Project no. FP6-034442

GridCOMP

Grid programming with COMPonents : an advanced component platform for an effective invisible grid

STREP Project

Advanced Grid Technologies, Systems and Services

D.CFI.05 – CFI prototype and early documentation

Due date of deliverable: 31 May 2008

Actual submission date: 07 July 2008

Start date of project: 1 June 2006 **Duration**: 30 months

Organisation name of lead contractor for this deliverable: INRIA

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU

Keyword List: component framework implementation, GCM, ProActive
Responsible Partner: Denis Caromel, INRIA

MODIFICATION CONTROL			
Version	Date	Status	Modifications made by
0	DD-MM-YYYY	Template	Patricia HO-HUNE
1	25-06-2008	Draft	Cédric Dalmasso
2	27-06-2008	Draft	Bastien Sauvan
3	30-06-2008	Draft	Cédric Dalmasso
4	03-07-2008	Draft	Cédric Dalmasso
5	07-07-2008	Draft	Cédric Dalmasso
		Final	Denis Caromel

Deliverable manager

- Denis Caromel, INRIA

List of Contributors

- Denis Caromel, INRIA
- Cédric Dalmasso, INRIA
- Bastien Sauvan, INRIA
- Matthieu Morel, UCHILI

List of Evaluators

- Yongwei Wu, TU
- Igor Rosenberg, ATOS

Summary

This document describes the Component Framework Implementation (CFI) prototype. The D.CFI.05 prototype which is an improved version of the early prototype released in the D.CFI.02 deliverable. This prototype is the first implementation of the GCM component model [1].

The new features provided by the CFI prototype are listed and briefly described in the first part of this document. In addition, a separate file contains the early documentation fully describing all the CFI and showing how to make use of this prototype. The documentation contains:

- A user guide explaining how to create primitive and composite components.
- Documentation of each feature included in the CFI.
- Documentation of the GCM deployment framework which implement ETSI standards.

Table of Content

1 Introduction

This document describes the deliverable D.CFI.05, i.e. the prototype of the Component Framework Implementation (CFI). The current implementation of the CFI provides the basic features defined by the CoreGRID NoE project in the GCM [1]. This prototype is built upon the ProActive Grid Middleware [2] which features a framework for transparent asynchronous communication between Active Object [3].

CFI users do not need to know the Active Object programming model and the ProActive middleware to start to use the CFI since the creation of components and access to components rely on standard Fractal/GCM API. The current implementation of the CFI provides the basic features defined in the GCM model. The D.CFI.02 deliverable [4] contains a description of the features included in the previous CFI released. Furthermore, we explained how ProActive is used to provide an implementation of the GCM in the D.CFI.03 [7] which described the architectural design of the CFI. The source code, the binaries and the whole ProActive documentation are packaged in the D.CFI.05-bundle.zip file.

In this document, we describe the changes made in the implementation since the D.CFI.02 released. In addition, this deliverable contains the CFI early documentation.

2 CFI early documentation

To ease the distribution of the document in different format (PDF and HTML) we use the DocBook[1] technology. As a consequence we can not include the documentation in this document. The CFI early documentation is available in a separate file, D.CFI.05_CFI-early-documentation.pdf. The CFI features implemented until now are documented. The documentation contains:

- Documentation of the GCM deployment framework which implement ETSI standards.
- Technical documentation describing how to use each feature included in the CFI.
- A tutorial providing a user guide explaining how to create primitive and composite components along a simple example.

This documentation related to features developed in the frame of GridCOMP is also available in the ProActive user documentation.

Among all the features described in the documentation, we provide in this section an overview of the major changes included in the CFI during the second year of GridCOMP. The documentation contains in details the complete description of these features. Apart from a

[1] DocBook is an XML language for technical documentation, http://www.oasis-open.org/docbook/

consistent number of bugs fixed, a lot of new features have been implemented. As major improvements, we can list: the new GCM deployment framework; and improvements in the collective interfaces. The other new features are detailed in the last part of this section.

2.1 GCM deployment

Until now the CFI was using the ProActive deployment framework. This framework has several drawback and do not provides a fully and easy interoperable way to deploy application on a grid as required in the GCM definition. The main defects are the complexity to write deployment descriptors and the incapacity to easily reuse already written deployment descriptor files with another application or infrastructure. A new framework, named “GCM deployment”, has been designed and implemented to meet those requirements.

To ensure an interoperable deployment, we have defined and implemented two ETSI standards. These technical specifications divide the deployment in two parts:

- “GCM Interoperability Deployment” for grid administrators.
- “GCM Interoperability Application Description” for the application developers.

In the previous deployment used, i.e. the ProActive deployment, the grid part and the application part were not separated. Thus, this new deployment is useful in order to have a clear separation between grid administrators who know the grid and application developers who know their application.
With the "GCM Interoperability Deployment", the administrator describes all the resources provided by the grid and how they are acquired. For the "GCM Interoperability Application Description", the developer describes the needed resources, the way to launch the application and which deployment descriptor(s) must be used. This clear separation allows users to easily reuse the *same* deployment descriptors describing given grids.

The change of deployment framework slightly impacts the rest of the GCM implementation. The major changes were in the Fractal ADL [5] implementation. For backward compatibility it supports both ProActive and GCM deployment framework. Also, old methods using the `org.objectweb.proactive.core.descriptor.data.VirtualNode` class are no longer supported since it has been removed for implementation reasons and in order to avoid confusion with the new VirtualNode class. Those methods have been replaced and now use array of nodes. Now, you can still refer to a given virtual node defined in deployment files by its name and retrieve single node or array of nodes from it.

2.2 Collective interfaces improvements

2.2.1 Multicast interfaces

Within the context of the collaboration with the WP5 partners from University of Chile, multicast interfaces have been improved.

When calling a method on a multicast interface, if some of the parameters of this method are lists of values, we have seen in D.CFI.02 that there is various ways to distribute these values: broadcast, one to one, round robin or a custom mode. Two more modes have been added:

- Random, which distributes each element of the list of values in a random manner.

- Unicast, which sends one value of the list of parameters to only one of the connected server interfaces. The index of the argument to send and the server interface are specified by using a custom controller that extends org.object.proactive.core.component.controller.MulticastController.

As a reminder, collective interfaces give the possibility to manage a group of interfaces as a single entity. Therefore, collective interfaces use the group communication principle. First step is the partitioning of parameters according to the distribution mode. A set of tasks is generated, corresponding to the given partitioning scheme. The dispatch operation follows; it maps generated tasks to connected server interfaces, using one of the available dispatch modes, like for parameters dispatch: broadcast, round robin, random or custom. Now, a new mode is available: dynamic. With this mode, buffered tasks are statically allocated to connected server interfaces using the default allocation mode. Then, remaining tasks (un-buffered) are dynamically allocated to most appropriate connected server interfaces which increases the global performance of the execution.

Finally, the last improvement made is a reduction mechanism. Usually, when calling a method on a multicast interface, the provided result, if there is a result, is a list of values. But, with the reduction mechanism, developer can choose to reduce the received results, i.e. gather and/or perform some operations on the list of values; for instance compute the average on a list of int and eventually return a double as result. In order to use it, the specific annotation org.objectweb.proactive.core.component.type.annotations.multicast.Reduce must be set at the method level and must specify the mode to be used. Two modes are available:

- Select unique value, which considers that the list contains just one value and returns this value.
- Custom, which allows the developer to define its own reduction algorithm.

2.2.2 Gathercast interfaces

By default, an invocation coming from a gathercast interface can be created and executed when all connected client interfaces have performed an invocation on it. The timeout mode of org.objectweb.proactive.core.component.type.annotations.gathercast.MethodSynchro annotation already allows users to have a timeout in sight of avoid that gathercast interfaces wait indefinitely for requests from all connected client interfaces. An additional functionality has been implemented for the gathercast interfaces synchronization and the MethodSynchro annotation has been extended in order to relax the synchronisation constraints on gathercast interfaces. By specifying the new waitForAll mode of the MethodSynchro annotation to "false", the developer can choose to have a gathercast interface which will create and execute an invocation on the first request received from any of the connected client interfaces and therefore to not wait requests from other connected client interfaces. Actually, this provides to gathercast interfaces a symmetrical behaviour to the multicast unicast mode.

2.3 Other changes

In order to add the possibility of having Non Functional prioritized requests, a new controller has been implemented: org.objectweb.proactive.core.component.controller.PriorityController. This feature has been added to solve issues occurring for instance during reconfiguration or for autonomic features developed in the WP3. Using this feature, non functional requests may have a different priority and can pass other requests in the queue. Thus, in a first step, the request types have been extended and a priority order has been decided. Now, by using the priority controller to manage the priority of each method exposed by a component, requests can be:

- Functional requests, which always go at the end of the queue.
- Standard Non Functional requests (NF1), which also go at the end of the queue.
- Non Functional prioritized requests (NF2), which can pass the Functional requests but not pass the other Non Functional requests.
- Non Functional most prioritized requests (NF3), which can overtake all the other requests.

A first support for stream ports has been added. It is available by using the new Java interface StreamInterface as a tag on the java interface definition of a component interface. Now, during instantiation of a Fractal interface type, the implementation ensures for each interface implementing the StreamInterface that all methods it defined have a void return value, otherwise the type creation failed. At the moment, there is no specific communication optimization, the provided stream interfaces just allow to express in the design the stream behaviour of a port.

Another major improvement is the possibility to have non functional components, i.e. components managing non functional aspects put in the membrane. This feature allows developer to create controllers of a component as component themselves. It offers a way to structure the membrane with non-functional components instead of standard Java object. Using non functional components, developer takes advantage of the structure, the hierarchy and the encapsulation provided by a component-oriented approach.

Henceforth, composite components can have an attribute controller. Previously, composite components could not have an implementation class. Now, one exception is allowed: if a composite component has an attribute controller therefore an implementation class can be provided. This implementation class has to implement the AttributesController interface.

To simplify some part of the implementation the ComponentParameters controller has been removed. The methods it featured are now in the Component controller.

3 Perspectives & Conclusion

As listed in the D.CFI.02 deliverable and in this deliverable itself almost all the features defined in the GCM have been implemented. All basic GCM features, primitive and

composite components, single and collective bindings, ADL and deployment are implemented. During the upcoming final 6 months of the GridCOMP project, we will continue to refine the component framework prototype.
The ongoing works are:

- Extension of the component packaging information in addition to ADL.
- Enhancement of the collective interfaces, in particular the synchronisation and reduction capabilities for gathercast cardinality.
- Implementation of a component monitoring API which is needed for GCM autonomic features and will provide more information to user in the GIDE [6] tool.
- Completion of the GCM deployment framework to improve the interoperability.

4 References

[1] GCM, D.CFI.01 - Component model presentation and specification (XML schema or DTD), GridCOMP deliverable

[2] ProActive - Parallel, Distributed, Multi-Core Solutions with Java, http://proactive.inria.fr/

[3] Baude F., Baduel L., Caromel D., Contes A., Huet F., Morel M. and Quilici R., in "GRID COMPUTING: Software Environments and Tools", Jose C. Cunha and Omer F. Rana (Eds), Springer Verlag, January 2006

[4] D.CFI.02 - CFI early prototype, GridCOMP deliverable

[5] FractalADL, The Architecture Description Language of the Fractal component model, http://fractal.objectweb.org/fractaladl/index.html

[6] D.GIDE.01 - Grid IDE and architectural design (Report), GridCOMP deliverable

[7] D.CFI.03 - Architectural design of the component framework, GridCOMP deliverable