Grid Programming with Higher-Order Components (HOCs)

Philipp Kegel, Jan Dünnweber, and Sergei Gorlatch
University of Münster, Germany
http://pvs.uni-muenster.de/pvs/

2nd SCIEnce Workshop
Palaiseau, France
January 20th, 2009
HOCs - Basic Facts

- Higher-Order Component (HOC)
  ≈ Higher-order functions for grid programming
- HOCs (along with the HOC Service Architecture) are an add-on for the Globus® Toolkit
- Developed at the Parallel and Distributed Systems group (Prof. Sergei Gorlatch) in Münster
  - Actively developed between 2003 and 2008 by Jan Dünnweber
- HOCs are now part of the Globus® Toolkit as an incubator project hosted by the Globus® Alliance
  (http://dev.globus.org/wiki/Incubator/HOC-SA)
Programming Grids using Grid Middleware

- **Grids** connect HPC systems
- Grid application programmers have to cope with *heterogeneous*, *dynamic* computational resources and interconnects
- **Grid middleware** (e.g., Globus) allows for multiple different resources to interoperate via a common service protocol
**Programming Grids using Grid Middleware**

- **Grids** connect HPC systems
- Grid application programmers have to cope with *heterogeneous*, *dynamic* computational resources and interconnects
- **Grid middleware** (e.g., Globus) allows for multiple different resources to interoperate via a common service protocol
Configure services and resources (extended WSDL format)
Mapping service namespaces to implementation packages
Implement services, resources and "homes" (factories)
Writing deployment configuration (WSDD)
Deploying the Grid Application Archive
Example WSDL-interface for WSRF (shortened!)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="MasterService"
    targetNamespace="http://org.gridhocs/Master"
    xmlns:wsdlpp="http://www.globus.org/namespaces/2004/10/WSDLPreprocessor"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:import location="../../wsrf/properties/WS-ResourceProperties.wsdl"/>
    <wsdl:types>
        <schema targetNamespace="http://org.gridhocs/Master"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
            <complexType name="ArrayOf_xsd_double">
                <complexContent>
                    <restriction base="soapenc:Array">
                        <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:double[]"/>
                    </restriction>
                </complexContent>
            </complexType>
            <!-- more parameter type and resource property declarations -->
        </schema>
    </wsdl:types>
    <wsdl:message name="configureRequest">
        <wsdl:part name="in0" type="impl:ArrayOf_xsd_string"/>
    </wsdl:message>
    <wsdl:message name="configureResponse">
        <part name="parameters" element="impl:void"/>
    </wsdl:message>
    <wsdl:portType name="MasterPortType"
        wsdlpp:extends="wsrpw:GetResourceProperty
        wsrlw:ImmediateResourceTermination"
        wsrp:ResourceProperties="tns:MasterResourceProperties">
        <wsdl:operation name="configure" parameterOrder="in0">
            <wsdl:input message="impl:configureRequest"/>
            <wsdl:output message="impl:configureResponse"/>
        </wsdl:operation>
        <!-- more operation declarations -->
    </wsdl:portType>
</wsdl:definitions>
```
Idea: Programming using HOCs

- Select and parameterize suitable components (HOCs) with distributed implementation in form of services + resources
- Express application by composing components

HOC: Recurring algorithmic pattern or skeletal construct

Note: Program’s parameters are programs ⇒ higher-order
HOC-SA: Service Architecture for HOCs

- HOCs must be supplied with code units
- Role of the HOC-SA: Introduces grid-compliant means for code mobility
- The substantial elements of the HOC-SA are
  - The **code service**: to upload and download code units
  - The **remote code loader**: makes code units stored by the code service available to the Grid servers.
HOC-RA: Service Architecture for HOCs

- HOCs must be supplied with code units
- Role of the HOC-RA: Introduces grid-compliant means for code mobility
- The substantial elements of the HOC-RA are
  - The **code service**: to upload and download code units
  - The **remote code loader**: makes code units stored by the code service available to the Grid servers.

```plaintext
HOCclient

store("WorkerID", ...)

HOC

remote code loader

retrieved("WorkerID")

callHOC("WorkerID", ...)
```
Distribution of Roles in the HOC-SA

HOC-SA = Separation of concerns + Distribution of roles

- **grid experts** build and deploy efficient HOCs
- **application developers** select, parameterize and combine HOCs in an application
**An Example: The Farm HOC**

- A **grid expert** provides an architecture tuned, efficient **farm implementation** as a combination of distributed services (including configuration and deployment).
- The **application developer** selects hardware independent, application specific parameters for **master and worker**.
HOC-SA Applications Implementation

- All server-side interfaces are derived from a general HOC-interface

```java
public interface HOC<E> {
    public void configureGrid(String[] availableServers);
    public void setDatasource(String source);
    public void compute(E[] input);
    public E[] getResult();
}
```

- Grid experts provide architecture-tuned implementations of HOCs
- Application programmers access components via common remote interfaces, e.g., the Farm HOC:

```java
public interface FarmHOC<E> extends HOC<E> {
    public void setMaster(int masterID);
    public void setWorker(int workerID);
}
```

- Endpoint/Resource interactions are hidden in abstract *adapter classes*
Code parameters are hardware-independent, application-specific, and implement interfaces like e.g.,

```java
public interface Worker<E> {
    public E[] compute(E[] input);
}
```

```java
public interface Master<E> {
    public E[][] split(E[] input, int numWorkers);
    public E[] join(E[][] input);
}
```
Uploading Code Parameters

- Comfortable GUI can be used to upload code parameters
Invocation of a HOC in the HOC-SA

```java
farmHOC = farmHOCFactory.createHOC(); // returns adapter
farmHOC.setMaster("masterID");
farmHOC.setWorker("workerID");
farmHOC.compute(input);
```

1. Create HOC instance
2. Select code parameters
3. Start remote processing
HOC-SA: The Big Picture

Client

local code
... HOC(A, B); ...
... code param [A] code param [B]

Web service

① upload

Code Parameters

Code Service

② request

OGSA-DAI

HOC

execute

③ download

Remote Code Loader

HPC Hardware

④
Case Study 1: Julia Sets

- Calculating fractal images is a compute-intensive task
- Procedure can be applied to multiple independent tiles
- Straightforward parallelization possible via farm pattern
Computing Julia Sets using a HOC

Implementation is done using the **Farm HOC**

1. Programmer only provides a simple implementation of a “Fractal Worker” as follows:

```java
public class FractalWorker implements Worker<int> {
    public int[] compute(int[] tile) {
        for (int y = 0; y < tileHeight; ++y) {
            for (int x = 0; x < tileWidth; ++x) {
                ... // compute julia value for (x, y)
            }
        }
    }
}
```

2. Programmer invokes the HOC in the HOC-SA

```java
farmHOC = farmHOCFactory.createHOC();
farmHOC.setMaster("masterID");
farmHOC.setWorker("workerID");
farmHOC.compute(input);
```
Case Study 2: Sequence Alignment

Common task in bioinformatics:
Align DNA or aminoacid sequences

...GAATTCA\_G\_T\_A\_T\_A\_...
| | | | | |
...GGA\_TC\_G\_A\_A\_...

Kegel (University of Münster)
Case Study 2: Sequence Alignment

- Common task in bioinformatics: Align DNA or aminoacid sequences

\[ \ldots \text{GAATTCAGTTA}\ldots \]
\[ \ldots \text{GGA\_TC\_G--A}\ldots \]

- Sequence distances are described using a similarity matrix

\[ s_{i,j} := \max \begin{pmatrix} s_{i,j-1} + \text{penalty}, \\ s_{i-1,j-1} + \delta(i,j), \\ s_{i-1,j} + \text{penalty} \end{pmatrix} \]

\[ \delta(i,j) := \begin{cases} +1 & \text{if } \epsilon_1(i) = \epsilon_2(j) \\ -1 & \text{otherwise} \end{cases} \]
Every element depends on north-, northwest- and west-neighbor
Independent elements are grouped in “wavefronts” positioned along the matrix’ antidiagonals

The actual alignment is obtained via *traceback*

The Alignment HOC covers the whole alignment process
Using a HOC for Sequence Alignment

- Every element depends on north-, northwest- and west-neighbor
- Independent elements are grouped in “wavefronts” positioned along the matrix’ antidiagonals

The actual alignment is obtained via *traceback*

The **Alignment HOC** covers the whole alignment process
The Alignment HOC

HOC

DNA Sequences

HPC Hardware (RMI Communication)

one Submatrix per Server

inside the Matrices

Scoring Function

Alignment Function

Traceback Function

Code Parameters

user-defined Code

HOC Code Parameters

Kegel (University of Münster) Higher-Order Components
Case Study 3: Deforming Virtual Clay
Using a HOC for a simplified crash test

- Virtual objects are placed within a discrete 3-dim regular grid
- Every point in the grid (voxel) has a mass value
- Code parameter specifies how mass of a specific material is exchanged

- Step-by-Step, the Deformation HOC moves objects, distributes masses, and computes mass exchange
Using a HOC for a simplified crash test

- Virtual objects are placed within a discrete 3-dim regular grid.
- Every point in the grid (voxel) has a mass value.
- Code parameter specifies how mass of a specific material is exchanged.

Step-by-Step, the **Deformation HOC** moves objects, distributes masses, and computes mass exchange.

Single Layer Transform:
Simple density distribution, repeated until no congested voxels remain.
The Deformation HOC

1 Slice per Processor

HOC

Slice 1
Slice 2
Slice 3

Deformation Routine
Correction Routine

inside the Slices

Code Parameters

case 1
\( \Psi_1 \)
\( \Psi_2 \)

case 2
\( \Psi_3 \)
\( \Psi_4 \)

case 3

case 4

user-defined Code

\( \Psi \leq \text{threshold} \)
\( \Psi > \text{threshold} \)

obstacles

Kegel (University of Münster)
Higher-Order Components
Conclusion

- HOCs provide an abstract, component-based programming model for grids
  - HOCs are architecture tuned and application independent
  - Separation of concerns
  - Code reuse
- HOC-SA enables the use of HOCs on top of the Globus Toolkit
  - Provides the means for code mobility
  - Exposes HOCs as standard Web services
How to start using HOCs

- Take a look at the HOC-SA incubator project at
  http://dev.globus.org/wiki/Incubator/HOC-SA
- Download software
  (including Client API, examples, tutorials and HOCs)
- Contact authors and contributors
- Contribute own HOCs

- Visit our website at
  http://pvs.uni-muenster.de/pvs/forschung/hoc/
A challenging class of Applications: ROIA

Real-Time Online Interactive Applications (ROIA), e.g., online games or interactive e-learning, are a challenging class of Internet services:

- huge number of concurrent users in a single application instance (e.g., more than 40,000 simultaneous participants in Eve Online),
- high update rate (5-100 updates/sec),
- short response time to actions (< 100ms),
- high-latency and low-bandwidth Internet connections,
- variable and daytime-dependent user load,
- mechanisms for trust and security.
RTF is a novel middleware technology that enables:

- **Scaling interactive real-time applications like online games through a variety of parallelization and distribution techniques.**
- **Seamless experience for services running on multiple resources.**
- **Service adaptation during runtime to a changing user demand by adding resources transparently for consumers.**
Thank you for your attention!

Any questions?