Collaboratively Building Reusable Job Configurations for HPC

Jeremy Cohen
London e-Science Centre, Department of Computing, Imperial College London
jeremy.cohen@imperial.ac.uk

With thanks to:
John Darlington, Chris Cantwell, David Moxey, Spencer Sherwin & Jeremy Nowell

PRISM Seminar, Thursday 26th March 2015
Reusable Job Configurations

- Provide a **high-level** approach for end-users to **configure** their jobs

- Address **complexity** of **configuration files** that stems from complex **software methods** and **heterogeneous hardware**

- Improve **usability** of **software methods** and their availability to **scientists/researchers** not from a computationally focused background
Integrated e-Infrastructure

- End-users
- Computer Scientists
- e-Infrastructure Operators
- Hardware Platforms & Providers
- Method Developers
A Decoupled e-Infrastructure

End-users

Method Developers

E-Infrastructure

e-Infrastructure Operators

Computer Scientists

Hardware Platforms & Providers
Abstraction versus Efficiency
A Fundamental and Long-standing Problem

Simplicity and Ease of Use

Abstraction  ?  Complexity

Efficiency

Slide: J. Darlington, Imperial College London
Abstractions

Coordination Forms = Control Abstraction
Higher-order, functions as arguments

Abstract Components = Data Processing Abstractions
First-order, data as arguments

• Allows automated selection of optimal implementations

• Metadata plays a key role in enabling abstract => concrete mapping – maintains information
Coordination Forms

- A functional/mathematical approach to job specification
- Referentially transparent, Church-Rosser property
- Based on work by Darlington, et al.


- Can have multiple implementations – e.g. sequential/parallel
- Compositions of coordination forms can be used to describe application flow
Alternative implementations: Coordination Forms

Co-ordination Form: PAR

A co-ordination form that specifies the execution of one or more independent components/tasks.

**Inputs:**
- I1) A list of the tasks to be executed
- I2) A list containing the inputs for each of the tasks in (I1).

**Outputs:**
- O1) A list containing the outputs for each of the tasks from (I1)

---

**PAR Implementation**

**Sequential**
Undertakes each of the tasks listed in turn.

**Parallel**
Undertakes all of the tasks listed concurrently, spawning a new process for each task.

**Cloud**
Undertakes each of the tasks listed concurrently, provisioning a dedicated cloud server for each task.
Software Components

- Granularity varies
  - **Fine-grained**: small libraries, individual functions, command-line tools
  - **Coarse-grained**: Whole application!
- **Abstract** – metadata wrapper, no implementation
- **Concrete** – Runnable component, metadata + implementation
- Components can have **multiple implementations**
Alternative implementations: Components
Libhpc Projects

• Libhpc 2 runs to end October 2015
• Builds on Libhpc 1 which ran from July 11 -> Jun 13
• Developing framework model and a range of associated tools, services and demonstrators

• Imperial College London
  • Dept of Computing (LeSC/SCG)
  • Dept of Aeronautics
  • CISBIO / Bioinformatics Support Service
  • Epidemiology, School of Public Health

• University of Edinburgh
  • Edinburgh Parallel Computing Centre (EPCC)
Libhpc Architecture

User Interface

Libhpc Mapper

Libhpc Deployment Services

Hardware Metadata Repository

Hardware Metadata

Batch Cluster (e.g. PBS-based)

Public Cloud (e.g. Amazon EC2)

Private Cloud (e.g. OpenStack)

Standalone Local Resources

CF Repository

FARM
PIPE
PAR

Co-ordination Forms + Metadata

Software Component & Metadata Repository
Component + Metadata
Component + Metadata
Templates and Profiles
Libhpc software parameter templates

- Represent an application’s possible configuration parameters/decisions
- **Tree structure** with semantic parameter grouping
- Defined using **XML Schema**
- Does not contain values for any of the specified parameters
- Includes validation and documentation metadata
Templates & Profiles

• Libhpc profiles
  • Provides an instantiation of a template’s parameters
  • XML document – profile structure can be validated against template

• May be:
  • **Partial**: contains a subset of the required values from template
  • **Complete**: Contains a full set of required values and can be used to run a job
## Profiles

<table>
<thead>
<tr>
<th>Name</th>
<th>Solver</th>
<th>Valid</th>
<th>Editable</th>
<th>Created by</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default profile - Cardiac Electrophysiology</td>
<td>CES</td>
<td>✓</td>
<td>✓</td>
<td>jhc02</td>
<td>25 Jul 2014, 4:30 p.m.</td>
</tr>
<tr>
<td>Default profile - Incompressible Navier Stokes</td>
<td>INS</td>
<td>✓</td>
<td>✓</td>
<td>jhc02</td>
<td>12 Aug 2014, 2:25 p.m.</td>
</tr>
<tr>
<td>test profile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 Oct 2014, 11:41 p.m.</td>
</tr>
</tbody>
</table>

---

```xml
<?xml version="1.0" encoding="utf-8"?>
<IncompressibleNavierStokes>
  <Physics>
    <KinematicViscosity>1</KinematicViscosity>
  </Physics>
  <ProblemSpecification>
    <SolutionMethod>VelocityCorrectionScheme</SolutionMethod>
    <EvolutionOperator>Adjoint</EvolutionOperator>
    <Geometry>CylinderGeometry.xml</Geometry>
    <InitialConditions>
      <Constant>-81</Constant>
    </InitialConditions>
    <Expansion>
      <PolynomialOrder>7</PolynomialOrder>
      <BasisType>MODIFIED</BasisType>
    </Expansion>
  </ProblemSpecification>
  <NumericalAlgorithm>
    <Projection>ContinuousGalerkin</Projection>
  </NumericalAlgorithm>
</IncompressibleNavierStokes>
```
Templates & Profiles

• Templates defined and built by developers / domain experts

• Partial profiles may be saved; extended by different entities

• End-users may be provided with an almost complete profile and then finalise this to run their job(s)

• Helps to decouple interactions required for configuration of complex applications for heterogeneous resources
Examples and Demos
Bioinformatics: Genome Read Pre-Processing/Mapping

Input files –
Reference Genome – FASTA file
Reads from sequencing machine - FASTQ

\((sr_1, sr_2), u) = \text{PAR}([\text{fastq} \_\text{split}, \text{bwa} \_\text{index}],
\quad [(\text{short} \_\text{read} \_\text{file}, \text{None}, \text{None}), (\text{ref} \_\text{genome} \_\text{file}),])\)

\((v, w) = \text{PAR}([\text{bwa} \_\text{aln}, \text{bwa} \_\text{aln}],
\quad [(\text{ref} \_\text{genome} \_\text{file}, sr_1, \text{None}),
\quad (\text{ref} \_\text{genome} \_\text{file}, sr_2, \text{None})])\)

result = \text{PIPE}([\text{samtools} \_\text{index}, \text{samtools} \_\text{sort},
\quad (\text{samtools} \_\text{import}, \text{ref} \_\text{genome} \_\text{file}),
\quad \text{bwa} \_\text{sampe}],
\quad [\text{ref} \_\text{genome} \_\text{file}, [v, w], [sr_1, sr_2], \text{None}])
Nekkloud: Simplifying Access to Nektar++

For more info see: J. Cohen, D. Moxey, C. Cantwell, et al., "Nekkloud: A software environment for high-order finite element analysis on clusters and clouds," IEEE Cluster 2013, Sep 2013, Indianapolis, IN, USA. DOI: 10.1109/CLUSTER.2013.6702616
Molecular Dynamics: GROMACS

- GROMACS is a high performance molecular dynamics package providing a range of MD algorithms – http://www.gromacs.org
- Ideal example of an application that includes both tightly coupled parallel processes but also a higher-level pipeline of tools
Nekkloud Demo
Thank You

Questions?

Acknowledgements:

• Nektar++ team – http://www.nektar.info/wiki/Latest/Team – including Chris Cantwell, David Moxey and Spencer Sherwin
• London e-Science Centre – John Darlington, Peter Austing
• EPCC – Jeremy Nowell, Xu Guo;
• Bioinformatics Support Service, Imperial College London – Sarah Butcher, James Abbott and Filippo Mortari
• Additional thanks to members of the above groups who have provided content for this presentation