Ensuring an effective user experience when managing and running scientific HPC software

(POSITION PAPER)

Jeremy Cohen[*], David Moxey[†], Chris Cantwell[†], Peter Austing[*], John Darlington[*] and Spencer Sherwin[†]
[*]Department of Computing, [†]Department of Aeronautics
Imperial College London, South Kensington Campus
London SW7 2AZ, UK
Email: jeremy.cohen@imperial.ac.uk

***Abstract*—With CPU clock speeds stagnating over the last few years, ongoing advances in computing power and capabilities are being supported through increasing multi- and many-core parallelism. The resulting cost of locally maintaining large-scale computing infrastructure, combined with the need to perform increasingly large simulations, is leading to the wider use of alternative models of accessing infrastructure, such as the use of Infrastructure-as-a-Service (IaaS) cloud platforms. The diversity of platforms and the methods of interacting with them can make using them with complex scientific HPC codes difficult for users. In this position paper, we discuss our approaches to tackling these challenges on heterogeneous resources. As an example of the application of these approaches we use Nekkloud, our web-based interface for simplifying job specification and deployment of the Nektar++ high-order finite element HPC code. We also present results from a recent Nekkloud evaluation workshop undertaken with a group of Ph.D. student users of Nektar++.**

I. Introduction

As computing hardware continues to evolve, it is clear that the future of high-performance computing is diversifying to include a variety of multi- and many-core architectures. The methods under which we access these resources are also rapidly changing, with Infrastructure-as-a-Service (IaaS) cloud platforms and related technologies, such as containerisation, becoming increasingly popular as an alternative to traditional on-site HPC facilities and the support and infrastructure that is required to maintain them. Moreover, the range of different platforms and the different development techniques that they require, together with the process of deploying and configuring jobs to run on these platforms, are now placing increasing demands on the knowledge and skill-sets of developers and users alike. Investigations have been undertaken into how to support the requirements of such environments, for example in the context of e-Science [1], however, new approaches governing how users interact more widely with HPC facilities and how developers create their applications for them are still required.

Users face two main challenges when using modern HPC codes. Firstly, as software continues to evolve and becomes more elaborate, the complexity of the configuration also increases, with a plethora of performance and hardware-specific options and parameters often becoming available. Secondly, as hardware platforms become more diverse, so does the process of building and configuring software to run on these platforms. In this paper we outline our ideas and current efforts to improve the end-user experience, in terms of simplifying and managing complex software configurations, as well as providing a unified approach to deployment across heterogeneous architectures. We are particularly interested in improving access to alternative computing platforms for end-user scientists and researchers.

Traditionally, end-users may have relied on developers to handle the complexities of configuring simulations to run on new platforms. Preparing input data for HPC codes can be difficult and is often not intuitive. If users frequently run the same jobs on the same resources, input data files can often be reused, but where input data needs to be changed to modify the algorithms or target different platforms, this may be challenging and error-prone. Our approach is to use parameter trees to represent and group application parameters. These trees not only allow end-users to understand the options available to them and the decisions that need to be made in order to run code efficiently, but also support developers in understanding code structure to support maintenance and extension of applications. We call these trees *application parameter templates* and they are discussed in more detail in Section II-A. Rather than just providing a direct representation of an application's input file(s), the tree structure and parameters are intended to represent decisions based on application semantics and how they relate to different application structures.

As computing has become an integral part of almost all scientific domains, scientists and researchers generally require some familiarity with HPC infrastructure and deployment to clusters via batch scheduling systems such as PBS or Grid Engine. However, with the diverse range of platforms that are now available, traditional batch job submission is not always used, for example on cloud platforms where resources are instantiated as needed. The steps to run a simulation on an IaaS platform are generally significantly different to a local cluster, and many scientific end-users do not necessarily have the technical skills to construct and deploy the virtual machine images which are commonly used on these platforms.

We consider that a natural solution to these issues is an intelligent middleware that abstracts away the complexities of the hardware and computational methods, captures developers' knowledge of the software parameter space, and enables users to move seamlessly between platforms within a common environment. The material presented in Section II is based on our ongoing work developing *libhpc* [2], [3], a framework to support job specification and the flexible execution of HPC applications on heterogeneous infrastructure. In Section III we discuss the outcomes of a workshop held in December 2014 to introduce a group of Ph.D. student users of Nektar++ to the Nekkloud [4] environment and obtain feedback on the system and its usability. We provide concluding remarks in section IV.

II. Web-based job configuration and deployment

A. Job specification

While users have the potential to benefit from the new types of computational platform described in Section I, using these systems in a traditional manner adds complexity for users. Generally they must log in to a platform, transfer or create their input data files and then run their job(s). This is after they have organised accounts and credentials to access the platform(s). It is often necessary for them to learn a new environment and commands, whilst understanding details of the target platform's configuration in order to be able to use it correctly. This is especially true in cloud-based environments, where additional complex tasks such as constructing a virtual image of the software to be run need to be undertaken.

We believe that visual environments for job submission can provide a superior interface to platforms where the user may not be familiar with the underlying infrastructure tools. However, a common complaint is that such environments can remove flexibility or hide important functionality. When working with more complex software and targeting different types of platform, the need to abstract away from the underlying hardware and configuration process becomes more important.

We do this abstraction at two levels – the job specification and the deployment layer. For job specification we use the concepts of *application parameter templates* and *job profiles*. A template is designed to represent a series of parameters that cover all the possible decisions that can affect the execution of the target application. These parameters are grouped under headings relating to different application features or areas of functionality and displayed to the user in a suitable graphical manner that makes them easy to browse. A user fills in values in a template in order to specify their job. An example of this tree structure and the user interface is given in Figure 1.

A template that has some or all of its values specified can be stored as a *profile*. A valid profile – one that has a full set of required template values completed – can be used to run a job. Stored profiles, if set to be public by their creator, can be edited by other users in order to complete additional content until they ultimately become valid profiles that can be used to run a job. The ability to edit and extend partially complete profiles is a powerful concept, enabling collaborative job specification and the ability to store libraries of partially or fully complete profiles. It enables more experienced users to generate pre-built templates for jobs, so that less-experienced

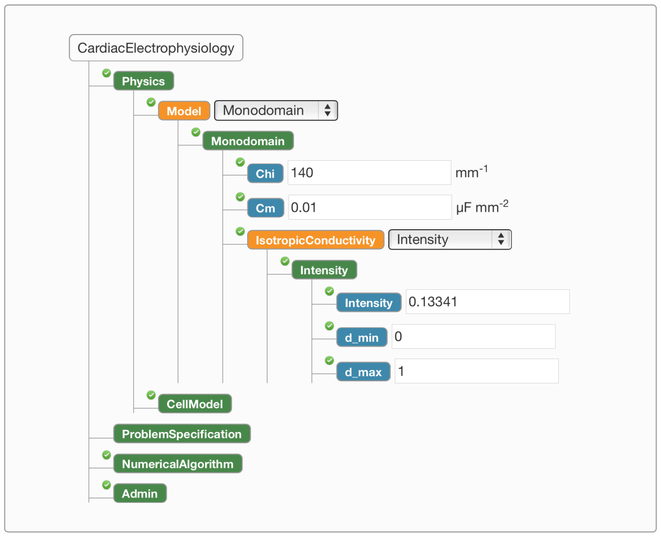

```
CardiacElectrophysiology
  Physics
    Model  Monodomain
      Monodomain
        Chi  140  mm^-1
        Cm  0.01  μF mm^-2
        IsotropicConductivity  Intensity
          Intensity
            Intensity  0.13341
            d_min  0
            d_max  1
    CellModel
  ProblemSpecification
  NumericalAlgorithm
  Admin
```

Fig. 1. Subset of a template for a Nektar++ solver

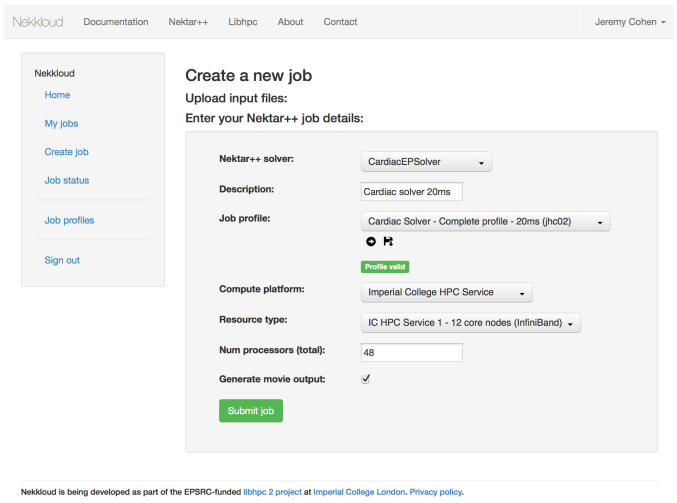
users can quickly get started with their intended simulations. Additionally, developers may provide profiles which select optimal performance parameters for different architectures, abstracting this away from the user.

Once a valid profile is selected to run a job, it is transformed from its initial XML-based form into the necessary input file format for the target application. The preparation of metadata and code for transforming a profile into the required input file format is handled by the application developers using guidelines and specifications defined by the libhpc framework.

B. Deployment

On the software deployment side, the libhpc framework makes use of a deployment service that is capable of deploying applications to different types of platform. Our initial focus has been on IaaS clouds and PBS clusters where we have the ability to build on existing tools that simplify access to such platforms for developers. Apache libcloud [5] provides a software layer that offers a simplified interface to access a number of different IaaS cloud providers' platforms. RADICAL-SAGA provides an access layer, based on the Simple API for Grid Applications, to a range of HPC cluster platforms [6], [7]. A software repository stores software metadata and, where necessary, binaries, enabling the deployment of code and job data to a supported platform on a per-job basis.

These approaches support the development of a user-friendly interface that, whilst only shown here for Nektar++, could be similar in design and structure for a range of different scientific HPC codes, reducing the complexity of presenting parameters to users in a manner that should be more intuitive and self explanatory. We believe this approach optimises the job specification process for end-users and, in combination with job deployment tools developed in our libhpc I and II projects [3], removes a number of barriers associated with choosing to run code on different hardware platforms.



Create a new job

Upload input files:

Enter your Nektar++ job details:

Nektar++ solver: CardiacEPSolver

Description: Cardiac solver 20ms

Job profile: Cardiac Solver - Complete profile - 20ms (jhc02)

Compute platform: Imperial College HPC Service

Resource type: IC HPC Service 1 - 12 core nodes (InfiniBand)

Num processors (total): 48

Generate movie output: ✓

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

Fig. 2. The Nekkloud [4] job submission page, that allows users to easily start Nektar++ simulations on a variety of resources through a web-based interface.

C. An interface for Nektar++

The focus of our development efforts has been centred around Nektar++ [8], a high-order finite element framework consisting of a number of solvers with applications in various domains including fluid dynamics, aeronautics and cardiac electrophysiology. This software is used by a broad spectrum of users from a variety of different backgrounds, making it an ideal exemplar application for the concepts we present here. Nektar++ users currently work with a number of different HPC infrastructures and so are susceptible to the problems of configuration and deployment we have outlined previously. To address these needs, we have developed Nekkloud [4], a web application for specifying and running Nektar++ jobs. The aims of Nekkloud are to tackle precisely these problems, providing a user-friendly, streamlined interface under which configuration is significantly easier, and deployment can be targeted to a variety of cluster and cloud based infrastructure. An example of the Nekkloud job submission page, in which users can select the solver to use, which architecture to target and specify job parameters, is shown in Figure 2.

III. Nekkloud Evaluation Workshop

An opportunity arose to investigate some of the thoughts and ideas set out in this paper by running a small evaluation workshop for Nekkloud in December 2014. The workshop brought together seven users, with varying levels of experience of the Nektar++ framework, to introduce them to the Nekkloud environment and gain initial feedback on the usability and operation of the system. The Nektar++ users had not seen Nekkloud before and we hoped to achieve the following in running the workshop with this group of users:

- Feedback on the design and structure of Nekkloud and means of specifying jobs.
- Understand how new users interact with the system. What problems do they have with the system? What is unclear?
- Identify any lack of clarity in instructions, processes or wording provided in the interface.
- Feedback as to the suitability of documentation.
- Suggestions of additional features or modifications that they would consider valuable.
- Test reliability when using our local test cloud platform.

A. Workshop structure

The workshop attendees were either Ph.D. students or researchers undertaking work in different scientific domains, making use of Nektar++ to run simulations to support their research. The workshop was designed to allow us to obtain as much feedback as possible from a group of new users who were previously unfamiliar with the Nekkloud system.

The workshop consisted of two sessions. In the first session, users were given the URL of the Nekkloud environment and invited to spend approximately 45 minutes to see what they could achieve with the system using only the resources available in the web application. This included documentation and a sample mesh geometry for each of the solvers Nekkloud supports. Our aim was to investigate whether users could navigate the system, without external interaction, to submit their own jobs and find any shortcomings with the interface. On completion of this stage of the workshop, a feedback discussion was held to allow each attendee to report on what they achieved, what challenges they faced and any suggestions they had for possible improvements.

A second session was then run in a similar fashion to the the first, following a short presentation providing more information sabout the system and its operation. Attendees were asked to bring a small sample geometry of their own to the workshop and the aim for this second session was to run a job based on their own geometry rather than one of the samples provided. A further feedback discussion was then held to obtain any additional comments based on having gained further knowledge of the system.

B. Workshop outcomes

The workshop served as a valuable exercise in assessing user perspectives on our model of using templates and profiles to specify Nektar++ jobs and to see how intuitive users would find this approach. The vast majority of user feedback was positive, but important issues were raised and these will be addressed in ongoing development of the system. The comments received can be broadly grouped into three areas: technical, usability and feature-related. We now look at the feedback provided in each of these areas.

1) Technical: The majority of issues raised by users were in relation to technical issues or observations, such as the use of a self-signed certificate for SSL-based access, downloading or viewing data files, and whether files should be displayed or trigger a save request when clicked. These are all expected to be addressed in future development.

2) Usability: By far the most important feedback obtained was the usability of the system. Feedback was generally very positive but some particular feature requests and general comments were made. Documentation was considered to be clear and the job submission page was felt to be intuitive. The profile system, lying at the heart of the features we felt users could most benefit from, was deemed to be very useful. Whilst some minor technical glitches, which we were aware of before the workshop, tarnished the experience slightly, users could appreciate the benefits of easily changing parameters within their jobs and the use of pre-determined profiles. Some comments were made about a lack of clarity, when viewing job profile trees, about where required values needed to be entered. This can make it difficult for the user to identify the specific values that they need to add when working with a partially complete tree.

3) Features: The final area where comments were received was in terms of feature requests for the Nekkloud platform. Particular requests included:

- more detailed resource specification information;
- clearer job execution progress information;
- improved output file naming; and
- marking default values when editing templates.

While this study was small and was not backed by a questionnaire or other material to enable us to more easily tie the outputs to specific metrics, we consider that as a first stage in evaluating our software and identifying the most significant areas to focus ongoing development, it was a valuable exercise. Users appreciated the simplicity of deploying their code to a cloud platform, potentially giving them immediate access to a much wider range of resources when running their jobs. We plan to undertake further interactive workshops in due course and hope to get more detailed input from a larger number of users as our system progresses.

IV. Conclusion

Configuration of HPC jobs can be a challenge for users running complex scientific codes. When working with a single platform such as a local cluster and regularly running similar jobs, users may go through the task of creating a base configuration once and then be able to re-use this configuration on many occasions. While modern, flexible HPC infrastructure offers a number of potential benefits to scientists and researchers, it raises the prospect of having to make configuration changes to support different target platforms on a per-job basis. The configuration processes and means of interaction can differ significantly between platforms and require a range of knowledge in order to be able to use them effectively. Furthermore, in an environment where many users are not from a computer science background and may have limited HPC knowledge, providing an effective user experience is considered to be the most important aspect to ensure take up of new environments.

This paper has described some of the challenges of working with complex scientific software on heterogeneous infrastructure and our approaches to overcoming them. It has discussed our use of application parameter templates and job profiles to help end-users undertake job configuration, both independently, and collaboratively with colleagues or other stakeholders. To address the related challenge of simplifying the process of deploying codes to heterogeneous infrastructure we have discussed the use of web-based interfaces and associated software services to provide an abstraction layer between the end-user and the complexities of software deployment. While this approach has so far been demonstrated in the context of the Nektar++ finite element framework and the Nekkloud web application, being developed as part of the *libhpc* project, we believe it to be an effective solution to ensuring users can keep pace with advancing technology and consider that it is equally applicable to a wide range of other scientific areas.

To obtain feedback on our approaches to job specification and execution, a small-scale Nekkloud evaluation workshop was carried out to introduce the prototype Nekkloud system to a group of scientific HPC users. While this highlighted a number of specific technical concerns, the overall feedback in relation to both the deployment and job specification aspects was positive and participants said that they would be willing to use such a platform for their research.

While efficient management and use of modern HPC infrastructure models is an ongoing challenge, we believe that with further development, our approach has the potential to offer HPC users greater flexibility and a more acceptable user experience when undertaking their HPC jobs.

Acknowledgment

The authors would like to thank the UK Engineering and Physical Sciences Research Council (EPSRC) for funding the projects libhpc: Intelligent Component-based Development of HPC Applications (EP/I030239/1) and libhpc Stage II: A Long-term Solution for the Usability, Maintainability and Sustainability of HPC Software (EP/K038788/1) that have supported this research.

References

[1] L. Ramakrishnan, K. R. Jackson, S. Canon, S. Cholia, and J. Shalf, "Defining future platform requirements for e-science clouds," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 101–106. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807145

[2] J. Cohen, J. Darlington, B. Fuchs, D. Moxey, C. Cantwell, P. Burovskiy, S. Sherwin, and N. C. Hong, "libhpc: Software sustainability and reuse through metadata preservation," in *First Workshop on Maintainable Software Practices in e-Science*, Chicago, IL, USA, Oct. 2012, position paper. [Online]. Available: http://www.software.ac.uk/sites/default/files/softwarepractice2012_submission_8.pdf

[3] libhpc: Intelligent Component-based Development of HPC Applications. http://www.imperial.ac.uk/lesc/projects/libhpc. Accessed 22^{nd} Jan 2015.

[4] J. Cohen, D. Moxey, C. Cantwell, P. Burovskiy, J. Darlington, and S. J. Sherwin, "Nekkloud: A software environment for high-order finite element analysis on clusters and clouds," in *IEEE International Conference on Cluster Computing*. IEEE, 2013, poster paper.

[5] Apache Libcloud. https://libcloud.apache.org/. Accessed 30^{th} Jan 2015.

[6] S. Jha, H. Kaiser, A. Merzky, and O. Weidner, "Grid interoperability at the application level using saga," in *e-Science and Grid Computing, IEEE International Conference on*, Dec 2007, pp. 584–591.

[7] RADICAL Cybertools: RADICAL-SAGA. http://radical-cybertools.github.io/saga-python/. Accessed 30^{th} Jan 2015.

[8] C. D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. de Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. M. Kirby, and S. J. Sherwin, "Nektar++: An open-source spectral/hp element framework," *Computer Physics Communications*, vol. (Accepted), 2015.