Optimisation of usage of mathematical libraries in the Grid environment

Authors: Maciej Brzeźniak[1], Norbert Meyer[2] {maciekb}{meyer}@man.poznan.pl
[1, 2] Poznan Supercomputing and Networking Center

Abstract

The paper describes the idea of optimising the usage of mathematical libraries exploited by applications running in the distributed, heterogeneous Grid environment. The main feature of approach presented is a distributed execution of individual function calls. The paper presents the current state of the art and the concept on which PSNC is working. As the approach is based on GridRPC mechanisms proposed in the NetSolve and Ninf projects, the paper discusses main aspects of these systems and it presents the improvements and extensions PSNC is working on. These extensions address the issues that are crucial to demands of the polish scientific environment. The area of work includes the tools for seamless adaptation of the existing user applications for using the GridRPC mechanisms and techniques that would improve the fault-tolerance of the remote function calls. PSNC develops techniques for incorporating the computational resources working under the control of the resource managers (such as queue systems or Grid brokers) into the system. PSNC also work on the extensions of the system scheduling mechanisms. They would give the scheduling unit the possibility to automatically adapt to characteristic of the given computational environment and to plan the execution of user computations in queue systems.

1 Introduction

The emerging development of the Grid technology gives the users ability to exploit the advanced computing infrastructure. The components of this infrastructure are administered by individual (district) organisations. Many tools, mechanisms and technologies are worked out in the frameworks of national and international projects all over the world.

Although there is still an open issue, how to give the users easy, seamless and transparent access to the resources of the Grid without withdrawing from them the possibility of exploiting advanced features of the Grid elements and mechanisms realised by them.

This issue is general and applies to all the applications of the Grid systems. However, considerations made in this paper focus only on the computational applications using mathematical libraries.

1.1 Mathematical libraries in the Grid environment

The main areas where Grid systems are needed and used are the scientific researches. Additionally, the scientific researches are the main driving force of many Grid projects.

Majority of scientific researches is the computations. Many computations are composed of the standard, frequently repeated mathematical operations such as matrix transformation, solving the systems of equations etc. As the scientists represent numerous disciplines (physics, biology, chemistry etc.) and they generally are not computing science experts, they do not want to implement the numerical algorithms by themselves. Therefore, there is a need to provide the ready-to-use implementations of such mathematical functions to programmers. Therefore, C and Fortran function packages are developed (e.g. BLAS – Basic Linear Algebra Subprograms, LAPACK - Linear Algebra PACKage) and delivered as programming libraries.

There are many versions and implementations of individual mathematical packages: from free, open source, “general” versions to the highly optimised commercial ones (e.g. from NAG Software Ltd.). The platform- and architecture-optimised versions are able to use the power of underlying computing systems (SMPs, MPPs, clusters etc.). The programmer gains access to the power of well-designed numerical algorithms and well-equipped computing nodes without any specific (computing science-related) knowledge simply by linking his application with mathematical libraries.

However, in many research areas computing power of the single machine (even if it is the supercomputer) or cluster of machines is not sufficient. In order to satisfy the application needs, it is necessary to use a large number of computing machines. The computing resources (both hardware and software) may belong to many organisations (institutions) and they may be distributed logically and geographically. Therefore, the suitable computation running and management system is required. Grid structures seem to be an appropriate environment to it.

The main issue is how to run the user application in the distributed Grid environment and, as a consequence, how to programme Grid-enabled applications. There are some different ways to tackle with this problem.

1.2 Programming models for the Grid

The most common approach to running the application in the Grid structures is known from the queue systems e.g. LSF (Load Sharing Facility) [1], NQE (Network Queuing Environment) [2], PBS (Portable Batch System) [3]. It is also adopted in Grid systems that rely upon Globus Toolkit [4] and/or Legion [5]. The distribution of application is performed by placing deploying its copies over the Grid nodes and launching these instances using appropriate running services (e.g. Globus GRAM). The essential feature of this approach is that the application is put into system, run on it and migrated between its nodes as a whole. In case of applications using mathematical libraries, the source codes of its particular instances are the same. However the application’s instance must be linked with the local (i.e. installed on the given computational node) copy of mathematical library. Thus, this approach to distributing the computational application requires the complicated code maintenance.

There is also an another model of distributed computing – the main part of the user application is left on the selected computing node (e.g. user workstation) while the others are distributed over the several computing nodes. These parts could be activated using the remote function calls (RPC-like solutions) or the invocations of methods of remote objects (e.g. CORBA, Java RMI). This approach is more suitable to perform the distribution of the computational applications that exploit the mathematical libraries than the method mentioned in above paragraph. This approach is accepted by the authors of NetSolve [6] and Ninf [7] projects. These systems deliver the means to distribute the mathematical application in an easy and efficient way. Both solutions have the specific features but they are based on the similar idea. The generalisation of these mechanisms was worked out and it is called Grid RPC [8],[9] technology. The main advantage of this approach is that the code maintenance on the client side is simple.

1.3 Grid RPC

The leaders of *Ninf* and *NetSolve* projects define the Grid RPC as following [8]: *Grid is an RPC mechanism tailored to the Grid. At a very high level view the programming model provided by Grid RPC is that of standard RPC plus asynchronous coarse-grained parallel tasking. In practice there are a variety of features that will largely hide the dynamicity, insecurity, and instability of the Grid from the programmers.*

The details of Grid RPC technology will not be discussed in this paper as it could be found in [8] and [9]. The incorporation of the Grid RPC mechanisms into the Globus Toolkit is planned, for further information on the status of that technology please refer to [10].

Grid RPC is the generalisation of techniques implemented in reference Grid RPC systems: NetSolve and Ninf. The mechanism of executing the remote calls in the Grid RPC system is shown on the Fig. 1 (NetSolve example). The user application calls the NetSolve client module function (1). The module requests the NetSolve agent to assign resources for call execution (2). The task running facility of the agent serves the request. It invokes the scheduling unit that selects the resources for serving the call. It uses the data residing in the resource and performance databases. The decision is returned to the client module (3). Next, the module invokes the remote computations on the assigned resource. The input data of the task is transmitted to the NetSolve server module (4). The server module launches the computations, using the copy of mathematical library installed on the remote machine (5), (6). Then the results are returned to the client module (7) and through it to the application (8).

The system agent includes resources and performance databases. The resource database stores the list of the mathematical problems' solvers registered to the systems through the resource discovery mechanism.

The performance database includes the information concerning the current and predicted state of computational resources incorporated into the system, e.g. load, amount of memory available, network links state (observed latency and bandwidth provided) etc. These data are delivered by the monitoring and prediction mechanisms that operate continuously.

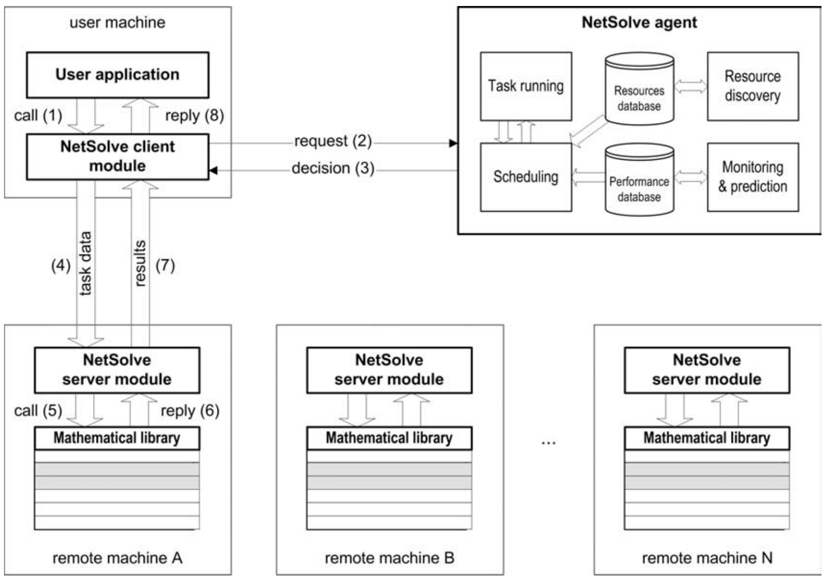


Fig 1: The remote call execution schema

In the next part of the paper, reference Grid RPC systems (Ninf and NetSolve) will be called *the system* in short. Their clients modules will be refereed as *the client module*. The central system module controlling the execution and managing of the resources will be called *the agent*. The elements of system that are installed on the computing machines (called *the servers*) will be referred as *server modules*.

1.4 How does the Grid RPC optimise usage of mathematical libraries?

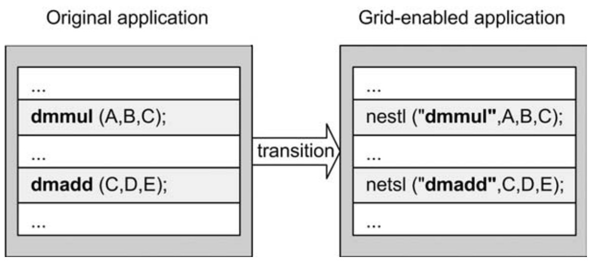
The short overview of Grid RPC systems provided above shows that these systems optimise usage of mathematical libraries existing in the Grid environment. There are several aspects of that optimisation.

Firstly, the Grid RPC systems provide simple-to-use access to complicated software libraries. That is one of the best motivations for using this technology, as main users of mathematical libraries are the programmers not familiar with low-level distributed programming techniques. Using Grid RPC technology they get access to software and hardware, that they were not able to use before in as easy manner.

Secondly, they may use remote copies of libraries directly from their machines even if the library is licensed only for the given remote machine. Moreover, the installation of all the mathematical packages on all the Grid nodes is not necessary. That may lead to considerable saving, as the software licenses are very expensive, especially for highly optimised, specialised mathematical packages. Therefore, from both the user's and computing systems owners' point of view, using the Grid RPC systems is profitable.

Thirdly, the systems give the user application access to the numerous, powerful computing resources. This affects the performance of the computations. Moreover, the access method is very efficient. First, mechanisms used in NetSolve and Ninf are very lightweight. They do not bring too many overheads. Second, the system agent implements scheduling techniques and selects the most suitable resources to perform given computation.

Fourthly, although providing the access to the remote resource the Grid RPC mechanism remains easy to exploit. The local function calls in the application source code shall only be replaced with the remote execution calls. The transition itself is quite easy. The example transformation of the fragment of the application source code (C language case) is shown in the picture below.



```
...
dmmul (A,B,C);
...
dmadd (C,D,E);
...
```

```
...
nestl ("dmmul",A,B,C);
...
netsl ("dmadd",C,D,E);
...
```

Fig 2: Transition from local call to the Grid RPC call (NetSolve case)

The remote mathematical function call is very similar to the local one. The mathematical functionality is specified by its name (identical to the name in the local library). Then the problem input and output data are specified. No additional parameters are required, e.g. referring to the details of the remote call.

Fifthly, as the Grid RPC API is so simple, there is no need for client-side code maintenance, i.e. no RPC stubs or client-side IDL data (Interface Description Language) must be provided by the programmer. The remote function is called by its name without considering the underlying operations needed to realise this call. Marshalling of the task input arguments and results of computations are made automatically by the system client module on the base of information acquired from the system agent. This simplicity is a characteristic feature of the Grid RPC mechanism.

Sixthly, getting access to *the system* is also simple. User must download the system client module and exchange the local function calls with the remote ones. Then, he has to recompile the code and link it with the system client module (instead of the local mathematical library module). From this moment, mathematical functions can be executed remotely.

Seventhly, the Grid RPC reference systems provide API for both C and Fortran languages. Interfaces for Matlab, Unix shell, Mathematica and Web-based Java GUI (NetSolve) as well as for Mathematica and Excel (Ninf) are developed. Thus, apart from programmers also "general" users can take advantages of Grid RPC technology and the optimisation of usage of mathematical libraries that this technology provides.

Eighthly, although the scheduling policies realised by the system agent may be set up to fulfil user needs, it is also possible to match the scheduling policies to computing resources owner's requirements. The scheduling policy can be adjusted to the given environment needs. It is possible to realise scheduling algorithms that minimise the request service time from the one hand or techniques optimising resources usage (load balancing) from the other. In short, various resource management policies can be realised by the system agent.

To recapitulate, putting the Grid RPC systems into practice make the usage of mathematical libraries much more easy, efficient and rational. The Grid RPC systems bring the benefits to both the end-users and computing systems owners. End-users gain the simple, consistent and efficient interface to various and numerous mathematical libraries as well as access to the numerous and heterogeneous hardware resources (that mathematical libraries are installed on). The owners of computing systems gain the possibility of managing the resources (especially the mathematical software licences) in an easy and economical (rational) manner. Giving the access to these resources to a large number of users becomes quite simple and seamless when using Grid RPC systems.

2 Putting Grid RPC systems into practice

The reference Grid RPC systems optimise the usage of the mathematical libraries. They were invented in order to challenge requirements of many scientific environments. The system mechanisms are simple, lightweight and universal

PSNC accepts these systems as the basis of its own *system*. We put these systems into practice, test and evaluate them. However, we realise that these systems do not fit ideally to requirements of polish Grid environment. Therefore, we propose some improvements of these systems.

2.1 Client-side mechanisms

The extensions proposed and developed by PSNC include the client-side tools that help the users to adapt their existing applications to the Grid RPC system usage. The mechanisms that increase the fault-tolerance of the remote calls are also designed and tested by PSNC.

Auto-adaptation tools

Many applications that use mathematical libraries exist in our environment. As mentioned in the introduction, the Grid RPC technology provides the user the means to perform the computations on the remote distributed resources in an easy and efficient way. However, the source code of the existing application must be edited manually by the programmer to give the application access to the Grid resources. This can discourage scientists from using the Grid RPC for the existing applications.

Therefore, we propose the source code pre-compiler, that will exchange local calls with Grid RPC calls in the application source code automatically or semi-automatically.

The recognition of the mathematical function calls inside the application source code take place by comparing each function call with the list of known mathematical functions. If found known mathematical function call, it will be converted to the remote call. The list of known mathematical functions will be acquired from the system agent's resources database. In the basic version, the pre-compiler would be a command-line tool. However, we consider giving the user the possibility to confirm each local-remote call exchange. Therefore, pre-compiler should be an interactive text or graphic tool.

Conversion of the local calls to the API of both Grid RPC reference systems will be possible. The decision whether to use the NetSolve or Ninf API instead of the local could be made either for the whole source code (before pre-compilation) or per single function call (if the interactive version of pre-compiler would be developed). PSNC also consider the possibility of translation to the Grid RPC API.

Presently, this tool is in the design phase. In our opinion it will make the adaptation of existing computing applications to the use of Grid RPC systems easy and seamless. Thereby, it will help the user to exploit the system optimising the usage of mathematical libraries in heterogeneous Grid environment.

Increasing the fault-tolerance of remote calls

The existing Grid RPC systems realise server-side fault-tolerance mechanisms including detection of failures of the computational servers, restarting interrupted computations on another system nodes etc. [11], [12], [13].

However, no fault-tolerance mechanisms are realised by the client module of existing systems. Moreover, the decision whether to perform the execution of the given function locally or remotely is made by the programmer, in the moment of exchange of the local function call with the remote one. If the programmer decides to perform the computations through the Grid RPC system, he has the chance to gain a better performance. However, if the system agent becomes unavailable during the application execution (e.g. due to its failure or the network link breakdown), then the execution might hang until the agent become available.

Therefore, we propose the emergency local function call execution mechanism for Grid RPC systems. It could be exploited by linking the user application with our own version of the client module. This version of the client module will check the availability of the system while the application calls the mathematical function. If the agent is accessible, the modified client will invoke the remote computations by calling function of original NetSolve/Ninf API. Otherwise, it will use the local copy of mathematical library as long as the copy remains on the client machine.

The modified client module will be worked out by PSNC and put into practice. This solution will improve the fault-tolerance of the application execution. It requires the extension of the client module, so the client lightweight will be lost. Nevertheless, in our opinion, it is acceptable since it is beneficial.

2.2 Incorporation of specific resources

PSNC believes that to optimise access to the libraries means also to give the user application access to all the computational resources that can be used. Many computational resources in the polish scientific environment work under the control of the queue systems (e.g. LSF, NQE and PBS). While the existing Grid RPC systems have the plug-ins for the Condor and Globus resource brokers [13], [14], they have no plug-ins for the queue systems. Therefore, such mechanisms for both Ninf and NetSolve systems should be developed.

PSNC decided not to use the Globus nor Legion interfaces and their interfaces to the queue systems. It is caused mainly by the fact that these interfaces bring considerable overhead while the reference Globus RPC systems was designed to be very lightweight and highly efficient. Therefore, the Ninf- and NetSolve-dedicated mechanisms seem to be useful.

In a general case running computational server modules of the system “directly” on the resources controlled by the queue systems would be possible. Unfortunately, each site has its own resource management policies and these policies must be respected. Yet, computations launched on such machines without the usage of queue system’s interfaces might violate these policies. Moreover, in many sites (including PSNC) running the interactive applications on resource that belongs to queue systems is not allowed at all.

The plug-ins we develop will make possible the running of mathematical function calls as the jobs in queue systems. They include the mechanisms for running such queue system jobs and the techniques for preparing mathematical libraries to use in Grid RPC systems. The effective scheduling mechanisms are required for the mathematical function calls that are going to be executed through queues. This issue will be discussed in details in the next subsection of paper concerning the extensions of scheduling mechanisms for system agent.

PSNC plans to develop plug-ins for LSF system firstly. The experiences gained in that stage of work are going to be used while working out similar solution for the NQE system. The implementation of plug-ins for PBS is also considered.

Recapitulating, Ninf and NetSolve plug-ins for LSF and NQE will give the user application access to the computational resources that work under the control of the queue systems. This will widen the range of operation of the user application. Additionally if the access method will be effective, the performance of the application will be improved.

2.3 Scheduling techniques

We propose some improvements of the scheduling techniques used in Grid RPC systems. We also design the scheduling techniques that will be appropriate to effectively run Grid RPC system tasks in the queue systems.

Knowledge-based scheduling mechanisms

The Grid is in its nature heterogeneous. Various types of computing systems (SMPs, MPPs, clusters and queue systems) are working in polish computational centres. Each of them has a “predisposition” to solve different classes of mathematical problems. Also, various implementations of mathematical libraries (open source, free versus highly-optimised, commercial: e.g. from NAG ltd.) are used. The environment is very dynamic – new resources are adding, the configuration (number of processors, memory amount etc.) and management policies of existing resources are changing.

Thus, the computational resources in Grid are diverse. This fact is considered by scheduling techniques exploited in NetSolve and Ninf systems. However, in the opinion of PSNC these mechanisms are too general, as they use simple performance models to predict the amount of time needed to solve given mathematical problem on the given resource. What is more important, these models are static – they do not consider the dynamic configuration changes and do not use the historical data.

Therefore, PSNC proposes some improvements of scheduling techniques in Grid RPC systems. The main idea of these improvements is the collection and analysis of the historical data concerning the executions of functions of mathematical libraries. Two different solutions are considered.

The first is based on the computational complexity function and the performance models. The solution relies on the fact, that the execution time of the mathematical function call is much more predictable than the execution time of the general task. The general tasks perform not only mathematical computations but also other operations, like opening files, making network transfers etc.

The computational complexity function of the given implementation of mathematical operation is usually known (it is often specified by the library supplier). Otherwise, it can be discovered, e.g. by using the measurement taken during the series of executions of the given mathematical task implementation. The unknown *a priori* function can be reconstructed, e.g. using the polynomial interpolation performed on series of measurements. The unknown coefficients of the known a priori complexity function could also be determined in that way.

However, such reconstruction requires a series of measurements of task execution times to be collected. These measurements could be acquired by performing the test computations using particular library functions. Unfortunately, the cost of such "learning phase" is not acceptable in the production environment. Its scale should be large to provide enough data to conclude the complexity function or its coefficients.

Above statements apply to accurate methods. However, many mathematical libraries implement the approximate methods. Nevertheless, the convergence coefficient of these methods is known or can be discovered on the base of data collected during their executions.

In short, the number of elementary operations performed by the accurate methods could be foreseen on base of the algorithm computational complexity function knowledge. Similarly, the number of iterations of approximate algorithm that will be done by the given call execution could be foreseen by using its either known or discovered convergence coefficient. PSNC will develop the mechanism for reconstructing the complexity functions and/or convergence coefficients from the data collected during the executions of mathematical library functions. This mechanism will give the scheduling unit of the Grid RPC system the ability to recognise the characteristics of the environment. That knowledge will be achieved while the system work. Therefore, the mechanism will also give the scheduling unit of the agent the possibility to adapt to changing configuration and state of the Grid environment.

Before the mechanism will be realised, first PSNC must face numerous problems. The influence of the resource state (e.g. load, amount of memory available) on the performance of computations launched on it should be took into account. It applies to both processes: analysing the function "behaviour" and the process of predicting the call execution time. The measures of execution times, used to recognise the "behaviour" of the function in the given environment, are affected by the state of the resources in the moment of task operation. The performance gained by the user computations in future is also going to be impacted by the state of the resources in the (future) moment of execution. Therefore, the relevant resource performance models should be exploited. The models developed in several projects (e.g. RCS - Remote Computation System [15], [16]) might be used, after its evaluation in the real environment or there should be evolved a new models. They should be able to express the features of different computing resources despite their different architectures.

The discussed so far techniques are complicated. They may bring substantial overhead on scheduling the client requests. Therefore, putting these mechanisms into practice will be proceeded by analyse and evaluation of them in the test installation.

The second mechanism of exploiting the collected historical data (concerning the executions of mathematical libraries functions) to predict the future executions is based on the artificial intelligence techniques. We consider the mechanisms that would consider simple relationships between state of the resources and the performance of computations performed on them. The discovered rules can be as simple as that: if the load of the resource A is smaller than x, the performance offered by it is about a, but if the load exceeds y, the observed performance is not greater than b. Such simple performance model can be exploited by the scheduling units. These rules can be introduced to the expert system that will support the scheduling unit in predicting the amount of time needed for the execution of the given task. However, such rules could be also automatically constructed during the system operation. PSNC will also consider the usage of neural networks as the mechanism for recognising simple relationships (between state of the resources and the performance of computations run on them) during the system work.

The mechanisms discussed will give the agent of the Grid RPC system the ability to recognise the predisposition of the individual software and hardware resources to solve the particular mathematical problems.

These solutions will be analysed and tested in real environment. The evaluation will focus on the effectiveness of the decisions made using the mechanisms worked out. The overhead that they introduce will also be evaluated. If results are promising, the knowledge-based scheduling mechanisms will be put into practice.

Scheduling mechanisms for queue systems

PSNC plans to incorporate the resources controlled by the queue systems into Grid RPC system. Therefore the appropriate mechanisms of predicting the task execution time in the queue systems should be provided to scheduling units of the system agent. The total time needed to serve the user computation request in the queue system is consisted of the execution time and the amount of time spent by the task in queue (waiting for the resource availability). While the former element is predictable (e.g. using techniques mentioned above), the latter is hard to foreseen. The degree of unpredictability depends strongly on the configuration of the queue that the Grid RPC system has access to. If the given queue runs only the tasks that belong to the Grid RPC system, the amount of time needed to complete the job execution is foreseeable. If the queue serves not only Grid RPC system's tasks, the prediction is impossible (since the amount of time other tasks are going to spend "on resources" is unknown).

The data needed to foreseen the waiting and the execution time of task will be acquired by the agent from the queue system broker. They include the static information concerning the configuration of the queue (priority, assigned resources: processors, memory, disk storage etc.) and the dynamic information e.g. the state of the system, number of tasks waiting in the queue etc.

3 Related work

Initially NetSolve and Ninf were referred to as NES systems (Network Enabled Solver). Some systems implement similar functionality. NEOS [17] is a Condor-driven optimisation solvers server accessible through the web, e-mail interfaces and special job submission tool. The RCS (Remote Computation System) [15], [16] is a remote procedure call facility providing the uniform access to a variety of supercomputers. It is focused on load balancing, implemented on top of PVM and accessible for the clients through the specific mechanisms of calling library functions.

Mechanisms for broadening the range of computing resources that the Grid RPC application can use are developed in the frameworks of Ninf and NetSolve projects. The Ninf-NetSolve gateways were designed and implemented [12]. In addition, the Condor interfaces were developed for these systems. Additionally, both Ninf and NetSolve migrate to the Globus services usage. The Globus-enabled versions of these systems are worked out, e.g. Ninf-G [8].

The resource management policies (i.e. scheduling mechanisms) for Grid systems are developed within the confines of numerous projects. Some of them are able to make the multicriteria decisions concerning the assignment of the resource to execute the user task e.g. the multicriteria resources broker developed in the GridLab project [18]. However, majority of them uses the RSL descriptions (Resource Specification Language) provided by programmers as the input data for scheduling process. Therefore, such scheduling techniques cannot be exploited in Grid RPC systems since the user of such systems do not specify the resource requirements.

The techniques that realise the prediction of the amount of time the batch job is going to spend in queue (its wait time and its run time) are also developed in several projects. The prediction agent of the GridLab's resource broker is supported by heuristic algorithms, fuzzy and rough set techniques etc. [18].

4 Conclusions

There are some technologies that optimise the usage of mathematical libraries in the Grid environment. The most promising is the Grid RPC technology. It brings the benefits to both end-users and computing systems owners. User gains the simple and uniform method of accessing numerous mathematical libraries. Additionally, the access method is effective. The owners of the computing systems gain the possibility of managing the software resources (especially licences) in a rational manner.. Implementation of various policies of managing computational resources (both software and hardware) is also possible (e.g. minimisation of system response time versus resources load balancing).

PSNC accept these systems as the basis of its own *system*. The existing Grid RPC systems are put into practice, they are tested and evaluated. PSNC realise that these systems do not fit ideally to the polish Grid environment needs. Therefore, some improvements and extensions are worked out in order to put these systems effectively into practice.

Client-side fault-tolerance mechanisms are designed and tested. The tools for seamless adaptation of existing applications to use the Grid RPC systems are also drawn up. The implementation of client-side mechanisms does not require any change in the structure of existing Grid RPC systems since these mechanisms are external.

However, PSNC develop also new technologies for the Grid RPC systems. These technologies include the mechanisms that make possible the incorporation of the resources working under the control of queue systems into the Grid RPC system. New scheduling techniques are also developed and tested. The improvements of scheduling techniques apply mainly to the techniques of predicting the time needed to solve the mathematical problem on particular resources. PSNC consider two solutions. The first use the computational complexity function (or convergence coefficient) of algorithms that the solvers implement and the performance models of computational resources. The second solution employ the artificial intelligence techniques to recognise the characteristics of the resources, i.e. relationships between the state of the resource and the performance offered by it.

The Grid RPC systems allow to optimise the usage of the mathematical libraries in the Grid environment. The extensions of these systems designed in PSNC are intended to make the usage of libraries even more effective and easy.

5 Future work

The techniques developed in PSNC will be introduced in the Grid RPC reference systems (Ninf and NetSolve). These techniques are invented to fulfil requirements of polish Grid environment. However, the main point of interest of PSNC is the design of the new technologies for the Grid systems and the improvement of existing technologies.

PSNC introduces and tests the concepts in real systems but research results might be used in a wider perspective. This mainly relates to the area of scheduling mechanisms and resource management techniques.

The mechanism for reconstructing and using the computational complexity function (or convergence coefficient) of the algorithms implemented in mathematical libraries could be adapted to more general applications. This technique could be employed by the brokers of Grid systems to recognise “the behaviour” of library functions, methods of objects or programmes run in the Grid environment. The performance models could also be exploited by Grid resource brokers.

Discussed mechanisms would give the resource brokers of Grid system the ability to recognise the predisposition of the individual software and hardware resources to execute the given functions or programmes. Additionally, the statistical analysis of the data concerning performance of computations made on the given node of Grid could make possible the conclusion of some quality parameters of this node. The value of some quality meters could be computed using that historical data. The quality meters might include availability rate, probability of fault etc. Such factors could be taken into consideration by the Grid brokers while assigning the resources to serve the user requests.

Generally, the experience of distributing the mathematical applications could be used to draw up the techniques for distributing the other classes of programmes in the network environment. Such solution could be applied to the applications that are consisted of the parts with well-defined functionality.

References

1. Load Sharing Facility. http://www.platform.com/products/wm/LSF/index.asp.
2. Network Queuing Environment. http://www.cray.com/products/software/nqe.html.
3. Portable Batch System. http://www.openpbs.org.
4. Globus: A Metacomputing Infrastructure Toolkit. I. Foster, C. Kesselman. Intl J. Supercomputer Applications, 11(2):115-128, 1997.
5. Grids: Harnessing Geographically-Separated Resources in a Multi-Organisational Context. Anand Natrajan, Marty Humphrey, Andrew Grimshaw. Presented at High Performance Computing Systems, June 2001.
6. Network-Enabled Solvers and the NetSolve Project. H. Casanova, J.J. Dongarra, and K. Moore, SIAM News, January, 1998, Vol 31, No. 1.
7. Ninf: A Network based Information Library for a Global World-Wide Computing Infrastracture. Mitsuhisa Sato, Hidemoto Nakada, Satoshi Sekiguchi, Satoshi Matsuoka, Umpei Nagashima and Hiromitsu Takagi. HPCN'97 (LNCS-1225), pp. 491-502, 1997.
8. GridRPC: A Remote Procedure Call API for Grid Computing. Seymour, K., Nakada, H., Matsuoka, S., Dongarra, D., Lee, C., Casanova, H. ICL Technical Report, ICL-UT-02-06, June, 2002.
9. Overview of GridRPC: A Remote Procedure Call API for Grid Computing. Keith Seymour, Hidemoto Nakada, Satoshi Matsuoka, Jack Dongarra, Craig Lee and Henri Casanova. Grid Computing - Grid 2002, LNCS 2536, pp.274-278, November, 2002.
10. Global Grid Forum - Advanced Programming Models Research Group. http://www.eece.unm.edu/~apm.
11. Deploying Fault-tolerance and Task Migration with NetSolve. Henri Casanova, Jim Plank, Micah Beck and Jack Dongarra. http://icl.cs.utk.edu/netsolve/files/pubs/fault-tolerance.ps. 1997.
12. NetSolve: A Network Enabled Server, Examples and Users. Henri Casanova, Jack Dongarra. Proceedings of the Heterogeneous Computing Workshop, Orlando, Florida, 1998.
13. Fault Tolerance on the Ninf System. Satoshi Shirasuna, Hidemoto Nakada, Satoshi Matsuoka. IPSJ SIG Notes Contents, High Performance Computing, No.087.
14. Performance Evaluation of a Firewall-compliant Globus-based Wide-area Cluster System. Y. Tanaka, M. Hirano, M. Sato, H. Nakada and S. Sekiguchi. 9th IEEE International Symposium on High Performance Distributed Computing (HPDC 2000), pp.121-128, (2000).
15. The Remote Computation System. P. Arbenz, W. Gander, and M. Oettli. Parallel Computing (23): 1421-1428, 1997.
16. The Remote Computation System. P. Arbenz, W. Gander, and M. Oettli. High Performance Computing and Networking, H. Liddell, A. Colbrook, B. Hertzberger and P. Sloot (eds.). Springer-Verlag, Berlin, 1996, pp. 820-825. (Lecture Notes in Computer Science, 1067).
17. NEOS and CONDOR: Solving Optimization Problems over the Internet. Michael C. Ferris, Michael Mesnier and Jorge J. More. Mathematical Programming Technical Report 96-08, October 1996.
18. GridLab – A Grid Application Toolkit and Testbed. http://www.gridlab.org.