Understanding Resource Selection Requirements for Computationally Intensive Tasks on Heterogeneous Computing Infrastructure

Jeremy Cohen[1(✉)], Thierry Rayna[2], and John Darlington[1]

[1] Department of Computing, Imperial College London, London, UK
{jeremy.cohen,j.darlington}@imperial.ac.uk
[2] Novancia Business School Paris, Paris, France
trayna@novancia.fr

Abstract. Scientists and researchers face challenges in efficiently configuring their scientific computing tasks so that they can be run in a timely and cost-effective manner. While the increasing availability of different types of computing platforms provides many opportunities to users, it can further complicate the job configuration process. In this paper we present work-in-progress to develop an approach to assist with identifying the most suitable computing platform and configuration for a computational task based on a user's financial and temporal constraints, using a decision support system. We use Nekkloud, a web-based tool for running computations via the Nektar++ spectral/hp element framework, as an exemplar and build a table that scores a range of properties for four example computing platforms to help select the most suitable platform for a job. We demonstrate our approach using three sample task scenarios.

Keywords: Resource selection · Decision support · Heterogenous platforms

1 Introduction

Computing platforms have evolved significantly over recent years with the emergence of multi-core processors containing increasingly large core counts and many-core architectures such as those used in GPUs. In addition, new models of accessing resources, such as the pay-per-use, on-demand access provided by Infrastructure-as-a-Service (IaaS) clouds, offer individuals additional opportunities for access to significant computational power.

While the capability to make use of new and different types of resources to undertake computations can offer scientists and researchers greater flexibility, it also presents a range of challenges. From a user perspective, these include selecting the most suitable resource(s) to use and correctly specifying the complex parameters and configuration files that are often required to run high performance scientific codes. Computing platforms have different costs, availability and

© Springer International Publishing AG 2017
J.Á. Bañares et al. (Eds.): GECON 2016, LNCS 10382, pp. 250–262, 2017.
DOI: 10.1007/978-3-319-61920-0_18

reliability and choosing the most suitable platform is a complex matter because it necessarily implies that trade-offs have to be made. Yet, it is not just an issue of 'time vs money' and there is most likely not a straightforward continuum of preferences that could be used to simply determine the most suitable platform considering the characteristics of the project at hand. The large number of parameters that are often involved in configuring High Performance Computing (HPC) codes mean that standard optimisation approaches can be impractical. This difficulty is compounded by the availability of multiple platforms and collaboration between different groups of individuals with different expertise when specifying task parameters. In this paper we describe work-in-progress to develop an alternative approach, relying on the expertise of those undertaking the job configuration and supporting this with a decision support framework that can assist in making the most appropriate decisions in a timely manner. We present the first stage of our research into approaches to help users in selecting a suitable target platform for running HPC jobs in environments where multiple resource types are available. Our use case is based on the Nekkloud [8] tool which, in combination with another tool, TemPSS [6], provides a web-based application for running spectral/hp element [12] computations via the Nektar++ [5] software. Using example scenarios, we demonstrate the challenges users can face in selecting a suitable target platform for their research computations. We then introduce the decision support framework, which will be implemented in the second stage of the research, and show, using the same scenarios, how it may be applied to help users in selecting suitable computational platforms for their tasks. As this work develops, we consider that the main contribution will be the ability to offer users of the Nekkloud tool, and others that adopt the decision support framework, recommendations of the most suitable computational platform to use to undertake their scientific HPC tasks. Ultimately we aim to enable automated selection of resources to support a user's required task configuration.

Related work is presented in Sect. 2 and we then introduce the HPC code and tools that provide our use case in Sect. 3. Section 4 presents and ranks four example computing platforms based on a set of properties. Section 5 presents the user scenarios and Sect. 6 discusses the properties of a decision support system to assist in efficient resource selection and looks at its application to the scenarios. We present our conclusions and details of future work in Sect. 7.

2 Related Work

Resource management and queuing systems such as TORQUE [1] and Grid Engine® [24] are widely used on HPC clusters to handle queuing and scheduling of jobs, and extensive work has been undertaken looking at approaches to job scheduling and resource management/selection. The emergence of computational grids raised additional challenges and [13] provides a survey of resource managers that support grid computing environments. Systems such as these handle a job when a platform has been chosen and the job submitted to it. As new models and hardware for undertaking computations have emerged, such as

Infrastructure-as-a-Service (IaaS) cloud platforms, GPUs and other accelerators, users have more opportunities to access computing power but also more challenges in understanding how best to select the most suitable platform(s) for their tasks. Our work focuses on the process of assisting users in selecting a suitable platform in an environment where multiple platforms are available.

Nekkloud users are currently required to select their chosen platform, resource type and related properties before running a job. However, it is often challenging for users to identify what they consider to be optimal and the level of risk they are willing to accept to achieve this. This is demonstrated in [4] which shows that where individuals have freedom of choice they take advantage of this even when its clear that the evaluation costs are high, possibly resulting in an inferior outcome compared to a case where choice is not available.

Describing platform capabilities and task requirements can be particularly challenging. Condor ClassAds [18], developed as part of the Condor workload management system [23], provide an approach to describing and matching requirements and capabilities. An extension of this mechanism, described in [15], provides a resource selection framework. In more complex heterogeneous environments, automating resource selection becomes especially challenging and [9] presents an algorithm that is designed to tackle this problem in a scalable manner. While it focuses on services, the SDL-NG framework [21] provides an alternative example of an approach, based on the use of domain specific languages, to provide formal descriptions of services. Our methodology defines a set of platform properties that are used to describe the capabilities of a platform, and could be represented using an approach such as ClassAds, and a decision support system that uses these properties to help address a user's requirements.

3 Software Environment

The spectral/hp element method [12], a type of high-order finite element method (FEM), can be used to solve systems of partial differential equations in order to carry out modelling of a range of physical mechanisms in a variety of scientific domains. Nektar++ provides an open source implementation of the spectral/hp element method and a set of solvers. It is well-documented but the complexity of such applications means it can still be challenging for end-users to work with.

Work to develop the libhpc framework [7,14], led to the development of a set of software tools and services to support and simplify running of scientific HPC codes on different computing resources. In this paper we base our investigations on two specific libhpc outputs, Nekkloud, and TemPSS (Templates and Profiles for Scientific Software). Nekkloud is a web-based user interface for running Nektar++ computations on different computing platforms and TemPSS is a service for managing sets of job configuration parameters for scientific applications. Figure 1 shows how the services are linked. TemPSS displays an application's configuration parameters as an interactive, visual tree. Users can build configurations by entering parameters into the tree nodes. The ability to store subsets

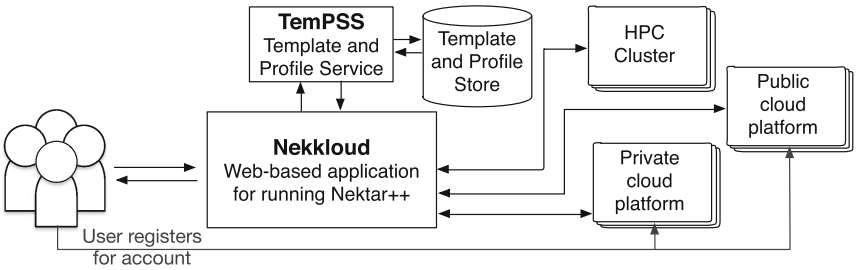

Fig. 1. The linking between Nekkloud, TemPSS, their users and computing platforms.

of parameters enables groups of individuals with different expertise to collaborate on building complete configurations. TemPSS can be used standalone or integrated into other applications and it has been built into Nekkloud.

The combination of Nekkloud and TemPSS offers an improved user experience and significantly simplified approach for running Nektar++ computations on remote platforms. It also helps to decouple the interactions between the different entities involved in running a computation. Nonetheless, in its current configuration, Nekkloud still requires that a user manually selects the computing platform on which they want to run a job, requiring some end-user knowledge about the pros and cons of running on different platforms and an understanding of the possible costs or issues of selecting one platform over another.

4 Computing Platforms

We introduce four example computing platforms and our methodology for comparing their capabilities. Table 1 shows the result of applying this methodology.

1. **Standalone server:** A multi-core x86_64 server with 2x16-core processors.
2. **HPC cluster using the PBS batch scheduler:** A large-scale HPC cluster consisting of several multi-core nodes interconnected with InfiniBand™ low-latency networking and accessed via the PBS job scheduler.
3. **OpenStack private cloud:** An OpenStack private cloud IaaS system offering on-demand provisioning of virtual servers of different specifications.
4. **Amazon EC2 public cloud:** The Amazon EC2 [3] public cloud platform that offers on-demand, pay-per-use access to a variety of resources.

Our methodology for comparing the platforms involves the use of eight properties that are assigned values to rank platforms based on their pros and cons. The use of these properties to help select a platform under different requirements forms a type of multiple attribute analysis problem and an overview of various approaches to such problems is described in [16]. The properties are:

– **Cost (purchase and usage):** The cost to the end user or their institution of the initial platform purchase and the costs incurred at the point of use.

- **Contention:** This is worse where there may be a large number of users or a long wait for a job to begin processing and better where there are fewer users or dedicated access to a resource.
- **Technical knowledge:** The amount of technical knowledge required to use a platform without the use of additional supporting tools such as Nekkloud.
- **Capacity:** Specification in terms of CPU cores, disk storage and memory.
- **Flexibility:** The ease of accessing different types/amounts of computing capacity and being able to scale requirements up or down on a per-job basis.
- **Reliability/maintainability:** How likely a platform is to fail or become unavailable and the potential difficulty of maintaining it.
- **Communications:** The expected inter-node communication performance.

Assigning numeric values to these properties allows straightforward summation or application of weights to the parameter values, as used in the "additive weighting" approach described in [16]. We use a scale of values between 1 and 9 (inclusive) with 1 being the worst and 9 being the best. The size of the scale was selected to offer flexibility in the assigning of scores for parameters where there is considered to be similarity between two or more platforms. A smaller scale would reduce the scope for highlighting small differences between platforms while a larger scale provides an unnecessarily wide range of options. It is, nonetheless, accepted that there is an element of subjectiveness in the assigning of the values but it is still felt that they offer a good representation of platforms' capabilities and similarities. At the initial stage of this work we have opted to give all properties equal weights rather than applying a weighting factor to give values more or less significance. With the introduction of the decision support system in Sect. 6, there is the opportunity to introduce weights to the property values and to automate their selection in order to take user requirements into account.

Table 1. Summary of the pros and cons of different properties of the four example platforms described in this section. P1 = standalone server; P2 = HPC cluster; P3 = OpenStack cloud; P4 = Amazon EC2; '-' = Parameter not applicable.

	Cost		Contention	Tech knowledge	Capacity	Flexibility	Reliability	Comms	Total
	Purchase	Usage							
P1	7	7	7	6	1	2	5	-	35
P2	1	**8**	5	**7**	6	7	7	**9**	**50**
P3	5	7	7	3	5	6	4	2	39
P4	**8**	3	**8**	3	**7**	**8**	**9**	4	**50**

The values in Table 1 are based on example deployment scenarios for each platform. These scenarios are considered from the perspective of a researcher working in fluid dynamics wanting to use the platforms to undertake their processing jobs. Platform 1 is purchased and operated by the researcher's own team. The team needs the expertise to deploy and manage the resource but the researcher is then likely to have relatively uncontended access to it – it is

likely to be shared only with other team members. The limited core count is an issue and maintenance is the responsibility of the researcher or their team. Platform 2, the HPC cluster, is designed for large-scale computations and is optimised for performance, providing high-speed, low-latency interconnects between nodes. Such a platform is very expensive to purchase but it may be supported through direct institutional funding and made available for researchers to use free of charge. Jobs are managed by the cluster's batch scheduling system and on a busy cluster, a job may be queued and take some time to begin running. The OpenStack private cloud platform offers much of the flexibility of a standalone server but with the potential to gain access to larger numbers of resources. A significant issue is that resources are interconnected using standard gigabit ethernet networking. This, combined with virtualised network interfaces on the cloud instances, can result in a significant performance penalty for parallel codes that undertake lots of inter-node communication. Reliability/maintainability may be an issue since this is, again, a locally operated platform. Platform 4, the Amazon EC2 public cloud, is accessed in a similar way to the OpenStack private cloud. However, EC2 offers a much wider choice of resource types and physical locations, which can be important if data is available in and must be processed in a specific geographic location. Access to Amazon EC2 resources is charged on a per-hour basis with a range of pricing options depending on the planned usage profile.

5 Usage Scenarios

We now look at three example usage scenarios, providing an initial platform recommendation for each based on the values in Table 1. The recommendations are revisited after the decision support system (DSS) is introduced in Sect. 6.

Scenario 1: A researcher is preparing a conference paper presenting analysis based on a 3-dimensional mesh that they have developed to support a fluid flow problem for a civil engineering project. The work is a collaboration between the researcher (a mathematician), and a group of civil engineering researchers. Around 3,000 CPU hours of computation are required for the analysis and the paper deadline is in two days. The lead researcher has limited funding for CPU time on their project which they would ideally like to retain for subsequent modelling tasks. Their institution operates an HPC cluster that is free at the point of use to researchers needing to undertake computationally intensive analysis.

Discussion: It is clear that the most important aspect here is the speed with which the job can be completed. From a pure computation perspective, the researcher would be best to undertake their job on the dedicated HPC cluster, requesting, say, 1024 or 2048 cores and having their computation completed in approximately 1.5–3 hours. However, this does not take into account job queuing time which, depending on the current cluster load, could be substantial. An alternative option would be to start a number of pay-per-use resources on the Amazon EC2 public cloud. The main issue with this approach will be the cost of purchasing around 3,000 CPU hours of compute time from the cloud service.

Platform recommendation: Local HPC cluster.

Scenario 2: An engineering team is developing a model of a gas pipe in a new structure. To test the correctness of their model and identify any issues with it, the team need to run a number of small-scale parallel tests. The number of time-steps to be run will need about 50 CPU hours per test and it will be necessary to undertake multiple runs with varying input parameters. The team are experts in their scientific domain but they need to collaborate with a software expert in order to select a suitable parallel computing environment and correctly configure their problem parameters to be used in the flow modelling solver.

Discussion: In this situation, the engineering team need the computation results as soon as possible in order to evaluate them and adjust the input parameters for the subsequent run. A platform where jobs are delayed by long queuing times is likely to be impractical, even if the ultimate computation performance is very quick. A better option could be to use a platform that may be slightly slower in terms of raw computation but where there is no contention for resource access. Such a scenario is likely to suit an in-house private cloud platform, or simply a multi-core standalone server. Software may need to be deployed and possibly built from source code to suit the target resources. The team are likely to require support from a computer scientist, if the software needs to be built from source code, and from the platform operator to identify the most appropriate parameters to suit the chosen platform when building and running the code.

Platform recommendation: Local, standalone, multi-core server

Scenario 3: A doctor wishes to model blood flow through an artery. They have collaborated with a CFD expert and a computer scientist and developed a one-dimensional model on which their computation will be undertaken using a Navier-Stokes solver. While the doctor understands the scientific problem they are tackling, their collaborators provide domain-specific knowledge to select or build a solver suited to the particular task. Computational requirements here are relatively low when compared to large three-dimensional models but funding is not available for computing time so low cost is the most important requirement.

Discussion: Time is not a significant constraint in this scenario. The lowest cost option depends on what resources are locally available to the doctor undertaking the modelling. We assume that a local server and institutional private cloud are available but not a cluster. For this computation, a local server offers the most straightforward option, despite the fact that computations may take some time to complete. However, a private cloud could offer the opportunity to scale computations and the developer working with the doctor should be equally capable of deploying their code to a local server or a remote cloud platform. The potential for complexity or delay that may result from having to work with a third-party platform is less of an issue with cloud infrastructure where a web based interface or API are normally available to start/stop and manage resources.

Platform recommendation: Institutionally-operated private IaaS cloud.

6 Decision Support System

6.1 System Model

As illustrated in the scenarios, constraints faced by users are heterogenous. While this means that access to different computing platforms is of critical importance, it also makes choosing the right platform quite complex, as all platforms have different properties. As a result, users are facing multi-dimensional trade-offs. There is obviously the matter of time and money (*cost* and *contention*). While this may, at first, look rather straightforward – there is a budget that cannot be exceeded and a deadline to be met – even such a trivial matter is actually not that simple. Indeed, what is the actual opportunity cost of a particular project? How much extra would a particular user or group of users be willing to pay to obtain their computation results a bit earlier? Conversely, how much money should they be offered to accept postponing completion of the computation for some period of time? In Table 1, property weights were considered to be equal. Yet, as with the property values assigned in Sect. 4, such weights are highly subjective and strongly depend on the project at hand. One task of the decision support system is to compute appropriate values for these weights given a particular scenario.

Previous literature [19] has emphasised the increasing complexity of choices in an environment when options are numerous and, as a result, the growing importance of decision support systems. In the case at hand, there is a clear need for a system that would enable identification of the most relevant option, based on the opportunity cost of each computational task. This question is particularly critical because the pricing of computing resources on certain types of computing platforms is often not stable (e.g. the 'spot' prices of computing units on Amazon EC2 [2] which evolve all the time). Being able to model the actual trade-off between time and money for a particular project provides the ability to identify opportunities as the project (and related computation) is being carried out. For instance, it may be possible to identify potential savings that can be achieved by delaying some of the computations. For some projects, this will be acceptable, but not for others. Hence the need to know the opportunity cost profile of the project. Determining this opportunity cost profile requires the following:

- knowing the 'super-boundaries': a maximum budget is sometimes not the actual maximum that can be spent on a project and even the strictest deadline might be extended by a small amount. Knowing what it would take for users to accept to 'go the extra mile' is critical to identify the best opportunities.
- knowing the 'boundary conditions': conversely, once a budget and time constraint have been assigned to a project, under which conditions is it acceptable to move away from these constraints?
- knowing the general opportunity cost of the project: considering the constraints of the project, what is the value of time (in money) for the users?

The first two points above can be determined through asking users a set of dedicated questions. The third point requires the use of an experimental methodology, such as the one used in [17] or described in [22]. When the project is carried

out by a group of users instead of a single user and there are several stakeholders (meaning that the opinion of several users – including some that may not be interacting directly with the computing environment – has to be taken into account), it is possible to use methods such as Discrete Choice Experiments [20] to infer the trade-offs of the group (weighting can also be introduced in case different stakeholders have different weight in the final decisions).

Another critical aspect of the optimal choice of platform relates to users' attitude towards risk. Indeed, depending on the project at hand, a failure of the computing platform can have more or less dramatic consequences. Since one can reasonably expect reliability and price to be correlated, the attitude towards risk can have an important impact on expenditure. Evaluating risk aversion is, however, quite straightforward: users are asked to answer a series of questions related to lotteries and their answers are used to build their risk profile [10,11].

The methodologies used to evaluate the opportunity cost and the risk aversion can be extended to cover the other properties of platforms. For instance, a Discrete Choice Experiment (DCE) can be built to encompass the properties listed in Sect. 4. A shortcoming is that when more parameters are included, users need to answer more questions for an accurate trade-off profile to be built. When only one or a few users are involved, this can become quite tedious. A way to alleviate this issue is to make use of gamification to help make the data collection process entertaining for participants.

6.2 Application to Scenarios

We now revisit the three scenarios introduced in Sect. 5 and look at how the application of the decision support system can help to improve on the naive recommendations made based on the values from Table 1 in Sect. 4.

Scenario 1: Analysis of a fluid flow problem for a civil engineering project.

Initial platform recommendation: Local HPC cluster.

Impact of the Decision Support System: Time and money are clearly conflicting in this scenario. knowing the *super boundaries* (i.e. investigating additional funding, however limited, that could be used towards the project) would enable the DSS to offer alternatives to the use of the HPC cluster, in particular if the job queue is long, e.g. by monitoring spot prices on platforms such as Amazon EC2 that offer variable or demand-based pricing. Knowing the user's *boundary conditions* would enable the DSS to advise on substituting local HPC resources for external resources, as the former become available. In this case, it is essential to estimate the actual 'exchange rate' between time and money: how much money saved makes it worth delaying the obtention of the results? Finally, risk is, in this case, worth considering. Facing both a nearing deadline and funding shortage, the user might be willing to consider less reliable, alternative options. Assessing risk aversion of the user would enable a DSS to advise on such options.

Scenario 2: An engineering team modelling a gas pipe in a new structure.

Initial platform recommendation: Local, standalone, multi-core server.

Impact of the Decision Support System: In such a scenario, choosing the right platform is no longer just a matter of time versus money, other characteristics such as contention have to be considered. It is in cases such as this that a DSS becomes particularly important. Firstly, contention boundaries can be defined by the team. Then, both *super boundaries* and *boundary conditions* related to contention versus other factors (e.g. time, money) are assessed at the team level (with each team member undergoing the testing procedure to assess the various trade-offs). This enables evaluation of each computational option and identification of the most suitable one. A further interesting aspect is that it may be the case that not all team members have the same objectives. For instance, a more suitable option for the engineers may require a significant amount of additional work for the support team. This is why the decision is based on the combined trade-offs of each stakeholder in the project and strict boundaries and property weightings can be set to ensure selection of the most relevant platform.

Scenario 3: A doctor wishes to model blood flow through an artery.

Initial platform recommendation: Institutionally-operated private IaaS cloud.

Impact of the Decision Support System: In this scenario, the role of the DSS is essentially to identify cheap substitute alternative options to the local private cloud as they become available. The issue is, of course, that while local resources are often free at the point of use, external resources may not be (e.g. Amazon EC2). In such a case, an optimal solution would most likely make use of internal and external platforms. A DSS can help determine the right balance between the two platform types. Indeed, an accurate estimation of the time-money tradeoff enables a calculation of how much computation can be diverted towards paid-for external platforms. Furthermore, since time is not a significant constraint in this case, less reliable (and presumably cheaper) options may be considered. However, the involvement of a technical team in the project is likely to impact on the risk profile of the project. Indeed, while the doctor may not bear the consequences of unreliable sources (since the local server is always available as a back up), switching between options leads to additional deployment work for the technical team. Consequently, as time is not of the essence, the trade-offs made by the technical team are given a greater weight in the project profile.

7 Conclusions

Running complex HPC applications in modern computing environments with a variety of available hardware platforms presents a number of challenges in selecting the most suitable resources to address a user's requirements. We have observed that one of the key aspects in helping to ensure that users obtain the most appropriate resources is ensuring that they understand their own requirements and risk profile in sufficient detail to make the most suitable choice from

the available options. However, it is clear that correctly identifying these requirements is key to obtaining the correct platform choice and to address this we introduce a decision support system. This system builds on approaches demonstrated in previous economics literature to help identify a user's risk aversion and the opportunity cost of different platform choices. The outputs of the decision support system can then be used to add weights to platform properties to help improve on the information provided by the platform feature matrix shown in Sect. 4 to obtain a platform choice.

In future work we aim to prepare an initial implementation of the decision support system and a framework for allowing straightforward addition of new platforms to the feature matrix. We intend to integrate this into the Nekkloud tool to provide a prototype platform recommendation feature for Nekkloud users.

Acknowledgements. The authors wish to acknowledge the Nektar++ team for their advice, particularly in relation to the scenarios and solvers. JC and JD acknowledge Imperial College London for funding the Pathways to Impact project "Simplifying High Performance Computing Access for the Nektar++ Framework" under Imperial's EPSRC Impact Acceleration Account. JC and JD also acknowledge EPSRC for their support of the completed libhpc (EP/I030239/1) and libhpc Stage II (EP/K038788/1) projects where Nekkloud and TemPSS were initially developed.

References

1. Adaptive Computing Inc.: TORQUE Resource Manager. http://www.adaptivecomputing.com/products/open-source/torque/. Accessed 19 July 2016
2. Amazon Web Services Inc.: Amazon EC2 Spot Instances Pricing. https://aws.amazon.com/ec2/spot/pricing/. Accessed 19 July 2016
3. Amazon Web Services Inc.: Elastic Compute Cloud (EC2) Cloud Server & Hosting - AWS. https://aws.amazon.com/ec2. Accessed 19 July 2016
4. Botti, S., Hsee, C.K.: Dazed and confused by choice: how the temporal costs of choice freedom lead to undesirable outcomes. Organ. Behav. Hum. Decis. Process. **112**(2), 161–171 (2010). http://dx.doi.org/10.1016/j.obhdp.2010.03.002
5. Cantwell, C.D., Moxey, D., Comerford, A., Bolis, A., Rocco, G., Mengaldo, G., de Grazia, D., Yakovlev, S., Lombard, J.E., Ekelschot, D., Jordi, B., Xu, H., Mohamied, Y., Eskilsson, C., Nelson, B., Vos, P., Biotto, C., Kirby, R.M., Sherwin, S.J.: Nektar++: an open-source spectral/hp element framework. Comput. Phys. Commun. **192**, 205–219 (2015). http://dx.doi.org/10.1016/j.cpc.2015.02.008
6. Cohen, J., Cantwell, C., Moxey, D., Nowell, J., Austing, P., Guo, X., Darlington, J., Sherwin, S.: TemPSS: a service providing software parameter templates and profiles for scientific HPC. In: 11th IEEE International Conference on e-Science (e-Science 2015), pp. 78–87, August 2015. http://dx.doi.org/10.1109/eScience.2015.43
7. Cohen, J., Darlington, J., Fuchs, B., Moxey, D., Cantwell, C., Burovskiy, P., Sherwin, S., Hong, N.C.: libHPC: software sustainability and reuse through metadata preservation. In: First Workshop on Maintainable Software Practices in e-Science, Chicago, IL, USA, position paper, October 2012

8. Cohen, J., Moxey, D., Cantwell, C., Burovskiy, P., Darlington, J., Sherwin, S.J.: Nekkloud: a software environment for high-order finite element analysis on clusters and clouds. In: 2013 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, Poster paper, September 2013. http://dx.doi.org/http://dx.doi.org/10.1109/CLUSTER.2013.6702616
9. Costa, P., Napper, J., Pierre, G., van Steen, M.: Autonomous resource selection for decentralized utility computing. In: 29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009), pp. 561–570, June 2009. http://dx.doi.org/10.1109/ICDCS.2009.70
10. Eckel, C.C., Grossman, P.J.: Men, women and risk aversion: experimental evidence. In: Handbook of Experimental Economics Results, vol. 1, pp. 1061–1073. Elsevier (2008). http://dx.doi.org/10.1016/S1574-0722(07)00113-8
11. Holt, C.A., Laury, S.K.: Risk aversion and incentive effects. Am. Econ. Rev. **92**(5), 1644–1655 (2002). http://dx.doi.org/10.1257/000282802762024700
12. Karniadakis, G., Sherwin, S.: Spectral/HP Element Methods for Computational Fluid Dynamics, 2nd edn. Oxford University Press, New York (2005)
13. Krauter, K., Buyya, R., Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. Softw. Pract. Experience **32**(2), 135–164 (2002). http://dx.doi.org/10.1002/spe.432
14. libHPC. http://www.imperial.ac.uk/london-e-science/projects/libhpc. Accessed 19 July 2016
15. Liu, C., Yang, L., Foster, I., Angulo, D.: Design and evaluation of a resource selection framework for grid applications. In: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC 2002), pp. 63–72, HPDC 2002. IEEE (2002). http://dx.doi.org/10.1109/HPDC.2002.1029904
16. MacCrimmon, K.R.: Decisionmaking among multiple-attribute alternatives: A survey and consolidated approach. Technical report MEMORANDUM RM-4823-ARPA, The RAND Corporation, Santa Monica, CA, USA (1968). http://www.rand.org/pubs/research_memoranda/RM4823.html
17. Payne, J.W., Bettman, J.R., Luce, M.F.: When time is money: decision behavior under opportunity-cost time pressure. Organ. Behav. Hum. Decis. Process. **66**(2), 131–152 (1996). http://dx.doi.org/10.1006/obhd.1996.0044
18. Raman, R., Livny, M., Solomon, M.: Matchmaking: distributed resource management for high throughput computing. In: Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, p. 140, HPDC 1998. IEEE Computer Society, Washington, DC, July 1998. http://dl.acm.org/citation.cfm?id=822083.823222
19. Rayna, T., Darlington, J., Striukova, L.: Pricing music using personal data: mutually advantageous first-degree price discrimination. Electron. Mark. **25**(2), 139–154 (2015). http://dx.doi.org/10.1007/s12525-014-0165-7
20. Ryan, M., Gerard, K., Amaya-Amaya, M. (eds.): Using Discrete Choice Experiments to Value Health and Health Care, The Economics of Non-Market Goods and Resources, vol. 11. Springer, Netherlands (2008). http://dx.doi.org/10.1007/978-1-4020-5753-3
21. Slawik, M., Küpper, A.: A domain specific language and a pertinent business vocabulary for cloud service selection. In: Altmann, J., Vanmechelen, K., Rana, O.F. (eds.) GECON 2014. LNCS, vol. 8914, pp. 172–185. Springer, Cham (2014). http://dx.doi.org/10.1007/978-3-319-14609-6_12
22. Smith, V.L.: Theory, experiment and economics. J. Econ. Perspect. **3**(1), 151–169 (1989). http://dx.doi.org/10.1257/jep.3.1.151

23. Tannenbaum, T., Wright, D., Miller, K., Livny, M.: Condor: a distributed job scheduler. In: Sterling, T. (ed.) Beowulf Cluster Computing with Linux, pp. 307–350. MIT Press, Cambridge (2002). http://dl.acm.org/citation.cfm?id=509876.509893
24. Univa® Corporation: Products Suite (2016). http://www.univa.com/products/. Accessed 19 July 2016