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

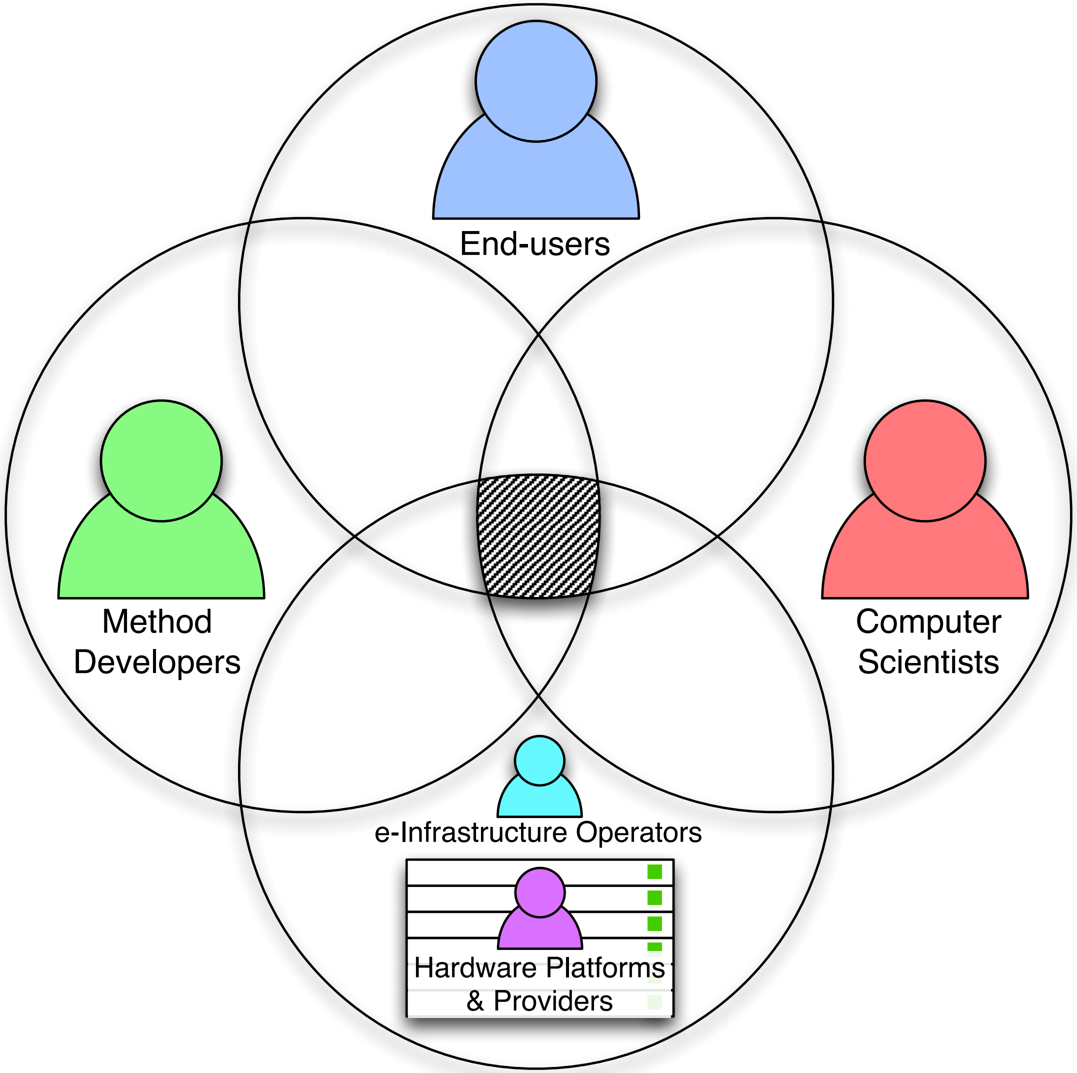
PRISM Seminar, Thursday 26th March 2015

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

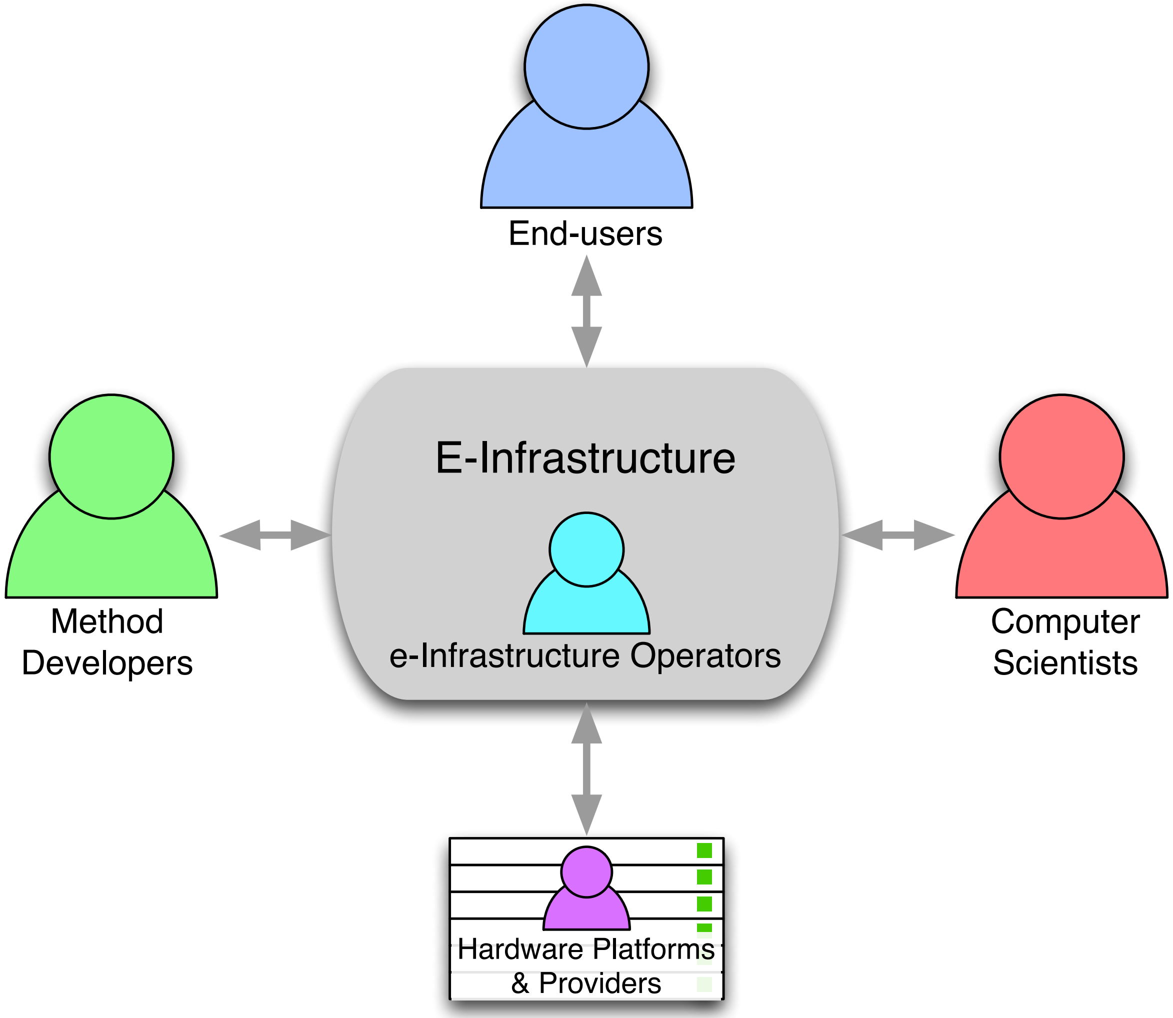
Imperial College
London

Integrated e-Infrastructure

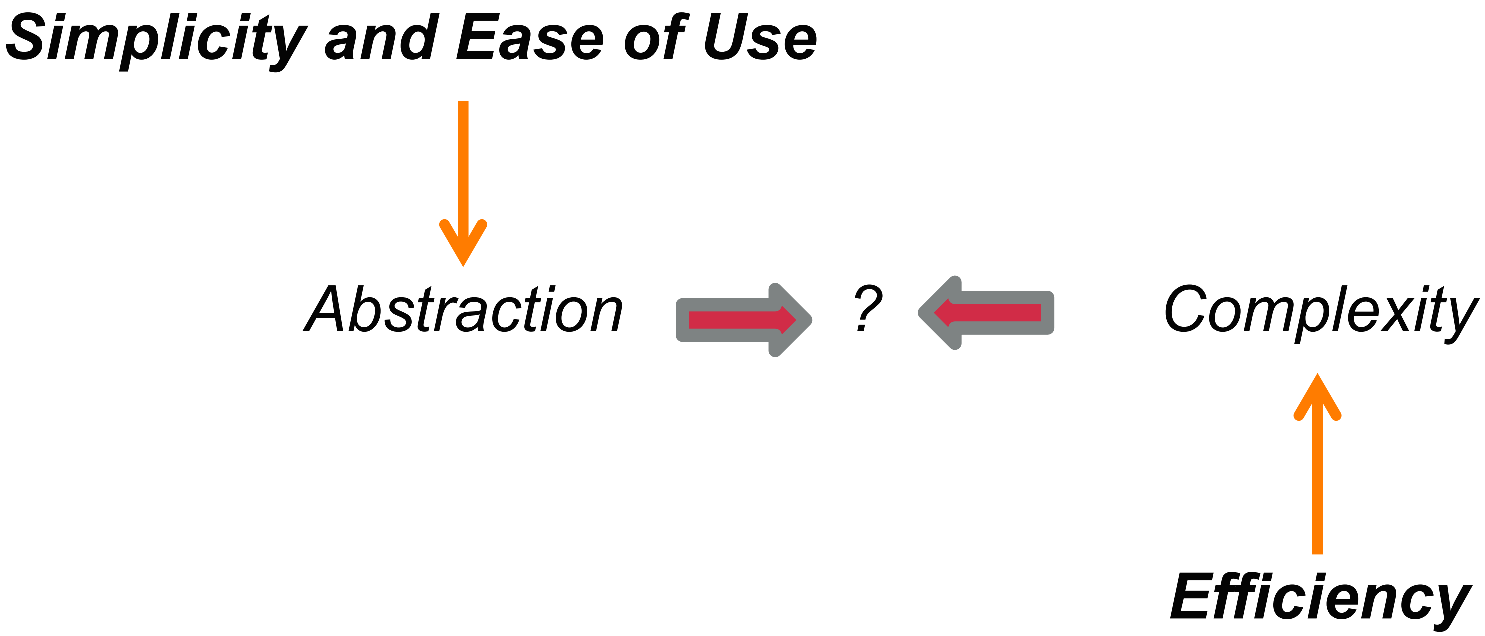


Imperial College
London

A Decoupled e-Infrastructure



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

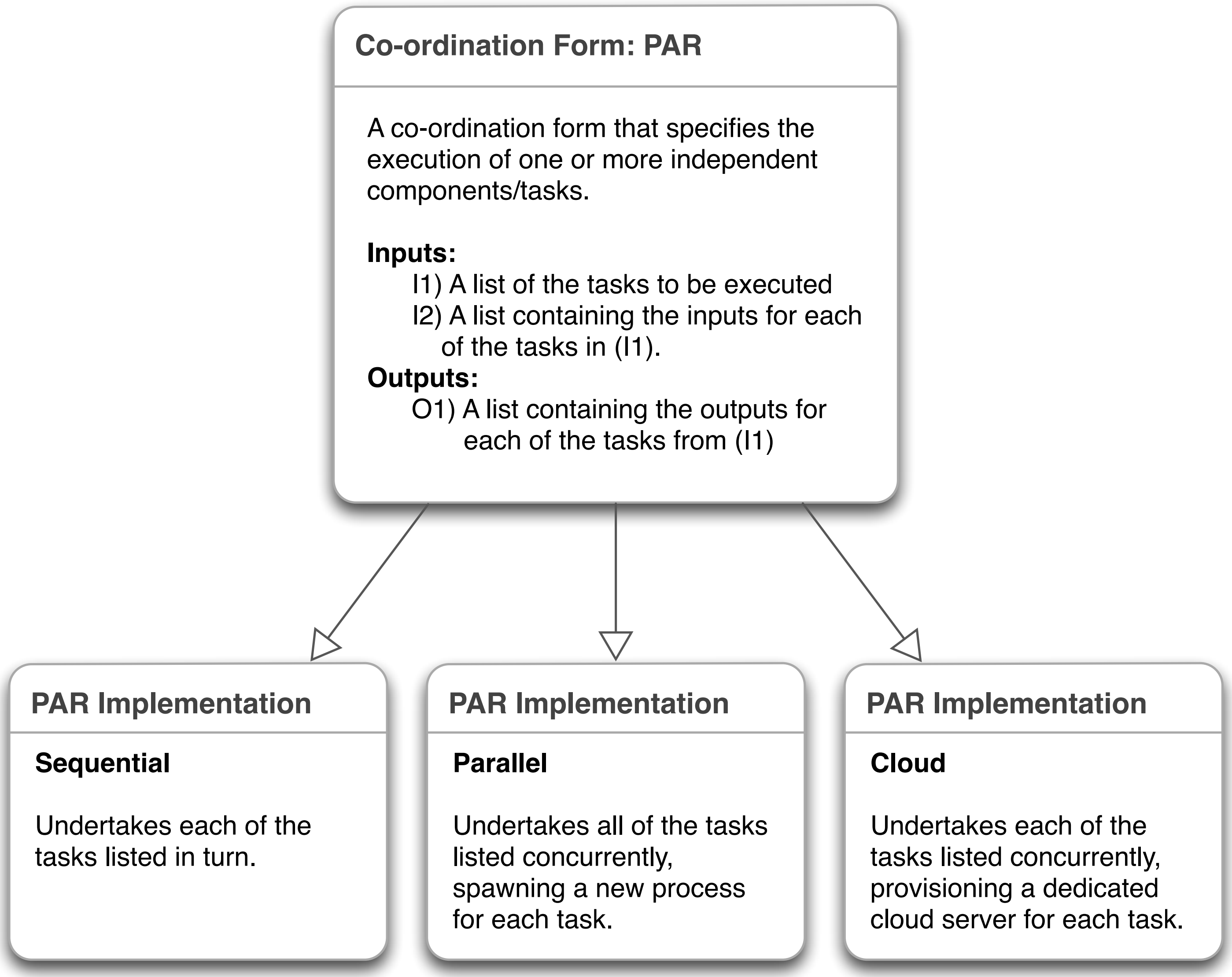
Coordination Forms

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

J. Darlington, Y. Guo, H. W. To and J. Yang. Functional skeletons for parallel coordination. In proceedings of EURO-PAR '95 Parallel Processing, LNCS 966/1995, p. 55-66, 1995. Springer Berlin/Heidelberg

- 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.

PAR Implementation

Parallel

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

PAR Implementation

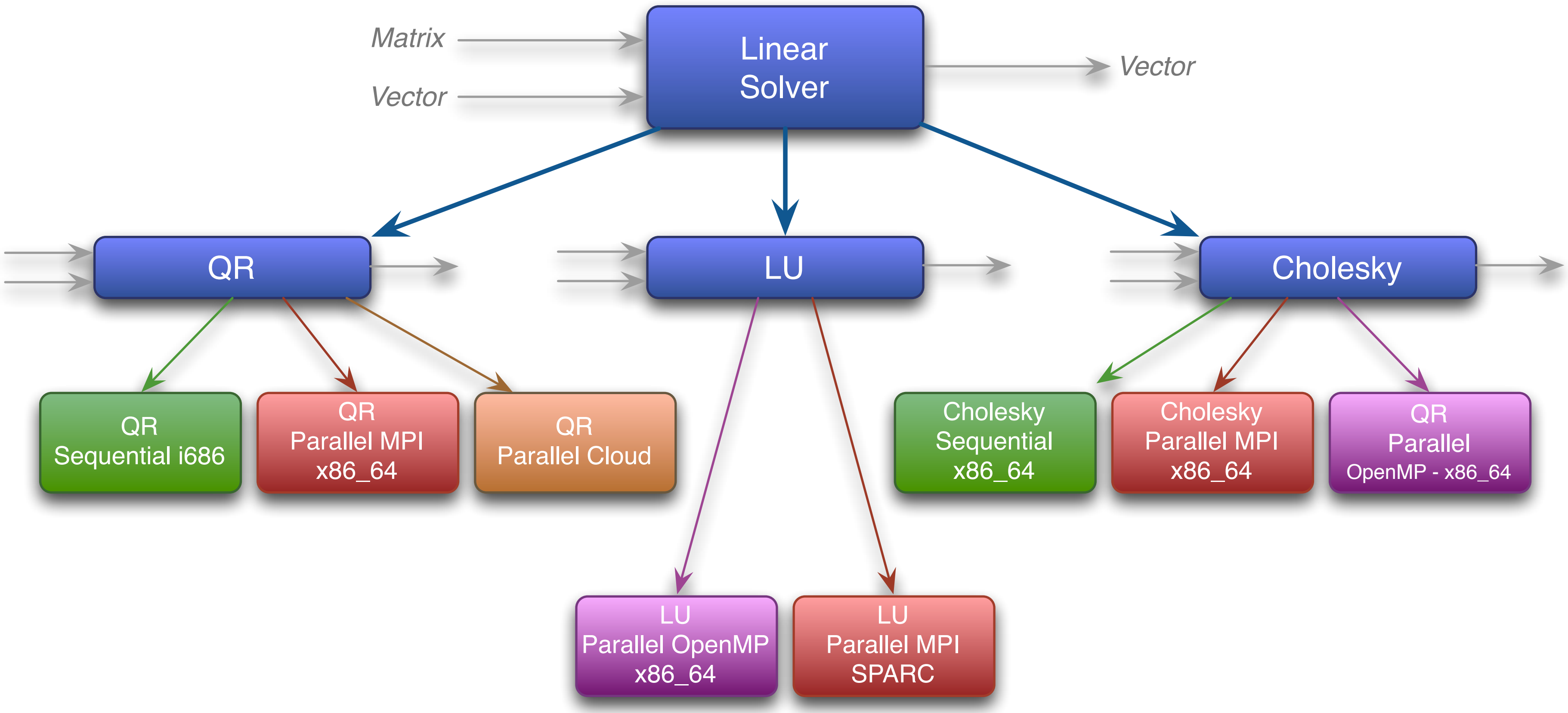
Cloud

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

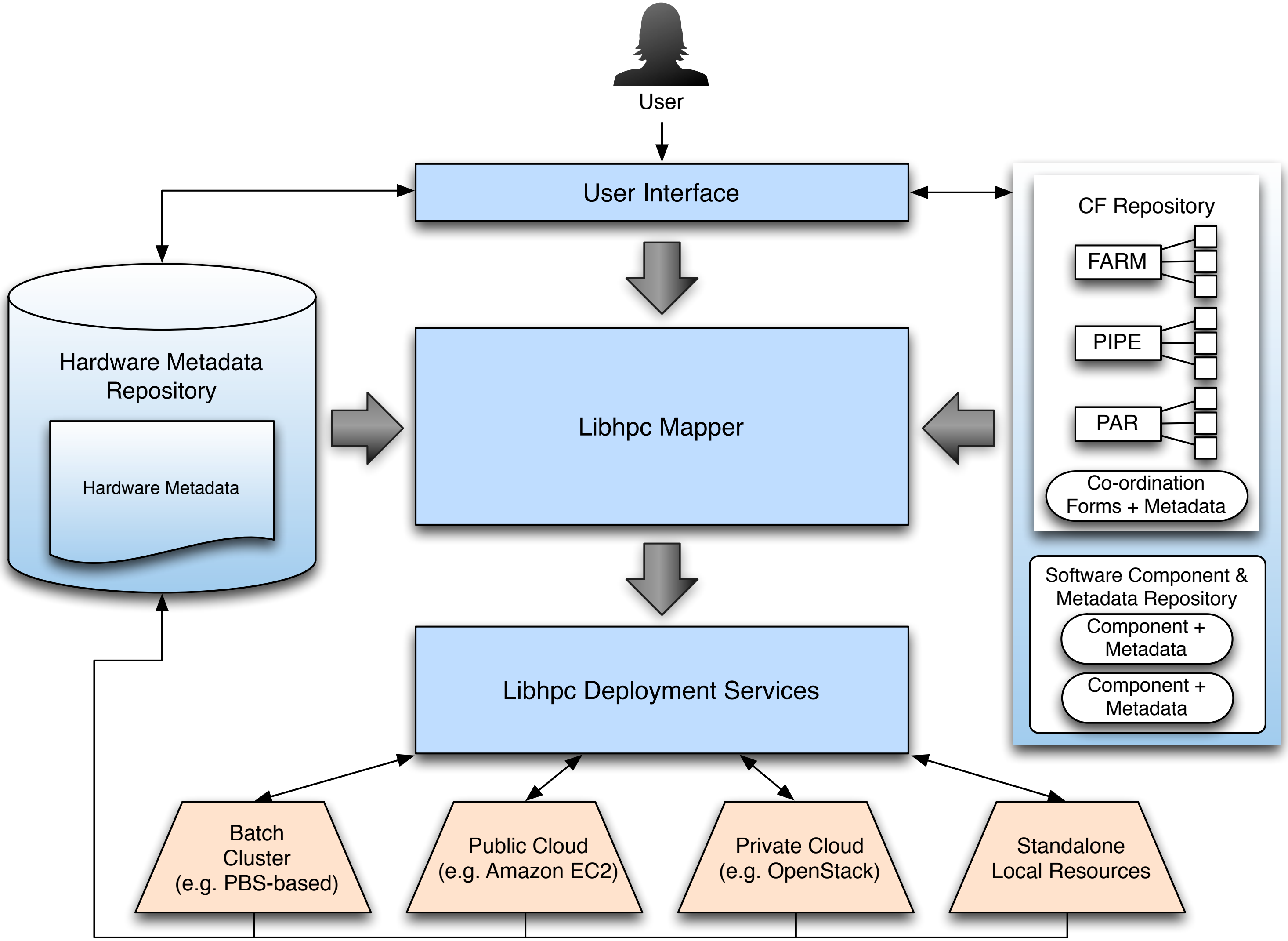
- Libhpc 2 runs to end October 2015
- Builds on Libhpc 1 which ran from July 11 -> Jun 13
- Developing framework model and a range of associated tools, services and demonstrators

- 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)

Imperial College London

Libhpc Architecture

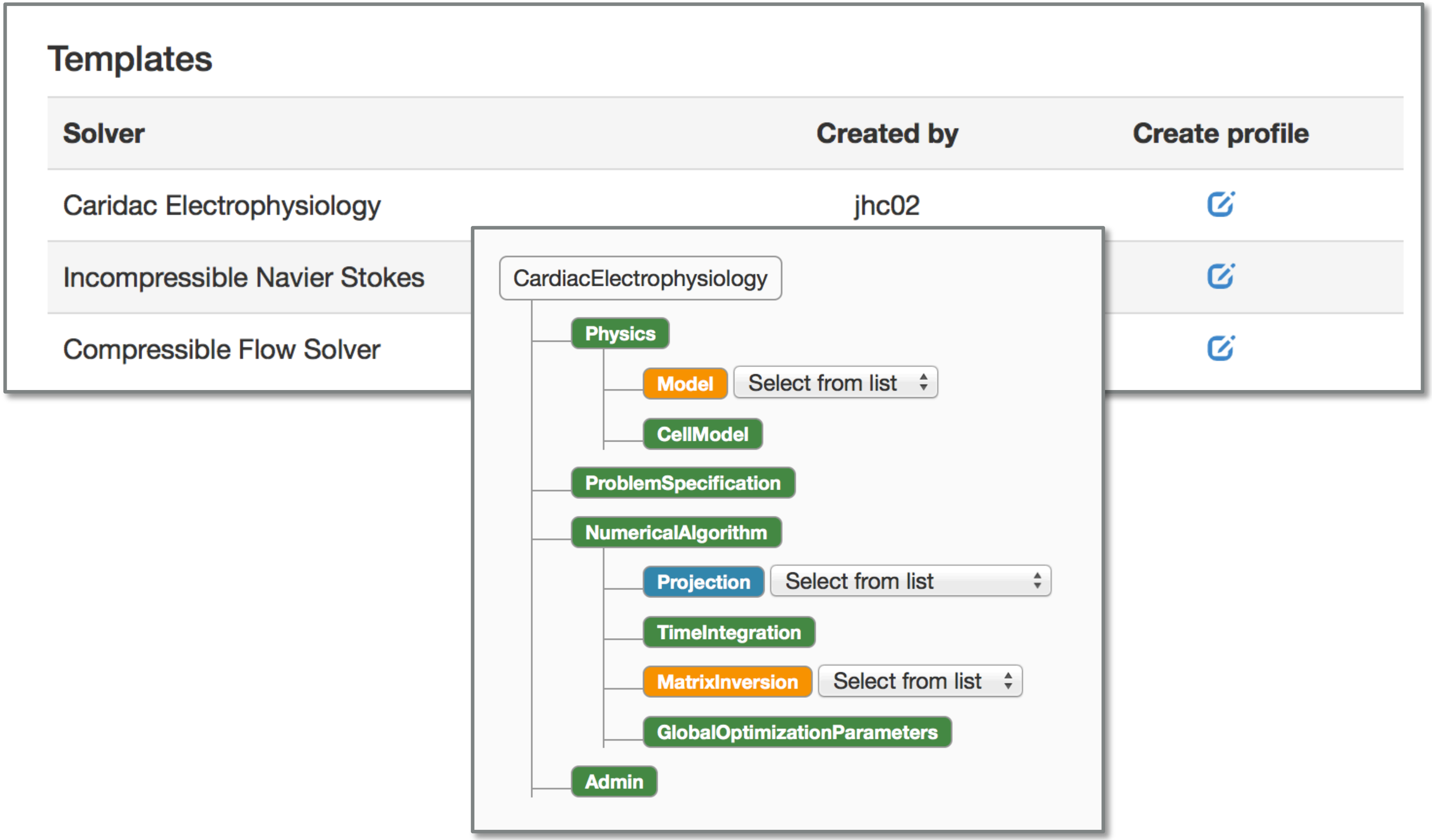

User
User Interface
CF Repository
FARM
PIPE
PAR
Co-ordination Forms + Metadata
Hardware Metadata Repository
Hardware Metadata
Libhpc Mapper
Software Component & Metadata Repository
Component + Metadata
Component + Metadata
Libhpc Deployment Services
Batch Cluster (e.g. PBS-based)
Public Cloud (e.g. Amazon EC2)
Private Cloud (e.g. OpenStack)
Standalone Local Resources

Templates and Profiles

Templates & 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



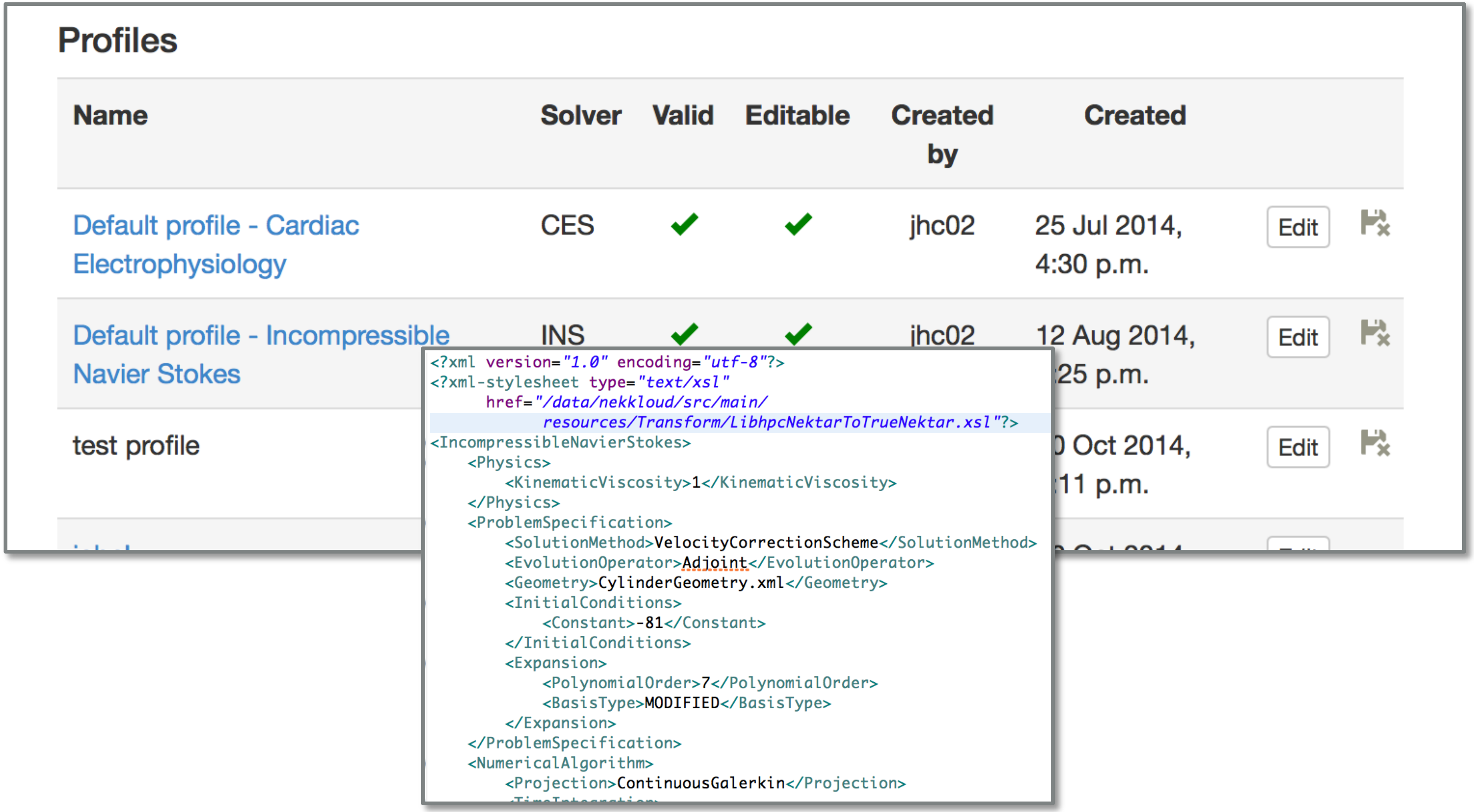
Templates

Solver	Created by	Create profile
Caridac Electrophysiology	jhc02	✎
Incompressible Navier Stokes		✎
Compressible Flow Solver		✎

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

Templates & Profiles



Profiles

Name	Solver	Valid	Editable	Created by	Created		
Default profile - Cardiac Electrophysiology	CES	✓	✓	jhc02	25 Jul 2014, 4:30 p.m.	Edit	
Default profile - Incompressible Navier Stokes	INS	✓	✓	jhc02	12 Aug 2014, :25 p.m.	Edit	
test profile					0 Oct 2014, :11 p.m.	Edit	

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl"
      href="/data/nekkloud/src/main/
            resources/Transform/LibhpcNektarToTrueNektar.xsl"?>
<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>
```

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 –

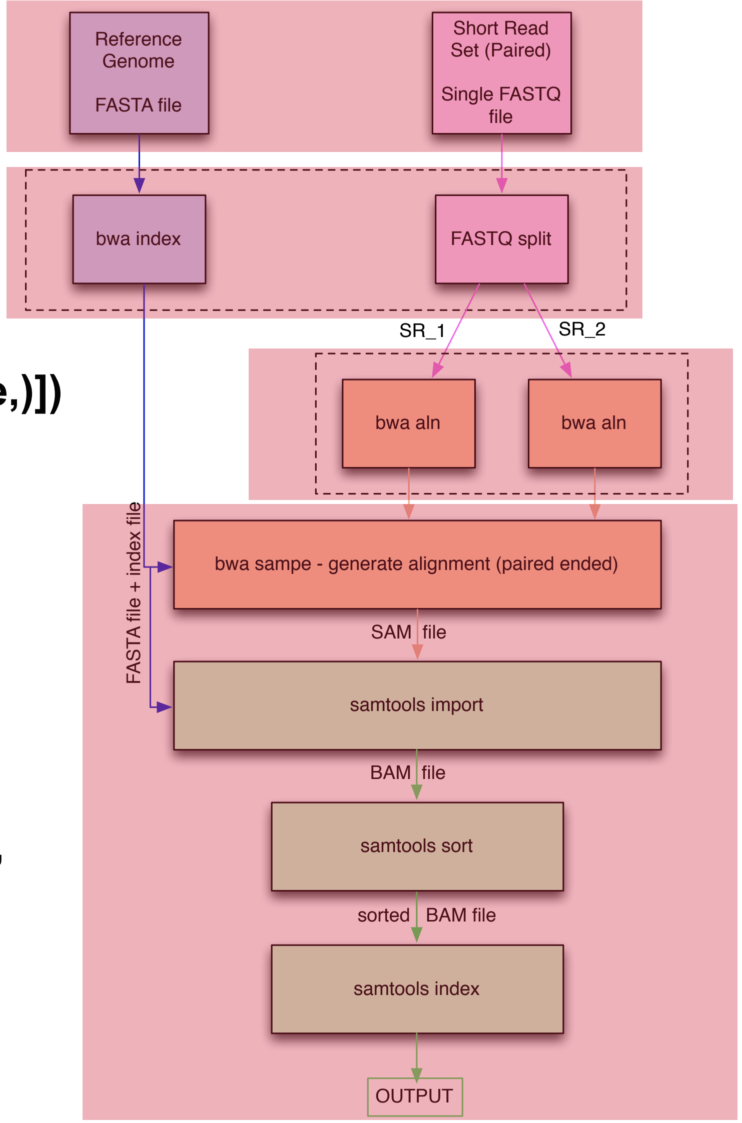
Reference Genome – FASTA file

Reads from sequencing machine - FASTQ

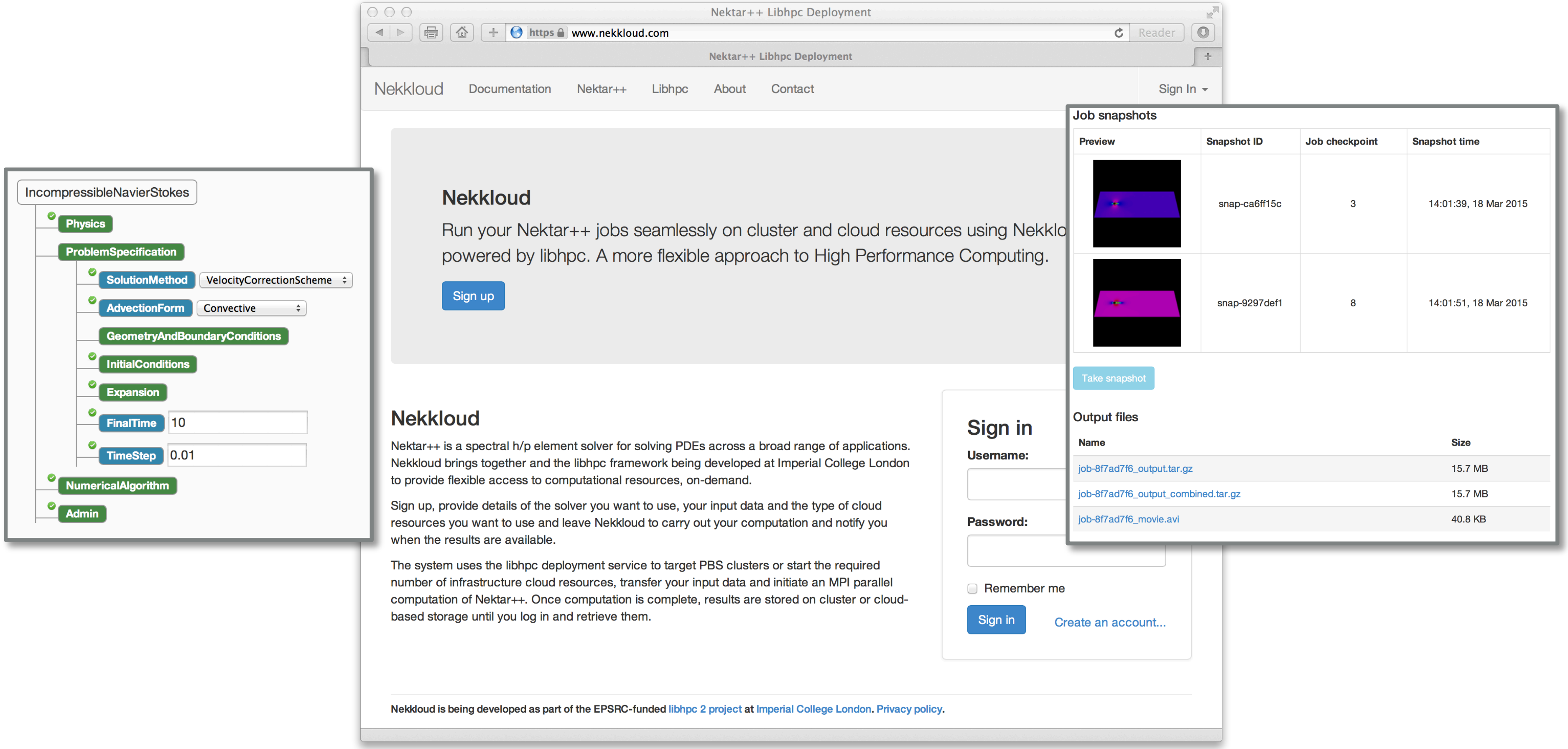
```
((sr1,sr2), u) = PAR([fastq_split, bwa_index],
     [(short_read_file, None, None),(ref_genome_file,)])

               (v, w) = PAR([bwa_aln, bwa_aln],
                       [(ref_genome_file, sr1, None),
                       (ref_genome_file, sr2, None)])

result = PIPE([samtools_index, samtools_sort,
                       (samtools_import, ref_genome_file),
                        bwa_sampe],
               [ref_genome_file, [v,w], [sr1, sr2], None])
```



Nekkloud: Simplifying Access to Nektar++



IncompressibleNavierStokes
- Physics
- ProblemSpecification
  - SolutionMethod: VelocityCorrectionScheme
  - AdvectionForm: Convective
  - GeometryAndBoundaryConditions
  - InitialConditions
  - Expansion
  - FinalTime: 10
  - TimeStep: 0.01
- NumericalAlgorithm
- Admin

Nekkloud

Run your Nektar++ jobs seamlessly on cluster and cloud resources using Nekklo
powered by libhpc. A more flexible approach to High Performance Computing.

Nekkloud

Nektar++ is a spectral h/p element solver for solving PDEs across a broad range of applications. Nekkloud brings together and the libhpc framework being developed at Imperial College London to provide flexible access to computational resources, on-demand.

Sign up, provide details of the solver you want to use, your input data and the type of cloud resources you want to use and leave Nekkloud to carry out your computation and notify you when the results are available.

The system uses the libhpc deployment service to target PBS clusters or start the required number of infrastructure cloud resources, transfer your input data and initiate an MPI parallel computation of Nektar++. Once computation is complete, results are stored on cluster or cloud-based storage until you log in and retrieve them.

Sign in

Username:

Password:

□ Remember me

Sign in　Create an account...

Nekkloud is being developed as part of the EPSRC-funded libhpc 2 project at Imperial College London. Privacy policy.

Job snapshots

Preview	Snapshot ID	Job checkpoint	Snapshot time
	snap-ca6ff15c	3	14:01:39, 18 Mar 2015
	snap-9297def1	8	14:01:51, 18 Mar 2015

Output files

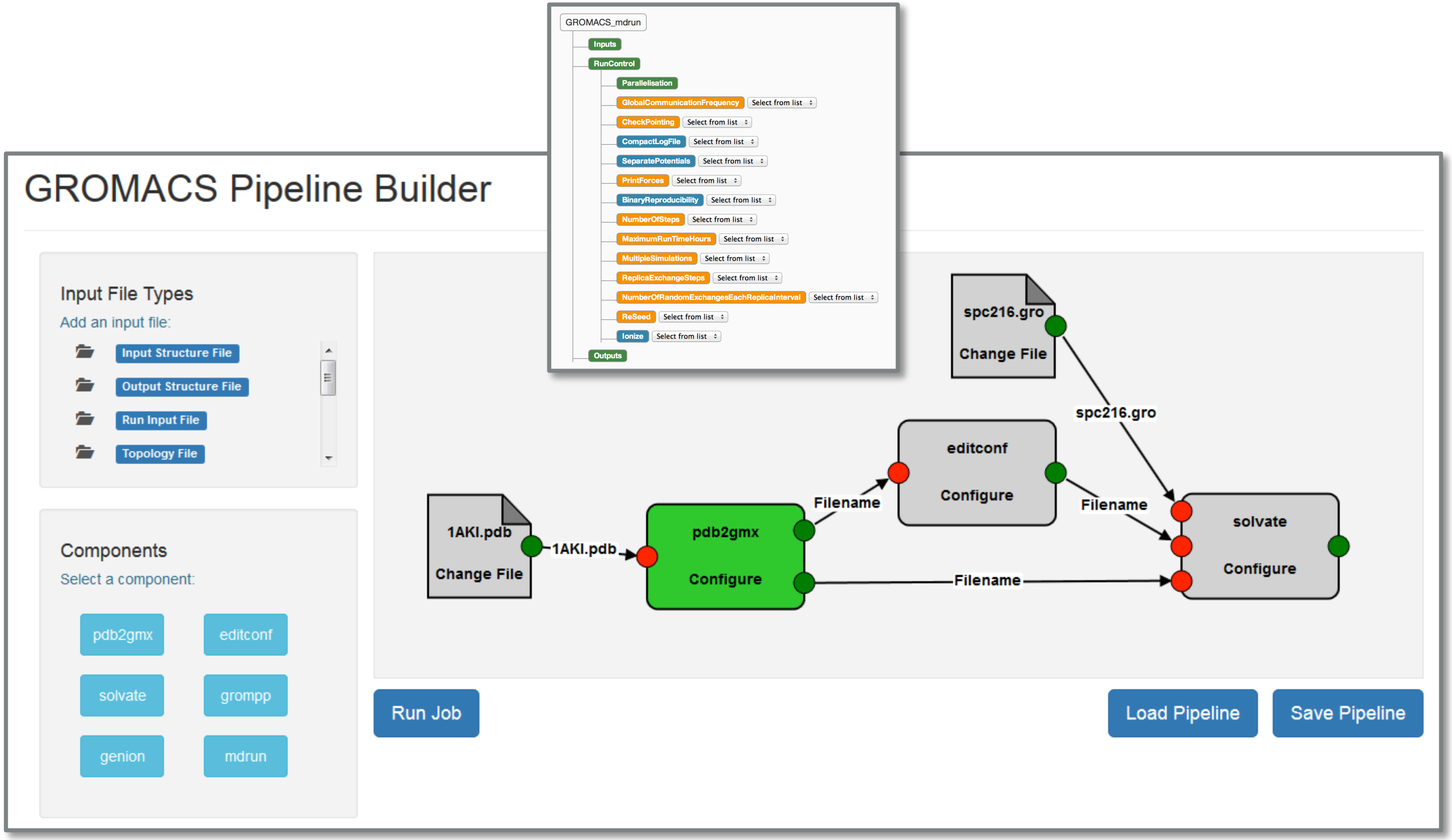
Name	Size
job-8f7ad7f6_output.tar.gz	15.7 MB
job-8f7ad7f6_output_combined.tar.gz	15.7 MB
job-8f7ad7f6_movie.avi	40.8 KB

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

Imperial College London

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



GROMACS_mdrun

- Inputs
- RunControl
  - Parallelisation
  - GlobalCommunicationFrequency: Select from list
  - CheckPointing: Select from list
  - CompactLogFile: Select from list
  - SeparatePotentials: Select from list
  - PrintForces: Select from list
  - BinaryReproducibility: Select from list
  - NumberOfSteps: Select from list
  - MaximumRunTimeHours: Select from list
  - MultipleSimulations: Select from list
  - ReplicaExchangeSteps: Select from list
  - NumberOfRandomExchangesEachReplicaInterval: Select from list
  - ReSeed: Select from list
  - Ionize: Select from list
- Outputs

GROMACS Pipeline Builder

Input File Types

Add an input file:

Components

Select a component:

Nekkloud Demo

Imperial College
London

Thanks & Acknowledgements

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