HPC on the Cloud

Author: Friedrich Grabner
Supervisor: Dr C. Cantwell

September 4, 2017

Imperial College London

Introduction

Context:

- Advancements in computing power has increased the feasibility of using high-order numerical methods to solve complex engineering problems
- Availability to IaaS systems has allowed those outside large institutions to gain access to high performance computing

Problem:

- Nevertheless most people still prefer to use low-order commercial software, on personal hardware

Introduction

Why do people prefer commercial codes?

- Complex and daunting interface of open-source softwares
- High-level of expertise required to understand how to run properly run a simulation. (numerical methods, submitting jobs, post-processing)
- Inadequate user support in open-source projects

Aims and Objectives

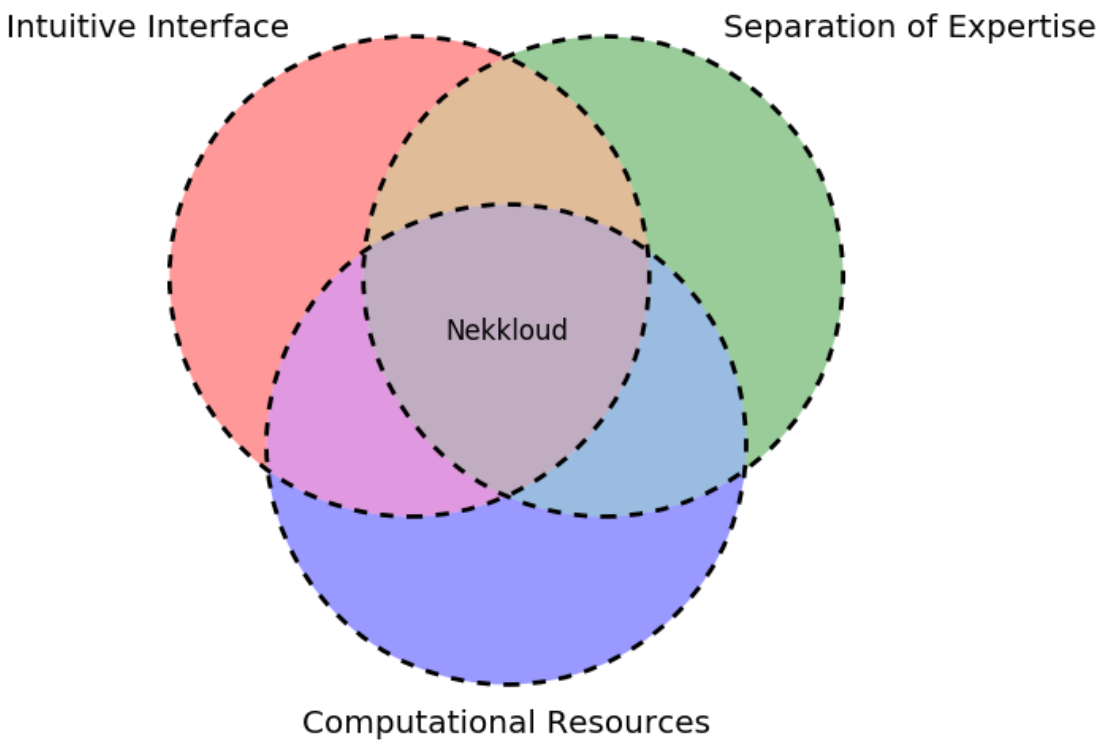
Aim: Build a web-based GUI that generates robust input XML files for Nektar++'s incompressible flow solver.

Objectives:

1. Map parameter space of incompressible flow solver in Nektar++
2. Understand and detail the constraints between parameters
3. Develop TemPSS transform templates that instantiate all parameters
4. Represent parameters and constraints within an intuitive interface
5. Perform a beta testing of the interface with Nektar++ users
6. Benchmark TemPSS using a selection of representative testcases

Nekkloud

- Nekkloud is a web based platform through which the Nektar++ solvers can be launched
- Nekkloud streamlines the running of simulations through access to IaaS systems



Templates and **P**rofiles for **S**cientific **S**oftware - **TemPSS**

- Integrated into Nekkloud
- Java based web environment hosting the user interface for Nektar++
- Key components:
  1. Templates - displays parameter space of solver
  2. Profiles - instatiations of templates
- Templates are defined by XML schema
- Profiles can be saved and shared with other users of Nekkloud/TemPSS

▸ TemPSS Interface

Mapping the Interface

To design the user interface first all the parameter space of the incompressible solver must be determined.

Additionally the parameter space must be divided into logical and intuitive sets.

▸ Incompressible Navier-Stokes

Users must also be prevented from making incorrect or injudicious choices for their Nektar++ input files.

Therefore constraints between the parameter space are mapped.

▸ Constraints Visualiser

Interface Framework

Template trees and Nektar++ input files are both defined using XML files.

Moving from the TemPSS tree to the final XML requires a number of stages.

- TemPSS tree is defined using XML schema
- Constraints are enforced using another XML sheet
- Values implemented in instantiated simulation profiles are parsed using an XSLT transform
- XSLT transform then generates a Nektar++ input XML

▸ TemPSS Interface

XML Schema

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2 <Force repeatable="true" paramType="choice" optional="true">
 3   <Absorption>
 4     <Coeff type="xs:string"/>
 5     <RefFlow type="xs:string"/>
 6     <RefFlowTime type="xs:string"/>
 7   </Absorption>
 8   <Body>
 9     <BodyForce type="xs:string"/>
10   </Body>
11   <Noise>
12     <Whitenoise type="positiveDouble"/>
13     <UpdateFreq type="positiveDouble" optional="true"/>
14     <Nsteps type="positiveDouble" optional="true"/>
15   </Noise>
16 </Force>
```

XSL Transform - Parent Node

```
 1 <xsl:template match="Force" mode ="AddForces">
 2     <FORCE>
 3       <xsl:if test="Absorption">
 4         <xsl:attribute name="TYPE">Absorption</xsl:attribute>
 5         <xsl:apply-templates select="Absorption" mode ="AddForces"/>
 6       </xsl:if>
 7       <xsl:if test="Body">
 8         <xsl:attribute name="TYPE">Body</xsl:attribute>
 9         <xsl:apply-templates select="Body" mode ="AddForces"/>
10       </xsl:if>
11       <xsl:if test="Noise">
12         <xsl:attribute name="TYPE">Noise</xsl:attribute>
13         <xsl:apply-templates select="Noise" mode ="AddForces"/>
14       </xsl:if>
15     </FORCE>
16 </xsl:template>
```

XSL Transform - Child Node

```
 1 <xsl:template match="Absorption" mode ="AddForces">
 2     <COEFF><xsl:value-of select="Coeff"/></COEFF>
 3     <REFFLOW><xsl:value-of select="RefFlow"/></REFFLOW>
 4     <REFFLOWTIME><xsl:value-of select="RefFlowTime"/></REFFLOWTIME>
 5 </xsl:template>
 6
 7 <xsl:template match="Body" mode ="AddForces">
 8     <BODYFORCE><xsl:value-of select="Body"/></BODYFORCE>
 9 </xsl:template>
10
11 <xsl:template match="Noise" mode ="AddForces">
12     <WHITENOISE><xsl:value-of select="Whitenoise"/></WHITENOISE>
13     <xsl:if test="UpdateFreq">
14       <UPDATEFREQ><xsl:value-of select="UpdateFreq"/></UPDATEFREQ>
15     </xsl:if>
16     <xsl:if test="Nsteps">
17       <NSTEPS><xsl:value-of select="Nsteps"/></NSTEPS>
18     </xsl:if>
19 </xsl:template>
```

Conclusions and Future Work

With respect to the initial aims and objectives the project can be deemed a success.

However for the project of Nekkloud there are still a number of areas which require further work, detailed below:

1. Improve and consolidate the user eco-system within Nekkloud/Nektar++
2. Continue to add templates to TemPSS
3. Develop the ability to load separate sections of profiles