Scalable Computing: Practice and Experience
Volume 9, Number 2, pp. 135–142. http://www.scpe.org

ISSN 1895-1767
© 2008 SCPE

WEB PORTAL FOR LARGE-SCALE COMPUTATIONS BASED ON GRID AND MPI

ASSEL ZH. AKZHALOVA[*], DANIAR Y. AIZHULOV[*], GALYMZHAN SERALIN[*], AND GULNAR BALAKAYEVA[†]

Abstract. The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. This concept is realized in the most popular problem solving environments (PSEs) such as NetSolve and WebPDELab. These systems use some approaches to computational Grids and Web browser interfaces to back-end computing resources. The aim of our work is to build PSE implemented as a Web portal that allows clients to choose the most appropriate services to solve some problems using of matrix-algebra and numerical methods based on MPI techniques. In addition, it is available to extend the library of the Web-portal by loading computational algorithms. The proposed system allows to users a rapid prototyping of ideas, detailed analysis, and higher productivity.

Key words: grid, web portal, online problem solving, MPI.

1. Introduction. The development in IT industry and manufacturing produce the most complex problems and the majority of them require finding exact results. Today the most actual problems are connected with such commercial applications as financial services, life sciences and manufacturing. These applications include seismic analysis, statistical analysis, risk analysis, and mechanical engineering, weather analysis, drug discovery, and digital rendering. To solve rising problems in these areas we need a large infrastructure consisting of supercomputers, advanced algorithms, and programming tools. The most appropriate tool to realize this infrastructure is the Grid computing.

The third generation of Grid computing introduced a service-oriented approach leading to commercial projects in addition to the scientific projects. The real and specific problems that underlie the Grid concept are coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. This concept is realized in the most popular problem solving environments (PSEs) such as NetSolve and WebPDELab. These systems use some approaches to computational Grids and Web browser interfaces to back-end computing resources.

As known, Grid systems can be classified depending on their usage as: computational Grid, data Grid, service Grid. For example, TeraGrid—NSF funded linking 5 major research sites at 40 Gbs (www.teragrid.org), European Union Data Grid—grid for applications in high energy physics, environmental science, bioinformatics (http://www.eu-egee.org), Access Grid—collaboration systems using commodity technologies (www.accessgrid.org), Network for Earthquake Engineering Simulations Grid—Grid for earthquake engineering (www.nees.org). Network Enabled Solvers (NetSolve) is an example of a Grid based hardware/software server. Its original goal was to free domain scientists from having to perform these tedious tasks when they needed to use numerical software, particularly on multiple platforms. The principle of NetSolve is client/agent/server model [2]. All requests from client are accepted by the agents, the agents manage the requests and allocate servers to service, and the servers receive inputs for the problem, do the computation, and return the output parameters to the client. NetSolve can be invoked via C, FORTRAN, MATLAB, or Mathematica interfaces. Fault tolerance and load balancing are supported within the system.

Another example of PSE is WebPDELab. WebPDELab is a Web server that provides access to PELLPACK [3], a sophisticated problem-solving environment for partial differential equation (PDE) problems. WebPDELab interface supports users by a collection of case studies such as flow, heat transfer, electromagnetism, conduction. Any registered user can upload PELLPACK problem-definition files, such as .e files, mesh files, and solution files from the previous sessions. WebPDELab is implemented with the help of virtual networking computing (VNC). It runs the application and generates the display, a viewer that draws the display on the client screen and a TCP/IP connection between a server and a viewer. Within WebPDELab it is realized security for the user, server, and payment for computing services. Despite of existence of the described systems more and more scientists face such problem as a result of computational resources' lack. Certainly, there are several powerful high performance computers that are allocated in developed countries. However most of supercomputers are local and joining to them is not trivial. Sometimes a scientist has to overcome a number of bureaucratic obstacles before getting an access to supercomputer and solve the problem. Moreover, only huge problems are usually

[*]Kazakh National University, Mechanics and Mathematics Faculty, Computer Science Department, Masanchi street 39/47,050012 Almaty, Kazakhstan ({ak_asel, aizhol}@yahoo.com, gseralin@mail.ru).
[†]Kazakh British Technical University, Tole be street 59, 050091 Almaty, Kazakhstan (g.balakaeva@kbtu.kz).

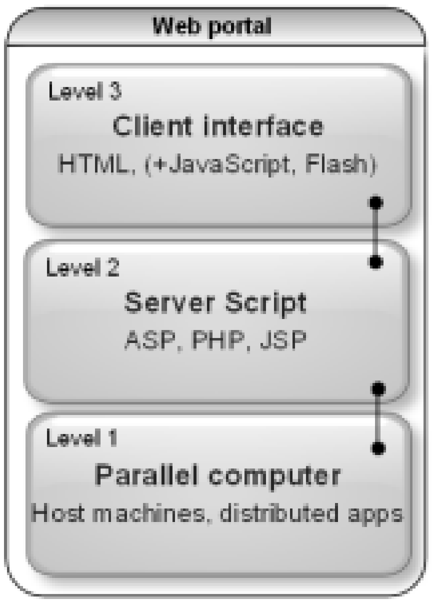

Fig. 2.1. *Three levels model of proposed system.*

introduced for the execution on such resources. In the same time, there are a lot of problems that should be solved by scientists over the world with the help of open computational resources. The emergence of networked computers has led to the possibility of using networks to launch parallel tasks remotely.

One of the problems of existing parallel systems is scalability. The rapid growth of computer networks and continuous changing of the network technologies such a huge number of different types of Web-services has a straight impact for reliable, flexible, and high-performing network software development.

The previous work [1] presented in LaSCoG'07 was concentrated on the architecture design of the proposed Web-portal. It was describing the principles of making large-scale computations through the Web-portal. This article is an extension of the part of Web-portal's functionality. Here, we propose the Service Manager component that supports the load balancing into the system and, improves the performance of the system. We consider the system performance problem as an optimal resource allocation problem. We formulate the problem of optimal resource allocation as finding an optimal sequence of servers at each time to keep the system on desirable response time and the total cost of the using servers would be minimal. This kind of problem belongs to optimal control problem. In this article, we propose to use a dynamic programming technique in order to solve the optimal control problem.

We propose a system that allows solving different scientific problems at any time through the Internet technologies. In this article we present a library that includes a parallel algorithm for two-dimensional non-stationary heat conductivity equation. The system offers an attractive alternative to expensive supercomputers and parallel computer systems for high-performance computing.

2. Design.

2.1. Architecture. The main idea is to merge server technologies and MPI programming tools in one system. The basic scheme of the system can be presented by three levels infrastructure. Figure 2.1 shows this model.

Data flows between levels from the third level to the first one. The second level is a middleware. The third level is a client interface. Client interface gives to the client Web form where input data can be uploaded to the server. There are two ways to input the data: completing the Web form or uploading the text file to the server. The second way is better when working with a large amount of data. Predefined templates allow to structure data before uploading. In addition, third level offers flash and Java scripts to check data and also display graphics. The second level is responsible for several tasks. Firstly, the information is processed

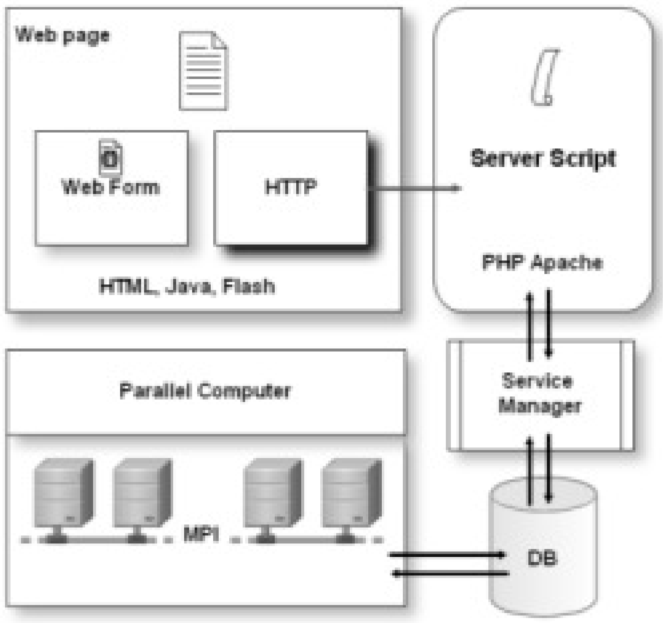

FIG. 2.2. *The architecture of proposed PSE.*

and converted to data according to the template that depends on the problem being used. The second level also provides security. The server scripts validate the data on the conformity and check whether user is going to use parallel computer or not with relevant purposes. The server is connected to the database that stores all information about previous executed tasks. The user can control and analyze the results with the help of diagrams and tables. Special agents find resources, control time execution and store it in the database. In fact, the database supports all needs starting from storage of authorization data and ending on control load balancing of parallel computer. In our case, the virtual parallel computer is the set of workstations connected by bus in P2P topology. The parallel computations are performing using MPI techniques. The main program, which is precompiled executable, reads the input data and calls for the appropriate task. Each task makes references to mpi.h library to implement communication and synchronization between processes. To implement parallel numerical calculations there is an additional library with prepared and ready for execution numerical methods. Synchronization is done by non-blocking sending and receiving routines [4]. If the same task is called by many users at once then the main program find free workstations and send them equal portions of tasks. There are different techniques to manage shared task set: cooperative lists manipulation and parallel access queue using fetch&add [5]. We offer a centralized load balancing technique. Figure 2.2 shows the architecture of the proposed PSE.

2.2. Functionality. The connection between third and second level is based on HTTP protocol. There is no need to make persistent connection between client and server as client has to make only two actions: to send the input data and receive the result. It is clear that the browser should be forced to check whether parallel computer has finished the work or not.

After the data had been posted to server, server script starts validating all received information. It authenticates the user, creates session, and validates the data, checks workload of parallel computer. After all these actions have been completed the server is ready to communicate with parallel computer. The communication is carried out by reading and writing files. This process does not waste much time, because reading and writing are made only once for each computation. The server starts an execution of the chosen problem. The agent starts an appropriate application ("calculator") on host-machines which were chosen according to strategy described in previous subsection. "Calculator" is one program from the set of given samples which are the set of programs allocated on hosts-machines. The agent transfers two arguments to the "calculator": amount of processes and input data. It calls MPIRun with required arguments and a parallel program [6].

The code of the program contains compiling directives with call functions from MPI library to spawn required quantity of processes and start implementation requested by the client. The output information after termination of calculations is saved in a special log-file that subsequently will be presented on a browser of the client. The client can also download the output data to a file. All input and output data are saved on the database. All data will be written down in a text file. In the failure case, the information with mistakes report will be written down in log-file. Server script opens the log-file with results, reads, and writes down in a database. Figure 2.3 shows the user interface of the Web-portal.

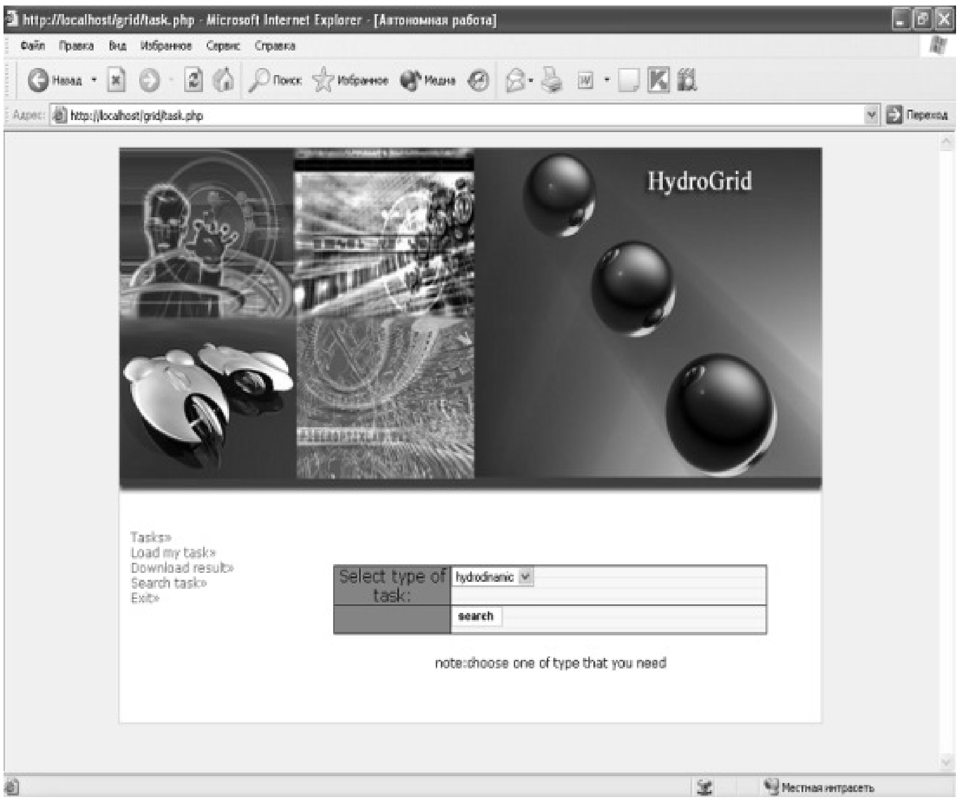
http://localhost/grid/task.php - Microsoft Internet Explorer - [Автономная работа]
Файл Правка Вид Избранное Сервис Справка
Назад
Поиск
Избранное
Медиа
Адрес: http://localhost/grid/task.php
Переход
HydroGrid
Tasks»
Load my task»
Download result»
Search task»
Exit»
Select type of task:
hydrodinamic
search
note:choose one of type that you need
Местная интрасеть

FIG. 2.3. *GUI of the Web portal.*

We developed the parallel algorithm for two-dimensional non-stationary heat conductivity equation in rectangular area. To solve the problem we create two-dimensional grid $M \times N$ that was divided on six parts (Figure 2.4).

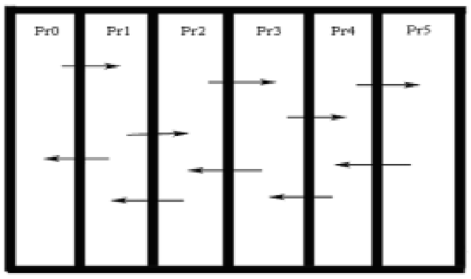

FIG. 2.4. *The divided area for the problem solution.*

We use the explicit scheme for solving of the equation in private derivatives. The parallelization technique involves not blocked data exchange between processes. MPI has a number of procedures for realization of asynchronous data transmission. In order to cancel an asynchronous exchange we call additional procedure that checks whether operation is completed or it is waiting for completion. We use send-buffer after all processes have terminated. Figure 2.5 shows us the computation time of the algorithm and it also confirms the benefit of parallel execution. The prototype of the algorithm was included to the library.

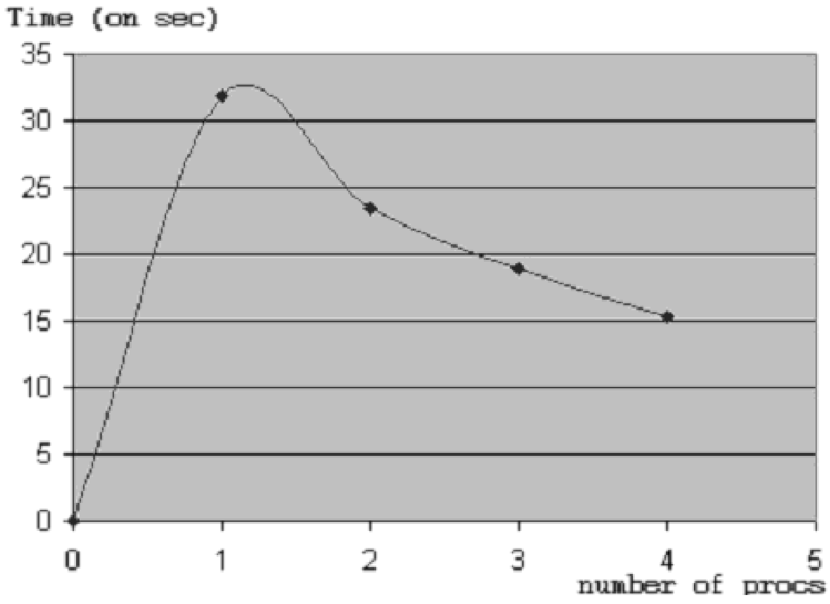

FIG. 2.5. *Execution time of the parallel algorithm.*

In addition, the server script generates a graphical representation of obtained results. When server script reads data from log-file it uses a system of flags. The flag will be equal to zero if "calculator" still works else it equals to 1 if "calculator" has finished work. The keeper of a flag is an output file, which is also provided in the termination detection service. It is available to extend the library of system by loading actual problem-solving algorithms. The system gives opportunity to view obtained results, save it in different text and graphics formats, and to analyze them.

2.3. Performance Control. The proposed PSE has some features of dependability and reliability. These features are realized on the second level by using of Service Manager component.

For instance, we face a situation when there is more than one request done simultaneously. It leads to the problem of sharing resources between the clients. To resolve this problem the system proposes a combination of the two decisions:

1) To block the server until the problem of one client is executed.

2) In regular intervals to distribute resources between clients, and supply them with full statistics about a quantity of clients that are on-line at the moment.

The task is analyzed by the following rules: how much time we need to spend to execute it and how many processes are required. Those functions are carried out by Service Manager's component. Here, we describe the principle of component functionality and the embedded algorithm.

Thus, we take a queued network model as the basis of Service Manager's component of the proposed system. We now consider queued networks that can evolve over discrete time. We treat the architecture of a queued network as abstract, in the sense that each service in the architecture defines a required functionality, but is yet to be instantiated with an actual service.

As we are only concerned with performance properties of evolving networks, we model such directories of alternative instantiating services by sets of service rates. From our performance oriented perspective, an evolving architecture consists of a sequence of service allocation choices to an abstract queue over time.

We now define our models of service communication and reconfiguration. An open queued network consists of a set of services connected to each other. We assume each service can receive messages, process the message and then send a new message. Each service takes a finite amount of time to process a message. Processing time may vary over the life of the network. Messages are sent between services via connections. Each service has its own queue and, if more messages arrive than can be immediately processed, they are stored in the queue and will be processed subsequently by the service according to a FIFO strategy.

The connections between services may take different forms. We use the same model as was proposed by [7]. Now we formulate the optimal control problem as reconfiguration of the system to be as near as possible to given desirable response time RT_{des} with minimum total processing cost of the system. We control the number of servers in each component at each moment of time.

The problem was solved by dynamic programming technique [8]. We have tested a simple example. We simulated our solution applied to two abstract services in sequence, with a Poisson distribution arrival rate. Both abstract services have (separate) sets of 6 equivalent services, each with different service rates and costs.

Table 2.1
Possible services for abstract service A.

Service chosen for A	**Service rate**	**Cost**
1	35	10
2	40	20
3	45	30
4	50	40
5	55	50
6	60	60

Table 2.2
Possible services for abstract service B.

Service chosen for B	**Service rate**	**Cost**
1	37	15
2	43	23
3	47	35
4	51	51
5	52	52
6	55	55

The arrival rate to the first component had Poisson distribution. The desirable response time was given as 0.4 sec. In general, the results showed us that it is optimal to use the first, second, and the fourth servers for the first component and the first server for the second component (Figure 2.6).

From Figure 2.7 we see that the total cost decreases. It can be explained by the fact that while a number of requests arrived to the first component was approaching to its maximum arrival rate, the Service Manager has used cheap servers. The servers were switched on the expensive ones only when a number requests became small, however, the response time was stayed on desirable level. We avoid using of heavy loaded servers which has a high cost when arrival rate of requests is extremely high and we allocate a small number of jobs to them if their quantity is low.

3. Conclusion. Using a Web portal has a number of well-enumerated advantages over specially designed multiprocessor systems such as: high performance workstations and PCs are readily available at low cost, the latest processors can easily be incorporated into the system as they become available, existing software can be used or modified. Grid infrastructure gives to end users interactive control, access to remote storage system at any time, and tool to solve a computational problem. The main idea of the given work was to build PSE implemented as a Web portal that allows clients to choose the most appropriate services to solve some problems with using of matrix-algebra and numerical methods based on MPI techniques. To provide desirable performance the Service Manager's component approach was developed. The main idea of Service Manager is to control the load of the system and satisfy to given requirements. To reach this goal dynamic programming technique was used. In addition, a parallel algorithm for two-dimensional non-stationary heat conductivity equation in rectangular area was included to the library of the Web-portal. In general, it is available to extend

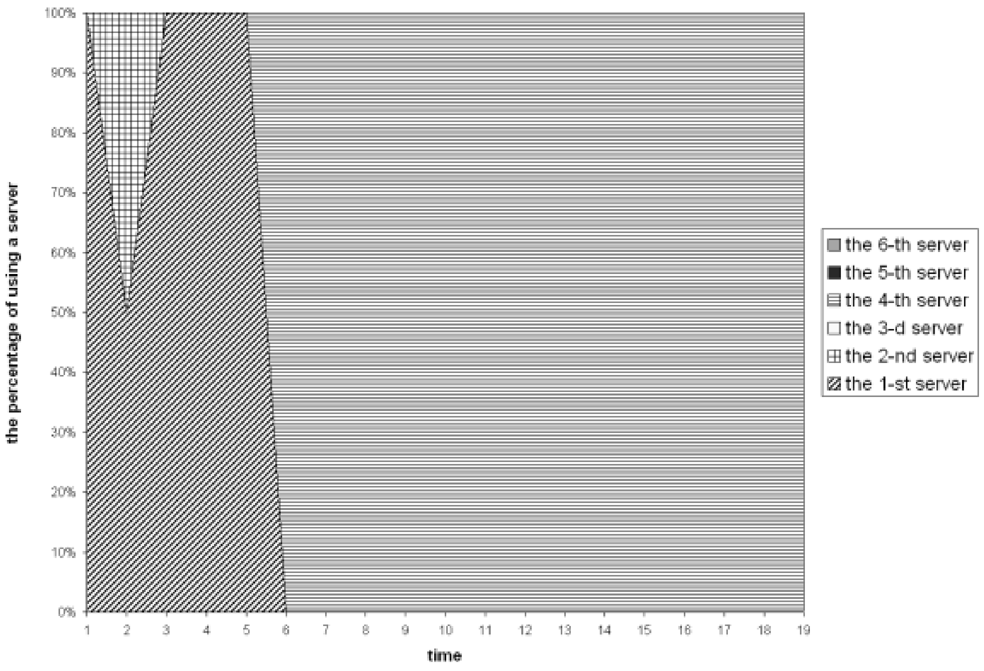

Fig. 2.6. *The percentage of using servers for the 1-st component.*

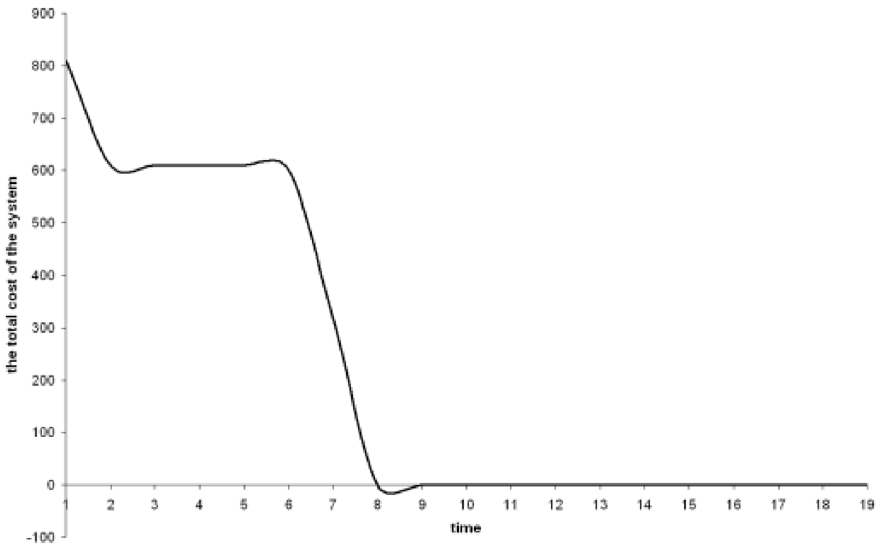

Fig. 2.7. *The total cost of the system.*

the library of system by uploading user's solutions. Thus, the proposed system allows the rapid prototyping of ideas, detailed analysis, and higher productivity.

In future, it is expected to make the proposed system self-adaptive by using of the Service Manager's component. In conclusion, the Web portal was created to solve different applied problems with use of MPI on the basis of the above described three-level platform. Any registered user can access the portal on-line with any Web-browser. Using of the proposed system will allow constructing the Internet services which would be able to solve vital applied problems.

REFERENCES

[1] Assel Zh. Akzhalova, Daniar Y. Aizhulov: Web portal to make large-scale scientific computations based on Grid computing and MPI, *Proceedings of PPAM 2007 Seventh International Conference on Parallel Processing and Applied Mathematics,* (LaSCoG-07) pp. 57–62.
[2] Jack Dongarra: Sourcebook of parallel computing. 2nd edn. Morgan Kaufmann Publishers, San Francisco (2003).
[3] A. C. Catlin, S. Weeravarana, E. N. Houstis, and M. Gaitatzes: The PELLPACK User Guide. Department of Computer Sciences, Purdue University, Lafayette, IN, 2000.
[4] Harry Jordan, Gita Alaghband: Fundamentals of parallel processing Pearson Education, Inc. New Jersey (2003).
[5] V. Wilkinson and M. Allen: Parallel programming: techniques and applications using networked workstations and parallel computers. Published by Prentice-Hall, Inc. (1999).
[6] S. A. Nemnjugin, O. L. Stesik: Parallel programming for high-efficiency multiprocessing systems. St.-Petersburg, (2002).
[7] Mahadevan Iyer, Wei Kang Tsai: Time-Optimal Network Queue Control: The Case of Multiple Congested Nodes, *Proceedings of 10th International Conference on Parallel and Distributed Systems (ICPADS'04),* pp. 709–718.
[8] P. Dmitri: Bertsecas Dynamic programming and optimal control. Athena Scientific, 2000.

Edited by: Dana Petcu, Marcin Paprzycki
Received: May 4, 2008
Accepted: May 24, 2008