Enhancing the performance of scientific workflow execution in e-Science environments by harnessing the standards based Parameter Sweep Model

23. July 2013 | Shahbaz Memon, Sonja Holl, Bernd Schuller, Andrew Grimshaw
Outline

- Introduction
- Taverna Workflow System
- UNICORE
- Motivation: Advanced Workflow Optimization
- JSDL-Parameter Sweep
- UNICORE Parameter Sweep Implementation
- Conclusions
Introduction

- Scientific workflows are extensively being used for defining and generating computer simulations
- Same application need rerun with different parameters – e.g. verifying and validating simulation models
- Parameter sweep concept can be used to represent an abstract job with iterative parameters
- An improved workflow optimization use case through the use of parameter sweep concept
- UNICORE is used as a Job submission and management middleware
Taverna Workflow System

- Open source written in Java
- Graphical Interface: Workbench
- Execution: Taverna Workflow Engine
- Data Flow Driven
- Activities represent Web services, Java classes, local scripts
- UNICORE plugin was developed in 2010
Integrated Architecture

Client

Server Tier

Target System Tier

Mitglied der Helmholtz-Gemeinschaft

23. Juli 2013
Client and Server software
Seamless and secure access to HPC resources
Based on open standards: X.509, SAML, XACML, WS-*
Service Oriented Architecture (SOA) and Web Services Resource Framework (WS-RF) compliant
Support Grid Standards (JSDL, OGSA-BES)
Use Case: Advanced Workflow Optimization

- The choice of parameters has a tremendous impact on overall scientific experimentation – performed via workflows
- Choose the best parameters instead of default ones
  - Default parameters ➔ Weak scientific results
- Proof of concept: Taverna – UNICORE plugin has been implemented to optimize workflow parameters by applying Genetic Algorithms
- Sub-workflows are considered for the optimization
- Several instances are executed in parallel
  - Each instance with one parameter sample
  - The resultant values are taken as fitness
Job Generation and Submission

Taverna

\[ a_{1} \ldots a_{n} b_{1} \ldots b_{n} \]

UNICORE Server

\[ \text{UNICORE/X} \quad \text{Service Orchestrator} \quad \text{XUUDB} \]

Target System
Limitations

- Client manually submits N job (same application) requests, with only difference in application parameters

- Comes with an additional overhead,
  - Client responsible of managing each sweep manually
  - N job status remote call outs
  - Data staging-in for each job
  - Waste of network resources
Parametric Job Generation and Submission

Taverna

UNICORE Server

Target System

UNICORE Job Instance

a) b) c) Sweep Generator

d) JSDL-PS Instance

UNICORE/X Service Orchestrator XUUDB
JSDL Parameter Sweep Extensions

http://forge.gridforum.org/sf/projects/jsdl-wg

1. Extension of Job Submission and Description Language (JSDL) specification [GFD.136]

2. A common requirement to select a job and submit it ‘10, 50, 300’ times, each time making some modifications to the ‘original/master’ JSDL (e.g. args, parameters, output dir, input file whatever…).

3. The JSDL + PS extensions allows you to group the master JSDL + the required modifications (which JSDL fields require sweeping);
   - Saves writing multiple separate JSDL docs.
   - Can be any value within the JSDL document itself,
   - Can be any value within a named file that is referenced by the JSDL (e.g. an input file).
   - Actually yields multiple separate jobs (rather than solely parameter sweeps).

Slide Courtesy: Geoff Williams et al.
JSDL Sweep Overview

1. Nest <Sweep> elements within a JSDL doc.

2. The <Assignment> identifies which set of <Parameters> should be swept/iterated using the given sweep <Function>.

3. <Parameter> + <Function> are abstract (can define different implementations as required).

4. Parameters:
   - DocumentNode
   - FileSweep

5. Functions:
   - Values, LoopInteger, LoopDouble

Slide Courtesy: Geoff Williams et al.
JSDL-PS Example

```xml
<jsdl:JobDefinition>
  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl:POSIXApplication>
        <jsdl1:Output>stdout</jsdl1:Output>
        <jsdl1:Environment>workflow</jsdl1:Environment>
        <jsdl1:Environment>results</jsdl1:Environment>
        <jsdl1:Environment>PRM</jsdl1:Environment>
      </jsdl:POSIXApplication>
      <jsdl:DataStaging>
      </jsdl:DataStaging>
      <jsdl:DataStaging>
    </jsdl:Application>
    <swe:Sweep>
      <swe:Assignment>
        <swe:DocumentNode>
          <swe:NamespaceBinding/>
        </swe:DocumentNode>
        <fun:Values>
          <fun:Value>157</fun:Value>
          <fun:Value>186</fun:Value>
        </fun:Values>
      </swe:Assignment>
    </swe:Sweep>
  </jsdl:JobDescription>
</jsdl:JobDefinition>
```
UNICORE Parameter Sweep Implementation

- Job execution service consume JSDL-PS request and pass it to the XNJS

- XNJS parses JSDL and forwards request to the Target System Interface (TSI)

- TSI translates the incoming XNJS message to batch system specific commands
XNJS Execution Management System

- Separated from core (Web) services tier
- Manage life cycle of atomic jobs (stage-in, execute, stage-out)
- Perform JSDL parsing
- Validation against resource and job requirements
- Manage user access to jobs
  - submit, start, stop
- Support file spaces (Job Working Directory, HOME, ROOT)
XNJS Sweep Job Handling

UNICORE
core services

[Diagram]

- UNICORE core services
- Local RMS (e.g., Torque, LL, LSF, etc.)
- Target System Interface
  - Local RMS

Job Submission

JSDLProcessor

Contains Sweep

Yes

SweepProcessor

Sweep Library

Resolve JSDL-PS instances

No

Simple JSDL

Submit generated JSDLs

Local RMS

qsub Job Script

TSI Submission

Simple JSDL Processing
XNJS Job Working Directory Structure

- Parent Job Working Directory is used for file staging-in
  - Files are staged-in once to the parent job WD
  - Soft links of the each staged-in file is created in the child job working directories
Client Performance Impact

Conventional Submission

Sweep Submission
Conventional vs Sweep

<table>
<thead>
<tr>
<th></th>
<th>File upload (small example)</th>
<th>File upload (large example)</th>
<th>Average CPU load on the client (%)</th>
<th>Max. CPU load on the client (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional Submission</td>
<td>400 kB</td>
<td>700 MB</td>
<td>12.89</td>
<td>360</td>
</tr>
<tr>
<td>Sweep Generator</td>
<td>20 kB</td>
<td>35 MB</td>
<td>7.79</td>
<td>244</td>
</tr>
</tbody>
</table>
Conclusions

- Parametric job run is optimized by submitting one remote job in lieu of hundreds or thousands
- JSDL-PS implementation will benefit wide range of applications which are relying on workflow optimization
- Taverna UNICORE plugin is extended to use the server side parameter sweep capabilities
- Future Directions
  - More intuitive file staging-ins
  - Combining standard out / err
  - Meaningful job status reports

23. Juli 2013
Questions?