Journal of Computational Science 5 (2014) 51–62

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Flexible composition and execution of large scale applications on distributed e-infrastructures

Stefan J. Zasada, David C.W. Chang[1], Ali N. Haidar[2], Peter V. Coveney*

Centre for Computational Science, University College London, 20 Gordon Street, London WC1H 0AJ, United Kingdom

ARTICLE INFO

Article history:
Received 18 February 2013
Received in revised form 9 September 2013
Accepted 24 October 2013
Available online 1 November 2013

Keywords:
E-infrastructure
High performance computing
Application virtualization
Usability

ABSTRACT

Computer simulation is finding a role in an increasing number of scientific disciplines, concomitant with the rise in available computing power. Marshalling this power facilitates new, more effective and different research than has been hitherto possible. Realizing this inevitably requires access to computational power beyond the desktop, making use of clusters, supercomputers, data repositories, networks and distributed aggregations of these resources. The use of diverse e-infrastructure brings with it the ability to perform distributed multiscale simulations. Accessing one such resource entails a number of usability and security problems; when multiple geographically distributed resources are involved, the difficulty is compounded. In this paper we present a solution, the Application Hosting Environment,[3] which provides a Software as a Service layer on top of distributed e-infrastructure resources. We describe the performance and usability enhancements present in AHE version 3, and show how these have led to a high performance, easy to use gateway for computational scientists working in diverse application domains, from computational physics and chemistry, materials science to biology and biomedicine.

1. Introduction

Today's computational scientists face a growing number of challenges which affect their ability to fully exploit the computational resources, made available to them via so called *e-infrastructures* (such as PRACE, EGI or EUDAT in Europe, or XSEDE in the USA). Firstly, they have an unprecedented amount of computational power available to them, which will continue to grow inexorably in the future, presenting many opportunities as well as challenges to an increasing number of scientific disciplines that rely on computer based modelling and simulation.

Secondly, the architectures of these large scale high performance computing (HPC) machines point to a growing trend of computers comprised of hybrids of scalar and vector processors [1,2]. This requires application scientists to ensure their code is optimized to take full advantage of the hybrid architecture of a specific machine. Grid computing [3,4] has sought to simplify end user access to and use of HPC resources, but the middleware tools developed to realize the computational grid concept have seldom provided the transparency and ease of use envisaged [5]. The challenges described above are compounded when one attempts to invoke multiple resources, in order to achieve more than just the sum of their individual parts [6].

Alongside grid computing we have witnessed the development of cloud computing. Cloud computing represents a fast growing business model that seeks to commoditize computational infrastructure, and provide access to various distributed resources such as CPU, memory and storage (known as infrastructure services) and applications (software as services). It is a rapidly growing area due to major strategic investments from global software companies such as Microsoft, Amazon, Google and IBM. Cloud storage today is growing in popularity, particularly due to its shared data at low cost capabilities. Nonetheless, there are many security and legal issues in cloud computing that are yet to be resolved.

The Application Hosting Environment [7,8] is a middleware layer designed to simplify the user's ability to exploit computational resources beyond the desktop, greatly facilitating the use of e-infrastructure. It has been deployed in support of a diverse set of projects, including HIV-1 protease modelling [9], immune system simulation [6], and large scale materials modelling [10]. AHE seeks to converge the Software as a Service model of cloud computing with high performance grid computing. In this paper we will discuss the concepts behind AHE, and describe in detail the latest version of the Application Hosting Environment, AHE 3.0, which has been reimplemented using RESTful services [11] rather than WSRF services [12]. We will demonstrate how the work we have done to redesign AHE 3.0 has led to a significant increase in performance

* Corresponding author.
E-mail address: p.v.coveney@ucl.ac.uk (P.V. Coveney).
[1] Now at: The Graduate School of Biomedical Engineering, University of New South Wales, Sydney, Australia.
[2] Now at: HSBC, Canada Square, London, UK.
[3] AHE is available to download under the LGPL license from: https://sourceforge.net/projects/ahe3/.

1877-7503/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2013.10.009

compared to AHE 2.0 [8], and we show how this new version of AHE is benefiting various ongoing research projects.

2. Service oriented computational science

Virtualization is a broad term used in computer science to describe the abstraction of resources. Application virtualization describes a range of technologies designed to separate an application from the operating system that it runs on. In many cases this is achieved by introducing compatibility layers around underlying operating system features and libraries, for example the WINE system used to run Windows applications on UNIX systems [13].

The key aim of virtualization is to abstract away all the details of an underlying hardware or software system from the concern of the user. The benefits are manifold: developers can code to a single virtualized interface or system rather than for a specific hardware implementation; multiple virtual instances of a system can often be run side by side on a single physical system (in machine virtualization for example); and physical resources can be protected.

The growth of virtualization technologies, along with service oriented architectures (SOA), has also driven the development of cloud computing. The use of virtualized interfaces and systems means that the specific details of a cloud's architecture are hidden from consumers of the cloud resources. Several cloud computing models exist; the Infrastructure as a Service (IaaS) cloud paradigm typically takes the form of virtualized servers running on hardware platforms managed by the cloud hosting company, where each user is given access to one or more virtual servers, solely under their control. This also provides a degree of *elasticity*, as the number of virtual machines in a cloud environment can be greater than the number of physical servers available to the hosting entities. The Software as a Service (SaaS) cloud paradigm delivers access to applications centrally hosted on a cloud platform, typically via a web browser.

While virtualization technologies certainly reduce the complexity of using a system, and especially when working across multiple heterogeneous computing environments, they are not widely deployed in high performance computing scenarios. As its name suggest, HPC seeks to obtain maximum performance from computing platforms. Extra software layers impact detrimentally on performance, meaning that in HPC scenarios users typically run the applications as close to the 'bare metal' as possible. In addition to the performance degradation introduced by virtualization technologies, choosing what details to abstract in a virtualized interface is itself very important. Grid and cloud computing support different interaction models. In grid computing, the user interacts with an individual resource (or sometimes a broker) in order to launch jobs into a queuing system. In cloud computing, users interact with a virtual server, in effect putting them in control of their own complete operating system. Both of these interaction models put the onus on the user to understand very specific details of the system that they are dealing with, making life difficult for the end user, typically a scientist who wants to progress his or her scientific investigations without any specific usability hurdles obstructing the pathway.

To address these problems, we have developed a software layer designed to implement the Software as a Service cloud paradigm for scientific applications that rely on high performance computing, mediated by the *Application Interaction Model* which we describe in Section 3, derived from the user requirements also discussed in Section 3. This model is based on the insight that many e-infrastructures impose a steep learning curve on the majority of end users, who do not possess the technical expertise for the most part to compile, optimize, install, debug and finally launch their applications; they simply want to run their applications, obtain results and focus on their scientific endeavours. While an application may consist of a single execution of a computational code, it could also consist of a complex set of operations involving multiple codes, connected as a workflow; AHE enables all kinds of applications to be treated as simple "atomic" units, helping realize the original vision of a grid as "distributed computing performed transparently across multiple administrative domains" [14].

3. User requirements

For supercomputer class applications, the user generally has to install his/her own application, if that application is not one of the few community codes pre-installed on the machine; it is not possible simply to stage an executable to the target resource as it requires too much bespoke tailoring to the particular hardware setup of the resource. Generally a group of researchers will want to use the same application on a resource. However, many users will not know where a particular application is installed on a target system, nor will they necessarily know the best way to run the application on a particular system. Often, with supercomputer class systems, applications have to be run in specific ways to achieve the best performance. The community's expert users must spend time educating other users on the vagaries of different queuing systems and machines. Typically, the end user will need to stage data to the supercomputer before he/she is able to execute her application. Therefore, the supercomputer must provide accessible interfaces over which data can be staged. In order to launch an application, the users have to prepare a description of the job that they want to run, which is submitted to the queue management system on their target resource, in a format that the queue management system understands and which is potentially incompatible with other instances of the same queue management system running on other resources. Once the job has been submitted, users monitor the progress of their jobs through the queuing system, using interfaces provided by the resources.

Distributed applications can consist of multiple computational codes launched on multiple resources, connected together as workflows of operations, as well as single codes launched on single resources. Applications can get their data from multiple sources, such as online data repositories and databases, and store their output data in similar resources. Typically, users will be given allocations of time on individual grid resources, or the e-infrastructure as a whole, through awards made to their project's principal investigator. These allocations will have a notional associated cost, the cost per CPU hour, derived by the resource operator from their running costs and a projected resource utilization. Such allocation models inhibit the most creative use of and ways of exploiting distributed e-infrastructure.

The scientific end user's primary concern is running their application in a timely fashion, in order to obtain results that further their scientific objectives. All the services and facilities provided by a grid should be subservient to this end. Typically, the user does not even care which machine on the grid their application is run on, as long as results are delivered within a time frame that makes them useful, whether that is the time to publish a scientific paper, or the time to conduct a potentially life-saving medical simulation [15].

A further problem faced by end-users and administrators of computational e-infrastructures arises in connection with the usability of the security mechanisms usually deployed in these environments, in particular identity management. Many of the existing computational grid environments use Public Key Infrastructure (PKI) and X.509 digital certificates as a cornerstone for their security architecture. However, it is well documented that security solutions based on PKI lack user friendliness for both administrators and end-users, which is essential for the uptake of any grid security solution [16,17]. The problems stem from the process of acquiring X.509 digital certificates, which can be a lengthy one, and generating proxy certificates to get access to remote resources as part of the authentication process [17]. As a