GAD Kit - A Toolkit for “Gridifying” Applications

Quoc-Thuan Ho, Yew-Soon Ong, Wentong Cai*, Hee-Khiang Ng, and Bu-Sung Lee
Nanyang Technological University
School of Computer Engineering
Nanyang Drive, Singapore 639798

Abstract. In this paper, we present a Grid Application Development toolkit called GAD Kit. It simplifies the “gridifying” process by providing mechanisms to enable existing applications as Grid services and their consumptions. These include features for automatic wrapping of applications as Grid services, service deployment and discovery, metascheduling and seamless access to Grid resources. To facilitate access to Grid resources, GAD Kit was developed based on the standard GridRPC API. Finally, we present the use of GAD Kit for “gridifying” a realistic aerodynamic design application.

Keywords: Grid computing, Grid programming tools, GridRPC.

1 Introduction

Over the recent years, there has been an increasing interests in executing high performance applications over the Grid. A lot of works have been done in the areas of core middleware support for building Grid computing environments, application specific problem solving environments and portal development [1]. However, at present, there has been a lack of user-centric tools and environments that facilitates seamless deployment of existing software components/applications as Grid services. Clearly, the tools to provide a secure yet simple consumptions of these grid resources has been lacking too.

“Gridifying” is defined as the process of transforming an existing application to execute appropriately on a Grid environment. This process involves handling the issues on parallelizing existing applications, deploying applications as Grid services, discovering and consuming of Grid services, last but not least, the scheduling of tasks on the Grid which would involves both at cluster and node levels. Clearly, having general users to perform all these tasks manually in order to be able to use the Grid is time consuming, tedious, and non-trivial.

In this work, we present the GAD Kit to simplify the “gridifying” process by providing mechanisms for seamless enabling of existing applications as Grid services and their consumptions. The rest of the paper is organized as follows.

* Contact Author: email – aswtcai@ntu.edu.sg
In the second section, we present the basic components of our GAD Kit. In addition, we demonstrate the “gridification” of a realistic application using the GAD Kit in section 3. Finally, the fourth section draws our main conclusions.

2 GAD Kit Architecture and Design

The GAD Kit comprises of a Graphical User Interface (GUI) module, Grid service provider module, Grid service consumer module, and the metascheduler module. Here, to facilitate a common application programming interface that conforms to Grid standards, the GridRPC API [2, 3] is used in the GAD Kit. The “gridifying” process in the GAD Kit is accomplished by two main components, namely the Service Provider and Service Consumer modules.

2.1 Service Provider

The service provider module in the GAD Kit provides automatic wrapping and deployment of existing applications as Grid services. To provide a platform-independent development environment for wrapping of existing applications, a standard format for specifying Grid service interfaces based on Grid Function Description (GFD) language is developed in the GAD Kit to accommodate different domain-specific interface. Currently, the problem description interface file format of NetSolve [4] is supported. The GFD language is similar in spirit to CORBA IDL and focuses on the arguments passed to the services. Once applications are wrapped as Grid services, they are deployed automatically onto Grid resources as specified by the users. Rather GAD Kit employs standard Grid technologies such as GridFTP and Globus remote commands to deploy Grid services, i.e., to distribute and install Grid services on resources.

2.2 Service Consumer

The service consumer module provides simple aggregation of Grid resources and offers mechanisms to facilitate consumption of the deployed Grid services. Principally, it produces Grid-enabled client programs that access deployed Grid services. To generate Grid-enabled client programs, templates for consuming Grid services are provided in the GAD Kit. This reduces programming time and eliminate programming errors. Further, a service consumer GUI is provided to user for customization of the templates, for example, selecting which Grid services to call in the client programs. To permit embarrassingly parallelism of Grid services, capabilities of making multiple asynchronous requests are also provided. Resource discovery and metascheduling mechanisms are also included in the GAD Kit. These mechanisms facilitate efficient service of requests across the dynamically changing resources based on the system configuration, speed, and workload information obtained from Globus MDS [5] and Ganglia monitoring toolkit [6].
3 Realistic Airfoil Analysis and Design using GAD Kit

![Diagram of GFD and Grid-enabled Client Program]

Fig. 1. “Gridifying” of Airfoil Analysis and Genetic Algorithm

In this case study, a Genetic Algorithm (GA) for aerodynamic airfoil designs was “gridified” using GAD Kit. The design optimization process was carried through a number of GA generations. In each generation, the Grid-enabled GA produces a population of designs for analysis using the “gridified” airfoil analysis service provided. Subsequently, these analysis results pertaining to the merits of each potential design point are used to generate the new designs for the next generation. The “gridifying” process of this application is outlined in Figure 1(a).

To implement the airfoil analysis Grid service, the users first specify the GFD of the service in the first stage (see Figure 1(b)). The domain-specific interface based on NetSolve is generated automatically. From here, the users can simply transfer their existing codes into the domain-specific interface and customize any output, input arguments necessary for the service. At stage 3, the Grid service is subsequently deployed on the user specified resources. Note that the GAD Kit registers the service on the Globus MDS automatically and all deployment tasks are completed with the use of standard GridFTP and Globus remote commands.

To consume the deployed Grid services, user first selects the airfoil analysis service from the list of Grid services discovered automatically by the GAD Kit via an in-built GUI. Subsequently, a Grid-enabled client program which takes into
account the configurations specified by the users for consuming the airfoil analysis service gets generated automatically. It is worth noting that the generated Grid-enabled client program contains built-in prototypes to perform resource discovery (i.e., locate where the airfoil analysis resides in) and metascheduling (i.e., select the cluster that gives maximum throughput). The users may perform further customization to this client program if necessary (see Figure 1(c)).

Traditionally, “gridifying” an application involves dealing with the low-level programming throughout all five stages. Using the GAD Kit, the development time is now spent mainly on specifying the GFD, customizing service and modifying the Grid-enabled program. With the GAD Kit, the “gridifying” process of applications is simplified considerably, contributing significantly to the success of Grid realistic applications.

4 Conclusions

The GAD Kit presented in this paper can simplify the “gridifying” process and offer a seamless and transparent access to Grid resources. It comprises of the GUI module and a set of mechanisms for providing and consuming Grid services. They cover most of mechanisms necessary for Grid application development and free users from tedious low-level programming and the complexity of underlying system. A case study on using the GAD Kit for facilitating embarrassingly parallelism in the aerodynamic design optimization application is also presented briefly.

In our future work, we plan to offer different models for applications. This requires mechanisms for seamless construction of hybrid services from existing services. In addition, extension of the present work to handle multi-level nested services would also be considered.

References