GridCOMP - Tutorial
GCM Component Programming

Cédric Dalmasso, Antonio Cansado and Denis Caromel

INRIA - OASIS Team
INRIA -- CNRS -- I3S -- Univ. of Nice Sophia-Antipolis, IUF

IV Grid@Work, Tsinghua University, Beijing
GCM Components

• GCM: Grid Component Model
  • GCM was defined in the NoE CoreGRID
  • GCM extends Fractal with Grid specificities

• Open Source ObjectWeb ProActive
  • implements a preliminary version of GCM

• GridCOMP takes:
  • GCM as a first specification,
  • ProActive as a starting point, and
  Open Source reference implementation.

  • Scopes and Objectives:
    • Grid Codes to Compose and Deploy
    • No programming, No Scripting, …
Introduction to Components

• What are software components?
  • Modules exposing the interaction with the environment
    • Provided (server) interfaces
    • Required (client) interfaces
  • Black-boxes (from outside)

• Advantages
  • Encapsulation (black-boxes)
  • Composition
  • Standardized Description $\Rightarrow$ ADL $\Rightarrow$ GUI, Verification
  • Units of deployment
  • Programming in the large vs. programming in the small (objects)

• Goal
  • Reuse and compose
  • Commercial Off-The-Shelf (COTS)
# Rationale: Grid applications

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Solutions with ProActive/GCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
<td>Distributed components</td>
</tr>
<tr>
<td>Multiple administrative domains</td>
<td>Handled by the middleware</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>Portable implementations, interoperability</td>
</tr>
<tr>
<td>Legacy code</td>
<td>Encapsulation, interoperability</td>
</tr>
<tr>
<td>Performance</td>
<td>Legacy code, parallelism</td>
</tr>
<tr>
<td>Complexity</td>
<td>Hierarchies, collective interfaces</td>
</tr>
<tr>
<td>Dynamicity</td>
<td>Adaptation and coherent reconfigurations</td>
</tr>
<tr>
<td>Tools</td>
<td>ADL, GUI, Packaging</td>
</tr>
</tbody>
</table>
Approach Based on the Fractal Model

- INRIA - France Telecom, V1 in '02
- General model, core concepts
  - Encapsulation
  - Strict Definition
  - Assembly and deployment units
- Simple, extensible, hierarchical, dynamic
- Separation of concerns (controllers)
- However:
  - Distribution ?
  - Deployment ?
  - Parallelism ?

⇒ Fractal requires extensions for Grid Computing

⇒ Specified in the Grid Component Model - GCM (CoreGRID)
Some important Fractal Concepts

- Content
- Controller (or membrane)
- Server Interface
- Client Interface

- Bind(ing)
- Functional interface
- Control (or non-functional) Interface
ProActive/Fractal

- Implementation of Fractal based on ProActive middleware Model
  - Based on MOP architecture: Component as Active Object
  - Distributed components, asynchronous communications (futures)
  - Benefits from underlying features of the middleware
    - Middleware services (Fault Tolerance, Security, Mobility etc..)
    - Deployment framework (in development GCM deployment, being standardized at ETSI)
  - Sequential processing of requests in each component
  - Main extensions to Fractal: deployment, collective interfaces
  - Configurable and extensible
ProActive/Fractal
Standard Fractal Interfaces

singleton interfaces

collection interfaces
dynamically created

Only 1 to 1 communications!
GCM Collective Interfaces

• \(\Rightarrow\) collective interfaces
  • Multicast
  • Gathercast
gather-multicast

• Simplify the design and configuration of component systems

• Expose the collective nature of interfaces

• Interface typing \(\Rightarrow\) Verifications

\(\Rightarrow\) The framework handles collective behavior
at the level of the interface
GCM Multicast interfaces

single invocation $\Rightarrow$ list of invocations

- Multiple invocations
  - Parallel
  - Asynchronous
  - Selective
  - Dynamic

- Data distribution
  - Automatic
  - Customized distribution function
    - Broadcast, scattering, reduction
  - Explicit typing,
    - Parameterized collections
    - Compatibility verified at runtime when binding
Multicast Interfaces Illustrated

- Configurable distribution policies
- Parallelism
- Strong typing
GCM Gathercast Interfaces

- Synchronization
  - ~ “join” invocations
  - Customizable: wait-for-all, wait-for-some
  - Timeout

- Data distribution
  - Aggregation / reduction of parameters
  - Redistribution of results
  - Symmetrical to multicast

list of invocations ⇒ single invocation
Gathercast Interfaces Illustrated

- Configurable distribution policies
- Synchronization
- Strong typing
Architecture Description Language (ADL)

• Specifies the system architecture
  • Components, subcomponents
  • Bindings
  • Interfaces (IDL)
• Used to configure and deploy component systems
Architecture Description Language (ADL)

- In GCM, the Fractal ADL has been extended:
  - allows to reuse ProActive-specific features like deployment
  - supports Collective Interfaces
Virtual Nodes

```xml
<virtualNodesDefinition>
  <virtualNode name="Dispatcher" property="unique_singleAO"/>
  <virtualNode name="Renderer" property="Multiple"
    constraintFile="RendererConstraints.xml"/>
</virtualNodesDefinition>
```

- Permits a program to generate automatically a deployment plan:
  - find the appropriate nodes on which processes should be launched.
Virtual Nodes in the ADL

```xml
<exportedVirtualNodes>
  <exportedVirtualNode name="VN1">
    <composedFrom>
      <composingVirtualNode component="this" name="myNode"/>
    </composedFrom>
  </exportedVirtualNode>
</exportedVirtualNodes>
...
<virtual-node name="myNode" cardinality="single"/>
```

- Renames a VN
- Exports a VN name

*final version of the GCM specification will precisely define the syntax for the virtual node definition, and their composition.*
Let's practice a little more!

http://proactive.objectweb.org
First-steps in GCM/ProActive Components

- Composite
  - Defined in ADL

- Primitive
  - Defined in ADL
  - Java class
    - implements server interfaces

- Interfaces
  - Cardinality (single or multiple) → ADL
  - Signed by Java interfaces
    - Distribution policy → Java annotations
Distribution Policy

- Given by Java annotations

```java
@ClassDispatchMetadata(
    mode=@ParamDispatchMetadata(
        mode=ParamDispatchMode.BROADCAST))

interface MyMulticastItf {
    public void foo(List<T> parameters);
}
```