Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

Buttressing volatile desktop grids with cloud resources within a reconfigurable environment service for workflow orchestration

Stephen C Winter[1*], Christopher J Reynolds[1], Tamas Kiss[1], Gabor Z Terstyanszky[1], Pamela Greenwell[3], Sharron McEldowney[3], Sandor Acs[1,2] and Peter Kacsuk[1,2]

Abstract

Cloud technology has the potential for widening access to high-performance computational resources for e-science research, but barriers to engagement with the technology remain high for many scientists. Workflows help overcome barriers by hiding details of underlying computational infrastructure and are portable between various platforms including cloud; they are also increasingly accepted within e-science research communities. Issues arising from the range of workflow systems available and the complexity of workflow development have been addressed by focusing on workflow interoperability, and providing customised support for different science communities. However, the deployments of such environments can be challenging, even where user requirements are comparatively modest. RESWO (Reconfigurable Environment Service for Workflow Orchestration) is a virtual platform-as-a-service cloud model that allows leaner customised environments to be assembled and deployed within a cloud. Suitable distributed computation resources are not always easily affordable and can present a further barrier to engagement by scientists. Desktop grids that use the spare CPU cycles available within an organisation are an attractively inexpensive type of infrastructure for many, and have been effectively virtualised as a cloud-based resource. However, hosts in this environment are volatile: leading to the tail problem, where some tasks become randomly delayed, affecting overall performance. To solve this problem, new algorithms have been developed to implement a cloudbursting scheduler in which durable cloud-based CPU resources may execute replicas of jobs that have become delayed. This paper describes experiences in the development of a RESWO instance in which a desktop grid is buttressed with CPU resources in the cloud to support the aspirations of bioscience researchers. A core component of the architecture, the cloudbursting scheduler, implements an algorithm to perform late job detection, cloud resource management and job monitoring. The experimental results obtained demonstrate significant performance improvements and benefits illustrated by use cases in bioscience research.

Keywords: Workflow; Cloudbursting; Hybrid cloud

Introduction

Cloud technology has the potential to increase access to a much wider range of powerful computational resources for advancing e-science research, but in practice, rapid uptake by scientists has been impeded by a number of factors, most notably that cloud computing, as with grid computing in former years, typically requires advanced understanding of underlying computing technologies which many scientists do not possess. Barriers to engagement with cloud technology are high, and as a result the potential benefits of high performance computing in the cloud are largely unrealised. Computational workflows can help overcome the barriers: they hide details of underlying computational infrastructure and are portable between different cloud platforms. They also provide an intuitive framework for expressing computational requirements, and as a result, are increasingly accepted within e-science research communities in both industry and academia.

* Correspondence: wintersc@wmin.ac.uk
[1]Centre for Parallel Computing, University of Westminster, London, UK
Full list of author information is available at the end of the article

Workflows are not without their own challenges. There are for example, many competing workflow systems available and different communities adhere to different systems. Developing and even using workflows can be daunting for non-computer scientists. These issues have been taken up by a number of projects including: SHIWA [1], which focuses on workflow interoperability and SCI-BUS [2], which provides web-based scientific gateways that include graphical workflow development support tools for different science communities. Still, deployment of such useful environments can be challenging – the systems tend to be large, addressing a wide span of computational targets that currently includes CPU-clouds, grids and local clusters, each with a huge range of possible operating system software and middleware. The resulting codes tend be rather cumbersome, especially in cases where the end-user requirements are actually quite straightforward. A solution to this problem is to decouple the code into components and deploy components only as needed. RESWO (Reconfigurable Environment Service for Workflow Orchestration) is an emerging virtual platform-as-a-service cloud model based on technologies developed originally in projects such as SHIWA and SCI-BUS. The cloud platform service permits customised environments to be assembled more precisely in accordance with the needs of the user, which then can be redeployed in the cloud as a software service.

Another barrier for scientists wishing to engage with the cloud is the ready availability of suitable and affordable computational resources. Desktop Grids (DGs), based on the principle of using spare CPU cycles within an organisation, are an attractively inexpensive type of distributed computing infrastructure (DCI) for many users. The EDGI project [3] has focused on the use of workflows in DGs and on the interoperability of DGs with Service Grids (SGs). DGs have subsequently been effectively virtualised as a cloud-based resource. However, DG hosts (usually desktop PCs) are volatile: users log on at random times, taking over the host and in most environments, pre-empting the local task contributing to the DG computation. Completion of the task is inevitably delayed, affecting overall performance. This can be overcome through cloudbursting: the dynamic deployment in real-time of durable (i.e. highly reliable and available) cloud-based CPU resources that execute replicas of jobs that have become delayed. The DG is thereby effectively buttressed with CPU-cloud resources, potentially improving its performance.

This paper presents the application of workflow within an experimental prototype RESWO environment that targets a DG that may be buttressed with cloud-based CPU resources, to support the aspirations and requirements of bioscience researchers. The performance enhancement achieved through cloudbursting is demonstrated experimentally, and the experiences of the bioscientists in using the system are reported. Computational workflows and cloud-based DCIs for scientific computing are discussed in Sections 'Workflow-oriented environments for scientists' and 'Desktop grids within the cloud ecosystem' respectively. In Section 'Buttressing desktop grids through cloudbursting', the architecture of a hybrid cloud-based DCI based on a DG buttressed with durable CPU resources to support two experimental use cases is introduced. In section 'Performance evaluation', the performance results, and end user experiences, are described for two bioscience use cases.

Workflow-oriented environments for scientists

Computational workflows allow program developers to focus on the composition of reusable legacy programs to create larger and more powerful applications and as a result have a strong impact on high-performance application development [4]. The use of workflows for orchestrating services and jobs within complex execution scenarios is becoming established in e-science research, but lack of interoperability between different workflow systems makes the reuse of workflows difficult, impeding the pace of adoption in academia and industry [5]. Approaches to interoperability include translation (e.g. CppWfMS [6]) and embedding (e.g. VLE-WFBus [7]). The SHIWA Simulation Platform (SSP) [8] enables both approaches and allows different scientific user communities to share and re-use workflows amongst themselves. SSP uses WS-PGRADE/gUSE [9], a derivative of the P-GRADE Portal technology [10] that provides a web-based, service-rich environment for the development, execution and monitoring of workflows on various DCIs. It therefore offers platform interoperability allowing individual workflows to execute across hybrid platforms comprised of multiple SGs and DGs. Adoption of workflows by scientists is also facilitated by availability of easy-to-use lifecycle support tools and environments. The SCI-BUS project extended the development of a generic gateway framework of WS-PGRADE/gUSE and created a methodology for customising gateways for the support of workflow-oriented development and end usage. The framework incorporates support for deployment, management and accounting/billing on various clouds supported by the CloudBroker DCI service broker [11]. The WS-PGRADE/gUSE platform and CloudBroker platform have been integrated in SCI-BUS enabling the development and execution of P-GRADE workflows on virtually any DCI. In both SCI-BUS and SHIWA, target DCIs could typically include any combination of SGs (e.g. ARC [12], gLite [13], Globus [14], UNICORE [15]), DGs (e.g. BOINC [16], OurGrid [17], XtremWeb [18], etc.) and clouds (e.g. Amazon EC2 [19], IBM SmartCloud Enterprise [20], OpenStack [21], Eucalyptus [22] and OpenNebula [23]).

Thus SCI-BUS and SHIWA provide core technologies that facilitate adoption by scientists in both industry and academia.

Deployment of SCI-BUS and SHIWA-based environments can be demanding, even where the end user requirements are comparatively modest. RESWO (Reconfigurable Environment Service for Workflow Orchestration) is a cloud-oriented model based on the core technologies of SCI-BUS and SHIWA, aimed at the construction of leaner customised environments. It exploits virtualisation to allow rapid assembly, customisation and deployment of complex, distributed infrastructures on demand. Workflow developers may then create customised configurations of SSP elements and host this configuration in a virtualised environment such as an Infrastructure-as-a-Service (IaaS) in the cloud. Through the graphical portal interfaces derived from SCI-BUS, various custom workflow development, sharing, executing and interoperability scenarios can be supported. The aim for each customised instance is to provide only those interfaces, workflow engines, workflow and workflow engine repositories, web services, discovery services, monitoring services and DCI services required by the workflows that end users plan to run. RESWO itself can be offered as a cloud-based service Platform-as-a-Service (PaaS), accessible via a portal. Workflow developers may then easily assemble their own customised environment for deployment on the fly, allowing workflows to be searched for, discovered, deployed, executed, and monitored. The dynamically generated workflow orchestration environment features all the advantages of SHIWA and SCI-BUS. For example, user communities traditionally working with a particular workflow system may easily accommodate workflows developed under a different system by a different community, using the workflow interoperability features of SHIWA. However, since the environment is dynamically customised it may be configured to include only the necessary components for the target user scenario. The RESWO approach also offers scientists considerable flexibility of provision at DCI level that may include combinations of clouds, clusters, service grids, and desktop grids. Each of these types of DCI has particular characteristics, attractions and restrictions for different user communities. For example, applications are often constrained to run only on particular infrastructures. Support for a hybrid multi-DCI allows such applications to be easily mapped onto an appropriate DCI within the system.

Desktop grids within the cloud ecosystem

DGs use the spare CPU cycles available within an organisation or a community, usually sourced from desktop PCs. In a *local* DG the resources are all within a single organisation and typically, and are managed centrally. In a *volunteer* DG, resources are provided on a voluntary basis by citizens, from around the world. In both cases, nodes are configured to be used within the DG according to the node owner's preferred policy. Typically this occurs whenever the nodes are not being used locally. The PC runs a client daemon that requests computation tasks from a centralized DG server that dispatches work. Virtually all institutions maintain a large number of networked desktop PCs, and together the power of these machines may be harnessed to provide cheap "off-peak" computing power in the form of a local DG. Universities in particular invest in large numbers of laboratory PCs that are practically unused outside of teaching hours. Generally DGs are well-suited to so-called embarrassingly parallel, or "bag of tasks", computations [24], since nodes communicate only with the server and not with each other. As a result of developments in the EDGI project, DGs have been effectively virtualised and as a consequence may be considered as a particular type of cloud-based resource.

A DG is an example of a very cheap DCI for scientific research (it is in effect "free", from the point of view of most researchers), but has performance limitations arising not only from intrinsic physical size limitations, but also the volatile nature of its underlying desktop PC nodes. In local DGs, the maximum number of compute resources is limited by the total number of PCs in the organisation. Actual availability (i.e. the number of machines available to the DG user in accordance with the DG use policy at any particular time) can vary hugely according to local usage – ranging from very high (for example during the night) to around very low (for example during a typical working day). There are occasions when any limitation, no matter how large or small, becomes restrictive. At such times, the ability to access additional resources for a short period, seamlessly and without perturbing the overall workflow, would be very useful. For example, in one common scenario, a scientific researcher based at a university has routine access to cheap local desktop grid (DG) resources, and uses these wherever possible, on the grounds of cost. Occasionally however, they also wish to access additional resources to achieve short bursts of highly intensive computation in response to deadlines, or to manage peaks of data throughput.

A second performance issue arises from the *tail problem* [25] - a well-known phenomenon in DG computing arising from the volatility of the underlying desktop PCs, which can seriously undermine the makespan (i.e. completion time) of a collection of submitted tasks. A parameter sweep computational stage within a workflow, or indeed any batch submission, is dependent upon the finish time of the last job of the batch. If some of the tasks become delayed, the completion time of the last job in the batch may be some while after the majority of the jobs have finished, thus seriously affecting the overall completion time of the batch. A late job in this context is called a

lagger, and the period at the end of the parameter sweep computation when only lagger results are outstanding is termed the *tail phase* of the computation.

The phenomenon is illustrated in Figures 1 and 2, which assumes a constant statistical pattern of lagger behaviour in two cases. For large batches (Figure 1) the impact of late jobs arriving can be fairly modest – but for small batches (Figure 2), this effect can be considerable. Figure 3 depicts the outcome of a concrete experiment in which a small batch of 50 jobs was submitted to the University of Westminster local campus DG. This DG comprises student laboratory PC hardware across the university, and utilizes a BOINC-based middleware that has been adapted for local DG computing. In the depicted run, the DG is offering approximately 1000 active nodes and employs a first-come-first-served (FCFS) scheduling policy and timeout values of 25 minutes, after which the job is considered late by BOINC and is replicated on another node. The jobs within the batch comprise a parameter sweep computation, in which the same function is applied to a range of prescribed input data sets - in this case employing the Autodock molecular docking tool [26] to find binding sites between a simple ligand molecule and a more complex protein. Clearly the makespan is affected adversely by the few late jobs towards the end, which had been scheduled to run on a number of nodes that were interrupted by students in the middle of the execution - the University's DG policy requires that a DG job shall be suspended when a student uses the host laboratory PC.

The DG scenario is a potent example of the need for a RESWO system to incorporate dynamic strategies for mapping tasks onto multiple DCIs; in the DG case to consider dynamically augmenting the volatile (albeit cheap) DCI with more reliable (albeit expensive) resources – that could be derived from the cloud.

Buttressing desktop grids through cloudbursting

Minimizing the makespan for batches submitted to DGs is discussed in [27-29]. The approach is typically to control the scheduling mechanism within the DG server, by manipulating task selection and node selection, optionally with the replication of laggers onto idle available resources. Whilst these algorithms can be efficient, for high throughput (when the batch is large) the FCFS algorithm employed by most DGs is near optimal [25]. However when the batch size is less than or approximately equal to the number of compute nodes, more sophisticated methods of resource selection and task replication are usually necessary. Strategies involving intelligent task and machine selection have been proposed to address the DG tail problem using DG resources only. For example, tail jobs can be replicated on historically more reliable and faster DG nodes. Alternatively, the number of compute jobs submitted for a given desired result can be increased above the minimum necessary, with a view to discarding (i.e. simply not waiting for) the last few results. However, the former is still constrained by the volatile nature of the DG in terms of resource availability, whilst the latter is inefficient since much redundant computation is performed. Both strategies are open-loop control strategies, in that they make no attempt to take account of the observable behaviour of the system as the computation progresses. Thus, using the DG alone, albeit with enhanced scheduling policies, is not always adequate to solve the tail problem.

SpeQuloS, a quality of service (QoS) model for enhancing the performance of "best effort" DCIs, i.e. DCIs in which service providers do not guarantee that resources will remain available during the lifetime of any running application, has been described in [30]. One particular scenario accommodated by SpeQuloS is the use of cloud (specifically cloud spot) instances as a source of durable CPU resources to buttress a DG, coupled with a task duplication strategy using cloud resources to mitigate the tail problem inherent to DGs. Evaluation of this scenario based on simulations has indicated very encouraging performance improvements. SpeQuloS is concerned with providing quality of service (QoS) for a given batch of

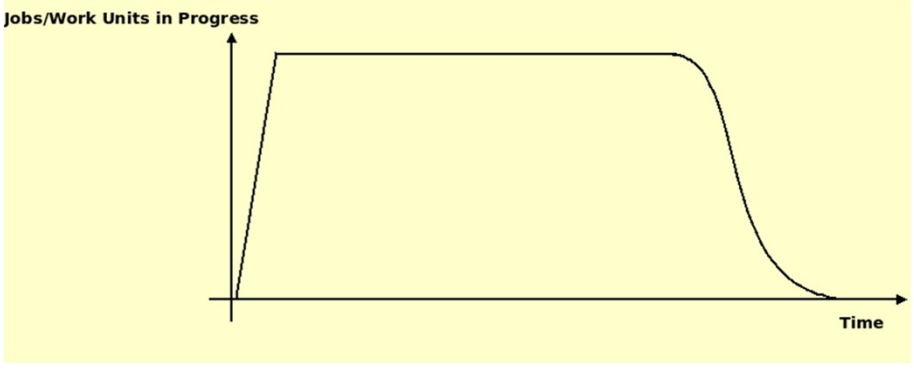

Figure 1 The tail problem in DG computing showing modest impact on larger jobs.

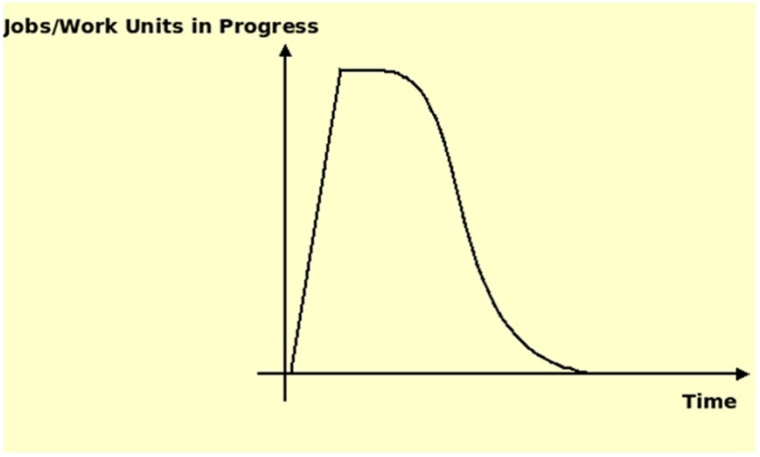

Figure 2 The tail problem in DG computing showing considerable impact on small jobs.

tasks through a computational credit exchange system, in which credits accrued from previous DG processing by one institution (the donor) on another institution's behalf (the requester), can be exchanged back for cloud resources at the requester's site when the donor requires them.

In the project "Optimal Scheduling of Scientific Application Workflows for Cloud-augmented Grid Infrastructures" (OSCA) [31], the authors of this article have investigated a similar scenario: the performance enhancement of DG applications obtained by buttressing the DG infrastructure with durable cloud resources. Specifically, a system has been designed and built to replicate laggers in an EC2-compatible cloud to avoid waiting for the return of late tail jobs that prolong the makespan. In contrast to SpeQuloS, the OSCA approach assumes the use of any public EC2 (Elastic Cloud Compute) compatible cloud such as Amazon or Openstack, and there are no limitations to the resources that can be bought. In addition, synchronisation between the DG and the CPU cloud is realised via a specific interface, the 3GBridge [32] (described below) that allows the DG scheduler to be the sole scheduler in the system; an implementation detail that modifies slightly the duplicated scheduler model assumed in SpeQuloS. The approach also incorporates the reuse of cloud instances, as a means of reducing

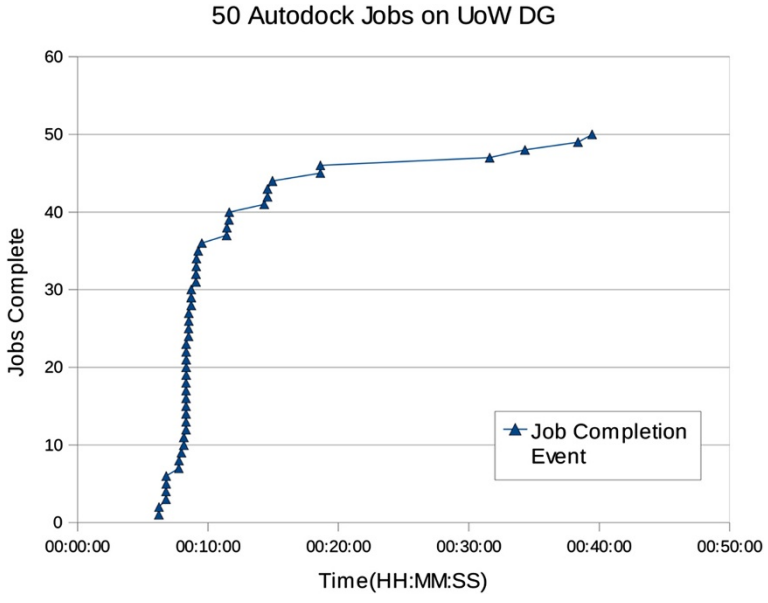

Figure 3 Submission of 50 Autodock jobs to UoW DG illustrating the tail problem. The makespan is increased significantly by the last 4 jobs.

overall cloud costs. A major aim of the project was to explore the DG/cloudbursting strategy in the context of realistic use cases, and to measure the performance of an actual physical deployment. In the experimental platform, durable cloud resources were sourced from an experimental cloud infrastructure at the University of Westminster: a private Openstack-Nova cloud consisting of 30 CPU cores, in contrast to the cloud spot instances assumed in SpeQuloS. Whilst cloud resources are generally billed by the hour, the OSCA system permits a more flexible method of resource provisioning and billing. This is due to the fact that the scheduler can maintain information about the state of the replicated jobs on Portable Batch System (PBS) clients/cloud instances. With this information, the scheduler can reassign those instances that no longer have jobs assigned. For example a free instance that has already been purchased may have used only 50% of its purchased time. The scheduler can reassign the instance to a new batch in its tail phase instead of simply releasing the instance that has already been paid for.

The architecture of the OSCA hybrid DG/cloud infrastructure is shown in Figure 4. In step 1, workflows developed by users in the P-GRADE environment are sent to the 3GBridge: a submission backend for P-GRADE developed in the EDGeS Project [33], conceived to achieve interoperability between DG and SG infrastructures. The 3GBridge is a fully decoupled submitter capable of creating jobs for different infrastructures including service grids, BOINC-based DGs (via a DG plugin), and EC2 clouds (via a cloud plugin that starts cloud instances). In step 2, jobs are submitted entirely to the DG, after which (in step 3) the scheduler component begins to monitor the status of the batch to determine whether the tail phase has been entered, according to a predefined metric, e.g. fixed percentage of tasks complete. Once the tail has been detected, virtual machine (VM) instances are instantiated in the cloud (step 4) following an instruction from the scheduler to the 3GBridge to use the cloud plugin. The VM instances connect back over a Virtual Private Network (VPN) to a PBS server (step 5), and register their instance identifier and hostname in a database. Since the instances are running PBS clients, the scheduler can submit replicated tail jobs to the PBS server, explicitly defining the instance to which a given replicated job is submitted. Finally, in step 6, the scheduler monitors both the replicated tail jobs in the cloud, and the tail jobs that are still processing (albeit running late) on the DG. Whichever returns a successful result first is used and returned to the portal. The *cloudbursting scheduler* algorithm at the core of the system is shown diagrammatically in Figure 5. A more detailed description of the algorithm and the developed system is given in [34].

Performance evaluation

Evaluation of the system was undertaken from two perspectives: the performance gains achievable in practice from the hybridisation of two formerly distinct classes of DCI; and the experiences of scientists newly engaged with e-science research objectives. The experimental platform used is based on a technologies developed in SHIWA and SCI-BUS, and complies with RESWO model principles.

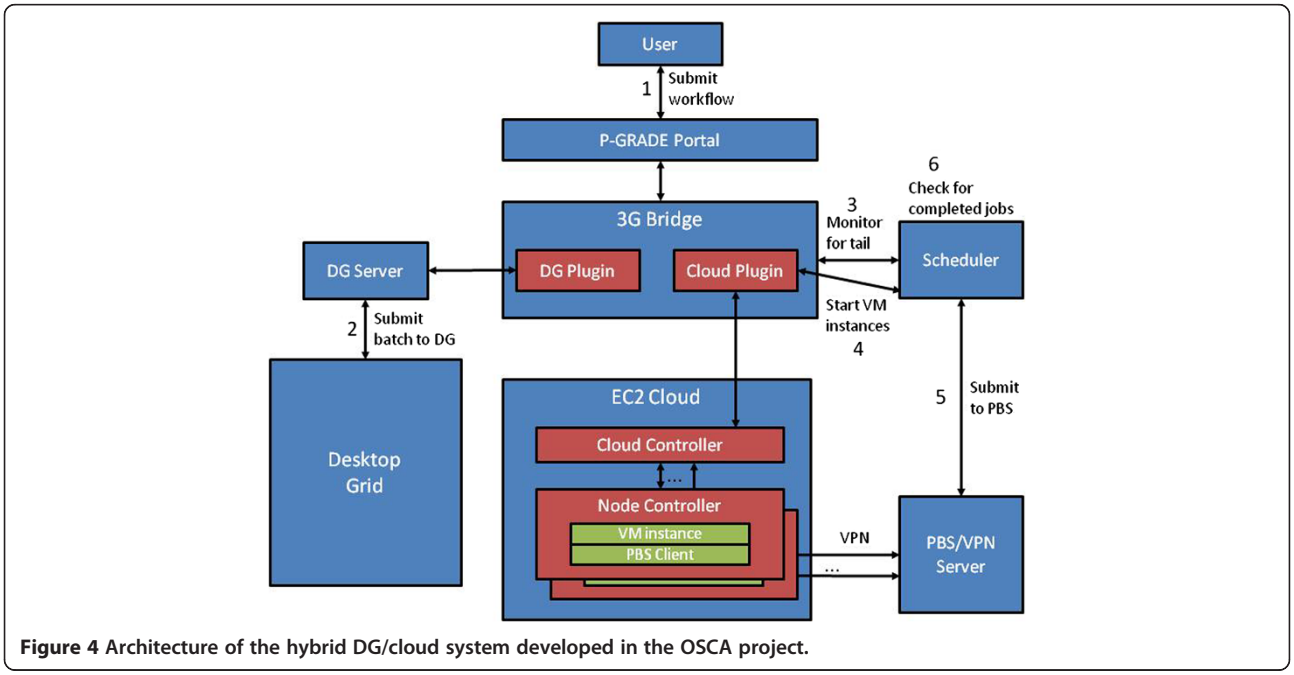

Figure 4 Architecture of the hybrid DG/cloud system developed in the OSCA project.

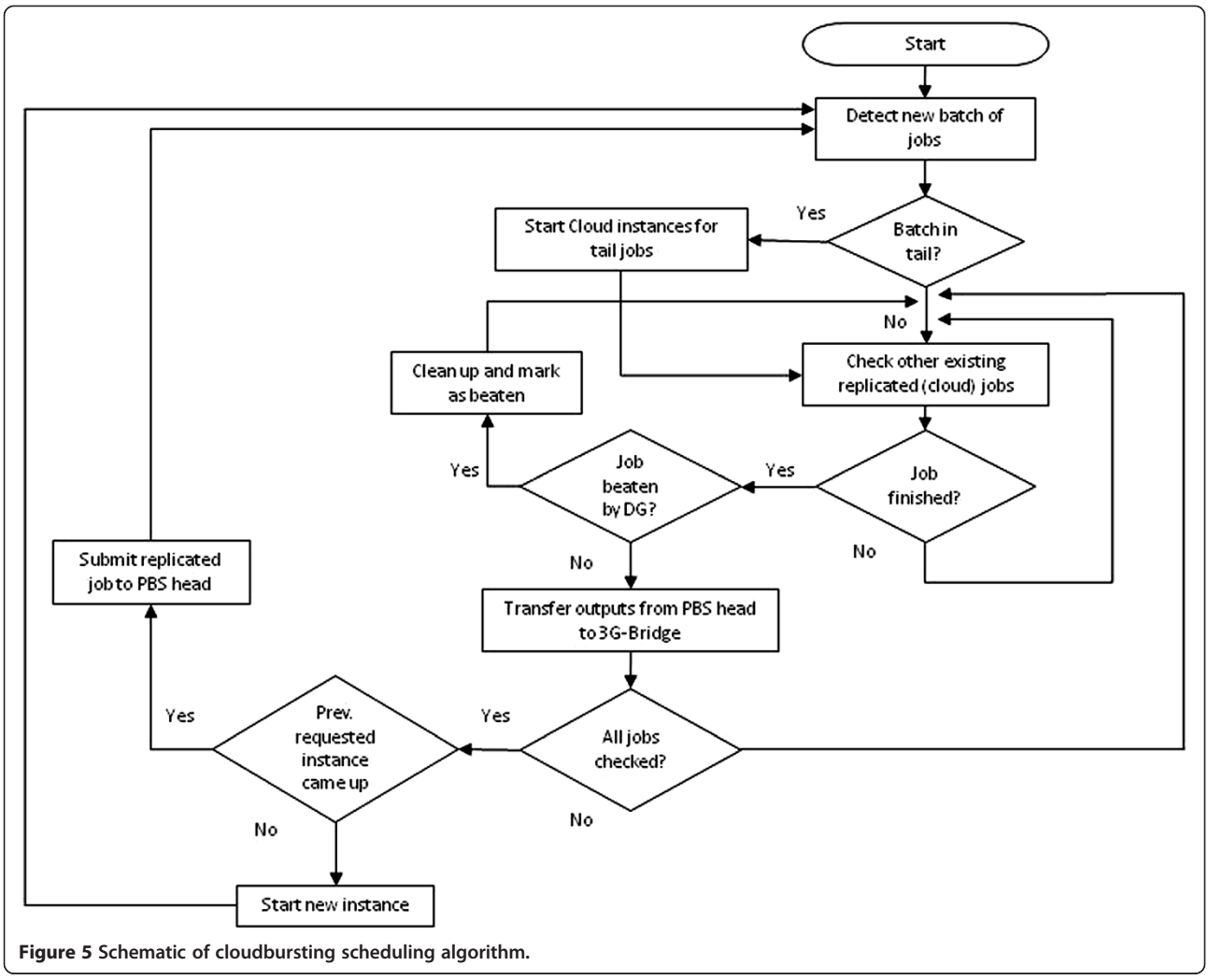

Figure 5 Schematic of cloudbursting scheduling algorithm.

System performance

The system was evaluated using a testbed comprising the P-GRADE portal, the 3GBridge, the University of Westminster (UoW) local DG [35], and an EC2 IaaS cloud of 30 CPU cores based on Openstack Compute (Nova) cloud software. The test workflow was the computationally dominant Autodock parameter sweep stage of the docking workflow shown in Figure 6 the computationally non-intensive pre- and post-processing stages being ignored. In this respect the system was evaluated based on a real-world workflow use case specified by UoW bio-scientists. Since the UoW DG was based on Windows PCs, the version of Autodock used in the experiments necessarily ran under Cygwin [36], a Linux simulation environment. A single instance of Autodock was mapped to a single CPU core in every case.

Many different batch sizes were investigated to explore the benefits of the system. In the experiments, the tail phase was pragmatically defined as the last 5% of the batch. Figure 7 shows a submission of a batch of 1000 jobs, for which the tail phase comprises the last 50 jobs. The figure illustrates the benefits of using cloudbursting, i.e. buttressing the DG infrastructure with cloud resources dynamically and on demand, compared with using the DG alone under a first-come-first-served FCFS scheduling algorithm.

Table 1 shows the mean percentage decrease calculated from 3 runs of the workflow. Performance is fairly consistent as indicated by the standard deviation value.

Autodock under Cygwin incurs a performance penalty: running approximately 3 times slower than in the native Linux environment for which Autodock was designed. Whilst this could be a significant issue in the case of serial execution, the problem can be readily overcome in a DCI based on DGs and clouds, for parameter sweep computations. For example, 3 parallel Autodock/Cygwin instances on 3 Windows nodes execute collectively in approximately the same time as 3 native Autodock

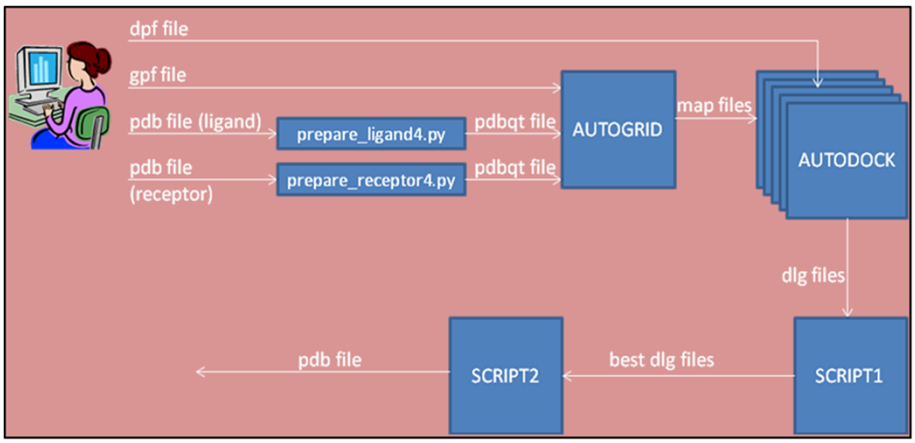

Figure 6 The workflow required by the bio-scientists. The Autodock parameter sweep stage was also the basis for system performance evaluation.

instances running serially on a single Linux node. Performance loss in the Windows environment can thus be recovered through the use of additional resources. In a cloud-buttressed DG, availability of resources is not a major problem, and the cost of using more hardware is not particularly prohibitive. For users, the DG is effectively "free" and under a pay-as-you-go model, cloud resources are also very cost efficient. A bonus for users in this scenario is that when a native Windows version of Autodock becomes available (as a result of a vendor upgrade), and is subsequent deployed on the DG, the benefits are immediately perceived by users as a 3-fold performance enhancement windfall.

Bioscience research Use cases

Biomedical research is increasingly being undertaken using *in silico* computer simulation – contrasting with conventional "wet laboratory" or *in vitro* methodologies - providing reductions in both experimental time and laboratory cost. *In silico* experiments, in common with their *in vitro* counterparts, are not typically constrained to a single stage or process; multiple stages of computation are

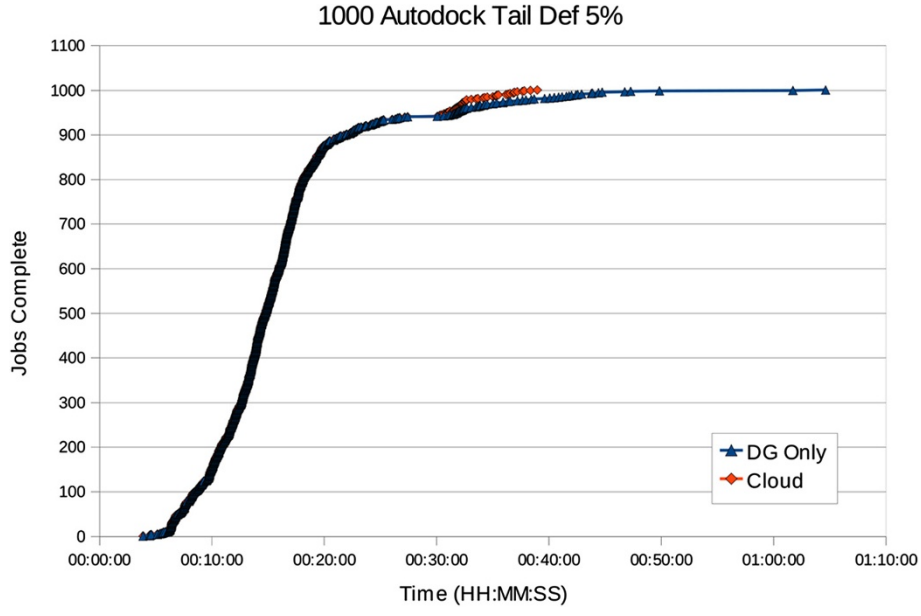

Figure 7 1000 Autodock jobs with tail defined at 5%. 30 CPU cores were used for the cloud. Each data point marks the completion of a job.

Table 1 Percentage decrease in makespan achieved by using 30 cloud CPU cores to process the tail (defined as the last 5% of jobs)

Run 1	Run 2	Run 3	Mean	Std Dev
34.3	39.6	39.7	37.9	3.1

common, usually with data needed as inputs to one stage provided by the outputs of a preceding stage.

To evaluate the utility of the hybrid DG/cloudbursting system as part of the *in silico* environment, two use case scenarios from bioscience research were explored.

Biological use case 1: design of carbohydrate-based vaccines

Carbohydrate recognition is critical to human biological functions. Examples include the highly specific responses of the immune system to pathogens and interaction and communication between cells. Understanding how pathogens bind to cell surface proteins can lead to the design of novel carbohydrate-based drugs and diagnostic and therapeutic agents. The bio-scientists were aiming to construct a library of tens of thousands of small molecule candidates available in databases such as DrugBank, that have been screened against known targets using molecular modelling and tools (in this case, Autodock). Such a small molecule library can be made available to other researchers for, say, *in vitro* validation, etc. The computational requirements here were to run Autodock many times for many small molecules screened for many docking sites.

Biological use case 2: effects of pharmaceutical compounds on aquatic organisms

Pharmaceuticals as environmental pollutants are detrimental to some organisms, e.g. vulture decline in India/Pakistan caused by veterinary drug diclofenac, and endocrine disruption of fish due to human contraceptives. Aquatic organisms are exposed to chronic low levels of pharmaceutical cocktails from sewage effluent. Major therapeutic classes are present in fresh water at environmentally relevant concentrations, e.g. analgesics, antibiotics, etc., but the lack of ecotoxicological data means effects on aquatic organisms of chronic, multi-generational exposure is scientifically uncertain. The bio-scientists wanted to establish whether currently available bioinformatics databases are a potential tool to predict the effects of pharmaceutical compounds on aquatic organisms. They needed to identify target species and chronic endpoints for the ecotoxicology testing of pharmaceuticals

In both use cases, the same *in silico* workflow pattern applies, and is shown in Figure 6. It comprises a preprocessing stages followed by a large parameter sweep computation, finishing with some post processing. This demonstrates the dynamically varying computational loads that often arise in scientific workflows. In Figure 8 the same workflow is shown as it represented within the P-GRADE environment.

In order to provide an even more convenient environment for bio-scientists to execute this workflow, which remains constant throughout the experiments performed by the scientists, a custom end-user interface has been built within the P-GRADE portal and is depicted in Figure 9. Using this interface, bio-scientists create individual workflow *instances* by uploading the necessary input files and setting input parameters. From the end user's point of view this will simply appear as a new task - the complexity of the underlying workflow (Figure 6) is completely hidden. After submission, the execution of the workflow can be monitored from the portal interface. The workflows are executed on the experimental hybrid DG/cloud infrastructure. Once the execution has finished, the resulting files (in this case depicting molecular representations) can be immediately visualised on the screen and or downloaded for further investigation and analysis.

The general observations made by the scientists were:

- Workflow development was natural, reflecting both intuition and previous experience based on *in vitro* methodology.
- The customised interface was highly appreciated, allowing quick and easy entry of data and parameters into the previously developed workflow.
- Individual workflow computations completed in a shorter time, allowing faster turnaround of the many experimental evaluations required.

A brief inspection of the two workflow representations in Figures 6 and 8 shows that the mapping between them is easy and very natural – one of the reasons that the bio-scientists have found the P-GRADE portal extremely usable. Such ease of use is a major requirement for acceptability in bioscience and other e-science communities. P-Grade has already stimulated growing interest within those communities that do not readily adopt computing-based scientific methods, or enjoy struggling with technologies perceived as arcane. The workflow-based portal has helped overcome barriers to take-up, enabling the bioscientists to concentrate on their scientific research. The computational benefits achievable by cloudbursting are greatly appreciated by the bioscientists, given that the total number of individual computations was extremely high. The requirement for hugely increased data throughput is expected to become very common within many scientific research processes. Typically, this will be driven by the quest for improved experimental reliability based on repeated runs, or, as in the case of molecular studies, the straightforward need to explore large data sets representing the myriad range of molecular structures.

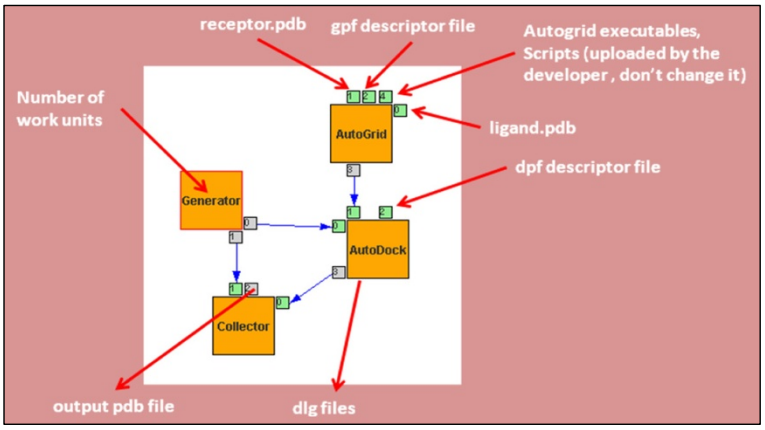

Figure 8 The same computational workflow as Figure 6 represented in the P-Grade portal.

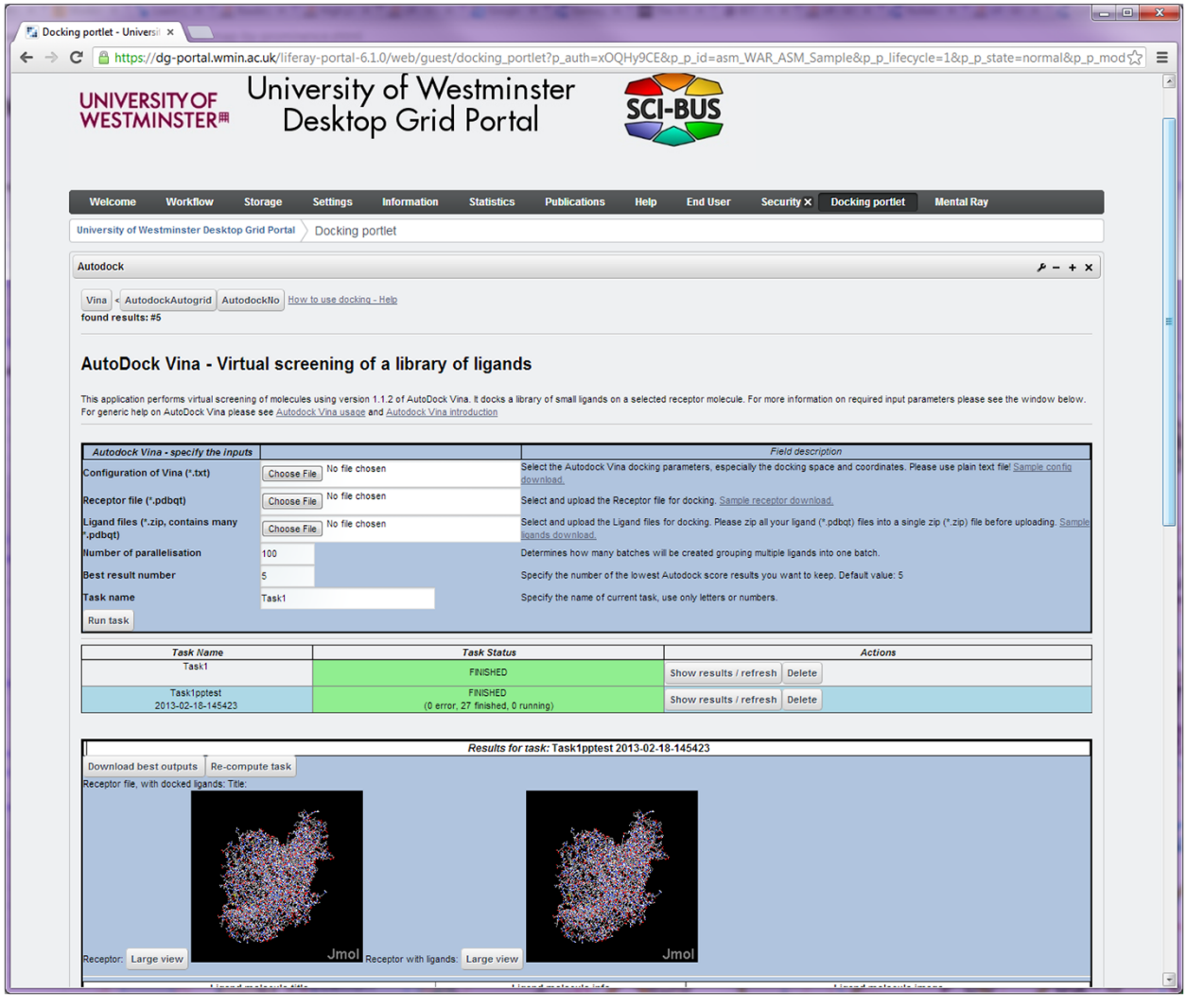

Figure 9 Custom graphical user interface for executing the workflow from the P-GRADE portal.

Further comment from the scientists regarding Use Case 1:

- Screening a large number of compounds for binding affinity against a single protein can be carried out *in vitro*, but is both time-consuming and expensive given the cost of buying the compounds to screen and the complexity of the experiments required. Novel microarray technology enables compounds to be placed on fabricated supports and these are screened using either labelled or unlabelled protein/peptide. However, the *in silico* approach exemplified by these experiments will enable researchers to screen hundreds of thousands of compounds and then determine the most likely drug candidates that could then be sourced and tested in vitro. This will be a more focused, rapid and cost-effective approach. However, in order to carry out thousands of docking experiments, grid- or cloud-based computing resources are required to deliver timely and robust data. In a related project to screen for novel neuraminidase inhibitors, an *in silico* workflow using Autodock was developed and ported. Molecular structures of small molecules were sourced and stored in the appropriate format in a library of 250,000 compounds. An additional 8,000 glycan structures were also sourced. It is now possible for a researcher to take a model of their protein of interest that has been energy minimised, and evaluate its binding partners within days compared to weeks/months using a single processor.

Further comment from the scientists regarding Use Case 2:

- In this scenario, interest is focused on the binding of known human and veterinary drugs and their homologues to specified proteins in selected vertebrates and invertebrates in order to help evaluate their potential toxicity. Currently, determining the acute and chronic toxicity of pharmaceuticals, and indeed industrial chemicals, in the environment is undertaken through ecotoxicity testing. These *in vitro* based approaches involve the exposure of the candidate species at different stages of their life-cycle to varying concentrations of potential toxins/drugs for varying times. It involves the use of animals and is a costly and time-consuming procedure. Utilising Gromacs (a molecular dynamics package primarily designed for bio-molecular systems such as proteins and lipids), an energy minimised structure is produced that can be used as a target in an Autodock experiment. This then can be interrogated with the drug of interest to determine whether a particular species would be a good candidate for *in vitro* trials. It would also enable us to broaden our search to look at potential impacts of specific molecules on the receptors from a range of environmental species rather than focusing on the species commonly used in ecotoxicological testing, which may not possess or express the proteins and hence would be poor indicator species for toxicity. In the past the *in silico* approach would have been time consuming, taking weeks to produce results. With the desktop grid/cloud based solutions such analysis can be completed within days or even hours rather than weeks or months.

Conclusions

Scientists operate in a multi-workflow, multi-DCI computational environment, but such environments are often difficult to engage with and inhibit wider take-up within mainstream scientific research. Interoperability solutions combined with portal technology offer solutions but are typically cumbersome; RESWO provides a model for efficient deployment of such environments. DCIs based on DGs are cheap, powerful and attractive to scientists. They are also volatile, but augmentation through cloud-based CPU resources, creating a multi-DCI platform, can overcome the drawbacks. Within the framework of the RESWO model, a system has been constructed that employs a novel algorithm for scheduling CPU resources in the cloud on demand to buttress the performance of a DG. The performance of the system has been evaluated, firstly from the point of view of technical performance improvements in overcoming the drawbacks of DG computation, and secondly from the scientific user perspective that includes ease of use, and research productivity. The results have demonstrated that considerable performance improvements on a simple DG can be obtained for realistic use cases. The scientists have been delighted by the environment, which they found easy to use for expressing their requirements, and delivering the results they required in a timely fashion.

Abbreviations

DCI: Distributed computing infrastructure; DG: Desktop grid; EC2: Elastic cloud compute; FGI: Fine grained interoperability; FCFS: First come first served; GUI: Graphical user interface; IaaS: Infrastructure as a Service; PaaS: Platform as a Service; PBS: Portable batch system; RESWO: Reconfigurable environment service for workflow orchestration; SG: Service grid; QoS: Quality of Service; UoW: University of Westminster; VM: Virtual machine; VPN: Virtual private network.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

SW, TK, GT, PK conceived of the study, and participated in its design and coordination and helped to draft the manuscript. CR, SA undertook software development, created the experimental framework, and performed the experiments PG, SM designed the experiments, and evaluated the results. All authors read and approved the final manuscript.

Acknowledgements
EPSRC/JISC for funding Project: Optimal Scheduling of Scientific Application Workflows for Cloud-augmented Grid Infrastructures. EPSRC Grant No. EP/I034254/1. EU FP7 for funding Projects: SHIWA (SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs), FP7 Project No. RI261585. SCI-BUS (SCIentific gateway Based USer Suppport). FP7 Project No. RI283481. EDGI (European Desktop Grid Initiative). FP7 Project No. RI261556. EDGeS (Enabling Desktop Grids for e-Science). FP7 Project No. RI211727.

Author details
[1]Centre for Parallel Computing, University of Westminster, London, UK. [2]Laboratory of Parallel and Distributed Systems, MTA SZTAKI, 1111 Kende utca 13, Budapest, Hungary. [3]School of Life Sciences, University of Westminster, London, UK.

Received: 2 April 2013 Accepted: 11 July 2013
Published: 13 January 2014

References

1. SHIWA (SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs), FP7 Project No., RI261585. http://www.shiwa-workflow.eu
2. SCI-BUS (SCIentific gateway Based USer Suppport)., FP7 Project No. RI-283481. http://www.sci-bus.eu
3. EDGI (European Desktop Grid Initiative)., FP7 Project No. RI-261556. http://edgi-project.eu
4. Taylor IJ, Deelman E, Gannon DB, Shields M (2007) Workflows for e-Science: Scientific Workflows for Grids. Springer, XXII. ISBN: 978-1-84628-519-6 (Print) 978-1-84628-757-2 (Online)
5. Kacsuk P, Kiss T, Sipos G (2008) Solving the Grid Interoperability Problem by P-GRADE Portal at Workflow Level. Future Generation Computing Systems: International Journal of Grid Computing. Theory, Methods and Applications 24(7):744–751, ISSN 0167-739X doi:10.1016/j.future.2008.02.008
6. Pellegrini S, Giacomini F, Ghiselli A (2007) A Practical Approach for a Workflow Management System. Proc. Core-GRID Workshop, Dresden
7. Zhao Z, Booms S, Belloum A, de Laat C, Hertzberger B (2006) VLE-WFBus: A Scientific Workflow Bus for Multi e-Science Domains. In: e-Science and Grid Computing, 2nd IEEE Int. Conf. on e-Science'06. IEEE Computer Society, Piscataway, NJ, p 11
8. Korkhov V, Krefting D, Montagnat J, Tram Truong H, Kukla T, Terstyanszky G, Manset D, Caan M, Olabarriaga S (2012) SHIWA workflow interoperability solutions for neuroimaging data analysis, Proceedings of Healthgrid'12, Studies in Health Technology and Informatics. In: Gesing S et al (eds) HealthGrid Applications and Technologies Meet Science Gateways for Life Sciences, vol 175. IOS Press, Amsterdam, pp 109–110
9. Kacsuk P et al (2012) WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities. Journal of Grid Computing 10(4):601–630
10. P-Grade Portal. http://portal.p-grade.hu
11. Cloudbroker GmbH. http://www.cloudbroker.com
12. ARC: The ARC Grid Middleware. http://www.nordugrid.org/middleware
13. Laure E et al (2006) Programming the Grid with gLite. Computational Methods in Science and Technology 12(1):33–45
14. Foster I (2006) Globus Toolkit Version 4: Software for Service-Oriented Systems. Journal of Computer Science and Technology 21(4):513–520
15. Erwin D (2002) UNICORE - A Grid Computing Environment Concurrency. Practice and Experience Journal 14:1395–1410
16. Anderson D (2004) BOINC: A system for public-resource computing and storage. In: Proc. 5th IEEE/ACM Int. Workshop on Grid Computing, 2004. http://boinc.berkeley.edu
17. OurGrid. http://www.ourgrid.org
18. XtremWeb, Fedak et al (2001) XtremWeb: A Generic Global Computing System. CCGRID2001 Workshop on Global Computing on Personal Devices. IEEE Press, Piscataway, NJ
19. Amazon EC2 (Amazon Elastic Compute Cloud). http://aws.amazon.com/ec2
20. IBM SmartCloud Enterprise. http://www.ibm.com/services/uk/en/cloud-enterprise
21. Openstack: Open source software for building private and public clouds. http://www.openstack.org
22. Eucalyptus: Open source software for building AWS-compatible private and hybrid clouds. http://www.eucalyptus.com
23. OpenNebula: Open source data centre virtualisation. http://opennebula.org
24. Candeia D, Araujo R, Lopes R, Brasileiro F (2010) Investigating business-driven cloudburst schedulers for e-science bag-of-tasks applications. In: Proc. IEEE 2nd Int. Conf. on Cloud Computing Technology and Science (CloudCom 2010). IEEE Computer Society, Piscataway, NJ, pp 343–350
25. Kondo D, Chien A, Casanova H (2007) Scheduling Task Parallel Applications for Rapid Turnaround on Enterprise Desktop Grids, Journal of Grid Computing., pp 379–405
26. Autodock automated docking tool. http://autodock.scripps.edu
27. Anglano C, Brevik J, Canonico M, Nurmi D, Wolski R (2006) Fault-aware scheduling for Bag-of-Tasks applications on Desktop Grids, 7th IEEE/ACM International Conference on Grid Computing. IEEE, pp 56–63. ISBN 1-4244-0344-8
28. Da Silva DP, Cirne W, Brasileiro FV, Grande C (2003) Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids. Proc Euro-Par 2003:169–180
29. Lee Y, Zomaya A (2007) Practical Scheduling of Bag-of-Tasks Applications on Grids with Dynamic Resilience. IEEE Trans on Computers 56(6):815–825
30. Delamare S, Fedak G, Kondo D, Lodygensky O (2012) SpeQuloS: Quality of Service Using Best-Effort Distributed Computing Infrastructures. In: Proc. 21st ACM Int. Symp. on High Performance Distributed Computing (HPDC'12). IEEE. ISBN 0-7695-1965-2
31. Optimal Scheduling of Scientific Application Workflows for Cloud-augmented Grid Infrastructures, EPSRC Grant No. EP/I034254/1. http://cloudresearch.jiscinvolve.org/wp/category/projects/scheduling-of-workflows
32. Marosi A, Kacsuk P (2011) Workers in the Clouds. In: 19th Int. Euromicro Conf. on Parallel, Distributed and Network-Based Processing (PDP2011). IEEE Computer Society, Los Alamitos, pp 519–526
33. EDGeS (Enabling Desktop Grids for e-Science), FP7 Project No. RI211727. http://www.edges-grid.eu/
34. Reynolds C, Winter S, Terstyanszky G, Kiss T, Greenwell P, Acs S, Kacsuk P (2011) Scientific Workflow Makespan Reduction Through Cloud Augmented Desktop grids. Proc 3rd IEEE Int. Conf. on cloud Computing Technology and Science (cloudCom 2011). IEEE Publications, Athens
35. Kiss T, Szmetanko G, Terstyanszky G, Greenwell P, Heindl H (2010) Molecular docking simulations on a combined desktop and service grid infrastructure. In: Proc. Third AlmereGrid Desktop Experience workshop: Desktop Grid applications for eScience and eBusiness. EnterTheGrid, Almere, The Netherlands, pp 23–27
36. Cygwin. http://www.cygwin.com

doi:10.1186/2192-113X-3-1
Cite this article as: Winter *et al.*: **Buttressing volatile desktop grids with cloud resources within a reconfigurable environment service for workflow orchestration.** *Journal of Cloud Computing: Advances, Systems and Applications* 2014 **3**:1.