An Adaptive Web Service based Loosely Coupled Distributed Computing architecture for solving Data Intensive Applications

Hemalatha Thangaraj, G. Athisha

Abstract— This paper focuses on Web Service based loosely coupled Distributed Computing Environment for Single Program Multiple Data and Multiple Program Multiple Data (SPMD, MPMD) type of applications in network. The Distributed Computing Environment (DCE) is formed by utilizing the underutilized Local Area Network (LAN) workstations in a Campus Network. The DCE is used to solve large problems. In this paper we propose a comprehensive approach to estimate the load of a workstation in order to decide whether the workstation can be a part of DCE to solve computational intensive and data intensive applications. The proposed work is implemented and two different applications are tested.

Index Terms— Compute node, Distributed Computing Environment (DCE), High Performance Computing (HPC) Applications, Simple Object Access Protocol (SOAP), Web Service (WS).

1 INTRODUCTION

Generally there are different kinds of applications ranging from High Performance Computing to High throughput computing. The High Performance computing applications require huge computational power and vast memory in order to carry out the task of solving engineering applications such as Finite element analysis, computational physics, Aerodynamic modeling and simulation, fluid dynamics, testing of virtual prototypes, processing of satellite images etc. In such environment computations are done in parallel over lots of Compute Elements (CPU -Central Processing unit & GPU – Graphics Processing Unit) and very fast network has to connect between the compute elements. HPC applications need networking of clusters and Grids [8]. The Complex or high performance computing applications requires an environment which has the capability to compute and move data across networks quickly and efficiently. Such networks should provide high throughput and low latency to execute such applications. High throughput refers to an environment which can deliver a large amount of processing capacity over a long period of time. Low latency refers to the minimal delay between processing input and providing output. When real time applications are executed in a desktop PC it takes longer time to get the expected output since it provides less throughput and high latency which is not suitable for execution. Hence we are in need of HPC when one wants to complete a very time consuming operation in less time or to complete an operation under a tight deadline or to perform high number of operations per second. During the past few years, a range of computational models have been developed as the basis for the design of fast and portable parallel algorithms [1]. To provide HPC we are in need of suitable Hardware and Software infrastructure. But the hardware is very costly and the software is either proprietary or open source. In case if software is open source it requires complex set of make utilities which is to be carefully executed one by one and configuring should be done parallel. The future needs of HPC are increasing and recently HPC has come to be applied to business uses of Cluster based supercomputers such as Data warehousing, Line of Business applications and Transaction processing. Since 93% of respondents stated that HPC was fundamental to the growth of their business, it is required to move away from expensive / specialized proprietary parallel computers to form clusters of desktop PC's mostly which are underutilized[10]. However, the design and implementation of efficient parallel algorithms for clusters is still a problematic issue [1].

Flynn's taxonomy is a classification of computer architectures proposed by Michael I. Flynn in 1966. It is a specific classification of parallel computer architectures that are based on the number of concurrent instruction and data streams available in the architecture [9]. They are Single Instruction Single Data SISD, Single Instruction Multiple Data SIMD, Multiple Instruction Single Data MISD and Multiple Instruction and Multiple Data MIMD [9]. MIMD is further classified into two categories they are Single Program Multiple Data SPMD and Multiple Program and Multiple Data MPMD, where in SPMD multiple autonomous processors simultaneously execute the same program of different data. The model was proposed by Frederica et. al.. In MPMD multiple autonomous processors simultaneously operating at least two independent programs in which one node is called host and other node is called manager. The manager runs a program which farms out data to all other nodes which run a second program. The nodes return results to the manager. In this paper we have used only the underlying concepts of SPMD and MPMD.

2 RELATED WORK

2.1 Distributed System

There is a need for more computational power with an increase of three orders of magnitude within five years and five orders of magnitude within a decade [3]. This dramatic increase in need for higher computational need can be achieved

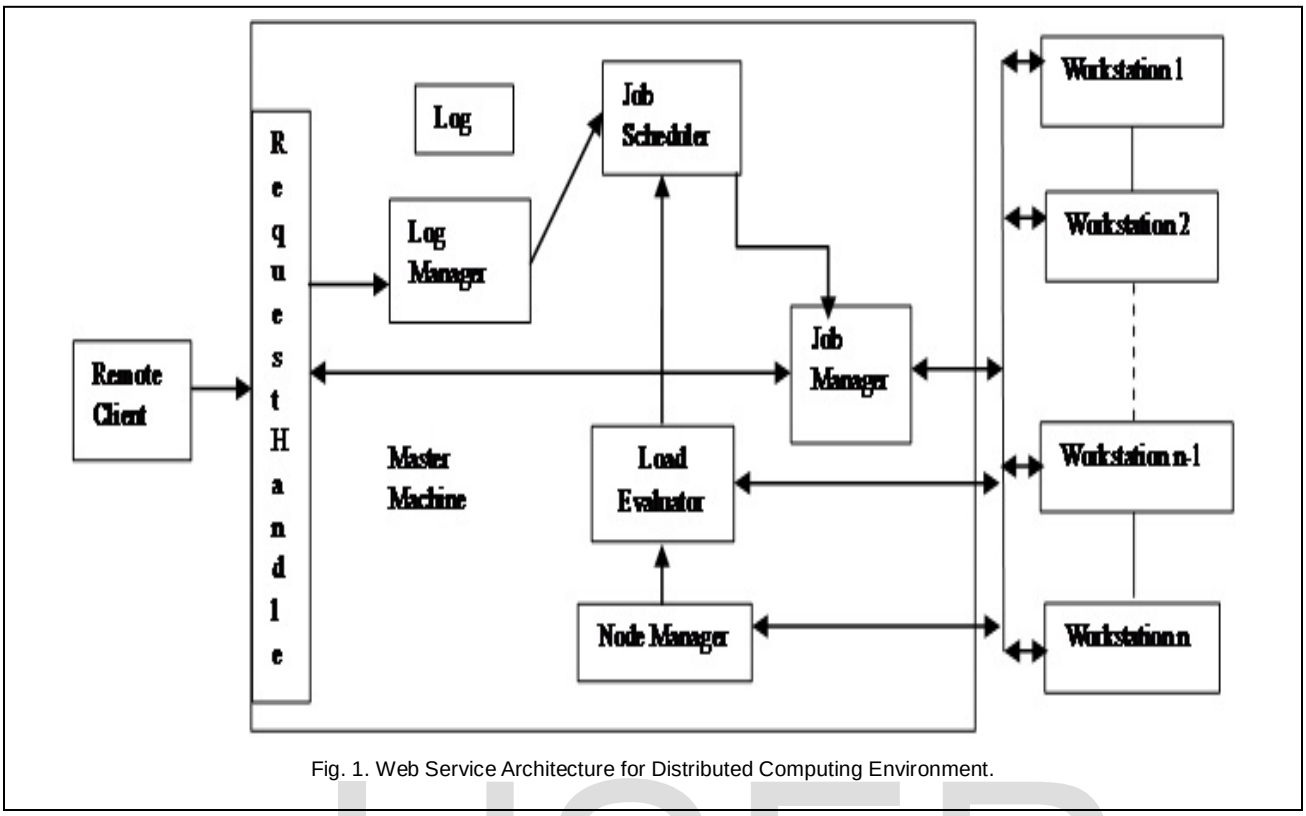

Fig. 1. Web Service Architecture for Distributed Computing Environment.

by technology improvement and increased utilisation of idle capacity. Grid is a distributed system with non interactive workloads that involve a large number of resources for computation. Grid provides a Distributed Computing environment that spans across multiple virtual organisations VO where each VO consists of either physically distributed Institutions or logically related groups. The goal is to enable resource sharing and coordinated problem solving in dynamic, multi Institutional virtual organisations. The approach taken by the de facto standard implementation – The Globus Toolkit is to build a uniform computing environment from diverse resources by defining standard network protocols and providing middleware to mediate access to wide range of heterogeneous resources. Grid is a massive, loosely coupled, distributed computing environment with extensive computing capability, communication bandwidth and data storage capacity. At present large number of grid infrastructures are available which are built with their own middleware. A middleware is a software abstraction layer used to aggregate the resources to form a grid. Different middleware are developed and they are widely used by various research institutes to build their grids based on their needs. A computational grid is a large scale pooling of compute resources. It requires significant hardware infrastructure to achieve the interconnection between several resources and software infrastructure to monitor and control the resulting ensemble. The environment provided by computational Grid is highly suitable for solving high performance computing applications. The end-systems that are used for setting up of HPC environments are classified into four classes. They are individual end systems like computers which are relatively small and a high degree of homogeneity and integrations. But they are lack in features which are necessary for integrations into large clusters. The next class of systems is Clusters. Clusters are like an individual end system which has homogeneous architecture and it is controlled by a single administrative entity that has control over each end system. But this has the complicating factors like less integrated and increased physical needs like alternative algorithms are needed for certain resource management and control functions. This leads to modification in software architectures and operating systems. The third class of systems is Intranets. Intranet comprises of large numbers of resources which do not belong to the single organisation. The complicating factors in intranet are heterogeneity; different administrative domains have different administrative policies and lack of global knowledge. The final class of systems is internet which adds additional issues in addition to intranet. They are lack of centralised control, geographical distribution and international issues since it has to cross international borders [3]. Current Grid middleware toolkits expose their functionality through services, programming model, command line tools, requiring much technical knowledge of grid backend and middleware systems [5]

2.2 Web Service Architecture (WSA) Standards

The next generation application is based on Autonomous Web services [13]. Web services are not limited to the Internet. They provide a powerful architecture for all types of Distributed Computing. It provides an open, interoperable and highly efficient framework for implementing systems. The loosely coupled architecture provides a new and promising solution for implementing complex collaborative applications. The WSA protocols is composed of Simple Object Access Protocol

SOAP for messaging, Web Service Description Language WSDL for describing the web services to the service consumer, Universal Discovery Description Integration UDDI is a directory through which the client can find any services. eXtensible Markup Language XML is a flexible and extensible data format standard for handling different universal data types in the form of text [14]. XML significantly reduces the burden of deploying many technologies needed to ensure success of web services. SOAP provides a standard, extensible, composable framework for packaging and exchanging XML messages which uses its own text encoding style. Web Service definitions can be mapped to any implementation language, platform, object model or messaging system. WSDL defines the abstract functionality of a service and concrete binding details for SOAP, HTTP and MIME. In addition the WSA provides following benefits a. promotes interoperability by minimizing the requirement for shared understanding b. enables Just in Time integration c. reduces complexity by encapsulation d. Enables interoperability of legacy applications [12]. Since the architecture is basically designed for highly dynamic program to program interactions we have chosen WSA. Hence in this paper we have designed a loosely coupled Web Service based architecture which is composed of dynamic components. Our objective is to provide an user friendly architecture to setup a loosely coupled DCE and to reduce the burden of deploying many technologies to form DCE for sharing of computational resources.

3 Proposed System

3.1 Model Design

The goal of the proposed model is to provide a framework for executing Coarse grained SPMD and MPMD processes enabling multi-platform operability without creating any specific middleware tool. It is based on web service architecture. The proposed framework establishes a loosely coupled distributed computing environment by utilizing the underutilized desktop workstations in Local Area Network (LAN). Several such nodes from different LAN in a Campus Area Network can be used to setup a loosely coupled DCE without using complex middleware. The framework consists of a DCE server and a set of Compute nodes. DCE server acts like a Master machine and compute nodes works like a client to serve the user. The Fig. 1 shows the design of the web service architecture for loosely coupled Distributed computing Environment. The framework consists of a master machine and set of compute nodes. The master machine consists of five modules. The request handler is the point at which the external user interacts with the framework. The client submits a job which consists of an executable code and an input to the executable code which is in the form of either a data file or a text file or in any other file formats which a user code can support. The request handler assigns a unique user Identifier ID for the job submitted by the client and it send back the ID to the client. A client can use this ID for further interactions with the Master machine in order to update the status of the submitted job. The role of the master machine is to establish a virtual organization for every remote client. The proposed system constitutes of loosely coupled software components. Each component is implemented as a web service. The following subsections covers in detail about the role of each component which is deployed as a service in the Web Service container.

3.2 Request Handler

It is the point at which the external user interacts with framework in order to create Distributed Computing Environment to execute the user job which is designed to satisfy the criteria SPMD or MPMD processes. This criterion is validated at the time of job submission in order to ensure whether the user submits a valid job or not. In case if it is not satisfied the job is rejected to avoid further conflicts. On submission the request handler will assign a unique Job ID and return the ID in a synchronous manner back to the user. Then the job is handed over to Log Manager.

3.3 Log Manager

Log manager registers the details of the job in a separate log file and stores the status of the job at every stage of its execution. It is a mechanism which is needed to track the status of the Job by monitoring the execution of the resulting processes. The status of the job is updated separately in a log file which is indexed on a Unique Job Id. Every entry in this log file consists of the following details viz. job ID, starting time, ending time, process initialization and its successful termination which is shown in Fig. 2.

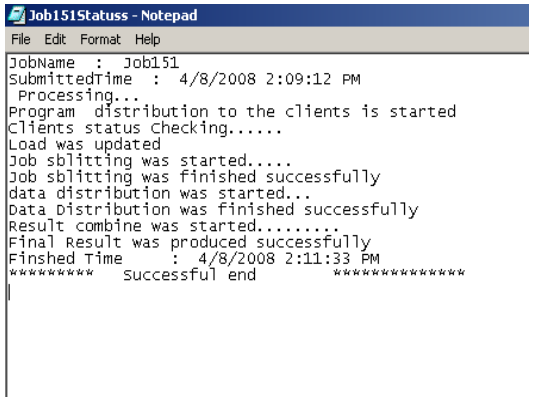

Fig. 2. A Sample Log file that shows the Details about the Job.

3.4 Node Manager

Using this module any new workstation which is connected to a LAN can be added to the DCE or existing workstation can be removed from the DCE on a demand basis. A software de-

- *Mrs. Hemalatha T. is currently working as an Associate Professor in Department of Computer Science & Engineering in P.S.N.A College of Engineering and Technology, Affiliated to Anna University, Dindigul, India, PH-09994980886. E-mail: hemashek@yahoo.com*
- *Dr. G. Athisha is currently working as a Professor in Department of Electronics and Communication engineering in P.S.N.A College of Engineering and Technology, Affiliated to Anna University, Dindigul,India, PH-09442462056. E-mail: hodece@psnacet.edu.in*

ployment task is introduced to install software components on a remote workstation using standard protocol while setting up a DCE. This is done from the Master node in order to create a DCE setup successfully. These files are totally obscure to the end user.

3.5 Load Evaluator

A novel Adaptive algorithm is proposed in this paper that estimates the load of remote LAN workstations that are connected to the DCE framework. The Load evaluator algorithm is implemented as a service in the target machine's operating system thereby it monitors the system resources periodically in every machine which is a part of a Virtual Organization's DCE. The system resources include the CPU load and RAM utilization. The CPU load estimation is computed by algorithm shown in Listing 1. Initially the details about number of processes running in the operating system are collected from the task manager. The load is estimated by considering the total number of processes, size of each process, memory utilized for each process and total memory usage. Scaling is performed in order to normalize these values and load is calculated using the algorithm shown below in Listing 1. Both scaling and load estimation is done at the workstation in order to reduce the overhead incurred at the master machine, since it is a time consuming operation at the Master machine to perform the periodical estimation for all machines which are connected to the DCE. The computed load is updated by the loadEval service which is running in the workstation. The listener in Master Node receives computed load and updates the computed load in the XML data store file which is accessed by the Job scheduler.

Algorithm:

1. Initialize the Load variable 'L' with value 0
2. Count the total number of processes under execution in the Workstation.
3. Based on the total number of processes adjust the 'L' Value
4. Check the Percentage of RAM utilization for the system and Application Processes.
5. Adjust the Value for the load variable 'L' based on the total RAM utilization.
6. Perform load Estimation by checking the value of 'L'

Listing 1: Algorithm to compute the load of a workstation in DCE

```
Program computeLoad (output)
// initialize the load variable with value 0
Set load variable L = 0;
//Count the number of processes running in the system
Begin
  If (no. of Processes < 35)
    Assign L = 1;
 Else
    Assign L = 2;
//Identify the number of Big processes / small processes/
//Mega processes based on RAM utilization for each
//process
  If (no. of big_Processes > 5)
    L = L + 3;
  End if
  If (no. of small_Processes > 10)
    L = L + 2;
  End if
  If (no. of mega_Processes > 2)
    L = L + 4;
  Endif
End;

//Check the percentage of total RAM used for entire
//System and application processes.

Program compute_RAM_Utilize (output)
Begin
 If (10 < total_RAM_utilisation < 30)
   L = L – 2;
 Elseif (31 < total_RAM_utilisation < 60)
  L = L – 1;
Else
  L = L + 2;

// Estimation of Load
 If (L < 4)
   Set status = "Lightly Loaded"
 Else
 If (4 < L < 8)
  Set status = "Medium Loaded"
 Else
  Set status = "Heavily Loaded"
Endif
 End.
```

3.6 Job Scheduler

This sub system plays a major role in this framework by selecting the best compute nodes for the execution of jobs and redistribute the job that is code file and data file to the selected compute nodes thereby it creates a new Virtual Organization. The selection of compute nodes is performed on the basis of the load computed by the loadEval service and the availability of the compute node. The availability of the node is validated from the policy of the individual workstation in a LAN or group of workstations in the same LAN that posses the same policy. Upon successful selection of compute nodes the input data file is splitted based on the dimension of the data file, the availability and load of compute nodes. The code file and data file is then transferred to the target host with the support of File Transfer protocol FTP.

3.7 Execution and Agglomeration

There is a listener at every node which will receive the job and schedule it for execution. On successful completion, the results are sent back to the master node. There is a listener running in the master node that agglomerates all the results which is received after execution and return the result back to the client either synchronously or asynchronously.

4 EXPERIMENTATION

A practical experiment is setup in order to test the Web Service based loosely coupled Distributed Computing Environment. Our objective is to develop a simple and easy to use software solution for setting up of a Distributed computing setup in order to execute a non interactive large data intensive computational application that are designed in the form of SPMD and MPMD. The implementation is tested in a LAN and CAN that consists of client workstations with the configuration of Pentium III or Pentium IV with a minimum of 512 MB RAM and above. Such workstations were automatically configured to setup a DCE to solve higher computational applications. Two objectives are pursued in the implementations. One is implemented to compute n X n matrix multiplication operations in C language where n ranges from 500 to 2000. Second objective is image processing based applications in which different stages were executed in the form of MPMD and otherwise same stage is executed in parallel for a sequence of different images.

4.1 Experimental Setup

Matrices play a key role in several domains like scientific and business applications. Some application of matrices in the physical science includes Electrical circuits, System of Linear equations, structural mechanics, fluid mechanics, stress analysis, multiple linear fitting and polynomial fitting[6], [7]. Besides the matrices are significant to solve a certain kind of problem like Markov Chain which is commonly used in certain kind of Businesses [6], [7]. In addition to this in the field of cryptography sophisticated methods of coding and decoding is done by using large matrix to encode a message and inverse of the matrix are used to decode the message in order to increase the strength of the encryption and decryption algorithm. In such applications it is highly required to accelerate the matrix operations so that we can get the faster response in such applications. Hence volumetric square matrices were selected with various sizes that vary from 500 to 2000 rows and columns respectively. Then the following operations were tested viz. Matrix multiplication, squaring & cubing of a matrix in the framework. Since Matrix satisfies the requirement of SPMD the code is handed over to all the cooperating set of nodes in the framework which are selected based on the least load to medium load. Then the input data file is split in to sub data files according to the available nodes and their corresponding load conditions. Then the sub data files are transferred to the listener running in each workstation in the framework. Hence the listener at every node will receive the job and schedule it for execution. On successful execution of the job, the results are sent back to the master node. The master node agglomerates all the results and returns the result back to the client either synchronously or asynchronously.
Then similarly a volumetric set of satellite images were selected and the different phases of image processing algorithms were executed in parallel in this framework in order to extract the feature or to use the processed images for pattern recognition. This is tested on a large set of images concurrently by executing a set of codes written in C-Sharp. A set of images are selected and executed in parallel in different machines in this framework. Once on successful completion the response time analysis was performed in order to analyze whether our objective is met or not.

4.2 Execution Results

The WS-DCE is developed by taking into consideration the different aspects which may affect the performance by introducing delay thereby resulting into the increase in response time. Hence in order to assess how useful this model is, the real execution times of the application described in section 4 were tested in the usual approach. Then the same application is executed on the proposed architecture presented in section 3.

Using these data it is possible to compare the response time. The comparative analysis was performed and the result is shown in Fig. 5. The response time is very less since the loadEval algorithm presented in Listing 1 estimates the workstations in the most adaptive and unique approach, thereby the best workstations are selected and the job is dispatched based on its workstation capacity. In addition to this since the loadEval algorithm is running as a Service along with other services in both Master Host and target host's operating system it does not introduce any additional overhead which is shown in Fig. 3 & 4. Likewise the Listener at the Master Node is implemented as a system service which runs along with other system services. Then the request handler described in section 3.2 was implemented as a Web service thereby any remote user can submit the job just through the browser and the result can be collected either synchronously or asynchronously.

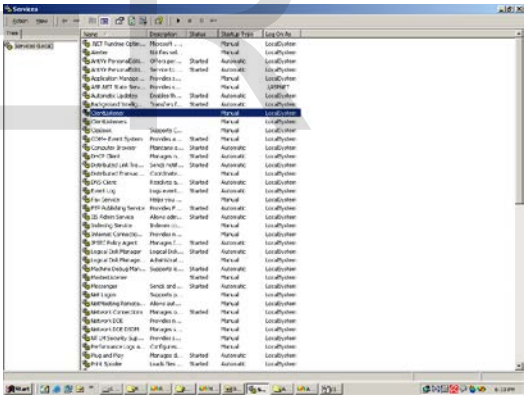

Fig. 3. Listener running in the Compute Node – Target Host.

6 CONCLUSION

In this paper a new Web Service based Computational Model for High Performance Computing - Single Program Multiple Data and Multiple Program Multiple Data type of applications has been proposed. The proposed work is validated by using a set of bench marks which is developed for this purpose. This proposed system does not require any higher configuration systems in order to execute high computational applications. Instead this architecture is developed in order to utilize the underutilized LAN workstations.

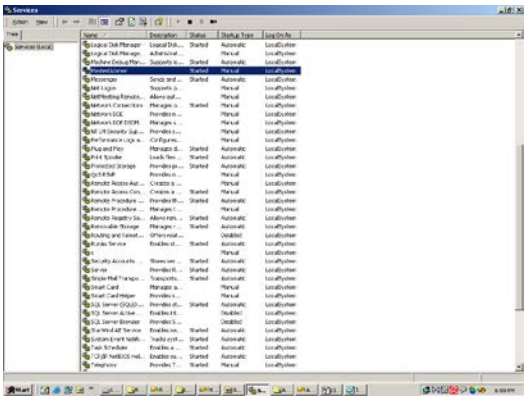

Fig. 4. Listener running in the Master Host of the Model.

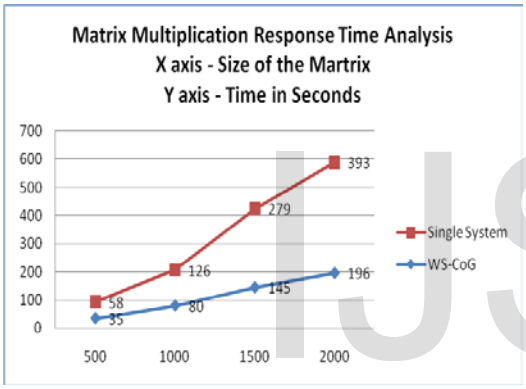

Fig. 5. Performance Evaluation – Response Time (in Seconds) analysis of Matrix Multiplication in WS-DCE Architecture comprises of 3 Compute nodes with that of the Single system.

Such workstations are estimated using a novel load estimation algorithm that measures the load of the client workstations by analyzing the processes running in the operating system and their corresponding resource utilizations. Additionally new workstations can join this WS-DCE easily since an automated procedure is implemented to add or remove a LAN workstation.

Any application that satisfies the criteria of SPMD and MPMD under the classification of Flynn's taxonomy can be applied to this System. This proposed model is tested by means of executing the set of benchmarks and the response time analysis is performed in order to validate the proposed model. At all conditions the setup is tested and it provides results at a faster rate with very less response time.

Future work in this area includes all the modules comprised in the system model will be converted into web service so that dynamic orchestration can be done on the fly in a regulated and coordinated environment which would be more flexible. The model can also be deployed in Cloud environment.

REFERENCES

[1] Jose Luis Bosque, Luis Pastor, "A Parallel Computational Model for Heterogeneous Clusters", *IEEE trans. on Parallel and Distributed System*. Vol 17, No. 12, 1390 --1400 2006

[2] Abramson D., Sosic R., Giddy J. and Hall B.: "A tool for performing parameterized simulations using Distributed Workstations". In: Proc. 4th *IEEE Symposium on High Performance Distributed Computing*. IEEE Computer Society Press, 1995

[3] Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1998).

[4] Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th *IEEE International Symposium on High Performance Distributed Computing*, pp. 181--184. IEEE Press, New York (2001)

[5] Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)

[6] David N. Bannard : Applications of Matrices, NCSSM, TCM Conference (2004).

[7] Gillberto E. Urroz: Matrix Applications with SCILAB (2001).

[8] http://en.wikipedia.org/wiki/supercomputer.

[9] http://en.wikipedia.org/wiki/category:Flynn%27s_Taxonomy

[10] www.panasas.com

[11] http://www.oracle.mobi/quickpage.html?page=36945.

[12] www.ibm.com/developerworks/webservices/library/w-ovr/#ibm-pcon.

[13] http://msdn.microsoft.com/en-us/library/ms996441.aspx

[14] www.w3.org/TR/w8-arch/#wsdl12