WorkWays: Interactive Workflow-based Science Gateways

Hoang Nguyen
David Abramson
Monash University, Australia

IEEE eScience 2012, Chicago
Outlines

• Background
  – Scientific Workflow
  – Science Gateways
• Motivation
• Design and Implementation
• Case study
• Conclusion & Future Work
Background

• E-Science
  – complicated process
  – involves different resources across multiple domains

• Grid middleware
  – bridge the hardware-software gap
  – powerful but rather complicated

• Scientific workflows & science gateways
Scientific Workflow (SWF)

• Automate, manage and execute various steps in scientific research

• Existing SWFMSs: Kepler, Taverna, Triana, Vistrails, etc.
  – Applied in various science domains
  – Unifying platform to build arbitrarily complicated computational applications by combining predefined components
Science Gateways

• A Web portal interfaced to a community-set of tools, applications, and data collection integrated

• Benefits
  – Accessibility
  – Usability
  – Sharing and collaborating

• LEAD, nanoHub, CAMERA
Science Gateways

• Computations behind the gateway
  – Single packages
    • nanoHub, GridChem
  – Workflows
    • LEAD, CAMERA, Crowdlabs

• Workflow-based gateways are more extendable than single applications/packages-based gateways
Outlines

• Background
  – Scientific Workflow
  – Science Gateways

• Motivation

• Design and Implementation

• Case study

• Conclusion & Future Work
Motivation

• Support man-in-the-loop workflows
  – Ability to perform IO operations with a continuously running workflow

• Different from most existing workflow-based gateways
  – execute the workflow in batch mode
  – users provide inputs, execute, and collect outputs

• Motivating example: virtual microscopy study
Motivation: Virtual Microscopy

• Based on Kepler scientific workflow
  – Capture images from Leica con-focal microscope
  – Process images
  – Visualize and archive processed images

• One application of the system is cancer research
  – Biologists study the effect of different drugs on tumor cells
  – Typically they only interested in a small area
  – Repeat scans to find the interested area
Motivation: Virtual Microscopy

1: Scan Images
2: Process Images
3: Display Images/ Acquire inputs
4: Further processing/ Storing results

Continue

True
False
Motivation: Virtual Microscopy

• Two approaches to implement the use case
• First approach
  – Construct acyclic workflow
  – IO is done before and after workflow execution
  – Need to run the workflow multiple times with different scan positions
Motivation: Virtual Microscopy

1: Scan Images
2: Process Images
3: Display Images
4: Further processing/Storing results

N times
Motivation: Virtual Microscopy

• Two approaches to implement the motivating example
• First approach
• Second approach
  – Consider a workflow as a continuously running service
  – Once initiated, the workflow pauses itself waiting for inputs
  – IO are performed while the workflow is running
  – Can be achieved by introducing a feedback loop
Virtual Microscopy

1: Scan Images
2: Process Images
3: Display Images/ Acquire inputs
   continue
   True
   False
4: Further processing/ Storing results

User is part of the execution
Outlines

• Background
  – Scientific Workflow
  – Science Gateways

• Motivation

• Design and Implementation

• Case study

• Conclusion & Future Work
WorkWays

• Inspired by the second approach

• Requirements:
  – Dynamic IO
  – Workflows as services
  – Extensibility
    • Support different IO mechanisms
    • Support different UI clients

• Kepler as the workflow engine
  – Its unique actor-oriented modeling approach
WorkWays

• Dynamic IO
  – Set of IO actors interacting with the portal
  – These IO actors allow data coming in/out of the running workflow

• Workflows as services

• Extensibility
WorkWays

• Dynamic IO
• Workflows as services
  – Workflows need to be run continuously, driven by user inputs
  – In Kepler, workflows with feedback loops and blocking IO under PN director
• Extensibility
WorkWays

• Dynamic IO
• Workflows as services
• Extensibility
  – Avoid contents coupling between IO actors and UI clients
  – Support different protocols
  – Connection topology
WorkWays

Web Portal
- Provenance Viewer portlet
- Workflow launch portlet
- IO portlet
- IO Library
- Other interface
  - IO Library

Provenance data
- Launch workflows

Provenance
- Workflow Run Engine
- Kepler Workflow Engine
- IO Actors
- IO Director

IO Library

New Component

Incomplete components

Existing component
Implementation: IO framework

• CometD framework
  – Framework for asynchronous message passing using either HTTP or web socket
  – Messages are routed via different channels
  – Publish-subscribe is the default routing mechanisms

• IO Server

• IO Clients: IO actors & UI clients
Implementation: IO framework

• CometD framework
• IO Server
  – Assigns each (data) channel to an IO actor
  – Multiple UI clients may subscribe to a channel
  – Reserves two special (meta) channels
    • “/publishers”
    • “/subscribers”
• IO Clients
Implementation: IO framework

- CometD framework
- IO Server
- IO Clients
  - IO actors: publishers
  - UI clients (Web client): subscribers
  - Handshaking between IO actors and UI clients are done via the two meta channels
    - IOPortlet
    - IOActor
**IOActor**

- Generic actor
  - Simplify the creation of (new) IO actors
  - Instantiate an *IOActor* & provide the actor definition

- IOActor definition
  - Actor name
  - Number of (supported) clients
  - Operation: input/output/inout
  - Additional information
IOActor

{ 
actor: "ImageInOutActor"
{
operation: "in/out"
input_type: "text"
prompt: "Choose an area in the image"
output_type: "binary"
display_type: "image"
action: "subarea"
wait_client: true
wait_for_input: true
}
}

IOActor declaration

Actor name: "ImageInOutActor"
Operation type: in/out/inout
One operation per actor
## IOActor

<table>
<thead>
<tr>
<th>Tag</th>
<th>Meanings</th>
<th>Possible Values</th>
<th>Applied to</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>type of the port</td>
<td>input/output/inout</td>
<td>all</td>
</tr>
<tr>
<td>input_type</td>
<td>type of input expected</td>
<td>Primitive types/text</td>
<td>input/inout</td>
</tr>
<tr>
<td>prompt</td>
<td>prompt string</td>
<td>String</td>
<td>input/inout</td>
</tr>
<tr>
<td>output_type</td>
<td>type of output</td>
<td>text/binary</td>
<td>output/inout</td>
</tr>
<tr>
<td>display_type</td>
<td>type of output display</td>
<td>text/image</td>
<td>output/inout</td>
</tr>
<tr>
<td>action</td>
<td>action to perform on output display to generate input</td>
<td>zoom</td>
<td>inout</td>
</tr>
<tr>
<td>group</td>
<td>group a port with other port with the same type.</td>
<td>other actor’s port /“all”</td>
<td>input/output</td>
</tr>
<tr>
<td>wait_client</td>
<td>blocking until a client connected</td>
<td>boolean</td>
<td>output/inout</td>
</tr>
<tr>
<td>wait_for_input</td>
<td>no output until input available</td>
<td>boolean</td>
<td>inout</td>
</tr>
</tbody>
</table>

**Supported declarative tags**
**IOPortlet**

- Web UI client
- Vaadin framework
  - framework for building rich Web applications
- JSR-286 portlet
  - UI elements generated based on requests from connected **IOActor**
  - Limited UI elements
Outlines

• Background
  – Scientific Workflow
  – Science Gateways
• Motivation
• Design and Implementation
• Case study
• Conclusion & Future Work
Case Study
Case Study: Scan Workflow
Case Study: **CombineChannels** actor

**input:** path of the folder contains images  
**output:** path of the combined image @channel0 and 1
Case Study: Scan Workflow
Case Study: Scan Workflow
Case Study: Scan Workflow

Display/Inputs

PN Director
CombineChannels
WSStartScan
SampleDelay

ImageInOutActor
inputs
Boolean Switch
Expression: input = "0"

DDF Boolean Select2
SegmentAndSkeleton
ImageOutputActor

Interaction Interface

Zoom Done
Case Study: Scan Workflow
Outlines

• Background
  – Scientific Workflow
  – Science Gateways
• Motivation
• Design and Implementation
• Case study
• Conclusion & Future Work
Future Work

• Support more objects
• Support asynchronous IO
  – Enhancement of Kepler directors
• Support different UI client(s)
  – Tiled display wall
• Multi-clients interacting with one workflow
QUESTIONS