DECENTRALIZED ORCHESTRATION OF DATA-CENTRIC WORKFLOWS USING THE OBJECT MODELING SYSTEM

Bahman Javadi
School of Computing, Engineering and Mathematics
University of Western Sydney, Australia

Martin Tomko and Richard O. Sinnott
The University of Melbourne, Australia

The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing
AGENDA

- Introduction
- Object Modeling System (OMS)
- AURIN Project
- OMS-based Workflows
- OMS Service Orchestrations
- Experimental Results
- Conclusions
INTRODUCTION

- Service-oriented Architecture
  - Web services

- Workflow Technologies
  - Coordinate a collection of services

- Workflow implementation approaches
  - Service Orchestration
    - Centralized engine → bottleneck for data-centric workflows
  - Service Choreography
    - Distributed control

- Goal: a new framework to implement data-centric workflows based on Object Modeling System (OMS)
OBJECT MODELING SYSTEM (OMS)

- A framework to implement science model
  - Object oriented (component-based)
  - Pure Java
  - Last version: OMS 3.0

- Main features
  - Non-invasive
    - Annotation of existing languages
  - Multi-threading
    - Able to be deployed on multi-core Cluster/Cloud
  - Domain Specific Language (DSL)
    - Groovy language
COMPONENTS IN OMS

- Components
  - PJO + annotation
- Annotations
  - @In
  - @Out
  - @Execute
  - ....
- Multi-purpose components
- Automatic manual generation

Listing 1: A sample OMS3 component

```java
package oms.components;
import oms3.annotations.*;

@Description("Average of a given vector.")
@Author(name = "Bahman Javadi")
@Keywords("Statictic, Average")
@Status(Status.CERTIFIED)
@Name("average")
@License("General Public License Version 3 (GPLv3)")

public class AverageVector {
    @Description("The input vector.")
    @In
    public List<Double> inVec = null;

    @Description("The average of the given vector.")
    @Out
    public Double outAvg = null;

    @Execute
    public void process() {
        Double sum;
        int c;
        sum = 0.0;
        for (c = 0; c < inVec.size(); c++)
            sum = sum + inVec.get(c);
        outAvg = sum / inVec.size();
    }
}
```
Workflow/Model Template in OMS

- **Components**: declaration of all components
- **Parameters**: input parameters
- **Connect**: connection of components

Listing 2: Model/Workflow template in OMS3

```java
// creation of the simulation object
sim = new oms3.SimBuilder(logging: 'OFF').sim(name: 'test') {
   // the model space
   model {
      // space for the definition of the required components
      components {
      }
      // initialization of the parameters
      parameter {
      }
      // connection of the different components
      connect {
      }
   }
   // start of the simulation to obtain the results
   results = sim.run();
}
```
AURIN PROJECT

- Australian Urban Research Infrastructure Network (AURIN)
  - National e-Research Project (2010-2014)
  - An e-Infrastructure supporting research in urban and built environment research disciplines
  - Web Portal Application (portlet-based)
    - A lab in a browser
    - Data discovery
    - Data visualization (Mapping service)
    - Access to the federated data source
      - Web Feature Service (WFS)
    - NeCTAR NSP and Research Cloud
    - RDSI Storage
THE AURIN ARCHITECTURE
OMS-based Workflows

- Annotation of existing code
  - Embedded metadata using annotations
  - Attached metadata using annotations (e.g., XML file)

- Basic Components
  - Web Feature Service (WFS) Client
  - Statistical Data and Metadata eXchange (SDMX) Client
  - Basic statistical functions

- Workflow Composition
  - A standalone portlet
  - Save a workflow through web portal
    - Save as an OMS script
OMS-based Workflows

- Workflow in the AURIN portal
OMS WORKFLOW WITH ONE WFS CLIENT

- WFS client example
  - Dataset: Landgate WA
  - Bounding box (bbox): geographical area
- DSL makes the workflow very descriptive

### Listing 2: An OMS workflow with one WFS client

```java
// this is an example for a wfs query
def simulation = new oms3.SimBuilder(logging:'ALL').sim(name:'wfs_test') {

    model {
        components {
            'wfsclient0' 'wfsclient'
        }

        parameter {
            'wfsclient0.datasetName' 'ABS-078'
            'wfsclient0.wfsPrefix' 'slip'
            'wfsclient0.datasetReference' 'Landgate, ABS'
            'wfsclient0.datasetKeyName' 'ssc_code'
            'wfsclient0.datasetSelectedAttributes' 'ssc_code, employed_fulltime, employed_parttime'
            'wfsclient0.bbox' '129.001336896,-38.0626029895,141.002955616,-25.99614648750003'
        }

        connect {
        }
    }

    result = simulation.run();
}
```
OMS Service Orchestration

- Workflow Enactment
  - Running OMS scripts by the OMS3 engine
  - Centralized service orchestration
OMS SERVICE ORCHESTRATION

- Take advantage of the OMS3 architecture
  - Flexible and lightweight (CLI for the OM3 core)
  - Decentralized service orchestration
CLOUD-BASED EXECUTION

- OMS3 Features
  - Supports component-level parallelism
  - Terracotta for distributed shared memory systems
  - Run on any Cluster and IaaS Cloud

- Developed Interfaces
  - NeCTAR Research Cloud
    - Small Instance: 1-core, 4GB RAM
    - Medium Instance: 2-core, 8GB RAM
    - Extra-Large Instance: 8-core, 32GB RAM
  - Amazon’s EC2
Experimental Setup

- AURIN Portal is deployed in NeCTAR NSP (4 VMs)
- Real workflow for typical urban analysis
  - Create topological spatial relationship
  - Relation: touch
  - Output: a topology graph shows the adjacencies of suburbs/LGA
- Input datasets

<table>
<thead>
<tr>
<th>State</th>
<th>No. of Geometries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Suburbs</td>
</tr>
<tr>
<td>Western Australia (WA)</td>
<td>952</td>
</tr>
<tr>
<td>South Australia (SA)</td>
<td>946</td>
</tr>
<tr>
<td>Tasmania (TAS)</td>
<td>402</td>
</tr>
<tr>
<td>Queensland (QLD)</td>
<td>2112</td>
</tr>
<tr>
<td>Victoria (VIC)</td>
<td>1833</td>
</tr>
<tr>
<td>New South Wales (NSW)</td>
<td>3146</td>
</tr>
</tbody>
</table>

TABLE II: Workflows for the experiments.
Experimental Setup

- Data-size for workflows
  - Data-centric Workflows

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Data size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Geometries</td>
</tr>
<tr>
<td>WA</td>
<td>33.02</td>
</tr>
<tr>
<td>WA, SA</td>
<td>66.44</td>
</tr>
<tr>
<td>WA, SA, TAS</td>
<td>119.75</td>
</tr>
<tr>
<td>WA, SA, TAS, QLD</td>
<td>170.35</td>
</tr>
<tr>
<td>WA, SA, TAS, QLD, VIC</td>
<td>244.97</td>
</tr>
<tr>
<td>WA, SA, TAS, QLD, VIC, NSW</td>
<td>399.04</td>
</tr>
</tbody>
</table>
RESULTS

- Execution time of Workflows on NeCTAR Cloud
  - Extra-Large Instance 8-core, 32GB RAM
RESULTS

- Execution time of Workflows on Amazon’s EC2
  - Hi-CPU Extra-Large instances 8-core, 17GB RAM
  - ap-southeast region (Singapore)
RESULTS

- Average performance improvement
CONCLUSIONS

- A new framework to implement data-centric workflows based on OMS
- Using decentralized service orchestration to bypass the bottleneck of centralized engine
- Substantially improvement the performance of data-centric workflows,
  - 20% on NeCTAR
  - 100% on EC2
- Future Work
  - Automate provisioning of Cloud resources for OMS-based workflows
Thank You