2008－MPS－69 (6)
2008／5／16

A Grid based Unified Framework for Optimization

ASIM MUNAWAR,[†1] MOHAMED WAHIB,[†1] MASAHARU MUNETOMO[†2]
and KIYOSHI AKAMA[†2]

This paper presents a Service Oriented Architecture (SOA) compliant Problem Solving Environment (PSE) that allows the user to implement any metaheuristics based algorithm over a Grid. We call this framework a Grid based Unified Framework for Optimization (GridUFO). GridUFO provides a unified approach for sharing and using metaheuristics algorithms (solvers) and objective functions over a Grid in an easiest possible "plug & play" manner. In this way the user can take all the advantages of the Grid without taking into consideration any of the complexities posed by the Grid environment. The framework is accessible to the user through a Web service or through a fully integrated 2nd generation Web portal. GridUFO provides well-defined interfaces between the user-programmed objective functions and solvers. We also present MetaHeuristics Markup Language (MHML), an XML based language that acts as an interface between the user and the framework. In this paper we will discuss the design and implementation of GridUFO in detail. Moreover, we will talk about our experience of working with Grids, and we will also make some recommendations for future research.

1. Introduction

In this paper we present GridUFO, a SOA compliant PSE that allows the user to run and share metaheuristics based algorithms (solvers) over a Grid. GridUFO is a step towards Virtual Innovative Laboratory (VIL), proposed by Munetomo[1] (2006). VIL seeks for realizing virtual laboratory that innovates automatically to find optimal solutions or designs. VIL intends to replace a part of human designer's trial-and-error process.

Many Grid based algorithms and frameworks were presented in the past. However, these projects are not in agreement with the true spirit of Grids as most of them do not allow the user to use all the functionality offered by Grids, and all of these projects completely ignore the service oriented aspect of the Grids. GridUFO addresses all the shortcomings of the work done in the past and provides a solution to these problems in consent with the true spirit of Grids. The framework provides a Web Services Resource Framework (WSRF) based Web service that offers the available algorithms as services to the users. This Web Service can be consumed by any client application independent of the operating system and architecture of the system. The framework is also accessible through a Web portal that allows the user to access all the services through any Web browser.

We also propose MetaHeuristics Markup Language (MHML), an XML based language for interfacing different modules of GridUFO. The main purpose of MHML is to provide a well defined interface between GridUFO and the users. Moreover, we define the interfaces between the solver and the objective function, both of which run independently in a distributed fashion over the Grid. This interface is defined with the help of GridUFO API, NinfIDL[2] and MHML.

Several projects have targeted this kind of framework in the recent years. We will only mention some of the most notable work done in past. Although most of the projects do not have one-to-one relation with GridUFO, we make a comparison with GridUFO in table 1. It is important to note that most of the Grid based optimization projects in the past talks about simulations or a specific algorithm for the Grid, and do not provide their work as a service to the users.

2. GridUFO

An optimization job consists of two basic modules, namely, an optimization algorithm (solver) and an objective function for the problem in hand. In a standard implementation of algorithms, these two modules are tightly linked with each other and are inseparable. However, GridUFO is unique in the way it treats these two modules as shown in Fig. 1. This approach of GridUFO is very different from the workflow tools approach. Typical workflow tools just define the dependencies and sequence of execution for two or more applications, while GridUFO's distributed execution allows the solver to use the objective function again and again directly from the code.

†1 Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan
†2 Information Initiative Center, Hokkaido University, Sapporo, Japan

Table 1 Comparison of GridUFO with the work done in the past.

	Scope	Black-box Optimization	Middleware	Information Exchange	Adding New Solver by User	Security	User Interface	User Guidance to Solvers
NEOS	Any Optimization	Not Supported	Non-Standard	Plain Sockets	Allowed (Through Management)	No	Web-based Job Submission	Text Describing Each Solver
Folding@Home	GROMACS or Protein Structure Optimization	Not Supported	SMP Client	Client-Server Sockets	Not Allowed	Yes (2048 bit Digital Signature)	–NA–	–NA–
Nimrod/O	Non-linear Optimization	Not Supported	OGSA Compliant	Active Sheets	Not Allowed	Yes (GSI)	Web Portal	Ontology
GEODISE	Fluid Dynamics	Not Supported	OGSA Compliant	Environment API	Not Allowed	Yes (GSI)	Matlab Tool box/ Portal	Ontology
OSP	Decision Support System	Not Supported	Aggregated Components	AMPL/MPL	Not Allowed	Yes	Web Portal	Designated Tools
GE-HPGA	Global Optimization with an Island Model GA	Supported	OGSA Compliant	Environment API	Not Allowed	Yes (GSI)	None	None
GridUFO	Global Optimization with different metaheurisits	Supported	OGSA Compliant	MHML (XML Based)	Allowed (Trough Portal)	Yes (GSI)	Web Portal & Web Service	SLD + SLA

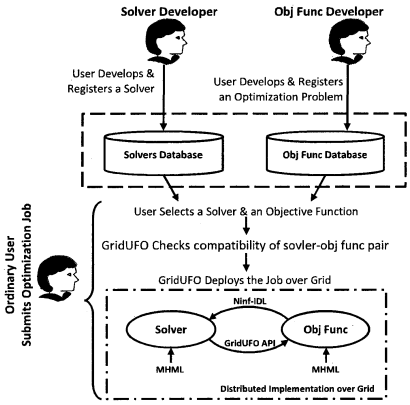

Fig. 1 GridUFO: The core concept.

2.1 Architecture of GridUFO

Figure 2 shows the architecture of GridUFO (only the most notable components and some important tools used by the framework). A very brief description of the main components of the framework is as follows:

(1) *GridUFO's Web Service:* is a WSRF compliant web service running inside a Globus Toolkit's[3] Web service container. This Web service can be consumed by any client application independent of the architecture and the platform that application is using. See Sect. 2.2 for further details.

(2) *GridUFO's Web Portal:* is a fully integrated 2nd generation Gridsphere[4] portal with JSR 168 compliant GridUFO portlets installed.

(3) *GridUFO's Directory Index:* is a PostgreSQL based database containing all the solvers and the objective functions registered with the framework. It also maintains logs/history of the jobs submitted to the framework.

(4) *GridUFO's Job Manager:* is responsible for maintaining a job waiting queue, and running the jobs over the available computational resources.

(5) *GridUFO's Resources:* refers to the computational resources only. GridUFO's resources can include any Globus based resource, or any other resource with a well known batch system installed.

Another important component is GridUFO's scheduler. It is a simple job scheduler which can be replaced by any other well known scheduler or super scheduler without any major modification in the system.

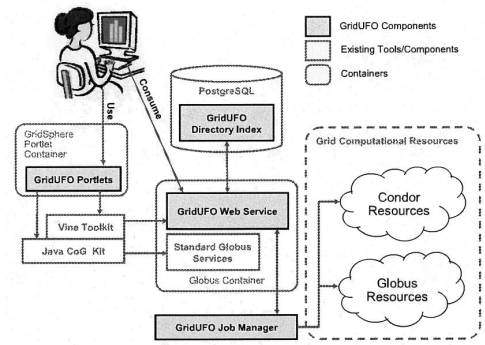

Fig. 2 Abstract level architecture of GridUFO (shows the most notable components of the system).

2.2 SOA Compliance

We have used GridUFO's WSRF compliant Web service to implement SOA, hence enabling the users to employ platform independent standard Internet protocols to access the services offered by GridUFO. GridUFO's Web service provides the core services that orchestrate all the components in the system. The four basic services offered by the framework are:

(1) *Register Service:* allows the user to register

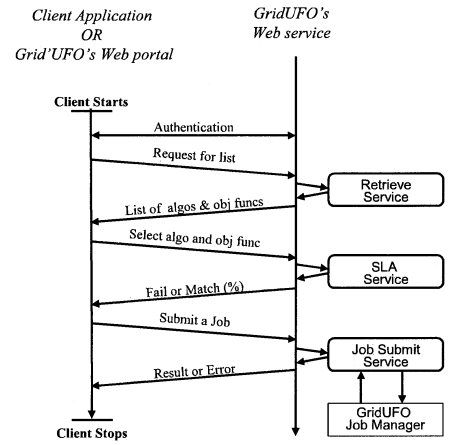

Fig. 3 Job submission scenario from user's perspective.

an algorithm or an objective function with the framework. The user is required to submit the SLD file.

(2) *Retrieve Service:* when queried, returns a list of all the algorithms and objective functions registered with the framework along with their SLDs.

(3) *SLA Service:* is a service of GridUFO that is used to check the compatibility of an algorithm with an objective function.

(4) *Job Submit Service:* allows the user to submit an optimization job to the framework through a job submission file. Job submission file contains the name of the algorithm, name of the objective function, and other information relevant to the job.

Note that both SLD and the job submission file are MHML files (see Sect. 3 for details on MHML).

2.3 Distributed Deployment of Jobs

GridUFO employs GridMPI, GridRPC and well known batch job schedulers for distributing the job over the available GridUFO computational resources. All these tools depend on the low-level Grid middleware (Globus in the case of GridUFO) for their working in one way or the other. Whenever GridUFO's Job submit service receives a new optimization job, it reads the job submission MHML file, the SLD files of the algorithm and the objective function. It then checks the compatibility of the algorithm and the objective function by using SLA service, creates a valid GridUFO job, and forwards it to GridUFO job manager. The job manager can then deploy the GridUFO job on the real physical resources.

An existing algorithm can be converted into a GridUFO compatible algorithm by using a simple API (GridUFO API) provided to the algorithm developer. Similarly objective function is a C code which is treated as a black box by the system. User is required to write a code for the objective function along with the Ninf-IDL[2] interface file.

2.4 User's Perspective

From the user's perspective GridUFO is a PSE meant for solving optimization problems over a Grid. There are four distinct types of GridUFO users: (1) Administrators, (2) Algorithm Developers, (3) Objective Function Developers, and (4) Ordinary users.

Figure 3 depicts the sequence of operations in the framework for submission of a new job submitted by an ordinary user.

We have designed JSR 168 compliant custom portlets for each of the service provided by the portal. The portal allows the user to access GridUFO through any Web browser without installing any extra client application.

3. MHML

All the communication between the user and the framework is done by using a proposed language that we call MetaHeuristics Markup Language (MHML). MHML can be considered as a modification/extension to E. Alba et al. (2003)[5]. MHML has the capability to represent: (1) Job Configuration, (2) Solver or Obj Func SLD, (3) Solver or Obj Func Config, (4) Client Information, and (5) Results/Errors. MHML is presented to the user as an XML schema. A top-level hierarchy of MHML is shown in Fig. 4. For a complete description see Munawar et al. (2007)[6].

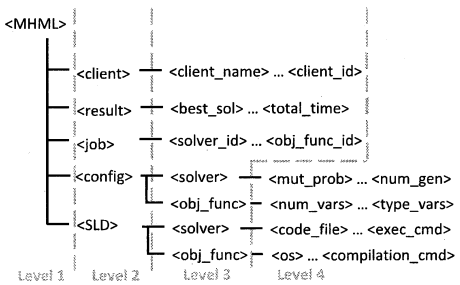

Fig. 4 Top level hierarchy of MHML with important tags.

4. Results & Discussion

4.1 Theoretical Analysis

An approximate maximum speed-up for a metaheurisitcs algorithm over GridUFO can be given as:

Table 2 Dummy problem over GridUFO using a Simple Real GA. Fitness function is called 990 times by the algorithm. Time is shown in seconds. T_f is the time taken by a single fitness value calculation. T_s/T_p is the Speedup achieved by implementation over GridUFO. O_{inter} is the approximate value for average overheads for one cluster. m is total number of clusters and its value is 15.

T_f (Sec)	T_s (Sec)	T_p (Sec)	T_s / T_p	u^{avg} (Sec)	O_{inter} (Sec)
0.001	3	241	0.012	0.0667	16.62
0.01	14	228	0.0614	0.66	15.1
0.1	102	240	0.425	6.6	15.56
0.5	499	233	2.141	33	13.337
1	993	295	3.366	66	15.26
2	1984	366	5.42	132	15.6
5	4955	771	6.42	330	29.4
10	9903	1218	8.13	660	37.205

$$S^{max} = \frac{T_s^{max}}{T_p^{min}} = \frac{mu^{max}}{mO_{inter}^{min} + u^{min}} \quad (1)$$

where, T_s^{max} is the maximum total time to execute an algorithm over a single cluster, T_p^{min} is the minimum total time to execute same algorithm over m clusters, O_{inter}^{min} is the minimum inter cluster communication overhead, u^{max} is the maximum total time taken by slowest cluster to execute the $(1/m)^{th}$ part of the algorithm, and u^{min} is the minimum total time taken by the fastest cluster to execute $(1/m)^{th}$ part of the algorithm. See Munawar et al.[7] (2008) for further details.

4.2 Empirical Results

We performed some experiments to give an idea of the overheads discussed in Sect. 4.1. We used a dummy optimization problem so that we can change all the parameters according to our needs. We map the results on Eq. 1 to find the overheads.

Table 2 shows the empirical results obtained by running the dummy problem over GridUFO using a "Simple Real GA" as the solver. We have used 15 clusters for the implementation and each cluster consists of only one IBM x3455 AMD® Dual-Core Opteron model servers (with 2GB of memory) Globus based resource. Each cluster has only one computational node, as we only want to observe the inter cluster overheads (O_{inter}). Note that the overheads are quite significant for small problems, however, we can obtain considerable speedups for the problems of larger sizes. Therefore, in terms of speedup GridUFO and other such environments are more suitable for problems of relatively larger size.

5. Conclusions & Future Work

In this paper, we demonstrated a Grid based optimization framework that can employ metaheuristics based algorithms in an efficient manner to solve different complex optimization problems. What makes our work unique is that the presented framework is the first framework of its kind i.e. it is the first SOA based optimization framework attempted for an Open Grid Services Architecture (OGSA) compliant grid middleware. In our experience of working with a Grid environment we have observed that a practical and stable Grid in its true sense is far from reality and a lot of research and advanced tools are required especially for deployment and debugging phases of software development.

In future we would like to work further on the Quality of Service (QoS) part and we would also like to add an economically feasible utility computing model to the system.

References

1) Munetomo, M.: Realizing Virtual Innovative Laboratory with Robust Evolutionary Algorithms over the GRID computing system, *Proceedings of the 6th International Conference on Recent Advance in Soft Computing*, pp.42–47 (2006).
2) Tanaka, Y., Nakada, H., Sekiguchi, S., Suzumura, T. and Matsuoka, S.: Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing, *Journal of Grid Computing*, Vol.1, No.1, pp.41–51 (2003).
3) Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems, *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp.2–13 (2006).
4) Novotny, J., Russell, M. and Wehrens, O.: GridSphere: a portal framework for building collaborations, Vol.16, No.5, pp.503–513 (2004).
5) Alba, E., Garc-Nieto, J. and Nebro, A.: On the Configuration of Optimization Algorithms by Using XML Files (2003).
6) Munawar, A., Wahib, M., Munetomo, M. and Akama, K.: Standardization of Interfaces for Meta-Heuristics based Problem Solving Framework over Grid Environment, *Proccedings of HPCAsia 2007*, Seoul, South Korea (2007).
7) Munawar, A., Wahib, M., Munetomo, M. and Akama, K.: *Linkage in Evolutionary Computation (to appear)*, chapter Parallel GEAs with Linkage Analysis over Grid, Springer (2008).