Component based Applications Programming within a Service-Oriented Grid Environment

Rainer Schmidt, Siegfried Benkner, Ivona Brandic, Gerhard Engelbrecht

Institute of Scientific Computing, University of Vienna

Nordbergstrasse 15/C/3, A-1090 Vienna, Austria

email: rainer@par.univie.ac.at

Extended Abstract

I. Introduction

We present an approach and prototype implementation that applies a component based programming model to a service-oriented computational Grid environment. Our system is based on the Vienna Grid Environment (VGE) [1], a Grid infrastructure for automatic HPC application provision over standard Web service technology. VGE services provide generic interfaces for remote job execution, monitoring, error recovery as well as for application level QoS support. A VGE Grid typically comprises multiple services and clients, one or more service registries as well as a certificate authority. VGE client applications are usually written using a Java API which allows the programmer to interact with remote VGE services. The VGE service provision framework is currently being utilized in the GEMSS Project [2] for the Grid provision of advanced medical simulation services which incorporate compute intensive methods such as Finite-element Modeling, Monte Carlo simulation, and Computational Fluid Dynamics. Key aspects of VGE include negotiable QoS support for time-critical service provision, flexible support for business models as well as end-to-end security. Most of these applications consist of various steps of execution for example mesh generation, data analysis, or visualization which can be deployed separately using VGE. VGE client applications are currently being constructed using a service-oriented programming model (registry-lookup, XML-RPC). By integrating a component based model we would like to provide the client with a simple plug-and-play environment (e.g. allowing visual composition) and move the handling of the service-oriented environment away from the client environment towards a component framework.

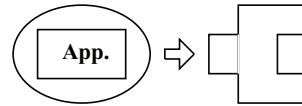It is our goal to provide a component-based program-

Fig. 1. Providing a Component Abstraction for Application Grid Services

ming model on top of a service-oriented infrastructure that allows programmers to compose client-side applications from deployed VGE components (Figure 1). We therefore incorporated several Common Component Architecture (CCA) based mechanisms into the system. In this paper we will briefly describe the CCA based component framework, the client programming environment and the mechanism used to integrate Web services with the framework. We further discuss to which extend such a component model can be applied to the functionality provided by VGE. Due to space limitations we omit a detailed description of CCA [3].

II. Related Work

Component based distributed programming models such as CCM [4], EJB [5] or DCOM [6] are widely used standards mostly in the context of commercial applications. The work presented in [7] uses parallel CORBA components within computational Grids. A hierarchical component model for a parallel and distributed framework is presented in [8]. CCA defines a component model for scientific applications wich is utilized by various projects: The Ccaffeine framework [9] focuses on supporting high-performance MPI based components using Babel for language interoperability. XCAT [10] is a distributed

framework that integrates CCA with Grid services e.g. OGSI [11]. It uses XSOAP [12] for communication and can use ssh or Globus GRAM [13] for remote component instantiation. Further distributed CCA based frameworks like SCIRun2 [14], LegionCCA [15], Mocca [16], or DCA [17] address diverse aspects related to problem solving, metacomputing, or component engineering.

III. Generic Application Services

At its core, a VGE application service follows a generic application service model and exposes a native application as a service to be accessed by multiple remote clients over the Internet. The application service provides common operations for remote job management, data staging, and optional operations for error recovery and QoS support. Figure 2 shows the interfaces and operations a VGE service provides.

The operations upload and download are used for uploading of input data to a service and downloading of output data. The operation push is provided for data staging between different services. The execution of a remote application can be initiated by calling start and stopped by calling kill. The operation getStatus allows the client to download an application-specific status file. The operations provided in the error recovery interface enable clients to control checkpointing and restarting of applications. The operations of the interface QoS are used during QoS negotiation. The behavior of these operations is customized for a specific application by means of an XML application descriptor.
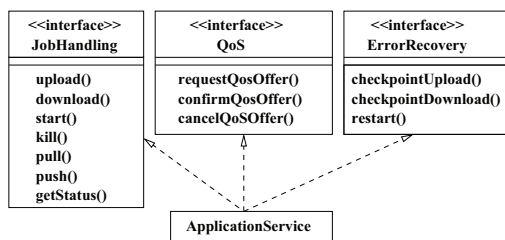

Fig. 2. Generic Application Service

VGE makes use of open-source frameworks such as Tomcat [18] and Axis [19] to provide HPC applications over standard Web Service technology. For large file transfers SOAP attachements are utilized. An operational PKI infrastructure based on X.509 certificates provides transport and message layer security.

IV. A Component Model for Grid Services

We aim to provide VGE services to client programmers based on the CCA component model. We therefore use CCA mechanisms like provides/uses ports, component interface, framework services and builder service. We currently do not implement the whole specification (e.g. events) and further did some slight modifications on the builder interface.

For integrating VGE application services with the component model we implemented three additional software packages. A distributed component framework that mediates between service-orientation and the component model, the service package provides interoperability between the Web services and the component framework, the client programming environment allows for component based application construction.

A. General System Overview

Our infrastructure (Figure 3) aims to map a component based application to services available in the Grid at runtime. VGE application services are therefore represented as components which can be accessed and interconnected.
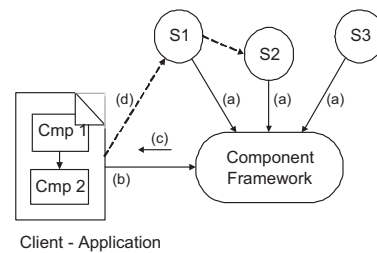

Fig. 3. General System Architecture

The component framework is implemented as Web service and acts as broker between application demands and available Grid services. We therefore extend the VGE services with component interfaces for interacting with the framework. The framework provides an interface for managing component ports, and services for client application construction. A service can register provided/used ports with the framework as well as request them dynamically (a) at execution time.

The client application may consist of several components, which can be instantiated, connected and invoked. The Builder service provided by the framework locates the services on behalf of the client application at execution time (b) and delivers runtime relevant service information back to the involved entities (c) which allows them to interact (d).

B. Providing VGE Services as Components

In CCA a component communicates with the component framework over an additional interface, called component interface. We therefore provide a library that implements the component functionality (Figure 4) which can be added easily to the Web Service by extending the deployment descriptor. A configuration file is used to specify meta-data on the wrapped scientific application (name, version, I/O file format) as well as uses ports (e.g. for data staging).

When such a Grid service is added to the pool of components that is maintained by our framework it is provided with a proxy, called services object in CCA, over the component interface. This proxy allows the service to manage `provided` and `used` port objects dynamically (register, retrieve).

In order to exchange the complex, CCA defined data types (e.g. Port, TypeMap, ComponentId), the current prototype uses SOAP messages utilizing Java serialization and Base64 encoding. Hence, the system represents an application service as a component and each Web service interface as a `provides` port. We have to note that this mapping imposes a certain limitation if multiple provides ports of the same type are exposed by one component as described in [20].
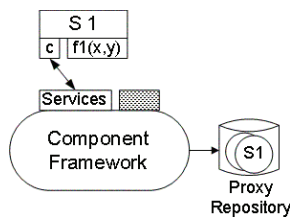

Fig. 4. Service - Component Framework Interaction

C. Remotely Accessible Component Framework

The Component Framework is implemented as Web service and provides the component management service, a simple Builder service and a Java proxy repository.

The proxy repository is used to provide the client environment with a high flexibility concerning component specific libraries. The idea is to enable client applications to stay compliant with different versions of VGE services, dynamically retrieve proxy updates, and also be able to incorporate non-VGE services. We therefore extended the Java classloader mechanism with a dynamic proxy pattern that allows the client to access remote libraries (Figure 5):
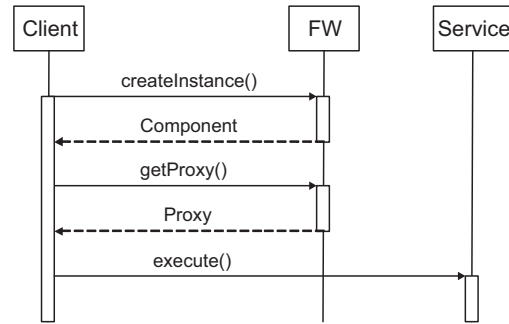

Fig. 5. Application Execution using Dynamic Proxy Pattern

If a port is registered with the component framework it can be associated with a proxy implementation which can be uploaded to the framework. At execution time the client environment contacts the framework services in order to fill the component representations with substantial runtime information (component id, service endpoint, proxy class name, class loader). Invoking a method on a port causes the runtime engine to generate a call from the client to a service using the local code base. If a fault occurs or the component is unknown to client-side code repository, the Java classloader dynamically retrieves an appropriate proxy file from the framework where the component is registered. This mechanism of interface based service discovery and proxy lookup which we have applied for Web services is well known from Jini [21] technology.

D. VGE Component based Application Programming

If VGE services are represented as CCA components every component looks equal as all services provide the same generic interfaces. Every VGE service also has the same dependency (uses port) to another VGE job handling interface which is required for data staging operations. As VGE services interoperate by exchanging files and don't use individual remote methods calls, all VGE components are technically interoperable.

Components are identified by meta-data descriptions of their underlying applications. The Builder Service, however does also consider the component interfaces when creating an instance in order to allow adaptor components to non-VGE services being incorporated.

For constructing composite applications our Builder Service basically provides following methods to the client environment:

createComponent: Requests a component from the framework based on application-specific meta-data, as well

as optionally a set of provided interfaces and causes the creation of a client session at the target resource. Components should be created at the beginning of a program as they could possible be not available and interrupt during application execution.

getComponentPort: Retrieves a specified Java interface from a component which can be invoked or connected to a `uses port`.

connect: Connects two components based on matching `provides`/`uses` ports. Using VGE services connect causes a push operation.

destroyInstance: Removes the component instance from the applicaton and invalidates the client session at the resource.

The prototype implementation currently uses Java as programming language for building composite applications. For future versions we plan to incorporate also a scripting language together with a graphical representation.

V. Future Work

VGE components communicate by exchanging files within the VGE infrastructure. In order to provide higher interoperability between the native applications we currently work on abstractions for scientific datatypes which serve as parameter types for `provides` and `uses` ports. These datatypes may allow automatic conversion of diverse file formats used between the connected application components.

Moreover we envision the provision of QoS aware components by integrating mechanisms provided by VGE that allow to negotiate on a Web service level agreement between service provider and consumer.

VI. Conclusion

We presented an environment that adds the abstraction of components to a Web service based Grid infrastructure. The system is based on VGE, a service provision framework for native HPC applications. We described a CCA based framework that allows the dynamical integration and management of VGE application services using Web service technology. We further provide a client environment which provides automatic library reconfiguration capabilities and allows the construction of composite applications based on the deployed application components.

References

[1] S. Benkner, I. Brandic, G. Engelbrecht, R. Schmidt. *"VGE - A Service-Oriented Environment for On-Demand Supercomputing"*, Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (Grid 2004), Pittsburgh, PA, USA, November 2004.

[2] Gemss Project homepage. http://www.gemss.de.

[3] R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski. *"Toward a Common Component Architecture for High-Performance Scientific Computing"* Proceedings of the High-Performance Distributed Computing Conference, August 1999, pp. 115-124.

[4] CORBA Component Model, v3.0, OMG. http://www.omg.org/technology/documents/formal/components.htm.

[5] Enterprise JavaBeans technology: http://java.sun.com/products/ejb.

[6] COM Component Object Model Technologies, Microsoft, http://www.microsoft.com/com/default.mspx.

[7] S. Lacour, C. Perez, and T. Priol. *"Deploying CORBA Components on a Computational Grid: General Principles and Early Experiments Using the Globus Toolkit"* In Wolfgang Emmerich and Alexander L. Wolf, editors, *"Proceedings of the 2nd International Working Conference on Component Deployment"* (CD 2004), number 3083 of Lect. Notes in Comp. Science, Edinburgh, Scotland, UK, pages 35-49, May 2004. Springer-Verlag.

[8] Francoise Baude, Denis Caromel, and Matthieu Morel. *"From Distributed Objects to Hierarchical Grid Components"* International Symposium on Distributed Objects and Applications (DOA), Catania, Sicily, Italy, 3-7 November 2003.

[9] B. A. Allan, R. C. Armstrong, A. P. Wolfe, J. Ray, D. E.Bernholdt, J. A. Kohl. *"The CCA core specification in a distributed memory SPMD framework"* Concurrency and Computation: Practice and Experience 14(5), 2002.

[10] Madhusudhan Govindaraju, Sriram Krishnan, Kenneth Chiu, Aleksander Slominski, Dennis Gannon, and Randall Bramley. *"XCAT 2.0: A Component-Based Programming Model for Grid Web Services"* Technical Report-TR562, Department of Computer Science, Indiana University. Jun 2002.

[11] Foster, I., Kesselman, C., Nick, J., Tuecke, S. *"The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration"*, Globus Project, 2002. Available at http://www.globus.org/research/papers/ogsa.pdf

[12] XSOAP toolkit. http://www.extreme.indiana.edu/xgws/xsoap.

[13] The Globus Alliance. http://www.globus.org.

[14] Keming Zhang, Kostadin Damevski, Venkatanand Venkatachalapathy, and Steven Parker. *"SCIRun2: A CCA Framework for High Performance Computing"* in Craig E. Rasmussen, editor, Proceedings of the 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments, IEEE Computer Society, 2004.

[15] Madhusudhan Govindaraju and Himanshu Bari and Michael J. Lewis. *"Design of Distributed Component Frameworks for Computational Grids"* in *"Proceedings of the International Conference on Communications in Computing (CIC)"*, pages 160–166, June 2004.

[16] Maciej Malawski, Dawid Kurzyniec, and Vaidy Sunderam. *"MOCCA - Towards a Distributed CCA Framework for Metacomputing"* in Proceedings of the 10th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS2005) (submitted), 2005.

[17] Felipe Bertrand and Randall Bramley. *"DCA: A Distributed CCA Framework Based on MPI"* in 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments, Santa Fe, NM, Apr 2004.

[18] Apache Tomcat. http://jakarta.apache.org/tomcat/.

[19] Apache Axis. http://ws.apache.org/axis/.

[20] S. Krishnan, D. Gannon. *"XCAT3: A Framework for CCA Components as OGSA Services"* Proceedings of the Ninth International Workshop on High-Level Parallel Programming Models and Supportive Environments, April 2004, pp. 90-97.

[21] *"J. Waldo. The Jini Architecture For Network-Centric Computing"* Communications of the ACM, 42(7):76-82, July 1999.