Coordinating Distributed Resources for Complex Scientific Computation[1]

Huashan Yu, Zhuoqun Xu, Wenkui Ding

School of Electronic Engineering and Computer Science, Peking University, Beijing, 100871, P.R. China
yuhs@ailab.pku.edu.cn, zqxu@pku.edu.cn, ding@ailab.pku.edu.cn

Abstract. There exist a large number of computational resources in the domain of scientific and engineering computation. They are distributed, heterogeneous and often too restricted in computability for oneself to satisfy the requirement of modern scientific problems. To address this challenge, this paper proposes a component-based architecture for managing and accessing legacy applications on the computational grid. It automatically schedules legacies with domain expertise, and coordinates them to serve large-scale scientific computation. A prototype has been implemented to evaluate the architecture.

1 Introduction

It is well known that there exist a large number of legacy applications in almost every scientific and engineering domain. They are unchangeable and too valuable to be given up. Each alone is very restricted in computability, due to both the target platform's limitation and the programming complexity. However, some of them are complementary in function and resolvable problem characteristics, and others are compatible. It is doubtless that their aggregation is almost powerful enough to solve every problem reliably and efficiently. The idea of coordinating these legacy applications and their target platforms with Grid technologies [1, 2] to solve large-scale and complex scientific problems is straightforward, and the advantages are obvious. However, despite the technical advances of Grid computing in recent years, this kind of coordinative computing remains a grand challenge.

We have attempted to devise an application framework *AOD* for coordinating and scheduling distributed legacy applications on the computational grid, so as to with. It is component-based and built on top of OGSA [2], supporting distributed but complementary legacies to be selected dynamically for solving large-scale complex scientific problems in a cooperative way. Our ultimate goal is to equip the computational grid with the mechanisms for coordinating and scheduling legacies automatically, and hence to create an on-demand computing environment. In this environment, legacies are augmented with domain expertise and abstracted as consistent services for performing specific computation on the computational grid.

[1] This work was supported by National Natural Science Foundation of China (No. 60303001, No.60173004)

H. Jin et al. (Eds.): NPC 2004, LNCS 3222, pp. 129-132, 2004.

Every service is automatically implemented with a collection of complementary and competitive legacies. These services are self-optimizing, self-healing and adaptive to problem characteristics. A grid application is a set services connected with each other by directed edges, and AOD provides the mechanisms for executing it on the computational grid.

The next section presents an approach for automatically managing and accessing legacy applications on the computational grid, and discusses the mechanisms for coordinating them to solve large-scale scientific problems. Section 3 introduces a prototype of AOD. Related works are overviewed in section 4, followed by a conclusion of this paper.

2 A Grid Environment for Large-Scale Scientific Computation

To manage and access legacy applications on the computational grid, we have proposed the concept of *grid-programming component* that provides an approach for incorporating domain expertise into the grid environment. A *grid-programming component* (GP component) is an autonomic and extensible entity exiting on the computational grid, aggregating a collection of legacies augmented with necessary domain expertise, and providing a set of functions for developing grid applications. Every function implies some kind of computation with its domain-customized interface, and is automatically implemented with the legacies. They are self-healing for failures occurred on the computational grid, self-optimizing according to problem characteristics and dynamic statuses of grid resources. We call every function as an *on-demand computing service* (OD service) of the grid-programming component.

Every GP component uses a generic configuration framework to specify its underlying computational resources and the augmented expertise. When it is registered, the configuration is interpreted by AOD to configure and implement its OD services on the computational grid. The configuration declares a list of IO ports and OD services as the GP component's interface. The IO ports are used by the OD services to input and output their arguments. Every port input or output one type of data objects in files, and specifies every transferred file's syntax and semantic in domain terms. Generally, an OD service has more than one candidate implementation. Every candidate is provided by some local platform independently, involving one or more legacies installed on the platform. Different candidates may differ in efficiency and resolvable problem characteristics. The configuration not only specifies an executing scheme of every candidate's underlying resources, but also details three kinds of domain expertise for dynamically selecting candidate. One is the annotation of every candidate's applicability to problem characteristics. The second is the methods for querying dynamic statuses of every candidate's underlying resources. And the third is the methods for detecting automatically a problem's characteristics from its data. With the expertise and the support of Grid middleware like Globus Toolkit [3], an OD service dynamically selects one optimal candidate to execute and complete the desired computation when it is invoked.

Based on the concept of GP component, AOD provides a Grid environment for combining distributed legacy resources dynamically to serve large-scale scientific computation. Every legacy in AOD is encapsulated in some GP component. In grid applications, a complex problem is divided into several concurrent and relatively

simple sub-problems, and every sub-problem is specified with a reference to some OD service. These references are connected with directed edges to specify the problem domain's concurrency. When the application is submitted to run, AOD will automatically create a formal sub-problem description for every reference, according to the application's arguments and the connected edges. The referred OD services are invoked concurrently, and each is provided a formal sub-problem description. AOD is also responsible for transferring data objects and communicating messages for the invoked OD services on the computational grid.

AOD consists of **repository**, **scheduler** and **broker**, and is built on top of Grid middleware for OGSA. The repository is responsible for configuring and managing all registered GP components and their OD services on the computational grid. It also provides an environment for every OD service to select and schedule its underlying computational resources. An OD service's behaviors on any underlying host are conducted by the local broker instance. The scheduler automatically invokes and synchronizes concurrent OD services when an application is executing.

3 Implementation and Experiment

Based on Globus Toolkit, we have implemented a prototype for AOD. In this prototype, the scheduler, the repository and every broker instance exchange have been assigned a local TCP/IP port respectively, in order to receive messages real-time messages with GlobusIO, so as to invoke OD services, perform computation and synchronize concurrent OD services. The local broker instances on every host are managed with GRAM. When an OD service is invoked, its arguments are transferred on the computational grid with GridFTP. The prototype provides three XML schemas for programmers. The first schema is for domain experts to define FC descriptors. The second one is for developing the configurations of GP components, and the last one is for developing grid applications. We also have developed a tool for running grid applications with Internet browsers.

Table 1. Experimental Result of a Demonstrative Example

GP component	computing host	working directory	Start time	End time
preProc	162.105.203.100	/home/chen/lyan/test1/	16:43:41	16:45:39
voiFilt	162.105.203.100	/home/chen/lyan/part1/	16:45:47	16:46:58
qCom	162.105.203.38	/home/aitest/oil/part2/	16:45:47	16:48:16
Synth	162.105.80.17	/home/globus/lyan/test2/	16:48:31	16:55:26

Table 1 is the experimental result of a demonstrative example performed on the prototype. The example consists of four GP components *prePrc*, *voiFilt*, *qCom* and *Synth*, performing some kind of simplified pre-stack migration for oil-prospecting data processing. Every GP component provides one OD service with the corresponding legacy application and several additional executables. *prePrc* accepts the primal sampling data, and its result is passed to *voiFilt* and *qCom* respectively. *Synth* creates the final result by synthesizing the results of *voiFilt* and *qCom*. The experimental sampling data is about 2 GB, consisting of two binary data files.

4 Related Works and Conclusion

In recent years, the challenge of developing grid applications has been investigated extensively. The OGSA is the first effort to standardize Grid functionality and produce a Grid programming model consistent with trends in the commercial sector. It integrates Grid and Web services concepts and technologies. In this architecture, resources are encapsulated to be Grid services [2, 4] with standard interfaces and behaviors. XCAT [5, 6] and ICENI [7] attempt to build an application component framework on top of OGSA for distributed computation, and support grid applications that require the collaboration of different Grid services. Neither OGSA nor XCAT takes account of complementary or competitive resources in resource scheduling. ICENI seeks to annotate the programmatic interfaces of Grid services using WEB Ontology Language, allowing syntactically different but semantically equivalent services to be autonomously adapted and substituted.

AOD provides the mechanism for scheduling complementary and competitive resources universally, according to problem characteristics and dynamic resource statuses. It abstracts distributed and heterogeneous computational resources to be services that are self-healing for failures occurred on the computational grid and adaptive to both problem characteristics and dynamic statuses of computational resources, and supports multiple services to serve complex scientific problems collaboratively. This kind of resource managing strategy not only improves the dependability and efficiency of grid computing, but also simplifies the complexity of developing grid applications. We are going to replace current candidate implementations of OD services with Grid/Web services, so as to simplify the complexity of developing GP components.

References

1. I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001.
2. I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.
3. Globus Toolkit. http://www.globus.org/toolkit/default.asp
4. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, D. Snelling. Open Grid Services Infrastructure (OGSI) Version 1.0. Global Grid Forum Draft Recommendation, 6/27/2003
5. Dennis Gannon, Sriram Krishnan, Liang Fang, Gopi Kandaswamy, Yogesh Simmhan, and Aleksander Slominski. On Building Parallel and Grid Applications: Component Technology and Distributed Services. http://extreme.indiana.edu/labpubs.html
6. Sriram Krishnan and Dennis Gannon. XCAT3: A Framework for CCA Components as OGSA Services. In Accepted for publication to HIPS 2004, 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments. IEEE Computer Society Press, 2004. http://extreme.indiana.edu/labpubs.html
7. J. Hau, W. Lee, and Steven Newhouse, Autonomic Service Adaptation using Ontological Annotation. In 4th International Workshop on Grid Computing, Grid 2003, Phoenix, USA, Nov. 2003.