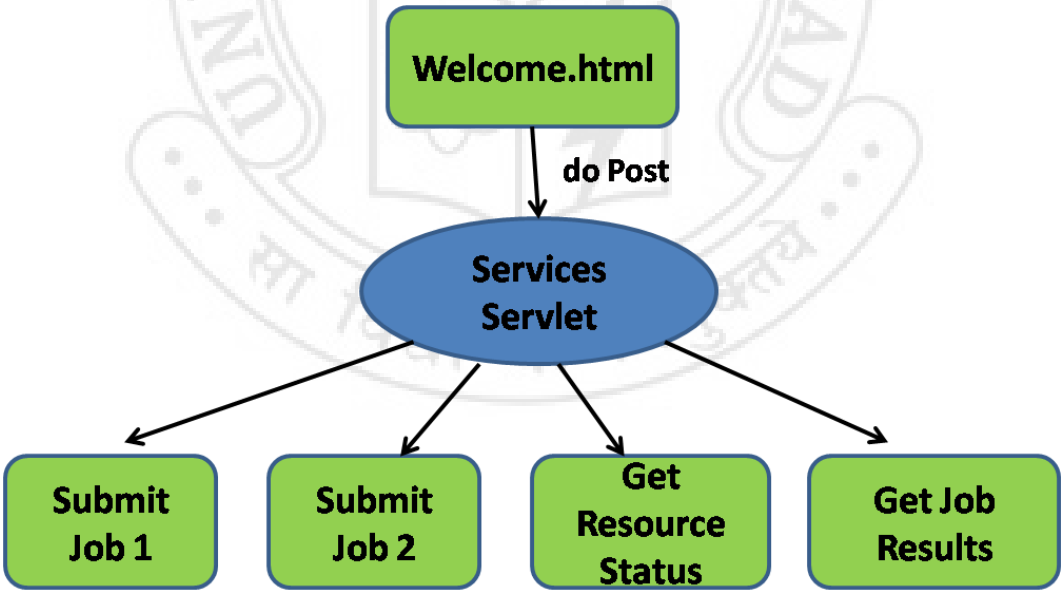


Figure 4.5: Grid portal application flow

Use case Diagram of Banking grid Portal:

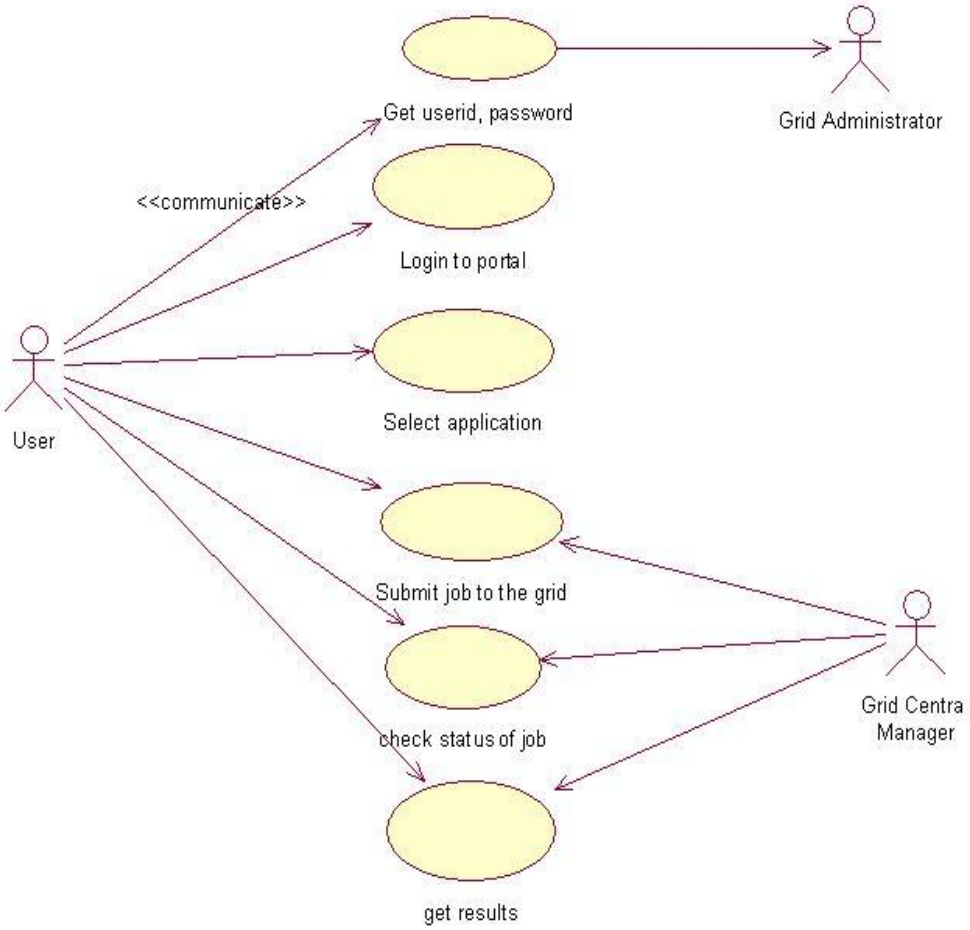


Figure 4.6: Grid portal use case diagram

User gets the username and password from the grid administrator. Then he will login to the portal using the same. The portal validates the user and grant access to the services of the portal. The portal forwards the request to the grid central manager. The central manager does the actual job submission, resource allocation and aggregation of results.

Sequence diagram of Banking grid portal:

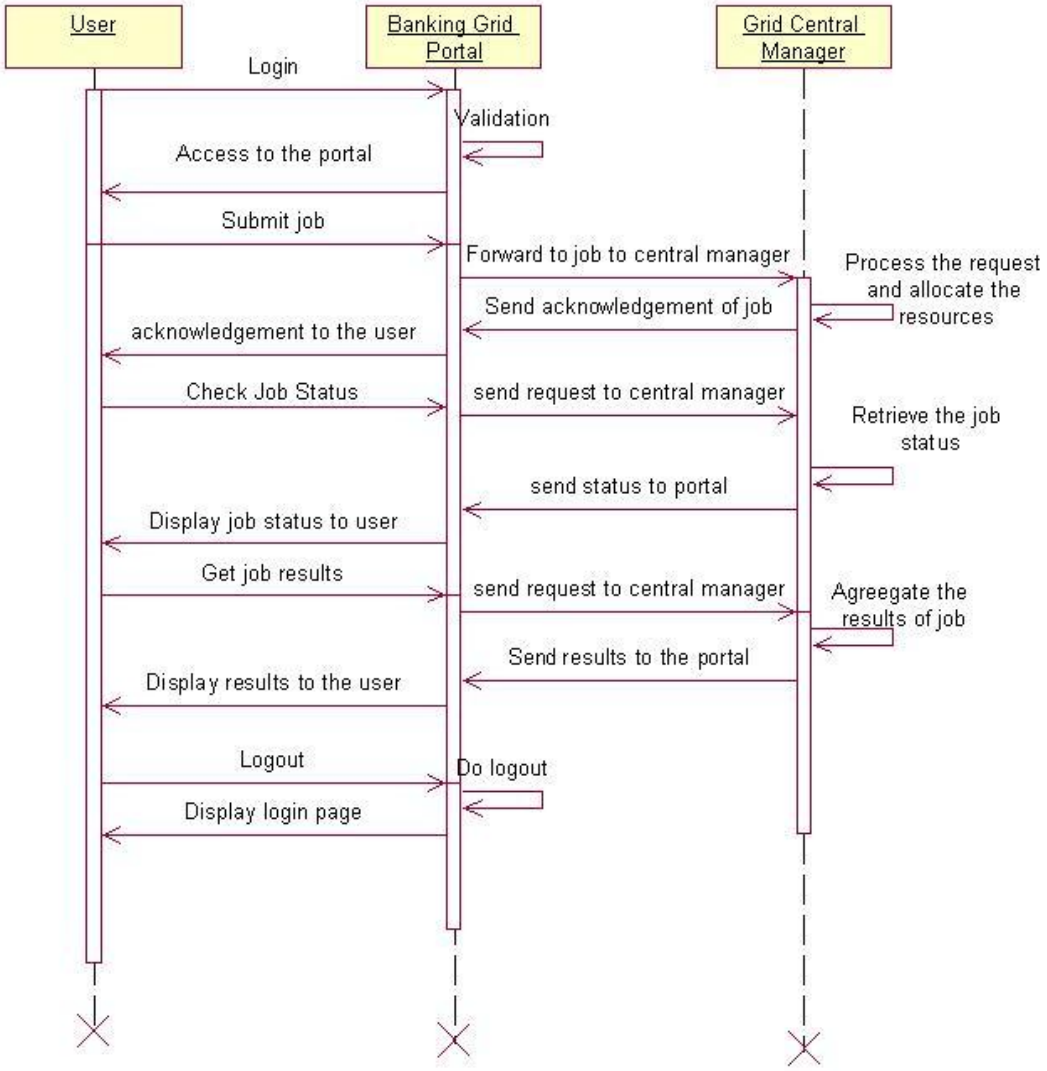


Figure 4.7: Grid portal sequence diagram

4.3 Implementation of Bank Grid Portal

The system requirements for the portal

Linux Operating System: Grid Portal was designed and tested on Linux but works as well on other platforms, perhaps with some minor changes. We used Linux Ubuntu 10.1 version.

Apache Tomcat 6.5: Apache tomcat web server was used for hosting the portal. It is commonly used open source web server. The portal can be hosted on other web servers also with minor configuration changes. Other popular web servers include Microsoft IS server, Web Sphere from IBM etc.

JDK 1.6: Java development kit 1.6 was used for programming the services in the portal. As java is platform independent, it is possible to run the services on any platform. We have tested the developed portal functionality on both Linux and Windows platforms.

MySQL: To store the persistent information of users and jobs we need database. MySQL is a most popularly used and supported open source database server. The usernames and passwords of the users are stored in users table. To store the information about jobs submitted by the users, a jobs table was created.

GTK with Condor: To setup the grid we have used popular grid middleware tools Globus toolkit and Condor system. These provide the basic services of the grid. These are developed by the open source communities and used worldwide in many grid projects. The services like resource management, job management, and file transfer are provided by these tools.

4.3.1 Services offered by the portal

The services offered by the portal are

- Remote Job submission and termination
- Job monitoring
- Resource monitoring
- Getting the results of job

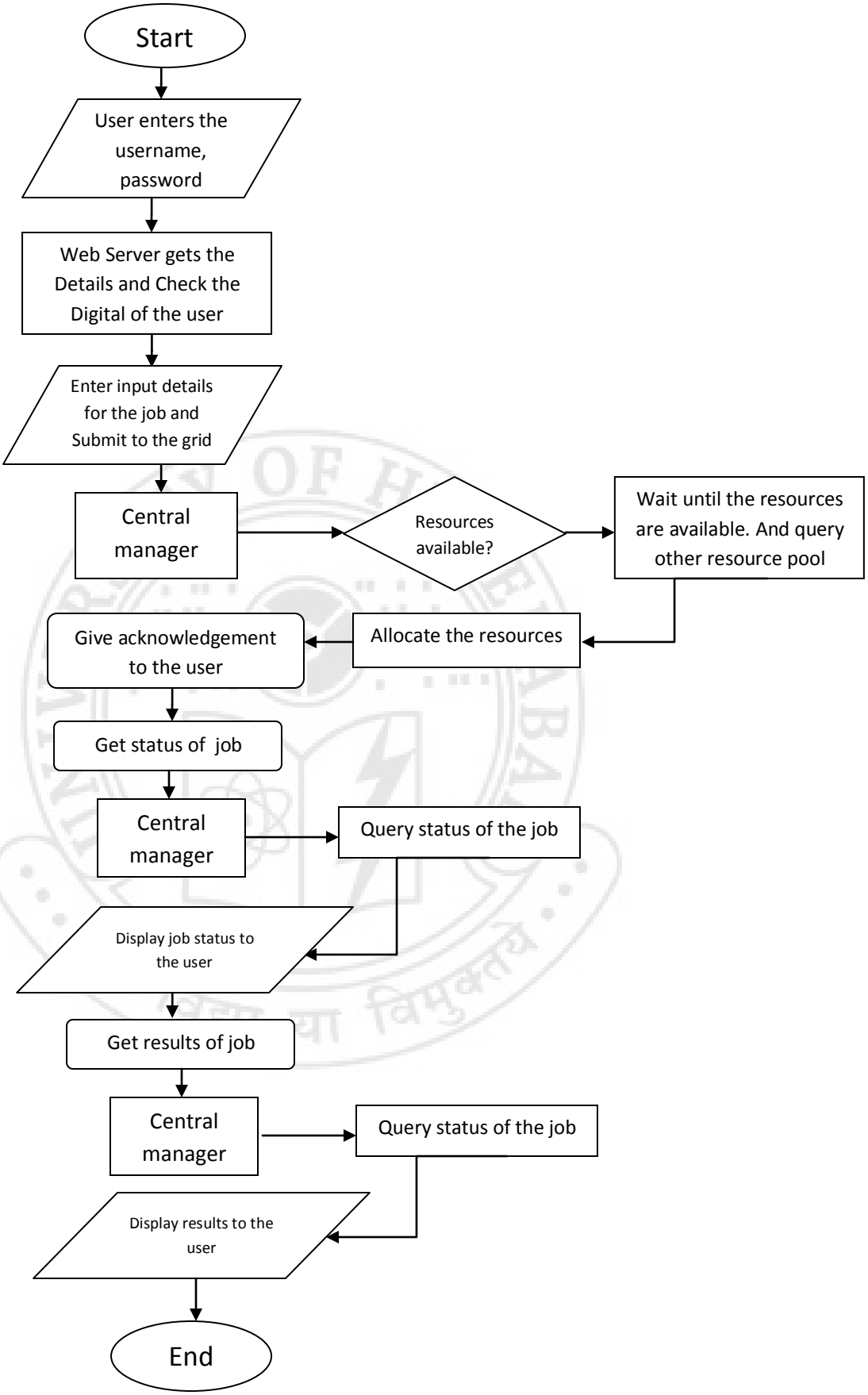


Figure4.8: Execution flow diagram

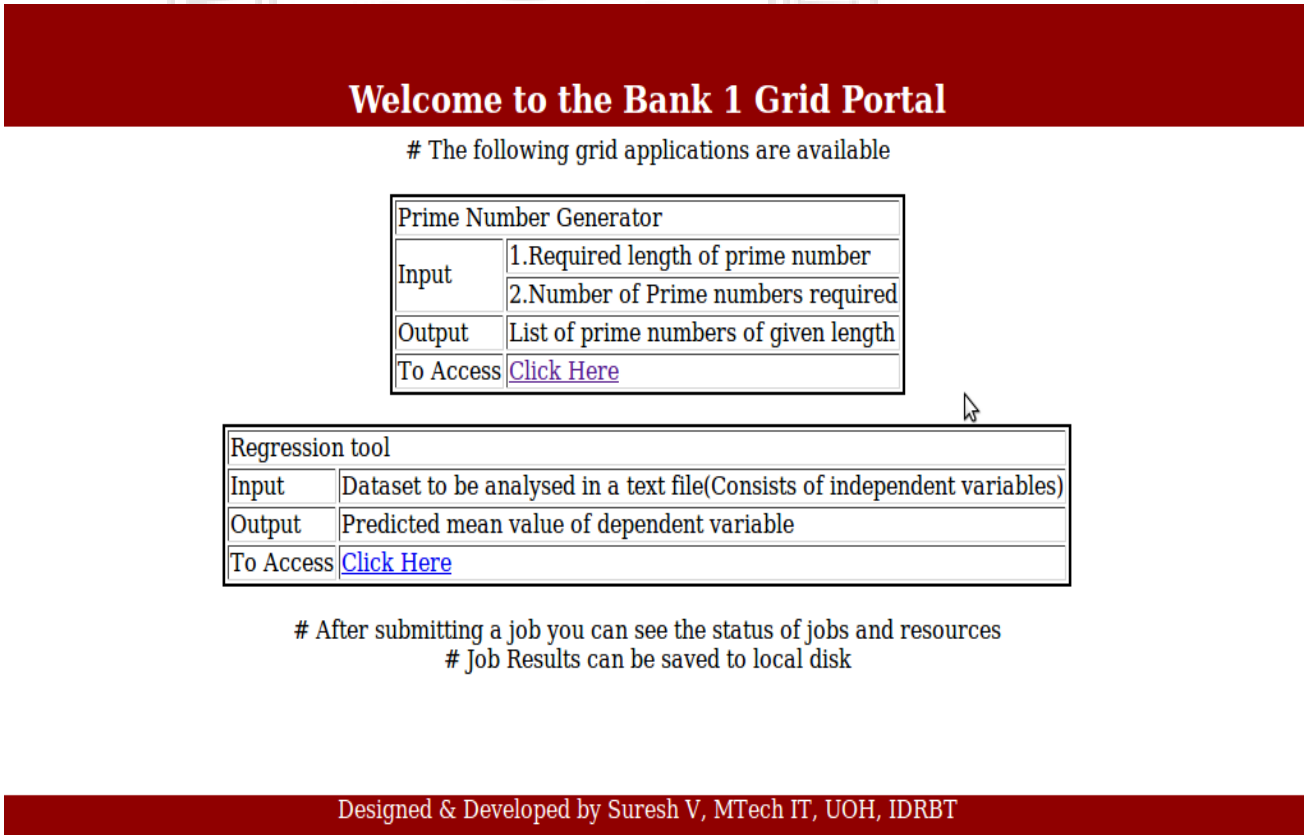
4.3.2 Accessing portal

To access the portal user need to follow below steps

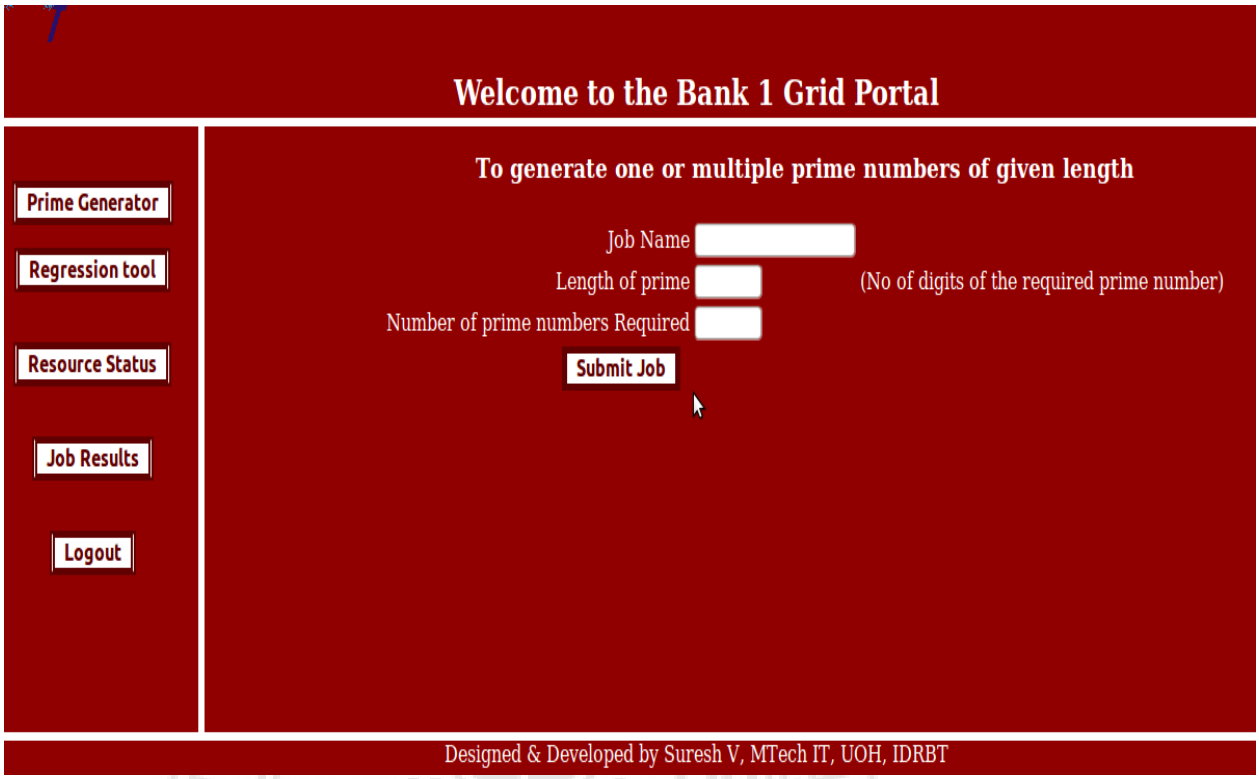
1. Open a web browser (Mozilla, Internet explorer, Chrome etc)
2. Enter the portal URL in the browsing tab as
   http://172.16.0.86:8080/gportal
3. User will be presented with Login page; here user will enter his valid username and password allotted to him by the administrator.
4. After successful login, portal home page appears with list of services and their description offered by the portal.
5. By clicking on appropriate service URL, user will be redirected to that service home page
6. On the service home page user can give input arguments, any input files etc required by the job.
7. After clicking on the "Submit job" button, user will get an acknowledgement about the successful submission of the job to the grid.
8. User can see the status of the resources and jobs running on the grid by clicking on the "Resource Status" button on the page. It will give the information about total available nodes, number of busy nodes and number of free nodes. Also it shows the total number of jobs, running jobs and number of jobs waiting.
9. User can see the status of his job by clicking on the "Job Results" button. It will show the status of each job user has submitted. If the job completed then the result link will be appeared to the user. User can save the results to his local disk. If the job is not yet completed, then the status will be shown as 'Running'.
10. User can terminate the jobs submitted by clicking on "Delete" button in the results page.
11. User can see how much time the job has took to complete on the results page. Both submission-time and completed time of the job are shown.
12. By clicking on "Logout" button, the user will be logged out of the portal.

4.4 Screenshots

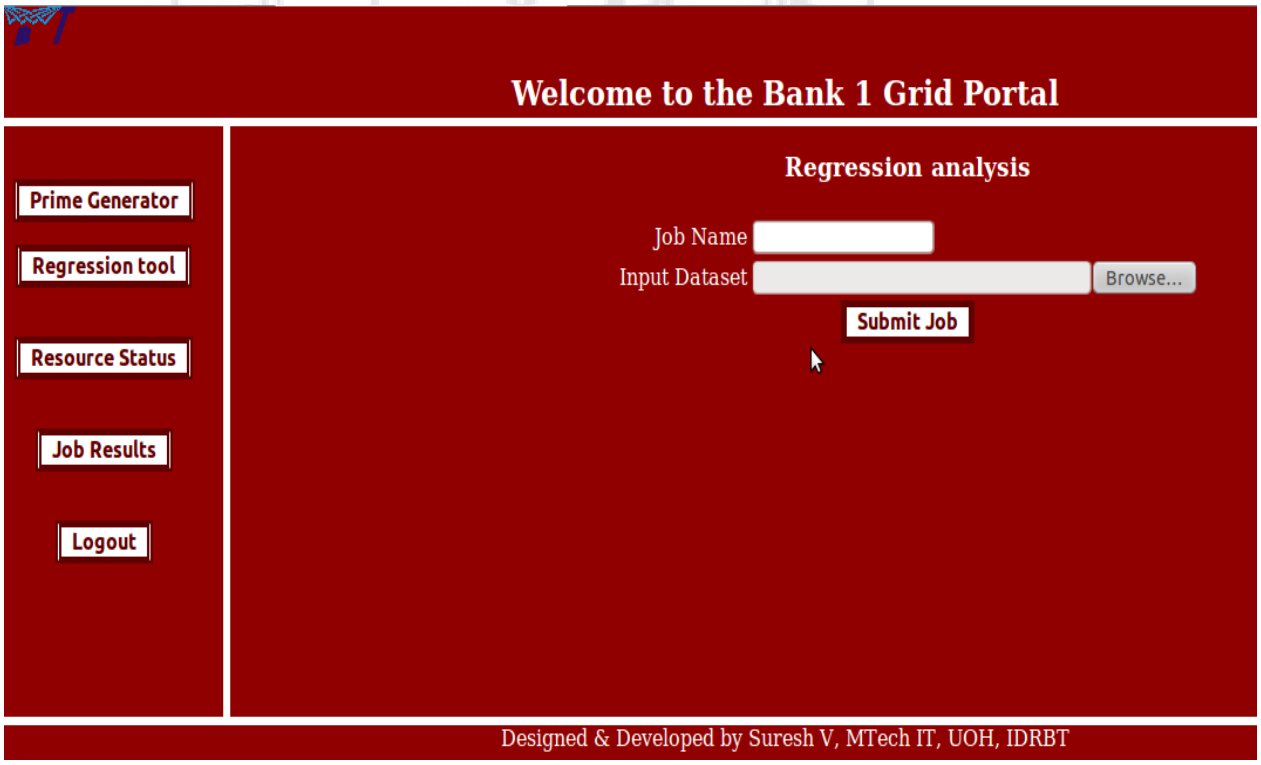
Portal Login Page



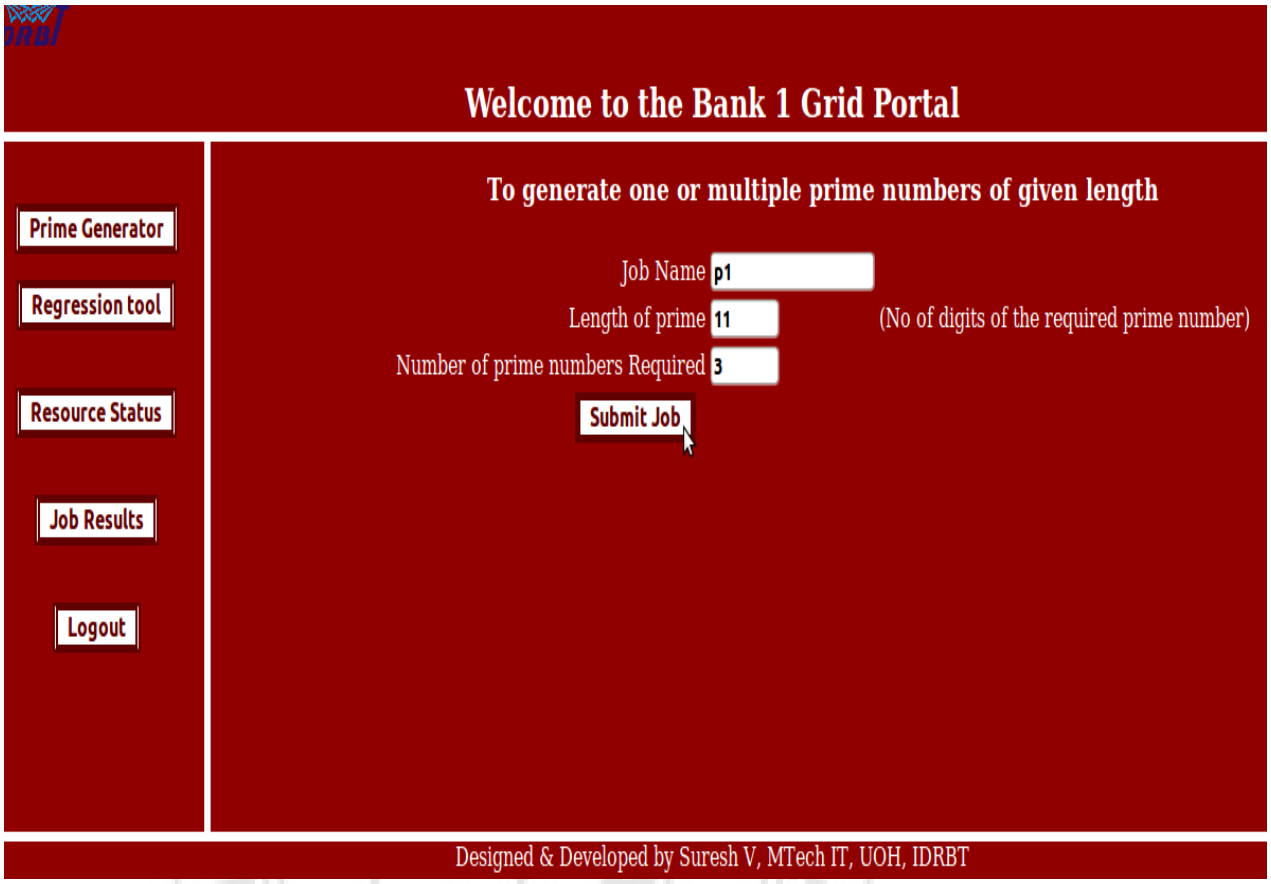
Portal home page



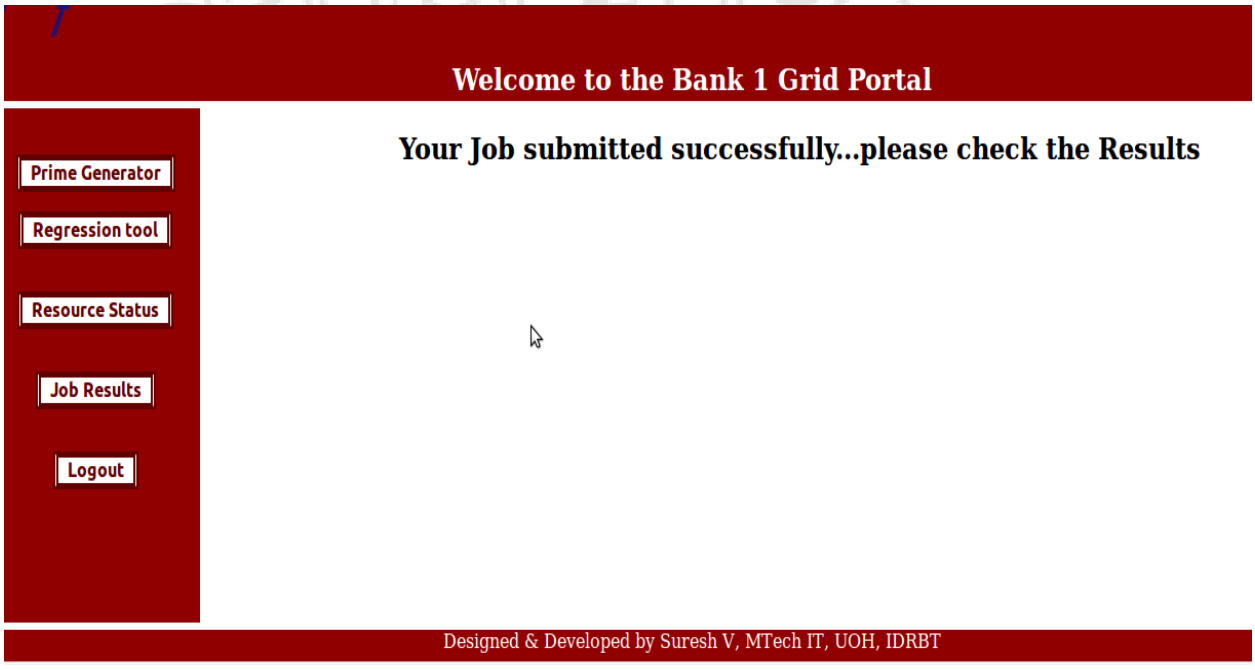
Prime number generator home page



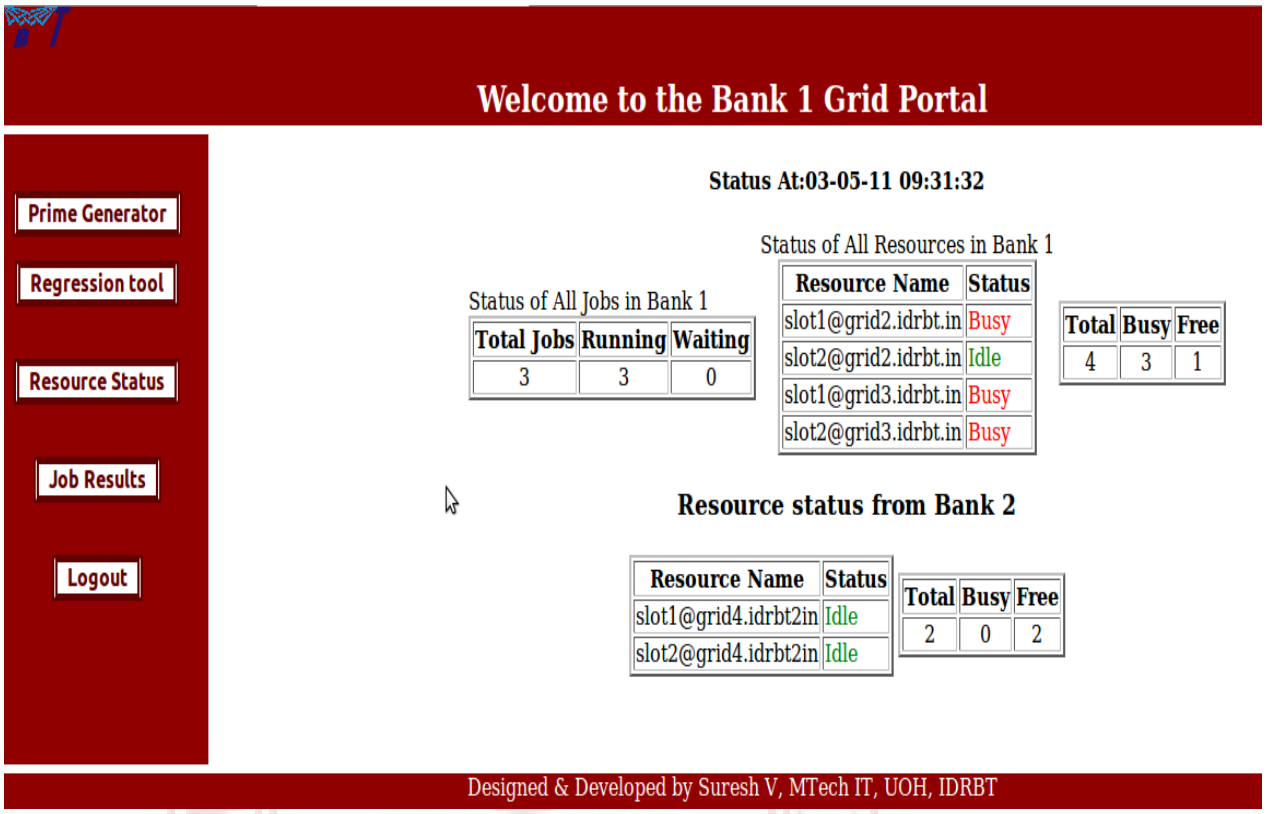
Regression tool home page



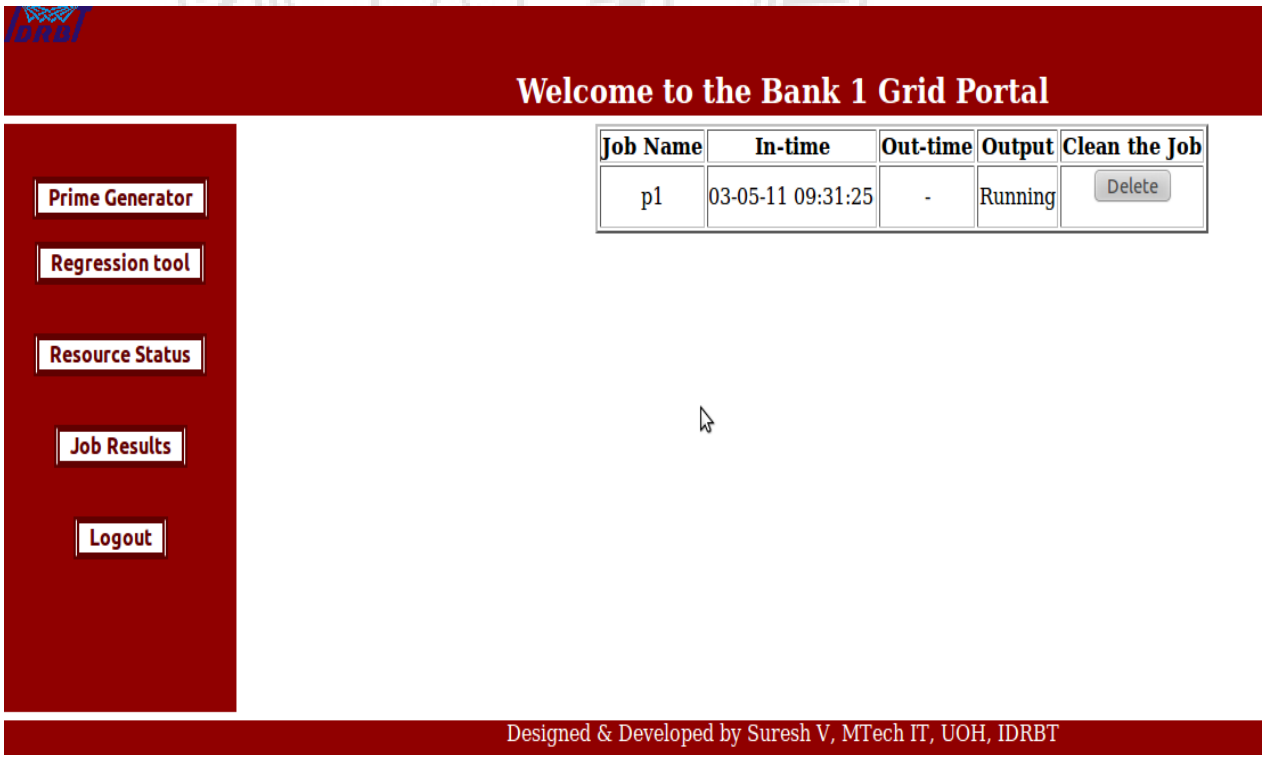
Input to the prime generator application



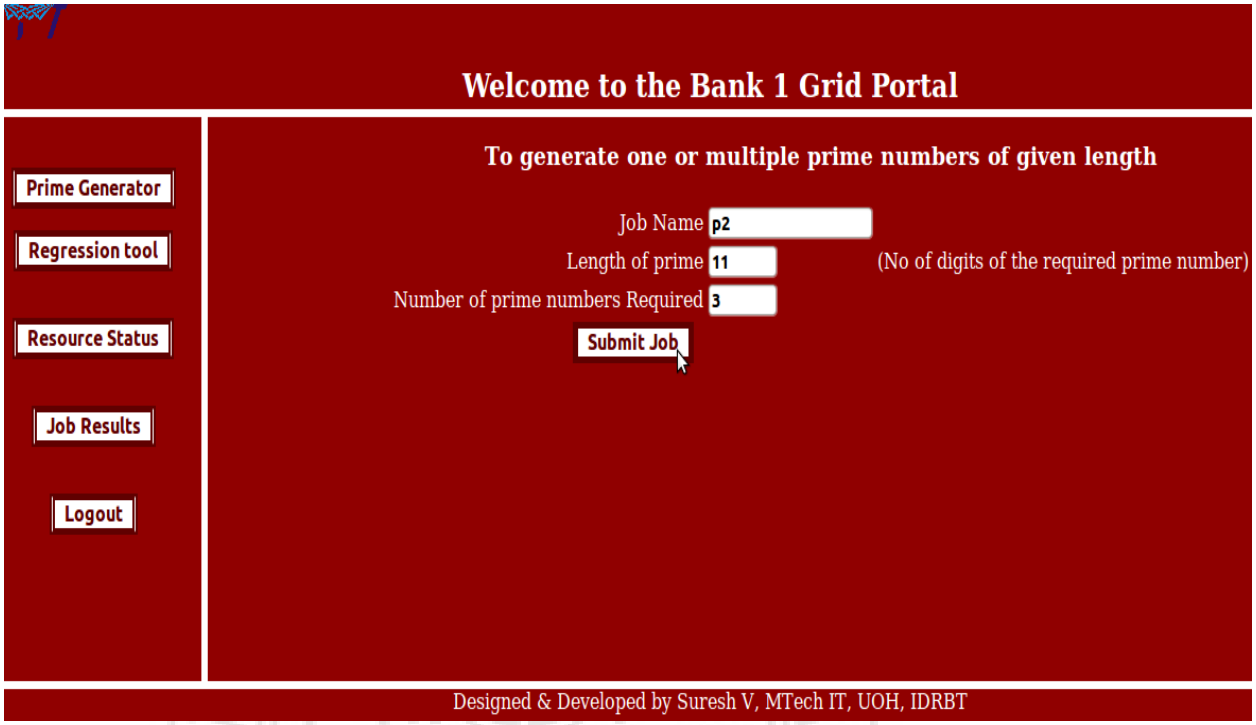
Acknowledgement after submission of job



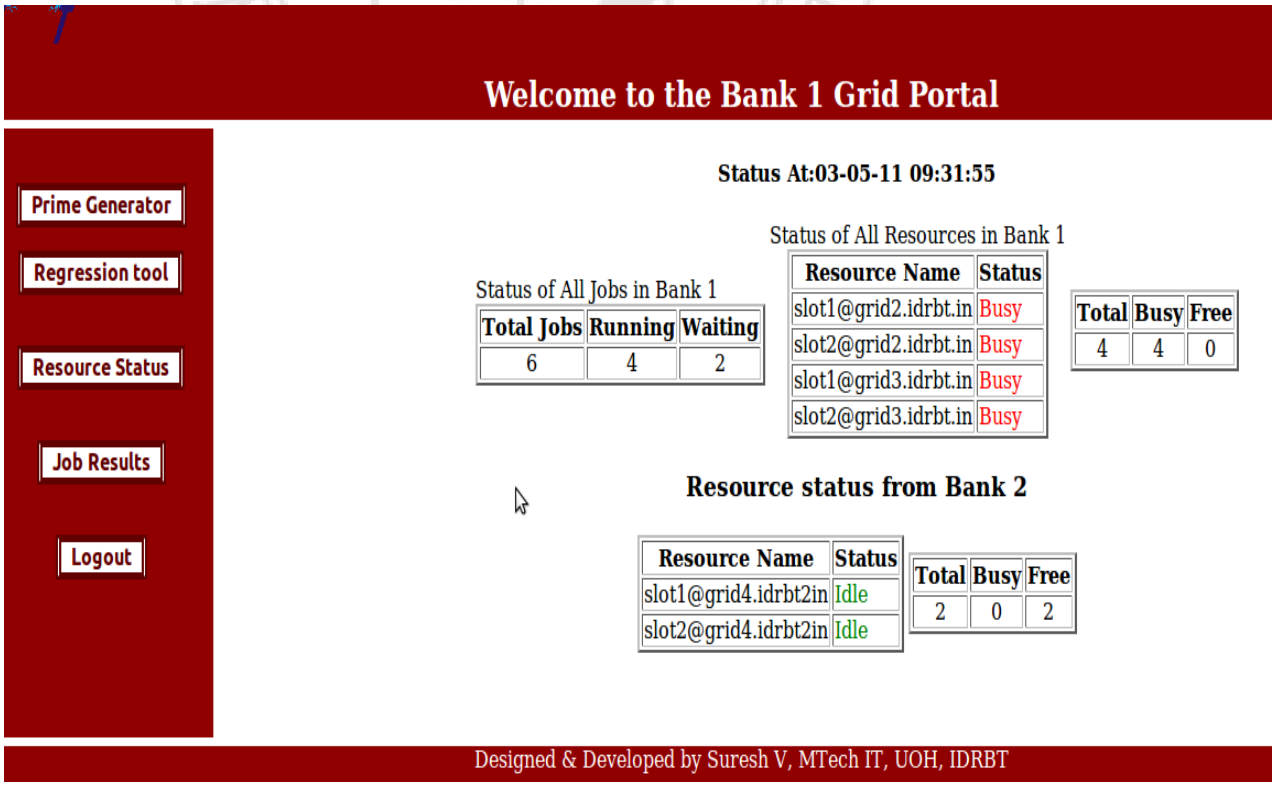
Resource status after submitting prime job with 3 instances



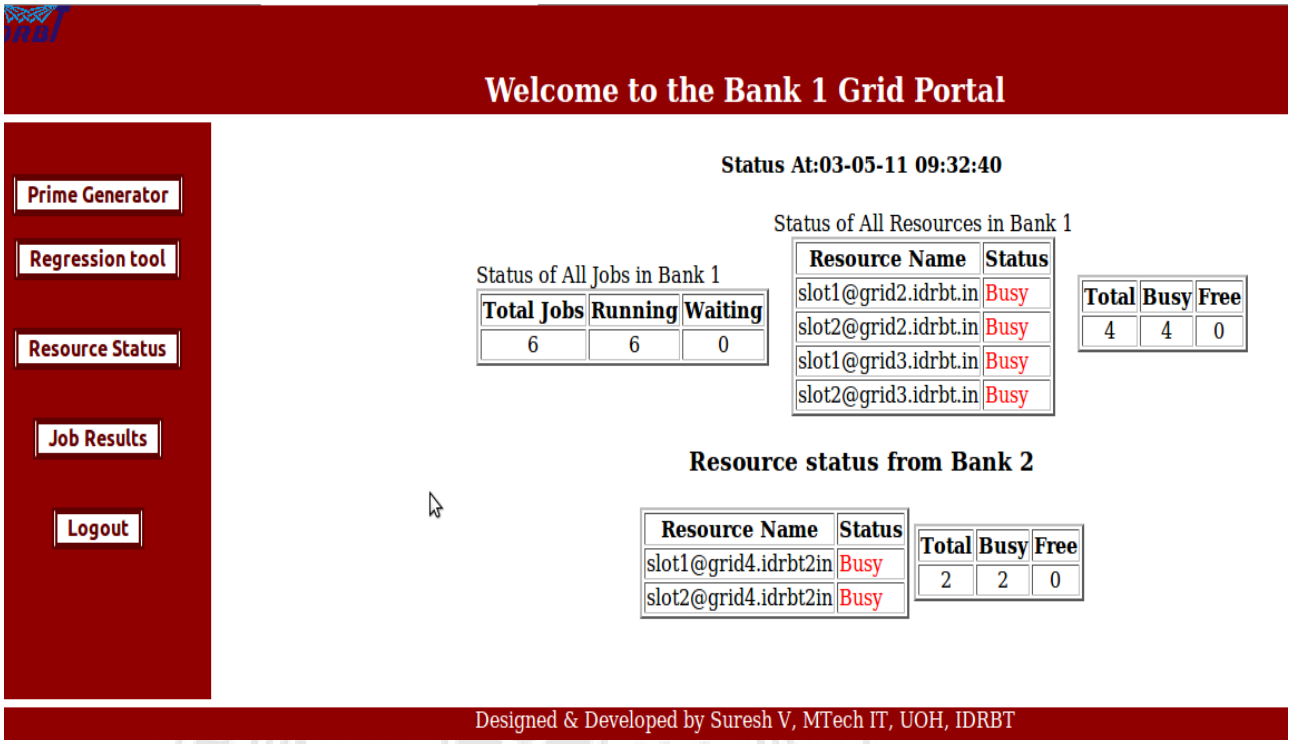
Job results showing the status as 'Running'



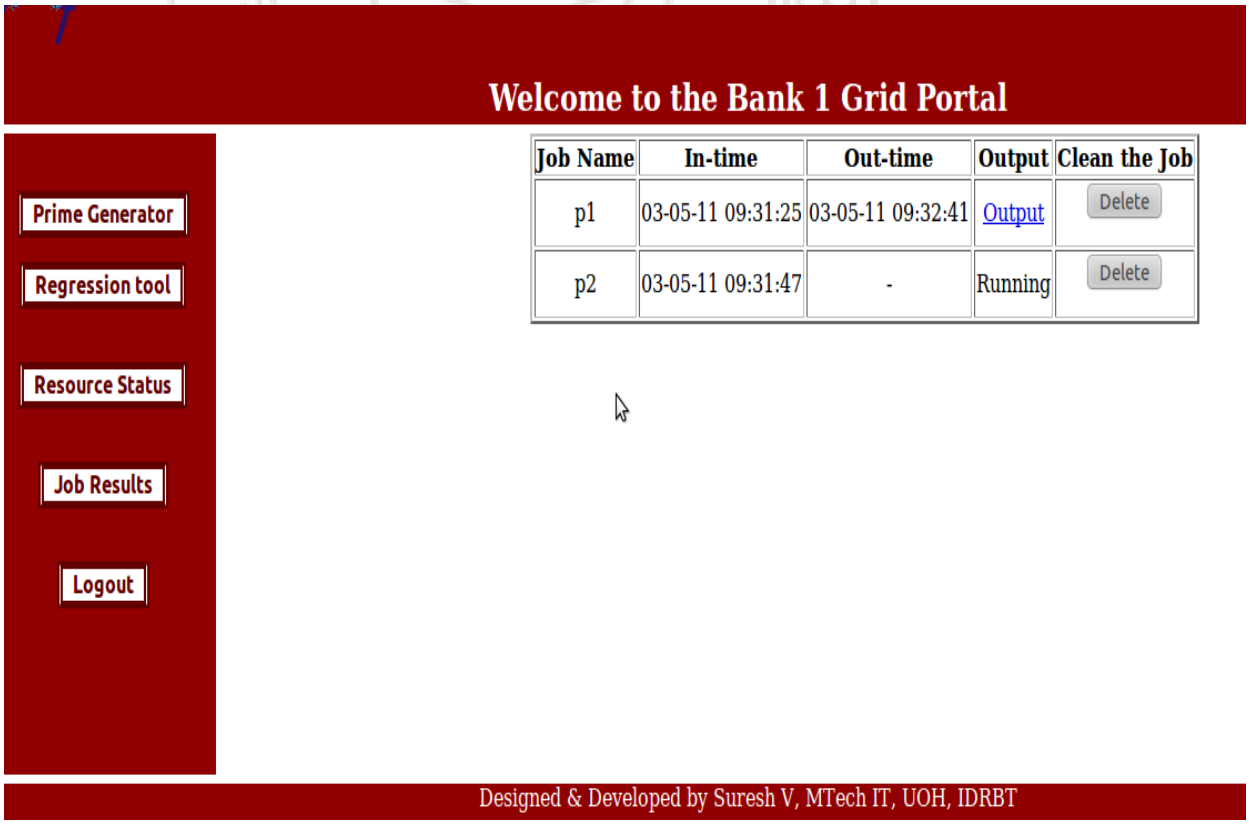
Submitting another prime job with 3 instances



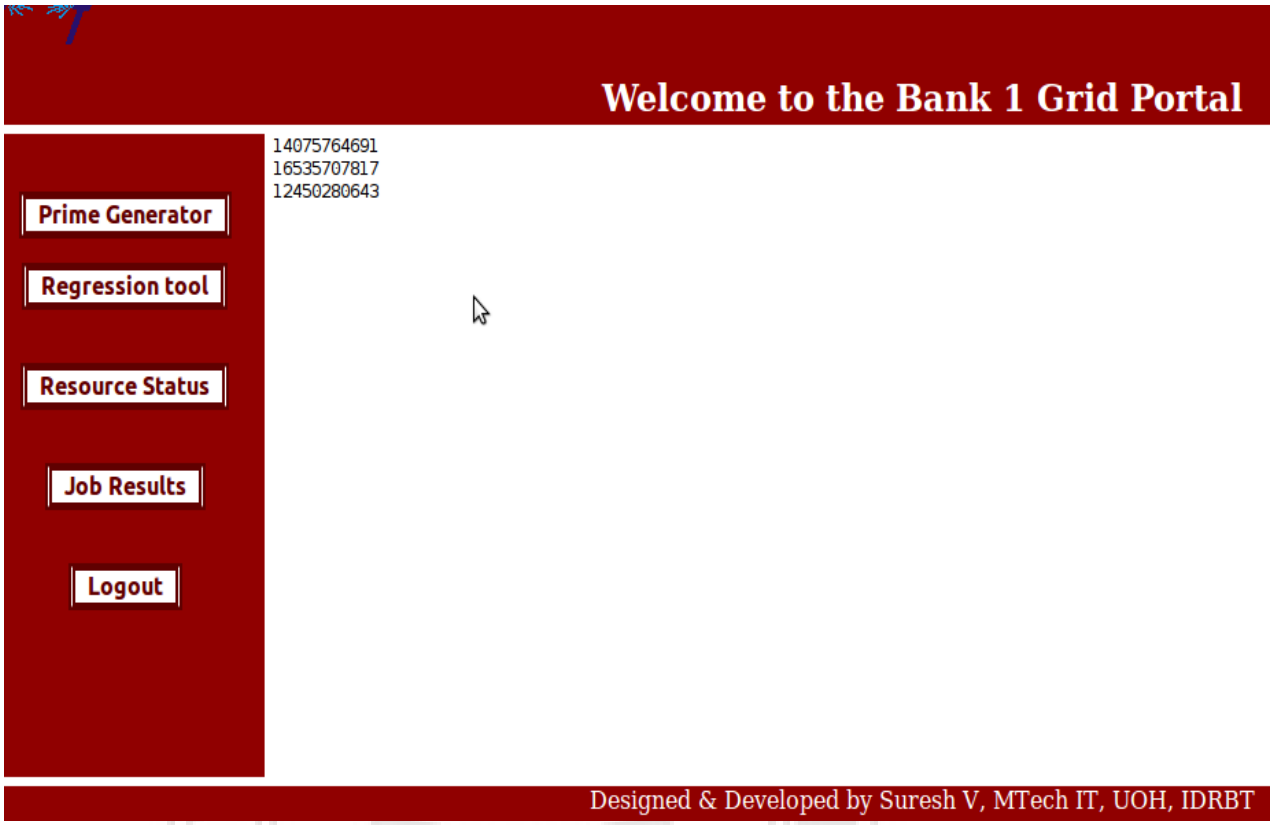
All the Resources in the local pool are busy



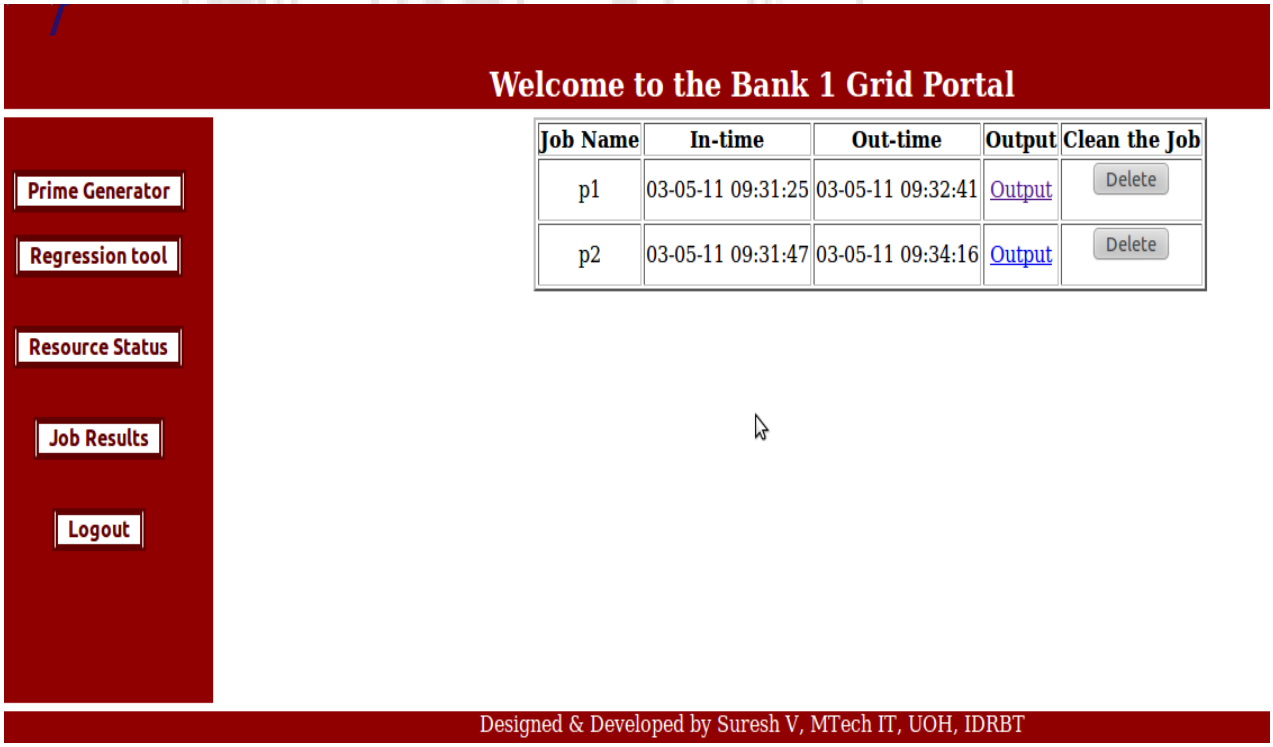
Resources are allocated in the other pool



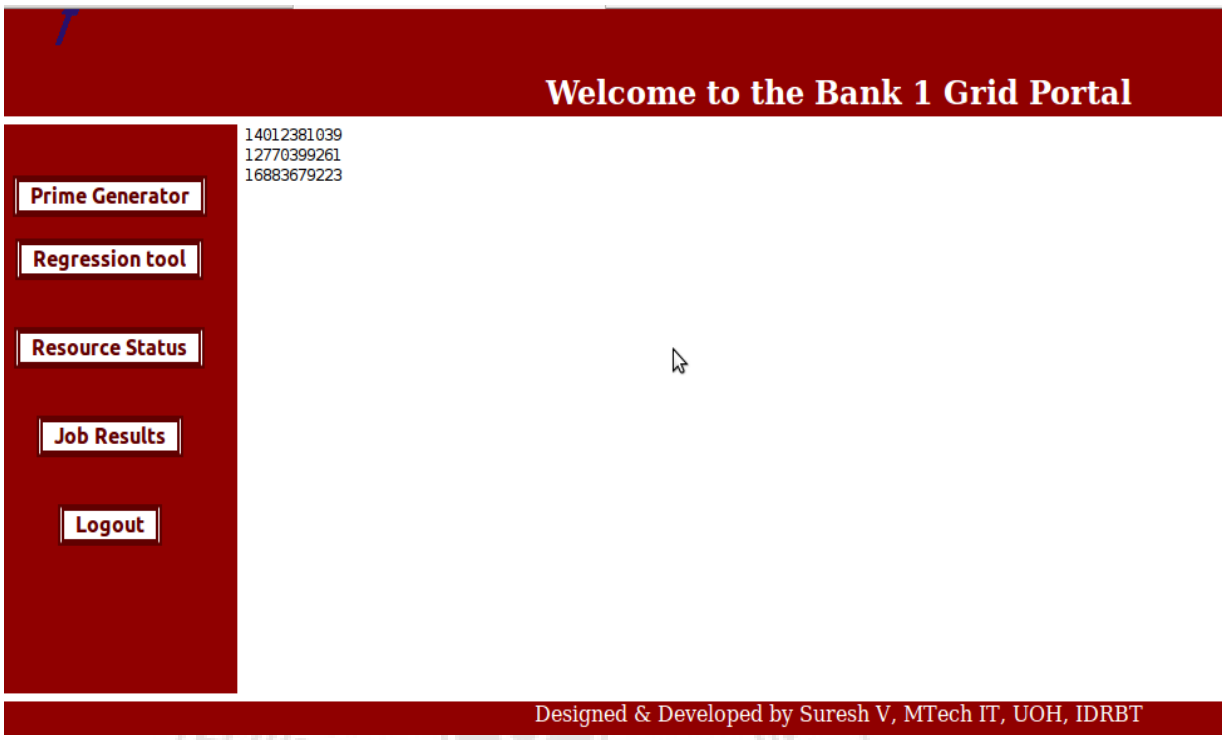
Status of the jobs, first job has been completed



Accessing the results of first job



Both the jobs are completed



Accessing output of the second job

4.5 Conclusion

In this chapter, we have discussed about the grid portal and implementation aspects of the portal along with their advantages in accessing the resources in grid. We analyzed different aspects of the portal. We have also discussed about various components of the portal. We have developed one application which runs in parallel on the grid using multiple nodes.

CHAPTER 5
CONCLUSIONS

Conclusions

Banking Grid is an integrated infrastructure that enables collaborative use of computational resources such as servers and work stations, databases, core banking software, security services and information storage platforms etc. owned and managed by multiple banks. When the resources are shared among multiple domains, coordination of the resources becomes the most important issue for efficiently using and managing recourses across domains. In this project we have proposed Banking Grid architecture and developed a working prototype, which enables coordinated sharing of underutilized resources in banks.

Also a Banking Grid Portal is developed through which all the authenticated users can access resources online and also make use of various grid services. Developed and integrated one application which will run in parallel on different nodes of the grid.

BIBLIOGRAPHY

[1] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Int'l Journal of Supercomputer Applications and High-Performance Computing*, vol. 15, no. 3, pp. 200–222, 2001.

[2] Fotis Georgatos, Vasileios Gkamas, Aristeidis Ilias, Giannis Kouretis, Emmanouel Varvarigos, "A Grid-enabled CPU Scavenging Architecture and a Case Study of its Use in the Greek School Network", *Journal of Grid Computing, Springer Publications*, vol. 8, pp. 61-75, 2010.

[3] Foster, C. Kesselmann, "Globus: a toolkit-based grid architecture, in: The Grid: Blueprints for a New Computing Infrastructure", Morgan Kaufmann, Los Altos, CA, 1999, pp. 259–278.

[4] Chen Yu, Dan C. Marinescu, "Algorithms for Divisible Load Scheduling of Data-intensive Applications", *Journal of Grid Computing*, Springer Publications, vol. 8, pp. 133-155, 2010.

[5] Viktors Bertis, Raphaël Bolze, Frederic Desprez, Kevin Reed, "From Dedicated Grid to Volunteer Grid", *Journal of Grid Computing*, vol. 7, pp. 463-478, 2009.

[6] Andrew Chien, Brad Calder, Stephen Elbert, and Karan Bhatia, "Entropia: architecture and performance of an enterprise desktop grid system", *Journal of Parallel and Distributed Computing*, Elsevier, vol. 63, pp. 597-610, 2003.

[7] R. Raman, M. Livny, M. Solomon, "Matchmaking: distributed resource management for high throughput computing", in: Proceedings of the Seventh *IEEE International Symposium on High Performance Distributed Computing* (HPDC7), Chicago, IL, July 1998.

[8] Yagoubi and Slimani, "Task Load Balancing Strategy for Grid Computing", *Journal of Computer Science*, vol. 3, pp. 186-194, 2007.

[9] J. Frey, T. Tannenbaum, I. Foster, S. Tuecke, "Condor-G: a computation management agent for multi-institutional grids", in: Proceedings of the Tenth *IEEE Symposium on High Performance Distributed Computing*, San Francisco, CA, August 2001.

[10] Chang Liu, Zhiwen Zhao and Fang Liu, "An insight into the Architecture of Condor-A Distributed Scheduler", IEEE, 1-4, 2009.

[11] R. Moreno-Vozmediano, K. Nadiminti, S. Venugopal, A. B. Alonso-Conde, H. Gibbins, and R. Buyya, "Distributed Portfolio and Investment Risk Analysis on Global Grids", Special Issue on Network-Based Computing, *Journal of Computer and System Sciences*, 73(8):1164-1175, Elsevier Science, Amsterdam, The Netherlands, Dec. 2007.

[12] R. Barbera, A. Falzone, "The GENIUS Grid Portal", *Computing in High Energy and Nuclear Physics*, 24-28 March 2008, California.

[13] Chervenak et.al. "The Data Grid:Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets", *Journal of Network and Computer Applications:Special Issue on Network-Based Storage Services*, (3):187–200, 2000.

[14] Eun-Kyu Byun, Jin-Soo Kim, "An Adaptive, Scalable, and Reliable Resource Provisioning Framework for WSRF-Compliant Applications", *Journal of Grid Computing*, vol. 7, pp. 73-89, 2009.

[15] Gabor and Peter, "Grid Application Development on the Basis of Web Portal Technology", *Journal of Grid Computing*, Springer Publications, vol. 8, pp. 472-480, 2006.

[16] Tevfik Kosar and Miron Livny, "Aframework for reliable and efficient data placement in distributed computing systems", *Journal of Parallel and Distributed Computing*, Elsevier, vol. 65, pp. 1146-1157, 2005.

[17] J. Nick I. Foster, K. Kesselman and S. Tuecke. "The physiology of the Grid. Technical report", Argonne National Laboratoty, University of Chicago, University of Southern California, IBM Corporation, 1999.

[18] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C.Kesselman, S. Meder, V. Nefedova, D. Quesnel, S.T. ke, "Secure, efficient data transport and replica management for high-performance data-intensive computing", in: *IEEE Mass Storage Conference*, San Diego, CA, April 2001.

[19] H. Takemiya, K. Shudo, Y. Tanaka & S. Sekiguchi, "Constructing Grid Applications Using Standard Grid Middleware", *Journal of Grid Computing*, Springer Publications, vol. 1, pp. 117-131, 2003.

[20] Anderson, D.P, "SETI@home: An experiment in public-resource computing", *Journal of Parallel and Distributed Computing*, Elsevier, vol. 46, pp. 146-157, 2003

[21] Grid Computing -Adoption in the Financial Sector, was written by 451 Group analysts William Fellows, Steve Wallage and Aidan Biggins, 2005.

[22] Ian Foster and Carl Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *International Journal of Supercomputer Applications*, 11(2): 115-128, 1997.

[23] Henri Casanova and Jack Dongarra, *NetSolve:* A Network Server for Solving Computational Science Problems, *Intl. Journal of Supercomputing Applications and High Performance Computing*, Vol. 11, No. 3, 1997.

[24] Ian Foster. What is the Grid? A Three Point Checklist. Grid Today, 2002.

[25] James Broberg, Srikumar Venugopal and Rajkumar Buyya. Market-oriented Grids and Utility Computing: The State-of-the-art and Future Directions. *Journal of Grid Computing*, (3):255–276, 2008.

[26] V. Bertstis. Fundamentals of Grid Computing. *Red Books, IBM Corporation*, 2002.

[27] Czajkoski et.al. "Grid Information Services for Distributed Resource Sharing", In *10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, 2001.

[28] Foster, I., Kesselman, C. (eds.): "The grid: Blueprint for a New Computing Infrastructure", 1st edn. Morgan Kaufmann, San Francisco (1998).

[29] Hidemoto Nakada, Mitsuhisa Sato, Satoshi Sekiguchi, Design and Implementations of Ninf: towards a Global Computing Infrastructure, *FGCS Journal*, October 1999.

[30] Fran Berman and Rich Wolski, The AppLeS Project: A Status Report, 8th *NEC Research Symposium*, Berlin, Germany, May 1997.

[31] Spyros Lalis and Alexandros Karipidis, JaWS: An Open Market-Based Framework for Distributed Computing over the Internet, *IEEE/ACM International Workshop on Grid Computing*, Dec, 2000.

[32] A. Grimshaw, The ROI Case for Grids, Grid Today, vol. 1, no, 27, 2002.

[33] Zhang, G.: Study on the Application of Service of Grid Computing Environment. Southeast University, Nanjing (2004)

[34] Tuecke, S. et.al. Grid Service Specification. *Global Grid Forum*, 2002

[35] Rajkumar Buyya, Grid Computing Info Centre: http://www.gridcomputing.com

[36] Open Source J2EE Frameworks http://java-source.net/open-source/j2ee-frameworks

[37] Java Community Process: *JSR 152 Java Server Pages 2.0 Specification*.

[38] http://www.jcp.org/en/jsr/detail?id=152

[39] http://www.globus.org

[40] http://www.banktech.com

[41] http://www.wikipedia.org

[42] http://www.cs.wisc.edu

APPENDIX

GLOBUS TOOLKIT

Installation of Globus Toolkit

This helps as a guide to Globus Toolkit 4.2.1 (GTK4.2.1) installation. It deals with basic installation that installs core services and a few base services namely: a security infrastructure (GSI), GRAM, GridFTP and RFT. After the installation and testing, we explore the possibility of using the CA of one machine on another machine.

1. Pre-requisites (Softwares/Databases)
   - Java
   - Apache Ant
   - Apache tomcat
   - Mysql database
2. Optional Software
   - IODBC

   Before starting the installation, first we need to create a globus user.
3. Building the Toolkit
4. Setting up the simpleCA
5. Get the host and container certificates signed by simpleCA
6. Get valid credentials from the CA by running grid-proxy-init

```
suresh@suresh-laptop:~$ grid-proxy-init
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-suresh-laptop/CN=suresh
Creating proxy ............................. Done
Your proxy is valid until: Fri Apr 15 01:10:33 2011
suresh@suresh-laptop:~$
```

7. Create grid-map file and add entries to it
8. Configure GridFTP
9. Install postgresql and configure RFT
10. Start the Globus container and test using the globus client

Condor

Installation of Condor

This helps as a guide to Condor 6.5 installation. It deals with basic installation that installs services on central manager and execute nodes.

1. Download Condor 6.5 binary from www.cs.wsic.edu/condor
2. Untar it using tar –xvf condor.xx.tar.gz
3. Now configure the make file
4. Change to sudo user mode in terminal using command 'su'
5. Run the following command

   ./condor -options
6. It will ask some questions about configuration
7. Now installation of condor completed
8. Install java and set the path in condor_config file

Condor commands:

1. To start the condor daemons run the command as a root user

   :> sudo condor_master
2. To see the status of the pool

   :> condor_status
3. To see the status jobs in the pool

   :> condor_q cluster_id_of_job

MySQL

Installation of MySQL

This helps as a guide to MySQL database installation. It deals with basic installation that installs database services on central manager

Install mysql client, server and the jdbc connector, either via synaptic package manager or by using the following

:> sudo apt-get install mysql-server

:> sudo apt-get install mysql-client

:> sudo apt-get install libmysql-java

Configuration

Set up MySQL default password

Set up the password for the root user as 'root' or whatever you want. The last entry is the password

:> mysqladmin -u root password root

Check you can connect to mysql

:> mysql -u root -p

then enter the password you just set and press return again. In this case it would be "root"

Creating database

Mysql> create database database_name;

Mysql> CREATE TABLE users (username varchar(15) NOT NULL, password varchar(15) NOT NULL, PRIMARY KEY (username));

Mysql> insert into users values('suresh','ok');