



## PROJECT PROPOSAL

**Project Supervisor :**

- Dr. Sanjiva Weerawarana

**Super Group Head :**

- Mr. Indika Perera

**Group Members :**

- Hiranya Jayathilaka (05185M)
- Aravinda Dassanayake (050071K)
- Eranda Angunawala (050018G)
- Dinusha Boteju (050049C)

**Project Title**

MOINC Server

**Introduction**

Over the last couple of decades, Information Technology together with Computer Science has reached a state that was never anticipated at the inception of Computer Technology. Today, the computer has transformed in to a versatile instrument, providing diverse services in the form of a powerful business tool, an unlimited source of information, a communication medium that connects people all around the world and a source of fun and entertainment. Two of the major driving forces behind this rapid revolutionary technological transformation are the improvements in computer hardware and the rapid development and wide adoption of the Internet. The Internet in particular has made a serious impact on people's day to day life by revolutionizing the way various commercial activities, business processes and technical activities are carried out. Consequently, the Internet which was nothing more than a network for publishing and retrieving information about one and a half decade ago, has now become the ultimate source of knowledge for those who enjoy learning, an unlimited space for advertising and communication for businessmen, a lovers park for those who fancy flirting and a world full of opportunities and vulnerabilities for hackers and cyber criminals. In a nutshell, the Internet is about everything in today's context.

The popularity of the Internet and the World Wide Web has given birth to a totally new paradigm of software engineering and distributed computing. Web services technology is probably the most significant element of this paradigm, which has evidently turned out to be the most popular approach for implementing the Service Oriented Architecture (SOA). It is based on a number of well formed platform independent and language independent standards. Web services allow machines and software applications to communicate, collaborate and process data in a distributed environment while adhering to the principals of SOA thus ensuring extended reusability, maintainability and encapsulation. Web services offer seamless interoperability among different systems and platforms which enables large organizations to further expand their businesses and combine their business processes with other organization's business processes.

**Problem Description**

As the Web services platform becomes more powerful and matured, more and more organizations around the globe have started adopting the technology to help improve their businesses. With Web services, complex business processes can be carried out easily in a collaborative environment while keeping the costs down. Also Web services give more visibility and far reach to the organizations that use it due to being loosely coupled to a specific platform, language or organization. Owing to these

numerous benefits of Web services, organizations are rapidly switching to large scale Web services deployments powered by enterprise grade middleware.

But unfortunately nothing is perfect and this rule applies for Web services as well. Web services deployments suffer a number of pitfalls as their usage levels and complexities increase. Poor availability, bad performance under large loads and low scalability are some of these issues. The trivial solution for all these issues is upgrading hardware used to deploy and host Web services. However this is never a sustainable solution because this can incur severe costs and on the other hand the initial problem can emerge once again when the business processes carried out by the Web services become more complex and the number of clients for the business increases. Hence a more robust, cheap, and dependable solution is required.

Fortunately, a concept that seems to be a remedy for this setback has already been discovered, and is in use successfully. That is to use the unutilized computing power of idling computers. Any large organization would have hundreds or may be even thousands of idling computers at any given instance. When the computing power of all these machines are combined, the collective computational capacity can deliver the performance and availability that no super computer or mainframe can possibly offer. This technique is used by a number of world renowned projects like SETI@home and World Community Grid to process complex algorithms employing enormous amounts of data. However, surprisingly it is still not used to gear Web services.

In this project we attempt to combine the Web services clustering model with the grid computing model to develop a Web services deployment platform that makes use of the computing power of the idling computers in a given network. Thereby we attempt to develop a long term solution for Web services availability, scalability and performance issues.

### **Web Services Clustering**

Any production deployment of a server has to fulfill two basic needs. They are availability and scalability. High availability refers to the ability of a server to serve client requests while tolerating faults and failures. That means the server should be available whenever the client wants some work to be done. The server down time should be at a minimum and the server should be up and running in good shape at all times. Scalability is the ability to serve a large number of clients sending a large number of requests without a significant degradation of performance and response time.

These two requirements make clustering a compulsory feature for any server application. There is no exception when it comes to Web services servers. Many enterprise scale Web services deployments have to process millions of transactions per day, or even more. A large number of clients, both human and computers, connect simultaneously to these systems and initiate transactions. Therefore, the servers hosting the Web services have to support that level of performance and concurrency.

It is impossible to support that level of scalability and high availability from a single server instance despite how powerful the server hardware or how efficient the server software. Web services clustering is needed to solve this, which allows to deploy and manage several instances of identical Web services across multiple Web services servers running on different server machines. Then we can distribute client requests among these machines using a suitable load balancing system to achieve the required level of availability and scalability.

There is a number of ways to implement Web services clustering. These different approaches are based on different algorithms and they have their own pros and cons. However in most of these clustering methods a shared repository plays a major role. The shared repository is the central storage for all the services and data. All the nodes (physical machines) in the cluster gets the necessary services and data from the repository. Also with Web services clustering load balancing techniques, node management algorithms and complex session management techniques come into play.

### **Grid Computing**

Technically speaking a Grid Computing system is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed 'autonomous' resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements. In simpler terms, Grid computing is applying the resources of many computers in a network to a single problem at the same time, which is usually a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data.

Although grid computing primarily focuses on processor cycles, they also enable the sharing, selection, and aggregation of a wide variety of geographically distributed computational resources such as supercomputers, compute clusters, storage systems, data sources, instruments, people and presents them as a single, unified resource for solving large-scale compute and data intensive computing applications.

Such resource intensive applications can be molecular modeling for drug design, brain activity analysis, and high energy physics etc. A well-known example of grid computing in the public domain is the ongoing SETI (Search for Extraterrestrial Intelligence) @Home project in which thousands of people are sharing the unused processor cycles of their PCs in the vast search for signs of 'rational' signals from outer space.

The name 'grid' computing comes through the analogy this sort of a system has to an electric power network where power generators are distributed, but the users are able to access electric power without bothering about the source of energy and its location.

Even though clusters and grids are treated synonymously in certain occasions, the key distinction between clusters and grids mainly lie in the way resources are managed. In case of clusters, the resource allocation is performed by a centralized resource manager and all nodes cooperatively work together as

a single unified resource. In case of Grids, each node has its own resource manager and don't aim for providing a single system view. It should be noted that autonomous resources in the Grid can span across a single or multiple organizations.

Grid computing requires the use of software that can divide and farm out pieces of a program to as many as several thousand computers. Grid computing can be thought of as distributed and large-scale cluster computing and as a form of network-distributed parallel processing. It can be confined to the network of computer workstations within a corporation or it can be a public collaboration in which case it is also sometimes known as a form of peer-to-peer computing.

Grid computing appears to be a promising trend for few reasons. The first is its ability to make more cost-effective use of a given amount of computer resources. This becomes important in saving the high cost incurred at acquiring a supercomputer, and eliminating its single point of failure issues. Secondly it becomes important as a way to solve problems that can't be approached without an enormous amount of computing power. In cases where a supercomputer seems very far away as an option and yet massive computational power is required, grid computing can come in handy to provide all the processing cycles needed. A third advantage would be that the resources of many computers can be cooperatively and perhaps synergistically harnessed and managed as a collaboration towards a common objective. This autonomy is very important in resource intensive applications.

### **Web Services Clustering with Grid Computing Model**

Using the grid computing model as a solution for Web services availability and scalability issues is an area of highest interest these days. Many large organizations like IBM for example are researching on potential methods for combining these two powerful technologies. Many visionaries of the industry look forward to an era where Web Services function as the software while Grid Computing becomes the hardware for it. However there has not been any work in this area with some practical value as yet.

This project focuses on designing and developing a Java Web services deployment environment, or a Java Web services application server that makes use of the grid computing model as the clustering technique. This will be a hybrid between the conventional clustering of Web Service application servers which primarily focus on replicating server instances and distributed computing harnessed by grid computing or voluntary computing.

The ultimate result of the project looks at a fully fledged framework which can perform distributed computing by pushing relevant services to the node computers of the cluster and create a new paradigm of executing web services over a distributed virtual supercomputer. With this approach the computers of tomorrow - no matter within an organization or at home, will be making good use of idle computing power while being part of a distributed

supercomputer and aiding higher quality of service for Web Services.

### **Client**

Generic

### **Objectives**

The main objective of this project is to design and develop a Java Web services deployment platform or a Java Web services application server that makes use of the grid computing model to facilitate clustering. Comprising the major functionalities of the MOINC platform, this naturally becomes the nucleus of the entire MOINC project. The MOINC Server is pretty much a self contained entity with its own functionality but it also relies on functionality of some external modules. These external modules could be modules in the MOINC platform (like the MOINC Server Enhancement module) or other third party modules. The bridging between the modules will be done via a well documented set of APIs.

The main functionality of the Server module would be, the distribution of client requests among idling computers in a specified network. As a computer in the network goes idle or whenever it is ready to serve requests from MOINC, that client needs to be added to the working cluster. Moreover when a computer in the working cluster needs to withdraw the processing power, again the MOINC has to release it. This crucial task is handled by the MOINC server module.

Since the MOINC platform renders services to clients by relying on the computing power of computers which are geographically dislocated and linked by the internet, network failures are inevitable. Thus the MOINC Server, as the module which handles addition and releasing process of subscribing computers to the cluster, another important objective is providing a mechanism which handles failures by redistributing the tasks allocated to the failed computer or the cluster of computers to other computers which are online.

Apart from the paramount objectives of the MOINC Server module mentioned above, the application is to fulfill the goals depicted below.

- Deploying , removing and modifying services
- Event logging
- Traffic monitoring
- Specifying web services and security policies
- Executing basic server administration options

## Methodology

In order to address project objectives here are the methodologies that we are using,

- Since this module is going to handle a large amount of transactions from thousands of clients, it will be used in conjunction with a high performance application server.
- Either this MOINC Server will be used into volunteer grid computing or into organizational level, it would need a mechanism to verify service requests. Furthermore, a load balancing system based on the history of the client's contribution record would be added to deliver more effectiveness to the whole system.
- The Server will be enabled to listen to notifications from the MOINC client nodes in order to process them. This includes adding idle clients to the cluster, removing clients from the cluster if required and committing services to clients.
- The entire MOINC platform is based on the contributor's system idle time. But the issue with depending on this is the system cannot predict about when it ends. Therefore a server-side roll back mechanism will be implemented to save the data of on going processes. This can be done by getting information regarding task status of ongoing processes and if needed relocating them to some other clients.
- Moreover, this Server module will be equipped with a robust error handling mechanism. This is due to the possibility that there could be unexpected node failures at the client side.
- All the administrative duties too have to be executed in the MOINC server. This includes functionalities such as event logging, traffic monitoring, deploying services, specifying web services policies, specifying security policies and executing basic server administration options.
- All the interfaces, APIs and processes will be based on existing standards.
- Due to the large development overhead associated with this project we will be using existing systems as much as possible to save time and refrain from reinventing the wheel.

## Project Scope

Being the main component of the MOINC platform, the MOINC Server needs to make sure that pretty much all the functionalities of the platform are handled properly. Due to this enormous burden on this application, we have to identify the key functionalities to ensure its smooth functioning. Here are those key functionalities that fall into the scope of the server side core application.

- A service execution environment to handle all the transactions.
- A default SOAP processing system to handle all incoming service requests.
- A mechanism of addressing notifications from the MOINC clients.
- A roll back mechanism to relocate tasks.
- An error management system.
- A server administration console featuring all the administrative functionalities.

Therefore summarizing the project scope of the MOINC Server module, we could say that it handles all the main administration functionalities, cluster management and service requests from Web service

clients. It does not handle any of the MOINC client side functionalities or optional functionalities related to client management.

**Project Plan and Schedule**

Activity	April				May				June				July				August				September				October				November				December				January				February				March			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
01) Preliminary research	█	█	█																																													
02) Project proposal		█	█																																													
03) Requirement gathering					█	█																																										
04) Requirement specification preparation						█																																										
05) System design							█	█	█	█																																						
06) System design document									█																																							
07) System development													█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█																				
08) System testing																																																
09) System integration and implementation																													█	█	█	█																
10) Project report submission and Demos																																													█	█	█	█