Tackling Complexity in High Performance Computing Applications

John Darlington, Tony Field, Layal Hakim

Imperial College London

July 2, 2015

Introduction

- Developing methods to tackle complexity inherent in HPC applications
- Two projects:
  1. MOSS - commercial video transcoding
  2. libhpc II - numerical simulations, aerodynamics, molecular dynamics and bioinformatics

Where does complexity originate?

- Inherent complexity of subsystems (application domains, methods, processors, machines)
- Choices compound on application construction
- Decisions made not represented in code

Our approach

- Abstract building blocks (functional) to explore choices
- Constraint solving (prolog) to make consistent choices

- Applications are specified by a functional task graph
- Nodes within the task graph may be defined to be *abstract*
- Constraint solving used to aid consistent selection of options
- By archiving workflows and their instantiations as they evolve from abstract to concrete, we naturally expose the provenance of a particular 'build' of the application.
- It is possible to specify additional *constraints* that describe the dependencies between various component implementations and parameters.

Decision Engine

We have introduced a decision engine which uses prolog as a constraint solver to guide mapping.

By automatically invoking a constraint solver at each parameterisation step we can ensure that it is impossible to construct a concrete workflow that is internally inconsistent.

Our approach combines the expressive power of high-level functional workflows, and logic programming, which we use to specify and manage constraints.

Main aim: make use of constraints to determine valid instantiations of a workflow.

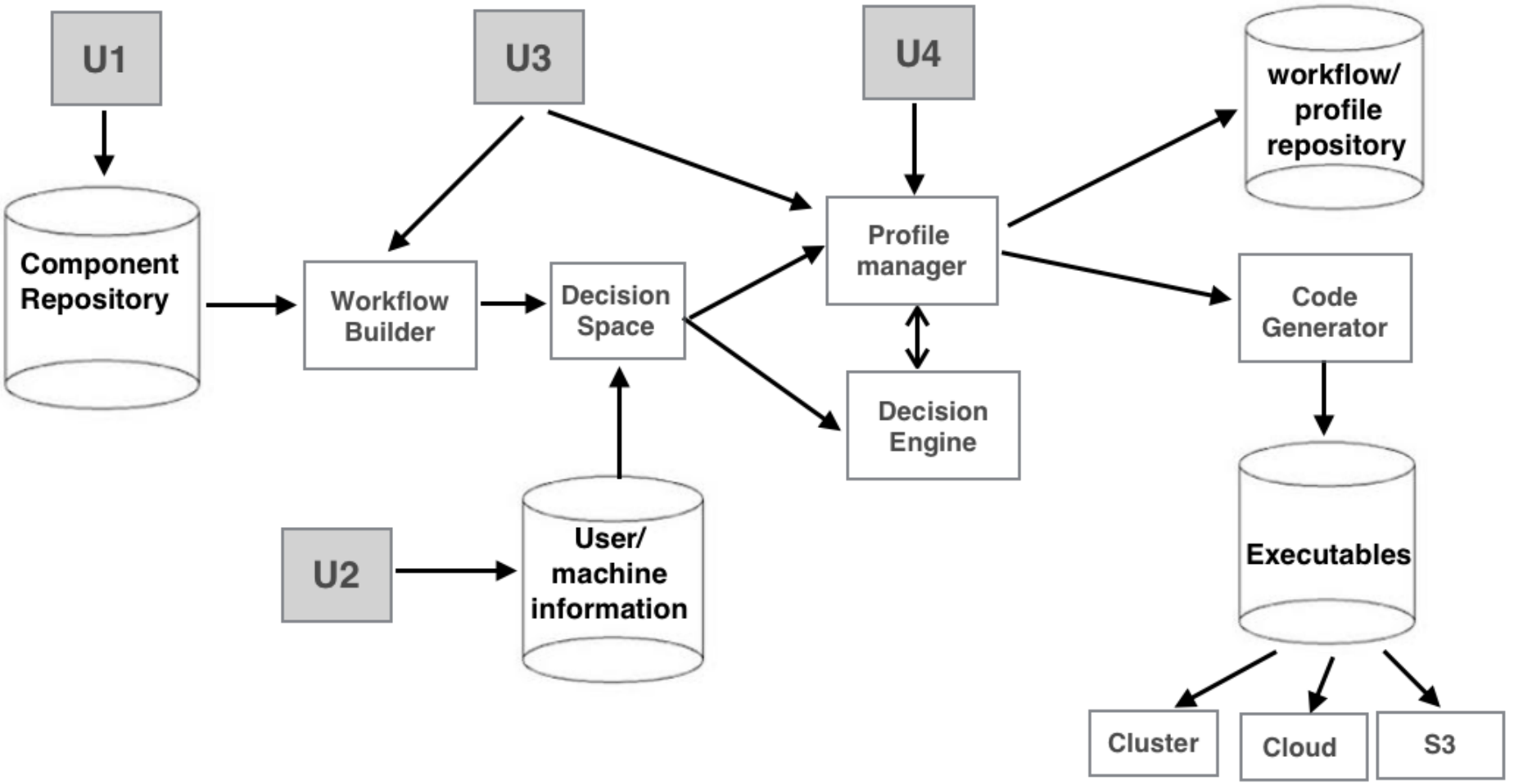


Figure: Computational abstractions

Main users

- Component developers
- Administrators
- Workflow/profile developers
- The end users

The profile repository acts as a decision store supporting:

1. end-use
2. system modifiactions
3. provenance

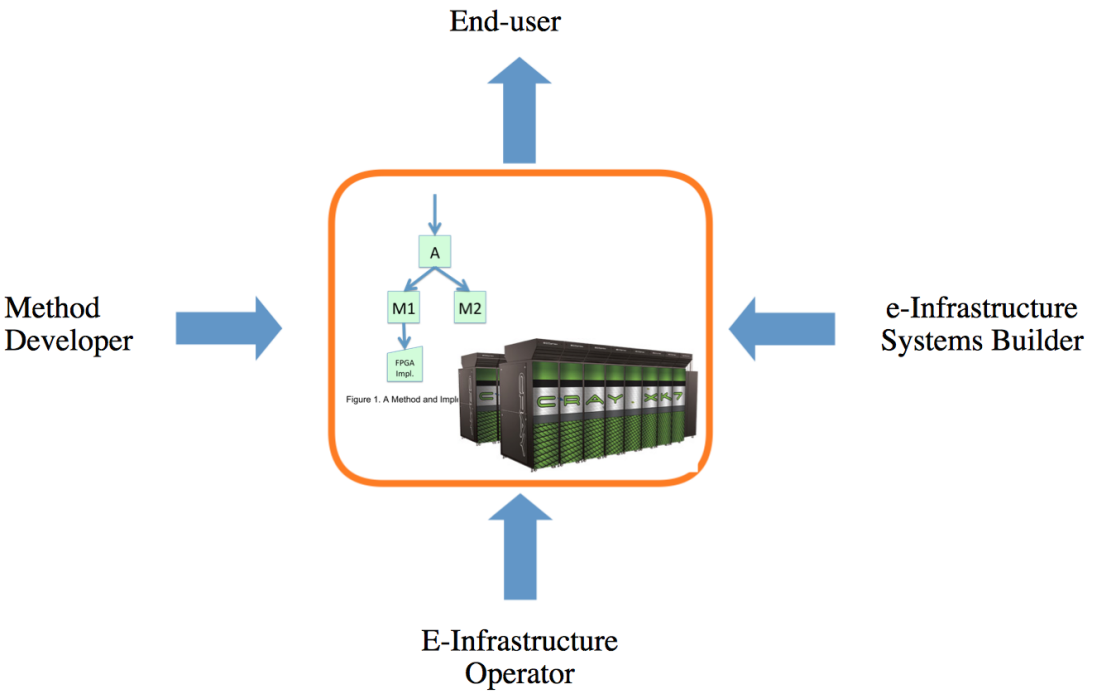


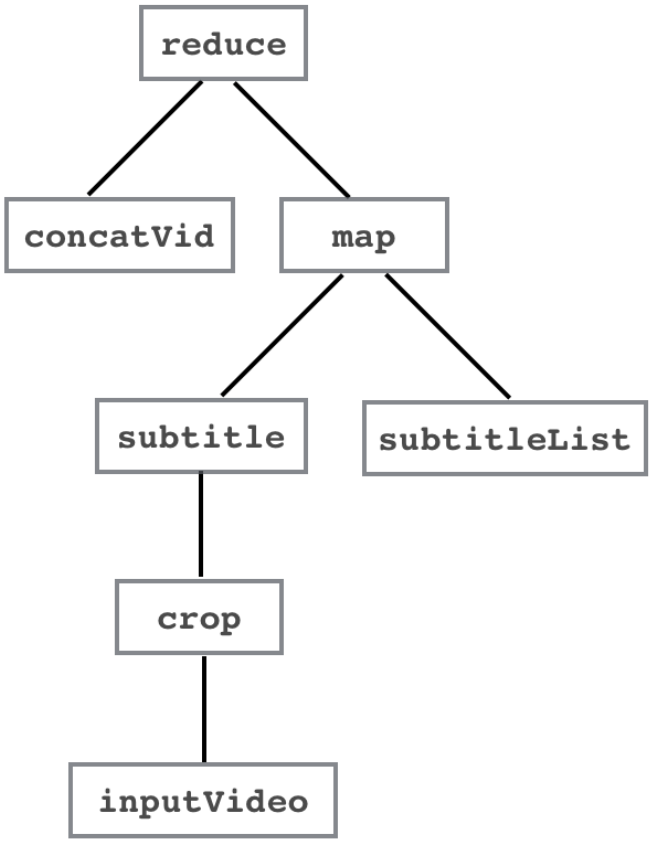
Figure: Decoupled e-infrastructure

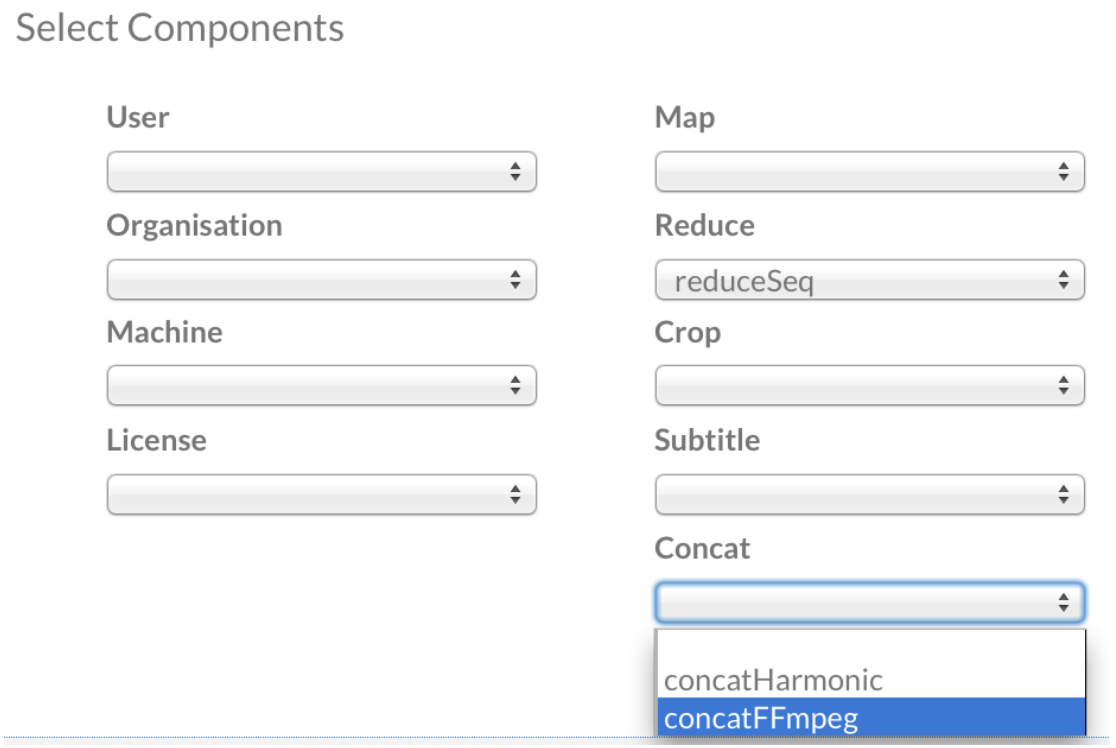
Example: Video Transcoding

- Main aim: To make video transcoding easier and to obtain a way of processing and archiving very large collections of media files (binary large objects, or blobs).
- Borrowed from Libhpc II:
  1. Functional Abstraction
     - coordination forms
     - components
     - *blob store*
  2. Profile store

Abstract Workflow Example

Files used in this example: InputVid (1 input video), SubList (8 srt files).
The abstract functions: mapAb, reduceAb, subtAb, cropAb, and concatAb.





$C = \text{reduce}(\text{concatVid}, \text{map}(\text{subtitle}(\text{crop}(\text{InputVideo})), \text{subtitleList}))$

Demo

MOSS Project Partners

Imperial College London

smoke&mirrors

Innovate UK Project
File Reference: 101778
Application Reference: 35941-254227
01/06/2014 - 30/05/2015

Thank you for your attention!