Supporting Interactive Computational Science Applications Within the JGrid Infrastructure

Szabolcs Pota and Zoltan Juhasz

University of Veszprem, Department of Information Systems,
Egyetem u. 10., 8200, Veszprem, Hungary
{pota, juhasz}@irt.vein.hu

Abstract. Future computational grid systems must support interactive and collaborative applications to help computational scientists in using the grid effectively. This paper shortly introduces how the JGrid system, a service-oriented dynamic grid framework, supports interactive grid applications. Via providing a high level service view of grid resources JGrid enables easy access to grid services and provides an effective programming model that allows developers to create dynamic grid applications effortlessly.

1 Introduction

Due to rapid advances in grid technology and the increasing number of production grid environments, scientists now can locate suitable remote computing resources without much difficulty, but programs are still executed on the selected resource by traditional batch runtime systems. Besides depending on platform-specific details, this grid execution model does not utilize the full potential of the grid. It is still a difficult problem to connect several resources (belonging to different administrative domains) to be used at the same time for solving one particular problem. Furthermore, grid environments based on batch systems provide limited support for dynamic resource discovery and runtime exception handling.

There are grid usage scenarios where static batch execution is not adequate. Interactive applications, ranging from scientific visualization, graphics rendering through computational steering and man-in-the-loop simulations to collaborative problem solving, could also benefit to a large extent from Grid technology. Interactive computational grids should support a wide range of applications, provide effective ways for service/resource orchestration, and should integrate into the everyday user work environment.

In this paper, we describe the JGrid [1] Java/Jini based service-oriented grid framework that provides a unified, high-level service view of computational resources, and besides traditional batch execution supports interactive/collaborative grid applications.

2 The JGrid System

The goal of the JGrid project is to develop a novel service-oriented grid system that aims to create a dynamic computational service fabric in which batch and interactive

applications can discover and use computing resources on-demand, and provides a high-level, effective service-oriented programming model for developers.

The JGrid system consists of a set of extensible services that provide a complete, dynamic grid infrastructure including wide-area service discovery, security support, and core computational services (batch, compute and storage services). Services can be easily shared among multiple clients and accessed concurrently. In addition, JGrid provides uniform, seamless access to diverse grid resources.

2.1 Grid Access

Most grid systems use the web browser as their primary user interface for accessing resources, typically via grid portals. Although web technology has developed significantly, web user interfaces cannot easily provide the rich functionality of desktop environments. This is especially problematic in interactive and collaborative applications.

In JGrid services are represented by Java interfaces and accessed via service proxies implementing these interfaces. As a result, services can be accessed programmatically in a similarly seamless, protocol independent way, by simply calling methods on the proxy object. A graphical, dynamically downloadable Service UI can be also attached to these proxy objects if the service is to use by end users. A Java service browser is also provided in JGrid for end users to access grid services and that provides rich desktop user experience and a single access point to the grid.

2.2 Key Computational Services

As the main intended use of JGrid is to solve large computational problems, we have placed great emphasis on developing computational services that support the widest range of sequential and parallel grid applications. To achieve this, we decided to create two different types of computational services. The *Batch* and *Compute Services* complement each other in providing JGrid users with a range of choices in programming languages, execution modes and inter-process communication modes.

The Batch Service [2] supports legacy applications and traditional batch execution by integrating batch runtime systems (e.g. Sun Grid Engine or Condor) in JGrid using a service-oriented interface. The integration is achieved via the standard DRMAA [3] interface, this way any batch execution environment that supports DRMAA can be integrated after re-configuration of the Batch Service.

The Compute Service supports the execution of sequential and parallel grid applications. It allows clients to execute Java programs using virtualized remote resources, which can represent single, multi-processor computers or clusters. The service, in fact, is a special Java runtime system that enables clients to execute programs in a secure and controlled way, and acts as core building block in dynamic grids that can execute interactive grid applications.

Interactivity is supported with a number of mechanisms built in the Compute Service. Compute Services use (i) preemptive scheduling to let submitted programs start immediately and remain responsive throughout the entire execution period. Furthermore, users can (ii) control (suspend, resume, cancel) program or task execution via dynamically generated proxies that connects to the remote task. The (iii) monitoring of applications running inside a Compute Service is also supported. Finally, users can

(iv) spawn processes inside the remote Compute Service that creates dynamic server objects accessed via remote method invocation.

2.3 Remote Data Access

The JGrid *Storage Service* provides access to remote data in a way identical to the Java I/O model, and represents a remote file space in a platform independent manner. It enables clients to access and manipulate their own directories and files. By using proxies for file access, the protocol used to access remote files is completely hidden from the user and is interchangeable. Sharing the appropriate remote file proxies among distributed application components ensures that everybody can access the same piece of data, i.e. sharing the same data.

3 Usage Scenarios

This section shortly introduces some usage scenarios to illustrate how JGrid can be used in various computational application domains.

3.1 Long Running Numerical Applications

We used JGrid to execute a biological application performing pair-wise alignment of biological sequences based on the BioJava library [4]. The application demonstrated the reliability of JGrid in executing long running numerical applications and also showed the appropriate computational performance of a Java-based computational environment. The program was executed on one of the available Compute Services of our research group. Running tasks could access the Storage Service storing the input sequence file and write the results via its proxy as it were a local file system.

3.2 Parallel Grid Applications

We used a simple SPMD-style parallel image processing problem to illustrate parallel program execution support in JGrid. The example performs simple image processing operations, e.g. edge detection, etc. in parallel, during which nearest-neighbour communication steps are required.

Tasks of the program are spawned at runtime on dynamically discovered Compute Services. Next, the client creates the required topology by exchanging task proxies between neighbouring tasks. Once the configuration is set, the client distributes the image segments, starts the computation and displays the returned results. A successful live demonstration was performed using a client in Hong Kong and Compute Services in Hungary.

3.3 Collaborative Applications

A number of case studies were developed to demonstrate more complex ways of interactions during collaborative work. One grid application is a modified version of the C3D collaborative and distributed ray tracing application [5] developed for benchmarking Java RMI and serialization. We modified this application to be able to run on

JGrid and added the flexibility of using dynamically discovered, possibly geographically distributed services.

Another collaborative application demonstrates the use of 3D visualisation in JGrid. Using a shared grid service storing a Box World data model, multiple clients can connect to the service and view and modify the model from different locations.

4 Summary

This paper introduced the JGrid environment, a service-oriented grid system developed to run computational grid applications and some usage scenarios supporting interactive applications. In contrast to most grid environments, JGrid was designed to create dynamic, interactive and collaborative applications. Its service-oriented nature provides users with a unified view of grid resources and simplifies the addition of future services with new functionality. We are continuing work on JGrid developing full-scale, real-world computational science grid applications.

References

1. JGrid: A Jini-based Universal Service Grid, <http://pds.irt.vein.hu/jgrid>
2. Pota, S., Sipos, G., Juhasz, Z., Kacsuk, P.: Parallel Program Execution in the JGrid System, In Proc. 5th Austrian-Hungarian Workshop on Distributed and Parallel Systems, Springer, Kluwer International Series in Engineering and Computer Science, Vol. 777, Budapest, Hungary (2004) 13-20
3. Distributed Resource Management Application API Working Group (DRMAA-WG), <http://www.drmaa.org/>
4. Pocock, M., Down, T., Hubbard, T.: BioJava: open source components for bioinformatics, ACM SIGBIO Newsletter, Vol. 20, No. 2. (2000) 10-128.
5. C3D, A distributed raytracer for benchmarking Java RMI and Serialization, <http://www-sop.inria.fr/sloop/C3D>