On the Easy Use of Scientific Computing Services for Large Scale Linear Algebra and Parallel Decision Making with the P-Grade Portal

Hrachya Astsatryan, Vladimir Sahakyan, Yuri Shoukouryan, Michel Daydé, Aurélie Hurault, Ronan Guivarch, Harutyun Terzyan, Levon Hovhannisyan

On the Easy Use of Scientific Computing Services for Large Scale Linear Algebra and Parallel Decision Making with the P-Grade Portal

Hrachya Astsatryan · Vladimir Sahakyan · Yuri Shoukouryan · Michel Daydé · Aurelie Hurault · Ronan Guivarch · Harutyun Terzyan · Levon Hovhannisyan

Abstract Scientific research is becoming increasingly dependent on the large-scale analysis of data using distributed computing infrastructures (Grid, cloud, GPU, etc.). Scientific computing (Petitet et al. 1999) aims at constructing mathematical models and numerical solution techniques for solving problems arising in science and engineering. In this paper, we describe the services of an integrated portal based on the P-Grade (Parallel Grid Run-time and Application Development Environment) portal (http://www.p-grade.hu) that enables the solution of large-scale linear systems of equations using direct solvers, makes easier the use of parallel block iterative algorithm and provides an interface for parallel decision making algorithms. The ultimate goal is to develop a single sign on integrated multi-service environment providing an easy access to different kind of mathematical calculations and algorithms to be performed on hybrid distributed computing infrastructures combining the benefits of large clusters, Grid or cloud, when needed.

Keywords Advanced trading · Sparse linear algebra solvers · P-grade portal · Parallel decision making

H. Astsatryan (✉) · V. Sahakyan · Y. Shoukouryan
Institute for Informatics and Automation Problems of the National Academy of Sciences of the Republic of Armenia, 1, P. Sevak str., Yerevan 0014, Armenia
e-mail: hrach@sci.am

V. Sahakyan
e-mail: svlad@sci.am

Y. Shoukouryan
e-mail: shouk@sci.am

M. Daydé · A. Hurault · R. Guivarch
University of Toulouse, INPT (ENSEEIHT)-IRIT, 2, rue Charles Camichel, 31071, Toulouse, France

M. Daydé
e-mail: Michel.dayde@enseeiht.fr

A. Hurault
e-mail: Aurelie.Hurault@enseeiht.fr

R. Guivarch
e-mail: Ronan.Guivarch@enseeiht.fr

H. Terzyan · L. Hovhannisyan
State Engineering University of Armenia, 105, Teryan str., Yerevan 0009, Armenia

H. Terzyan
e-mail: hterzian@seua.am

L. Hovhannisyan
e-mail: h_levon@seua.am

1 Introduction

Scientific computing aims at constructing mathematical models and numerical solution techniques for solving problems arising in science (including life and social sciences) and engineering. The solution of linear system of equations lies at the heart of most calculations in scientific computing. For the past twenty years, there has been a great deal of activity in the area of algorithms and software for solving linear algebra problems. It is often difficult for non expert users to take advantage of nowadays computational infrastructures (Grids, clusters, GPU, ...) or even to use the available advanced libraries, because they often require to be familiar with middleware and tools, parallel programming techniques and packages (MPI, OpenMP, CUDA, etc.) and numerical libraries (ScaLAPACK, LAPACK, BLAS, etc.).

The purpose of this article is to describe the scientific computing services that are deployed within our Portal and how their access has been made straightforward. One of the main components is an environment enabling the solution of large-scale linear systems of equations and algorithms on hybrid HPC infrastructures (heterogeneous Grids composed of clusters with and without GPUs), which is the extension of the previous work [3, 4]. The environment is responsible of scheduling the computing jobs to remote GPU and Grid computing facilities (workload management service) by matching the complexity of the user-submitted expression, the parameters of algorithms and the resource status information collected from information services. The computational resources of the Armenian National Grid (ArmGrid) infrastructure [5] has been used that unifies different architectures (multicore processors, GPUs) and network topologies (Myrinet, Infiniband, Gigabit). Currently ArmGrid consists of seven Grid sites located in the leading research and educational organizations of Armenia [6].

2 Related Work

Several projects that can be related to the Problem Solving Environments introduced in the 90's provide advanced functionalities but are typically restricted to a certain application area. They provide all the computational facilities needed to express and solve a target class of problems i.e. advanced solution methods enriched sometimes with an easy selection mechanism between alternatives. They often attempt to use the common language of the class of problems to simplify users life. Some examples are of course MATLAB, OCTAVE, NETSOLVE/GRIDSOLVE [7], CACTUS [8], DECIDE [9], NEOS [10], GRID-TLSE [11], PETSc, etc.

Several projects based on web portals are providing advanced features (workflow manager, easy deployment of services, authentication and authorization procedures, information services, etc.) for building problem solving environments or deploying software and tools for scientific communities, such as P-GRADE [12–14] and DIET [15]. Both P-GRADE and DIET are providing a workflow manager that allows an easy composition of services. They also supply facilities for deploying new services and provide APIs for generating web portals.

NetSolve/GridSolve aims at aggregating heterogeneous computational resources inteconnected by a network. It is a RPC based system that provides a seamless bridge between the standard programming interfaces and the use of the Grid. The library software deployed over the servers involved in the Grid is easily accessed from Fortran, MatLab, ... The NEOS server provides a large range of optimization software with a minimum input from users. Cactus is an open source problem solving environment designed for scientists and engineers that runs on many architectures. The Cactus user community has created and is maintaining toolkits for several research fields for example the Einstein Toolkit in computational relativistic astrophysics. GRID-TLSE is an expert site available through a web portal located at http://Gridtlse.org that provides an easy access to a large variety of direct solvers for sparse linear systems to enable comparative analysis on user-submitted problems, as well as on matrices from collections also available on the site. The site provides user assistance in choosing the right solver for a given problem and in setting the appropriate values for the control parameters of the solver. It is also intended to be a testbed for experts in sparse linear algebra which provides tools to create working groups, and a bibliography database on sparse matrices.

Our work makes use of P-GRADE for developing the web portal that allows users to enter their requests and recover the subsequent results.

Central to the work we describe in this paper, is the advanced trader we are using [16] and to our knowledge there is no generic approach capable of handling complex expression such as this tool available in a scientific computing environment. Our web portal mainly aims at simplifying the use of scientific libraries deployed as a set of computational services (i.e. each procedure of the library is seen as a service).

The globalization of resources (hardware and software) on large scale distributed computing infrastructures requires advanced trading mechanisms to identify the service or the composition of services able to fulfill a user request. Our advanced trader is using a sophisticated semantic-based service description similar to algebraic data type. The services (typically procedures of a library) are described in terms of their algebraic and semantic properties. This is quite similar to proceed to a description of algorithms in a certain application area, in other words it is based on the ontology of an application area covered by a scientific library (including description of the solution techniques). It is then possible with such a description to automatically compute the service or the combination of services that satisfy a user request expressed and then generate the list of library calls to be performed. The corresponding workflow with call to the local or remote procedures involved (depending on the amount of data or calculations) is then built automatically without the need of the user assistance. As a major benefit, users are not required to explicit call Grid-services but instead manipulate high-level domain-specific expressions similar to MATLAB to describe their calculations.

Similar issues are available within MATLAB or SciLab where user requests may occasionally be translated into a sequence of calls to a libray (e.g. $x = A \setminus b$ in MATLAB generates a workflow involving calls to LAPACK LU factorization followed by forward-backward substitutions for a dense square matrix).

Our approach is fully generic and can be used in almost all application areas. An experimental version of this approach has been implemented within GridSolve [17].

3 Grid Software Pool

Our software is structured into three main layers:

- User interface: it allows users to manipulate their data (initialization, uploading, suppression, ...) and to enter their computational requests using a MATLAB type of syntax.
- Analyzer/converter and advanced trader: the analyzer / converter is in charge of analyzing and converting the user expression into a XML format used as an input by the advanced trader in charge of computing the service or the list of services that fulfill the user request
- Job preparation modules: it converts the workflow generated by the advanced trader into a list of local or remote calls (depending on the location of services and on the size of the computations if services are available on several computers of different capacities), launches the calculations and recovers the results to be sent back to the user interface.

As a consequence, all the difficulties inherent to the use of the gLite middleware, MPI programming or construction of the library calls are then totally hidden to the user which is the main goal of our generic approach.

3.1 Overall Organization of the Grid Portal

The Web portal of our scientific computing environment is based on the P-GRADE Grid Portal. It is a web based, service rich environment for the development, execution and monitoring of workflows and workflow based parameter studies on various Grid platforms. It hides low-level Grid access mechanisms by high-level graphical interfaces; making even non Grid expert users capable of defining and executing distributed applications on multi-institutional computing infrastructures. The P-GRADE Portal can access multiple Grids simultaneously, which enables the easy distribution of complex applications on several platforms. Our scientific computing environment aims at providing a seamless bridge between the linear algebra calculations/algorithms and middlewares, tools, parallel programming techniques, linear algebra libraries, etc.

The Web portal is based on a module for managing the numerical objects, an analyzer and a converter, an advanced trader and the job preparation modules. It provides services for solving large-scale linear systems (including a range of dense and sparse direct solvers and a parallel iterative methods) and services for parallel decision-making.

The essential software components of the Grid Software pool include the portal, the gLite [18] middleware, an advanced trader, compilers, the MPI communication library and basic linear algebra libraries implemented on CPU and GPU. The deployment and the efficient use of the software components over a heterogeneous distributed computing infrastructure are quite challenging. In this context, our main option is to set up the environment only on the base of open-source technologies and software components that simplify the use of new computational resources and infrastructures.

gLite is a well spread middleware for Grid computing that provides a framework for building applications executed over distributed computing and storage resources across the Internet. The LFC (LCG File Catalog) data management of gLite is used, where the Grid Unique IDentifier represents a given file. The data are stored on the storage element of the target Grid site, which has a fast interconnection with the worker nodes. The interconnection between target architectures, numerical objects types and libraries is given in Fig. 1.

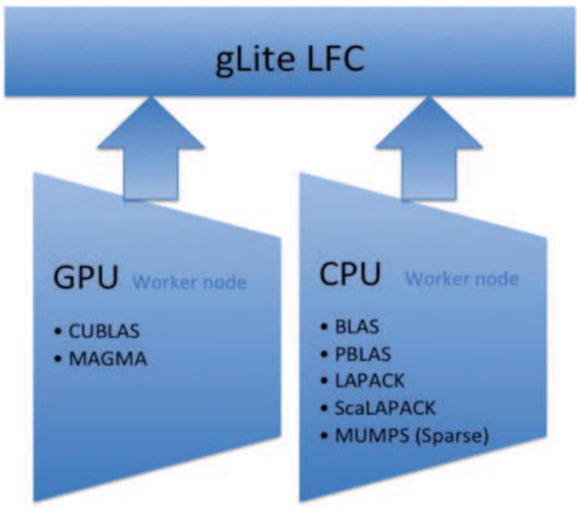


Fig. 1 Interconnection between the target architectures

3.2 Services and High-Level User Interface

One of the main challenges in our work is to provide non-expert users a seamless access to the scientific computing services that are deployed within our Web portal. This is achieved by allowing users to express their requests using mathematical expressions similar to MatLab or SciLab syntax. The identification and the management of the services is performed by the analyzer, the advanced trader and the job preparation modules.

The analyzer module (PHP code) carried out syntactic and semantic analysis of the expression entered by the user by checking the main parameters and analyzing the expressions and the numerical objects involved. If the expression is correct, the converter converts the expression into the corresponding XML format (required as an input for the advanced trader module) that fully defines the given expression including the parameters and their properties.

The advanced service trading module (based on the work described in [16]) computes the services required to perform the user request i.e. converts the mathematical expressions given by the user into a list of procedures calls (functions of libraries) that answer to its request.

Our service trading approach consists in adding additional semantic information to the services in order to allow finding the service or combination of services that provide relevant answers to a user request using equational unification to identify all the possible choices. This approach allows to significantly reduce the ambiguities of the current trading technologies for domain specific use. The user does not need to know the exact name of the service he is looking for, he just has to describe the mathematical operation he wants to compute in a given applicative area (here linear algebra) and to provide the data that the operation will use. The service trader finds an appropriate service or combination of services. It could also be possible to provide the user (through the portal) a list of possible choices and asks him to choose the best one according to his needs (this is not currently implemented). Depending on the problem size (typically if the order of the matrices involved into a matrix-matrix operation is over a certain size), we switch from BLAS to PBLAS in order to use the resources of Grid infrastructures. Practically, we switch to the Grid if the operation exceeds the memory of the server of the portal

Depending on the problem size, we switch from BLAS to PBLAS (if the operation does not fit into the memory of the server of the portal which is 8 GB) using the resources of Grid infrastructures. Practically for a matrix-matrix multiplication on the Armcluster we switch when the matrix size exceeds 8,192 (assuming they are all matrices are square of same size) [19].

The Job preparation module recovers the list of services computed by the advanced trader and selects the target infrastructure where the job will be executed taking into account the type, the complexity of the given expression (list of services) and the generated amount of calculation. Currently, the complexity of the expression is defined only by the size of the matrix but there is obviously space for setting up better strategies. In the case of using the Grid infrastructure as a target resource, a JDL (Job Definition Language) [20] file is automatically created by the Job preparation module to specify the resources that a job requires, then the file is submitted to the Grid infrastructure within the P-Grade Portal.

The main benefit of such an approach is that the user has only to type a mathematical expression and to provide the required data (matrices, vectors, and numbers) in order to get the results (see Fig. 2). All given expressions and results are saved into a centralized database. The real matrices are located in the storage element of the Grid infrastructure, which makes visible the data from the worker nodes of the Grid infrastructure. There is obviously space for further improvements in terms of scheduling considering issues such as machine load, energy saving, distributed storage of data, etc.

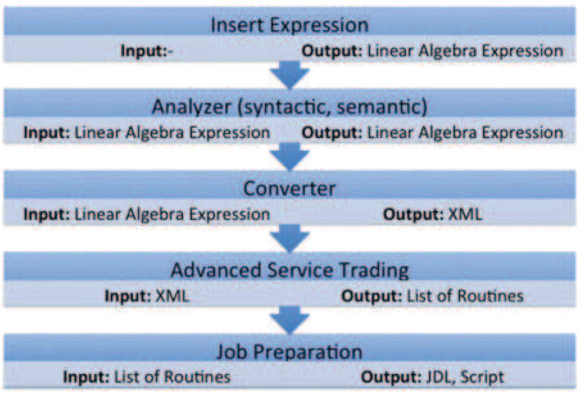
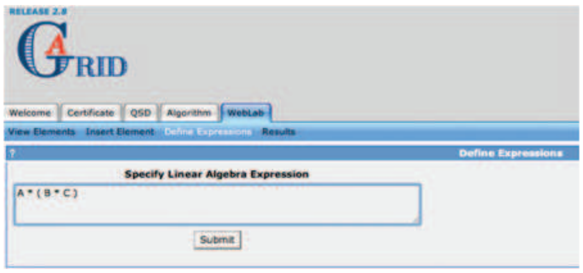

Insert Expression
Input:- Output: Linear Algebra Expression
Analyzer (syntactic, semantic)
Input: Linear Algebra Expression Output: Linear Algebra Expression
Converter
Input: Linear Algebra Expression Output: XML
Advanced Service Trading
Input: XML Output: List of Routines
Job Preparation
Input: List of Routines Output: JDL, Script

Fig. 2 Interconnections of modules

As an illustration of the interface, we consider a simple example. Assume that a user wants to solve the following algebraic expression C = A * (B * C), where A, B and C are general type 100,000-by-100,000 matrices. The user enters the mathematical expression to be evaluated that involves the numerical objects initialized as: A * (B * C) in the Web page for submitting user requests and then submits it for execution (see Fig. 3).

After the syntactic and semantic analyzes by the Analyzer, if the expression is correct, an XML file indicated below is created by the Converter that describes the expression.

The advanced trader module takes this XML file (fully defined expression) and computes the list of functions (or services) that can solve the requested expression. Since the matrices are quite large, the execution will be performed remotely using PBLAS (in case of smaller size matrixes it can be CPU or GPU realization of BLAS instead of PBLAS). The output of the service trader consists into two consecutive calls to PsGEMM (the



Specify Linear Algebra Expression

A * (B * C)

Fig. 3 Algebraic equation request entered by the user

routine that performs generalized matrix-matrix multiplication in the PBLAS):

```
<requete>
<domainName>LinearAlgebra</domainName>
 <term>
  <expression>
  <operatorName>*</operatorName>
  <terms>
   <term>
    <variable>
     <name>A</name>
     <sorte>Matrix</sorte>
    </variable>
   </term>
 <term>
  <expression>
   <operatorName>*</operatorName>
   <terms>
    <term>
     <variable>
      <name>B</name>
      <sorte>Matrix</sorte>
     </variable>
    </term>
    <term>
     <variable>
      <name>C</name>
      <sorte>Matrix</sorte>
     </variable>
    </term>
   </terms>
  </expression>
 </term>
 </terms>
 </expression>
 </term>
</requete>
```

PsGEMM ('N', 'N', 100000, 100000, 100000, 1., B, 1, 1, DESCB, C, 1, 1, DESCC, 0., p2, 1, 1, DESCp2) \\ $p2 = B * C$
PsGEMM ('N', 'N', 100000, 100000, 100000, 1 A, 1 1 DESCA, p2, 1, 1, DESCp2, 0., p1, 1, 1, DESCp1) \\ $p1 = A * p2$
p1

Considering the size of the matrices involved in the operation, the execution of these PBLAS procedures will be performed over the Grid, the Job preparation module will then generated an execution script launched over the Grid in order to recover the result of the user request that are sent back to the Web interface.

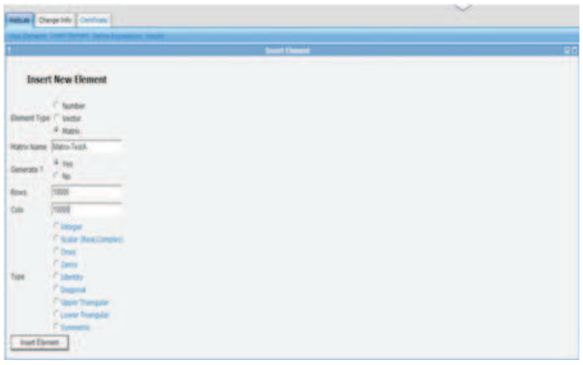
3.3 Data Management in the Portal

The users define and insert the new numerical objects (see Fig. 4) that are required in their mathematical calculations. Both uploading and generation mechanisms are available for the creation of numerical objects by specifying their types and properties, such as triangular, identity, symmetric, … for matrices. The interface provides to users different options (download, delete or change parameters) for the uploaded numerical objects. The parameters (types, properties, names, etc.) of the numerical objects are stored on the local mysql database of the portal and then the numerical objects are transferred to the storage element of the Grid infrastructure. Currently the portal uses the storage element of the ArmGrid infrastructure with the capacity of 8 TB.

4 Mathematical Services

The computational services currently deployed within our scientific computing environment include libraries for dense and sparse linear algebra (including direct and iterative solvers) and a solver for decision making problems. These libraries cover the major types of architectures (serial and parallel computers using the OpenMP and MPI programming paradigms and GPUs) and allow to solve a wide range of problems.

The services deployed within our scientific environment are procedures of the various scientific libraries. The C or Fortran codes of these proce-



Insert New Element

Element Type: Number / Vector / [✓ Matrix]
Matrix Name: Matrix-TestA
Generate ?: [✓ Yes] / No
Rows: 10000
Cols: 10000
Type: Integer / Scalar (Real,Complex) / Ones / Zeros / Identity / Diagonal / Upper Triangular / Lower Triangular / Symmetric

Fig. 4 Scientific computing environment based on P-GRADE: data initialization

dures are encapsulated within a shell script that allows to call them (input is the output of the trader module, output is the JDL file) from gLite when they are used remotely (which is not required when they are executed locally). Practically all ScaLAPACK procedures are encapsulated with that script and their functionalites, data input and output are described to the advanced trader.

4.1 Service for Large-Scale Linear Systems

Several libraries for dense linear algebra (ScaLAPACK, MAGMA, PSBLAS) in addition to the MUMPS package that provides a range of sparse direct solvers are available:

- ScaLAPACK [21] is a library of high-performance linear algebra routines for distributed memory message-passing MIMD (multiple instruction, multiple data) computers and networks of workstations supporting Parallel Virtual Machine and/or Message Passing Interface. With ScaLAPACK comes an additional library called the PBLAS, which can be seen as a parallel version of the BLAS (Basic Linear Algebra Subprograms) [22]. Using the PBLAS, simple matrix/vector and more complex operations can be performed in parallel.
- MAGMA (Matrix Algebra on GPU and Multicore Architectures) [23] is a collection of next generation linear algebra libraries for heterogeneous architectures. MAGMA allows applications to fully exploit the power of current heterogeneous systems of multi/many-core CPUs and multi-GPUs to deliver the fastest possible time to accurate solution within given energy constraints.
- The PSBLAS (Parallel Sparse Basic Linear Algebra Subprograms) [24] provides a framework to enable easy, efficient and portable implementations of iterative solvers for linear systems on parallel computers. It implements parallel versions of most of the Sparse BLAS computational kernels (sparse matrix by dense matrix multiplication, sparse block diagonal system solution, sparse matrix norms, vector norms, etc.), sparse matrix management functionalities and parallel implementations of preconditioned Krylov solvers.
- The MUMPS (MUltifrontal Massively Parallel sparse direct Solver) library [25] is a package for solving systems of linear equations of the form Ax = b, where A is a sparse matrix that can be either unsymmetric, symmetric positive definite, or general symmetric.

The matrix-matrix multiplication is one of the most important computational service for large-scale linear systems. Parallel matrix-matrix multiplication has been investigated and benchmarked using the computational resources of the ArmGrid infrastructure. Particularly, a series of experiments have been done on the high-performance computing Armcluster system, which consists of 128 3.06 GHz Intel Xeon processors and uses Myrinet 2000 network topology.

The PDGEMM double precision routine from the PBLAS in ScaLAPACK is used for benchmarking, as it is the one of most widely used computational kernel. Figure 5 reports the performance and scalability of PDGEMM for square matrices: the matrices ranks range from 600 to 12,000. The figure only includes the execution of the service, because the total execution time and the latency of the portal depends of the scheduling of jobs in the Grid environment which introduces strong variations depending on the overall load of the server (other jobs are running) and explain the superlinear speedup with the matrices of size 8,000 and the absence of gain when moving from 64 to 96 processors. The limits of the portal depends

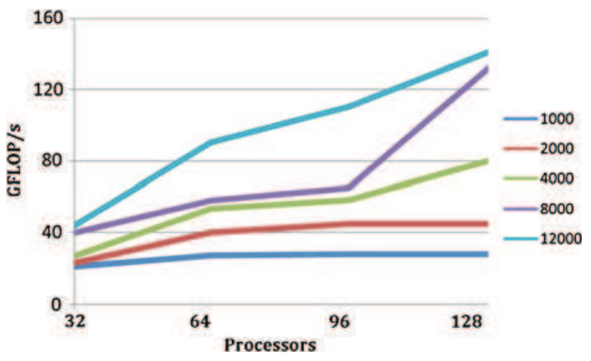


Fig. 5 ScaLAPACK PDGEMM benchmarking on Armcluster (performance in GFlops vs number of processors)

on the capacities of the storage and on the computational resources of the virtual organization supported the portal. In our case 128 cores and 8 TB of storage (the resources of Armcluster) are available for the portal. Currently the storage element of the ArmNGI is used for the experiments, which is interconnected with the Grid sites via 1 Gbps of dedicated bandwith.

4.2 Services for Parallel Iterative Methods to Solve Large Sparse Linear Systems

Iterative methods are, with direct methods, the other family of methods for solving general, large sparse linear systems $Ax = b$ [26]. The principle is generally to construct a sequence of iterates converging towards the solution of the linear system. Among the large variety of iterative methods, we focus on Krylov methods [26, 27] and Cimmino methods [28, 29]. In the future it is planned to benchmark these iterative methods, as well as to deploy other iterative methods.

4.2.1 Services for Krylov Methods

The Krylov methods are based on a projection processes onto Krylov subspaces, which are subspaces spanned by the vectors $\{b, Ab, A^2b, ..., A^ib\}$. In short, these techniques approximate $A^{-1}b$ by $p(A)b$, where p is a polynomial. The solution is constructed by many approaches that give rise to different methods [28]:

- The Ritz-Galerkin approach: full orthogonalization method (FOM), conjugate gradient (CG) methods,
- The minimum norm residual approach: GMRES method and its variants,
- The Petrov-Galerkin approach: Bi-CG and QMR methods,
- The minimum norm error approach: SYMMLQ method, ...

We focus on the following two methods: GMRES method for unsymmetric matrices and Conjugate Gradient for symmetric definite matrices. The services implemented in the platform are based on these two methods. These methods can be parallelized using some of the libraries we deployed over the platform such as PSBLAS.

4.2.2 Service for Block-Cimmino Method

The Cimmino method [28, 29] is a row projection method in which the original linear system is divided into subsystems. At each iteration, it computes one projection per subsystem and uses these projections to construct an approximation to the solution of the linear system. The Block-Cimmino method, as all the block algorithms, can be efficiently parallelized, for example, by distributing the different blocks on the available processors. However, the IRIT Team develops another approach [30] where the block distribution onto the processors is not explicitly performed within the code: the MUMPS multi-frontal sparse solver is in charge of handling the data distribution and parallelism.

4.3 Service for Parallel Decision Making

Searching algorithms for solving the non-linear multi-parametric experimental problems (optimization problems) contains series of control parameters, which defines the effectiveness of the extremum computational meanings that usually define the success of the search of extremum. Besides, the extremal problems usually contain gully situations, which make the extremum search process significantly complicated. In practice the passage of gully situations is implemented as a rule through the search from various start points. The service deployed (see Fig. 6) allows for not only the parallelization of the solution of extreme problem itself, but also to simultaneously solve the problem by using some set of values of control parameters and then select the best solution.

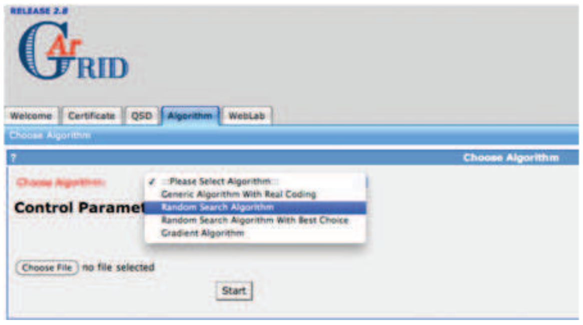


Fig. 6 Service for parallel decision making

As it will be shown, the development of mentioned approach solves the problem of adaptation in some manner and allows for a significantly increment of the speed and quality of the solution of optimization problems.

Let's consider a non-linear multi-parametric extreme problem [31, 32].

$$F(\overline{x}) = F(x_1, x_2, \ldots, x_n) \rightarrow \min_{x \in D} \Rightarrow \overline{x}^*$$

With the non linear limitations of inequality type $D : \varphi(\overline{x}) = \varphi_i(x_1, x_2 \ldots, x_n) \geq 0, \quad i = 1, 2, \ldots, l,$ where $F(\overline{x})$ is the goal function, $\overline{x}$is the vector of variable variables, z is the dimension of the range of the extreme search, $\overline{x}^*$is the vector of optimal values of variables, D is the allowable range formed by technical requirements in the form of l limitations. The transfer to the acceptable range D will be implemented through minimization of the non-negative function.

$$Q(\overline{x}) = \sum_{\varphi_i < 0} \frac{|\varphi_i|}{|\varphi_i| + 1}$$

The developed parallel algorithms have been benchmarked on the nonlinear problem of optimal projection of synchronic generator with obvious poles [33] using the service of the portal. For example, in case of random-search algorithm the total number of experiments are 27 (3^3), because three values of the following control variables are used: $\overline{x}_0$—vector of the start point coordinates, k$_u$—coefficient of the step division, μ—the count of maximum tries inside of hypersphere. In our benchmarks we have achieved a 8.64 speedup when simulations are run using 27 cores of the Grid site located in the State Engineering University of Armenia. The investigations of the developed system show that using the Web portal is essential, because big differences in the goal function optimal value occur depending on the best and worst set of control parameters.

The random search with the best selection, the classic genetic and gradient algorithms have also been implemented and tested. The experiments show that the discussed approach is also effective for the mentioned algorithms.

5 Conclusion

The full implementation of our environment for scientific computing over the Grid and GPUs will allow users, who are not familiar with the parallel programming technologies and software tools (Grid and cluster middlewares, parallel and GPU programming, queue system, linear algebra libraries), to use the services over the portal while hidding all the details of the underlying distributed computing infrastructure. The environment is a good tool for demonstrating how computations can be transparently performed over the Grid or GPU from the user point of view.

Acknowledgements This joint work has been done within the framework of the "Development of a Second Generation Scientific Computing Infrastructure Based on P-Grade Portal" (Centre National de la Recherche Scientifique, France and State Committee of Science of Armenia) Project.

References

1. Petitet, A., Casanova, H., Dongarra, J., Robert, Y., Whaley, R.: Parallel and distributed scientific computing: a numerical linear algebra problem solving environment designer's perspective. Handbook on Parallel and Distributed Processing (1999)
2. Parallel Grid Run-time and Application Development Environment. http://www.p-grade.hu (2003)
3. Astsatryan, H., Sahakyan, V., Shoukourian, Yu., Srapyan, M., Daydé M., Hurault, A., Grigoras, R.: Introduction of a Grid-aware portlet for numerical calculations. In: IEEE 1st International Conference on Parallel, Distributed and Grid Computing (PDGC'2010). Waknaghat, Solan (HP), India (2010)
4. Astsatryan, H., Sahakyan, V., Shoukouryan, Yu., Daydé, M., Hurault, A., Pantel, P., Caron, E.: A Grid-aware web interface with advanced service trading for linear algebra calculations. In: High Performance Computing for Computational Science—VECPAR2008, Lecture Notes in Computer Science 4395. Springer 2008, Toulouse, France (2008)
5. Astsatryan, H., Shoukouryan, Yu., Sahakyan, V.: Grid activities in Armenia. In: Proceedings of the International Conference Parallel Computing Technologies (PAVT'2009). Novgorod, Russia, March 30—April 3 (2009)
6. Armenian National Grid Infrastructure. http://www.Grid.am (2009)
7. Hardt, M., Seymour, K., Dongarra, J., Zapf, M., Ruiter, N.V.: Interactive Grid-access using Gridsolve and giggle. J. Comput. Inf. **27**(2), 233–248

8. Gabrielle, A., et al.: Cactus tools for Grid applications. Springer J. Clust. Comput. **4**(3), 179–188 (2001)
9. Ardizzone, V., Barbera, R., Calanducci, A., Fargetta, M., Ingrà, E., Porro, I., La Rocca, G., Monforte, S., Ricceri, R., Rotondo, R., Scardaci, D., Schenone, A.: The DECIDE science gateway. J. Grid Comput. **10**(4), 689–707 (2012)
10. Czyzyk, J., Mesnier, M., Moré, J.: The NEOS server. IEEE J. Comput. Sci. Engineer. **5**, 68–75 (1998)
11. Daydé, M., Desprez, F., Hurault, A., Pantel, M.: On deploying scientific software within the GRID-TLSE project. Computing Letters, VSP/Brill Academic Publisher, **1**(3), 85–92 (2005). doi:10.1163/1574040054861267
12. Farkas, Z., Kacsuk, P.: P-GRADE portal: a generic workflow system to support user communities. Future Gener. Comput. Syst. J. **27**(5), 454–465 (2011)
13. Kacsuk, P.: P-GRADE portal family for Grid infrastructures. Conc. Comput. Pract. Exper. J. **23**(3), 235–245 (2011)
14. Kacsuk, P., Farkas, Z., Kozlovszky, M., Hermann, G., Balasko, A., Karoczkai, K., Istvan Marton, I.: WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities. J. Grid Comput. **10**(4), 601–630 (2012)
15. Caron, E., Desprez, F.: DIET: a scalable toolbox to build network enabled servers on the Grid. Int. J. High Perform. Comput. Appl. **20**(3), 335–352 (2006)
16. Hurault, A., Daydé, M., Pantel, M.: Advanced service trading for scientific computing over the Grid. J. Supercomput. Springer Netherlands, **49**(1), 64–83 (2008)
17. Hurault, A., YarKhan, A.: Intelligent service trading and brokering for distributed network services in GridSolve, VECPAR 2010. In: 9th International Meeting on High Performance Computing for Computational Science, Berkeley, CA (2010)
18. gLite. http://www.glite.org (2005)
19. Astsatryan, H., Sahakyan, V., Shoukourian, Y., Dayde, M., Hurault, A.: Enabling large-scale linear systems of equations on Hybrid HPC infrastructures. ICT Innovations 2011. Springer Advances in Intelligent and Soft Computing **150**, 239–245 (2012). doi:10.1007/978-3-642-28664-3_22
20. Job Description Language HowTo—DataGrid-01-TEN-0102-02. http://www.infn.it/workload-Grid/docs/DataGrid-01-TEN-0102-0_2.pdf (2002)
21. Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK: a linear algebra library for message-passing computers. SIAM Conference on Parallel Processing (1997)
22. Blackford, S., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., Heroux, M., Kaufman, L., Lumsdaine, A., Petitet, A., Pozo, R., Remington, K.: An updated set of basic linear algebra subprograms. ACM Trans. Math. Softw. **28**(2), 135–151 (2002)
23. Tomov, S., Dongarra, J., Baboulin, M.: Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parallel Comput. **36**(5–6), 232–240 (2010)
24. Filippone, S., Colajanni, M.: PSBLAS: a library for parallel linear algebra computation on sparse matrices. ACM Trans. Math. Softw. **26**(4), 527–550 (2000)
25. MUMPS: A MUltifrontal Massively Parallel Sparse Direct Solver. http://mumps.enseeiht.fr (2000)
26. Saad, Y.: Iterative methods for sparse linear systems, 2nd edn. SIAM (2003)
27. van der Vorst, H.A.: Iterative Krylov Methods for Large Linear Systems, Cambridge Monographs on Applied and Computation Mathematics. Cambridge University Press, Cambridge (2003)
28. Arioli, M., Duff, I.S., Noailles, J., Ruiz, D.: A block projection method for sparse matrices. SIAM J. Sci. Statist. Comput. **13**, 47–70 (1992)
29. Arioli, M., Duff, I.S., Ruiz, D., Sadkane, M.: Block Lanczos techniques for accelerating the Block Cimmino method. SIAM J. Sci. Statist. Comput. **16**, 1478–1511 (1995)
30. Balsa, C., Guivarch, R., Ruiz, D., Zenadi, M.: An hybrid approach for the parallelization of a block iterative algorithm. In: International Conference on Vector and Parallel Processing (VECPAR 2010), 22/06/2010–25/06/2010, pp. 116–128. Springer, Berkeley (2010)
31. Terzyan, H.: Decision Making Algorithms in Electromechanics, pp. 18–27. Izvestia Univer-sities, Electromechanics (2009)
32. Terzyan, H.: Simulation of electromechanical systems. In: Numerical Methods and Solutions, p. 280. VDM Verlag (2009)
33. Terzyan, H., Hovhannisyan, L.: On solution of decision-making problem in distributed computing environment. In: Proceedings of the Conference Computer Science and Information Technologies, pp. 276–280. Yerevan (2011)