Optimization Problem Solving System using Grid RPC

Tomoyuki HIROYASU **Mitsunori MIKI** **Hisashi SHIMOSAKA**

Doshisha University, Kyoto, Japan, tomo@is.doshisha.ac.jp
Jack DONGARRA
University of Tennessee, Innovative Computing Lab.

1 Introduction

Optimization problems are to find design variables that minimize or maximize the value of objective function under constraints. Many problems that need decision making can be turned into optimization problems. For example, when structures are designed, this designing problem becomes an optimization problem by choosing a total volume of the structure as an objective and sizes of the structure elements as design variables. One of the models to solve optimization problems by computer simulation is shown in Figure 1. In this model, there are two modules: an optimizer and analyzer. An optimizer determines the search point and an analyzer derives the value of objective of its search point. Then, the optimizer determines the next search point again. After many iterations, the optimum result can be derived.

In this research, the problems whose calculation cost is huge are targeted; those are structural optimization problems, fluid dynamic problems, layout problems, protein folding problems, and so on. While this feature exists and this system needs many iterations, it takes a long time to perform this system. One of the solutions is to perform this system in parallel.

Recently, the computational Grid has been the focus. In the computational Grid, there are many calculation resources in the network and users can use them without consideration of the physical arrangement and special certification. There are many grid computing systems. Among them, there are Grid remote procedure calls (RPCs) [1]. In the GGF workshop, the work on standardizing and implementing the RPC mechanism for grid computing is carried out. This is the Grid RPC. NetSolve[2] and Ninf [3] are grid computing systems that are based on the Grid RPC API.

The goal of this research is to present the framework of the optimization system for general purpose in the computational Grid using Grid RPC. In this abstract, the concept and overview of the system are explained. At the same time, the APIs for the modules that build the system are explained.

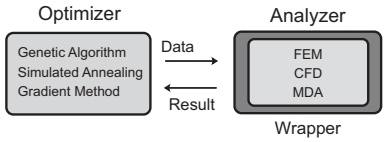

Figure 1: Optimization System

2 Overview of Proposed System

In Figure 2, the overview of the proposed system is illustrated.

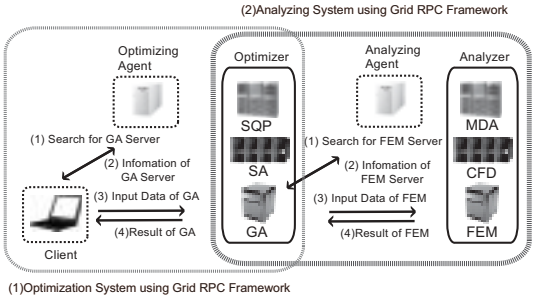

Figure 2: Overview of Proposed System using Grid RPC

2.1 System Construction Phase

There are five components in this system: client, optimizer server, analyzer server, optimizing agent, and analyzing agent. System developer has to be prepared optimizers and analyzers. These modules have to be followed the APIs that are explained later. After system developer has prepared optimizers and analyzers, the information of the optimizer server is registered to optimizing agent. The same thing is required for analyzer server and analyzing agent.

2.2 User Performing Phase

The basic procedure of using this system can be summarized as follows,

1. A user chooses the methods of optimization and analyzation.

2. The client asks optimizing agent who has the service of the user chosen optimization method. The optimizing agent has the information and he replies the server to the client. The client also asks optimizing agent who has the service of the analyzing. This question is transferred to analyzing agent who

has the information of the services of the analyzers. The analyzing agent returns the server information of the agent to the optimizing agent and the optimizing agent returns the answer to the client. Because of this mechanism, users need not have the information of the optimizer and analyzer servers.

3. The selected optimizing server starts optimization.

4. When the optimizer needs to have the values of objective function and constraints, he asks to the analyzer servers.

5. The analyzer server returns the values of the objective and constraints to the optimizer server.

6. The routines from 3 to 5 are performed until the terminal condition is satisfied.

7. When the optimization has finished, the result is returned to the client.

3 APIs for Optimization Problem Solving System using Grid RPC

In this abstract, APIs for analyzer and optimizer are prepared. System developer follows these APIs and users can operate the system properly.

3.1 API for Optimizer

The API for optimizer is defined as Figure 3.

```
int optimize(Name_of_Optimizer,
    Tag,Name_of_Analyzer,File1,File2,File3);
```

Figure 3: Optimizer API

Name_of_Optimizer indicates the method of optimizing such as gradient method, GA, and so on.

Tag is chosen from the following four options.

- Analyze_Init: When this tag is set, the configuration file for analyzer is sent.
- Analyze_Config: When this tag is set the input output relation file of the analyzer is sent.
- Run: When this tag is set, the optimization has started. The results are written in the output file.
- Finalize: When this tag is set, the optimizer performs the terminal procedures.

Name_of_Analyzer indicates the method of analyzer such as FEM, CFD, and so on.

```
int analyze(Name_of_Analyzer,
                Tag,File1,File2,File3);
```

Figure 4: Analyzer API

3.2 API for Analyzer

The API for analyzer is defined as Figure 4.

Name_of_Analyzer indicates the method of analyzer such as FEM, CFD, and so on.

Tag is chosen from the following six options.

- Initialize: When this tag is set, the configuration file for analyzer is sent.
- Config: When this tag is set the input output relation file of the analyzer is sent.
- Solve: Analyzer is performed with the specified input file. Results are written in the output file.
- Result: When this tag is set, all the results are returned.
- Finalize: When this tag is set, the analyzer performs the terminal procedures from optimizer.
- All_Finalize: When this tag is set, the analyzer performs the terminal procedures for Grid RPC.

4 System Demonstration

Because of the restriction of the pages, the explanations how to use these APIs and how to perform the system are skipped. These explanations will be performed at the poster session.

The proposed system is implemented using NetSolve. Some demonstration will be also performed at the session.

Bibliography

[1] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, and H. Casanova. Grid RPC: A Remote Procedure Call API for Grid Computing algorithms. In *http://www.eece.unm.edu/ apm/docs /APM_GridRPC_0702.pdf*.

[2] D. Arnold, S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Sagi, Z. Shi, and S. Vadhiyar Users Guide to NetSolve V1.4. *Computer Science Dept. Technical Report CS-01-467*, University of Tennessee, Knoxville, TN, 2001.

[3] H. Nakada, M. Sato, and S. Sekiguchi. Design and Implementations of Ninf: towards a Global Computing Infrastructure. *Future Generation Computing Systems, Metacomputing Issue*, 15(5-6):649.58, 1999.