Voss *et al. Journal of Cloud Computing: Advances, Systems and Applications* 2013, **2**:20
http://www.journalofcloudcomputing.com/content/2/1/20

Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

An elastic virtual infrastructure for research applications (ELVIRA)

Alex Voss[1*], Adam Barker[1], Mahboubeh Asgari-Targhi[2], Adriaan van Ballegooijen[2] and Ian Sommerville[1]

Abstract
Cloud computing infrastructures provide a way for researchers to source the computational and storage resources they require to conduct their work and to collaborate within distributed research teams. We provide an overview of a cloud-based *elastic virtual infrastructure for research applications* that we have established to provide researchers with a collaborative research environment that automatically allocates cloud resources as required. We describe how we have used this infrastructure to support research on the Sun's corona and how the *elasticity* provided by cloud infrastructures can be leveraged to provide high-throughput computing resources using a set of off-the-shelf technologies and a small number of additional tools that are simple to deploy and use. The resulting infrastructure has a number of advantages for the researchers compared to traditional clusters or grid computing environments that we discuss in the conclusions.

Introduction

Many problems at the forefront of science, engineering, medicine, arts, humanities and the social sciences require the integration of large-scale data and computing resources at unprecedented scales to yield insights, discover correlations, and ultimately drive scientific discovery. The explosion of scientific data produced by simulations, network-connected instruments and sensors, as well as the increase in social and political data available to researchers, promises a renaissance in many areas of science. To meet the data storage and computing demands of such data-rich applications, users in industry and academia are looking to cloud computing [1-3], which promises to realise the long-held dream of computing as a utility with users accessing resources and applications hosted by a third-party data centre. This model has several potential key advantages that have aided its rapid adoption:

- upfront costs are eliminated as resources are rented from a third-party provider;
- users only pay for the computing resources they need on a short-term basis (e.g., CPU by the hour, storage by the day);
- resources are elastic in nature and can dynamically 'scale out' to meet increasing demands and 'scale in' when demand is reduced;
- the cost of managing the underlying hardware and networking infrastructure is shared by many users of the cloud service and is reduced through virtualisation and automation.

Predictions of rapid growth of the cloud computing market abound (cf. [4]) but so far it would seem that adoption is often limited to specific applications. For example, while a recent report for the European Commission [5] suggests that cloud computing may "contribute up to €250 billion to EU GDP in 2020 and 3.8 million jobs", the same report highlights the fact that adoption is currently very uneven and 'shallow' (limited to specific applications such as email) and growth is dependent on a number of barriers being overcome. We know from a previous study [6] that the uptake of grid computing and similar e-Infrastructures for research was similarly uneven with a number of early adopter communities making regular use of these resources and indeed driving their development and the development of underlying technologies. At the same time, there is a 'long tail' of researchers working with more modest resources even when their work could potentially benefit from the use of more advanced e-Infrastructures. The (lack of) usability of tools and fit

*Correspondence: alex.voss@st-andrews.ac.uk
[1]School of Computer Science, University of St Andrews, St Andrews, UK
Full list of author information is available at the end of the article

with users' working practices played a crucial role in limiting the uptake of grid computing but there was also lack of a pathway from the demonstration of benefits through to early stages of skills acquisition and adoption through to routine use [6]. There is a danger that the adoption of cloud computing for research purposes might be held back by similar issues and might be limited to those who are already users of advanced e-Infrastructures such as clusters, HPC resources or grid computing.

Clouds and scientific computing

A recent report by the e-Infrastructure Reflection Group [4] suggests two models of adoption of cloud computing: for those researchers who are already users of existing e-Infrastructure services, those services can be extended to make use of cloud computing resources, keeping the user interfaces and usage models that these users are familiar with intact. For this group of users, who collectively make use of significant resources, the economics of resource provision favour a hybrid model of provision where some resources are provided through the community of users or their institutions while peak usage is burst out to a cloud infrastructure. "The exact threshold when the investment in [one's] own hardware and staff is more cost efficient than the usage of public clouds depends on the application and its utilization" [4].

At the same time, cloud computing offers new opportunities for researchers who – for one reason or another – are not currently users of existing e-Infrastructure services and whose requirements can be met by cloud computing resources [4]. As existing e-Infrastructures such as clusters and grids are often tailored to specific user communities, researchers outside those communities often find the costs of uptake prohibitive or find that their applications do not match the requirements profiles assumed by the resource providers. Cloud computing resources allow users to select instance types that better match the characteristics of their applications and to build runtime environments tailored to their needs.

The performance of cloud computing resources does not reach that of specialised high-performance computing (HPC) resources such as those that make up the Top-500 list of HPC resources [7] and their multi-tenant architecture and limitations on I/O performance may not make them suitable candidates for workloads that spread computation across multiple nodes [8]. However, offerings targeted specifically at compute-intensive applications such as Amazon's Cluster instance types [9] are beginning to address the performance gap [8], in terms of the compute performance of individual nodes but also the IO bandwidth and low latency required, through higher-performance network interconnects and the introduction of the concept of 'placement groups'. From the point of view of a provider of compute services, clouds are therefore primarily of interest as a way to absorb peak loads (through 'cloud-bursting'), extending the capacity of the infrastructure while physical resources support the base loads.

From the point of view of individual researchers or projects, other criteria may be of equal or even higher importance than performance alone. One key advantage of many cloud environments is that they provide virtual machine instances that are under full control of the client, from the kernel upwards. This means that researchers have full control of the software stack installed, thus avoiding problems with conflicts and changing configurations. Images used to produce results can be archived, providing the opportunity to replicate results at a later point. Cloud resources have the advantage of being available on demand and provisioned rapidly within public clouds – although performance can vary [10] and by default resource provision in many clouds is limited to a given number of instances. The economic model of clouds means that in ideal circumstances, the costs of runing n instances for an hour are similar to those for running a single instance for n hours – an effect called 'cost associativity', cf. [2]. In practice, overheads for instance creation and job submission as well as details of usage metering mean the costs will not be exactly the same [11]. However, in cases where a rapid turn-around time is desired, the on-demand and self-service nature of (public) clouds can be a signficant advantage over any alternative that cannot scale to the same levels or does not do so within the required timeframe.

Cloud computing interfaces

In principle the promise of cloud computing is a simple model of resource provision: virtual servers on demand. However, the practical issues involved in running research applications are significant. Current interfaces to Infrastructure as a Service (IaaS) clouds [2,3] are relatively low level and do not allow researchers to easily benefit from the elasticity that cloud infrastructures offer. The underlying logic for working with EC2-compatible clouds is the same in command-line tools such as the original EC2 tools [12], the equivalent Euca2ools [13] or graphical user interfaces such as the Amazon AWS Management Console [14] or Elasticfox [15]. Each of these interfaces deals with the same low-level concepts that are a direct representation of the EC2 cloud API. As a consequence, researchers have to deal with time-consuming and often error-prone tasks such as managing access credentials, selecting instance types, managing elastic IP addresses, as well as monitoring resource usage and starting, stopping and terminating instances in response.

In addition, the scientific applications and supporting components need to be deployed within instances and kept up-to-date. Input data needs to be staged in and output files retrieved. This keeps researchers from focusing directly on their scientific research. In essence, what is needed is not raw servers with a bare operating system but a runtime environment for the applications in question that is made available dynamically, on-demand and configured according to the researchers' needs; that is, IaaS needs to be turned into application-specific Platform-as-a-Service (PaaS) or Software-as-a-Service (SaaS).

Elastic virtual infrastructures for research applications

The Elastic Virtual Infrastructures for Research Applications project (ELVIRA, www.elvira-cloud.org) was funded under a joint EPSRC/JISC call for pilot projects for cloud computing in research. It set out to explore a novel way of using cloud resources by making use of the fact that clouds already provide a multi-tenant architecture that allows resource sharing between independent parties, opening up the possibility for the creation of *virtual private research environments* where groups of researchers can collaborate using relatively simple tools in an environment they have tailored to suit their specific requirements. This approach promises to avoid some of the problems often associated with traditional forms of sharing computational and storage resources such as the need for complex authentication and authorization mechanisms, restrictions on the available runtime environments, resource competition, delays in execution through job queueing as well as lack of support for interactive jobs.

In order to achieve this, we make use of functions already provided by cloud infrastructures rather than replicating them. Crucially, IaaS clouds provide a multi-tenant architecture at the infrastructure level using virtualisation and on-demand resource allocation. This is in contrast to grid infrastructures, which provide for multiple users at the operating system level and therefore give rise to issues of separating users, of allocating resources to them and of managing configurations (such as installed libraries). Using IaaS, we can move away from the model of a shared operating system environment and resources waiting for workloads to be submitted and instead create virtual private research environments on demand that are dedicated to a particular use and provide a user experience that differs in many important respects from that of current production grid infrastructures. Our aim in the project was to develop a set of tools, the ELVIRA tools, that enable the rapid creation of such research environments as high-level gateways to cloud resources.

In order to ensure that the development of the ELVIRA tools is informed by real-world requirements, we have worked closely with researchers from a number of disciplinary backgrounds. The case study we describe here involves the study of solar coronal loops but we have also supported computational algebra applications and studies of the use of Twitter during the Summer riots in England in 2011 as well as during the 2012 Olympic and Paralympic Games. Below, we will briefly discuss the Alfvén Wave Turbulence Simulation application used to study solar coronal loops and the requirements it gives rise to. This application is a good example of the kinds of code that are the target for the ELVIRA project – developed by two researchers with a background in mathematics rather than programming and under constant development in line with the ongoing development of the science behind it. Tracking changes to the code, ensuring its quality and the provenance of the results as well as adapting the code to more powerful compute resources have been key concerns for the researchers involved. Following this, we outline the different components our virtual private research environment used to support this application and discuss our experiences with using these tools.

Alfvén wave turbulence simulation

Magnetic fields play an important role in the heating of the solar corona. The energy for coronal heating likely originates below the photosphere. At the photosphere, magnetic elements are continually moved about by convective flows on the scale of the solar granulation. This results in magneto-hydrodynamic (MHD) disturbances that propagate upward along the magnetic field lines and deposit their energy in the corona. In coronal loops the magnetic fields may become twisted or braided on small spatial scales, and thin current sheets may develop where most of the heating occurs. Alternatively, energy may be injected into the corona in the form of Alfvén waves, which can be dissipated in a variety of ways. Alfvén waves have indeed been observed in the photosphere, corona and solar wind, but the role of such waves in coronal heating has not been clearly demonstrated [16].

Recently, van Ballegooijen et al. [16] developed a three-dimensional (3D) MHD model describing the propagation and dissipation of Alfvén waves in active region loops. The model includes a detailed simulation of Alfvén waves in the coronal part of the loop, as well as in the lower atmospheres at the two ends of the loop. As in earlier studies, the waves are generated by interactions of convective flows with kilogauss flux tubes in the photosphere.

The Alfvén wave turbulence model predicts that the properties of the Alfvén waves depend strongly on the profile of the Alfvén speed $v_A(s)$ as a function of the position s along the loop. If the coronal Alfvén speed is very high, there will be strong reflection of waves at the transition region (TR), and the fraction of energy entering the corona will be reduced compared to the case of low Alfvén speed. The Alfvén speed in turn depends on coronal density. Therefore, it is important to determine the coronal density and wave heating rate so they are consistent with each other.

The purpose of this work [17] is to construct more realistic models of Alfvén wave turbulence for various coronal loops in an observed active region. We use data from instruments on the Solar Dynamics Observatory (SDO) satellite for an active region observed on 2010 May 5 shown in Figure 1. Coronal images obtained with the Atmospheric Imager Assembly (AIA) in several EUV passbands show the presence of coronal loops in this region with temperatures in the range 1 - 3 million kelvin (MK). Our goal is to determine whether these loops may be heated by Alfvén wave turbulence, and if so, to predict the observational signatures of such waves and turbulence.

For each loop in Figure 1, we repeat our computations five times to ensure that the temperature $T_0(s)$, density $\rho_0(s)$, and average heating rate $Q_0(s)$ are consistent with the condition of thermal equilibrium. The iterative process is illustrated in Table 1 for field line F6. The time resolution is about 0.1s and the waves are simulated for a period of 3000 seconds, which is much longer than the Alfvén wave travel time along the entire loop ($\sim$200s).

The software used is a combination of interactive tools written in IDL and a batch part for solving the partial differential equations of the model that is written in Fortran. Because of the memory requirements of the IDL code used for analysing the results of runs and because of the software license required, data analysis was conducted on a server at St Andrews, which has been upgraded to 96GB of main memory. The Fortran code initially was similarly memory intensive and jobs were running for up to five days, making it difficult to use in many production HTC infrastructures such as the UK's National Grid Service [18] without negotiating tailored service provision. As part of the ELVIRA project, we have made changes to the code to significantly reduce the memory footprint and through the use of OpenMP [19] we are also making progress in reducing the running time by using multiple CPU cores, see Table 2.

As a result of these changes we are now able to run the Fortran jobs on the St Andrews StACC cloud, which provides a range of instance types, the largest of which provides 4 CPU cores, 4 GB of RAM and 40GB of instance storage. Where necessary, we can configure the system to use the Amazon EC2 cloud. The code scales well, so the larger instance sizes available there allow us to achieve significantly better turn-around times than would be possible on our local resources. The cc2.8xlarge instance type in particular is of interest as it provides the best overall performance while c1.xlarge provides a cheaper price per job (see Table 3).

Table 1 Iterations for model F6

Iteration	m	Q_{min} erg/cm^3/s	$\overline{Q}_{\mathrm{cor}}$ erg/cm$^{3/s}$	T_{max} MK	p_{cor} dyne/cm^2
1	0.524	7.99×10^{-4}	9.66×10^{-4}	2.28	1.65
2	0.567	8.15×10^{-4}	1.01×10^{-3}	2.29	1.68
3	0.532	8.38×10^{-4}	1.02×10^{-3}	2.31	1.71
4	0.514	8.22×10^{-4}	9.86×10^{-4}	2.30	1.69
5	0.553	8.41×10^{-4}	1.03×10^{-3}	2.31	1.72

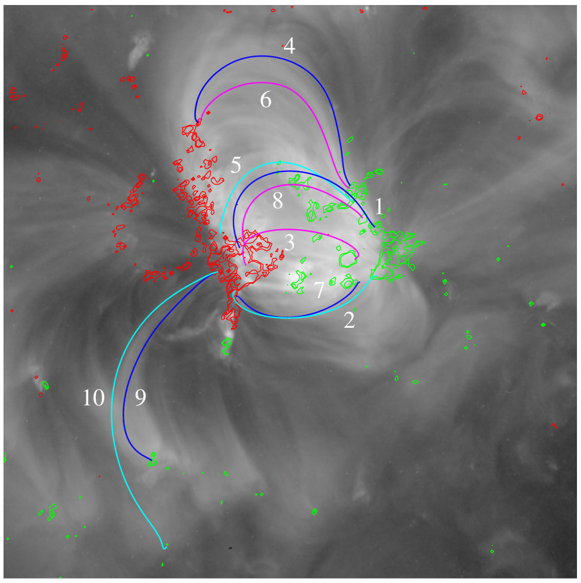


Figure 1 Ten magnetic field lines used for modeling Alfvén wave turbulence in coronal loops. The field lines were traced through the Non-Linear Force Free Field (NLFFF) model. The background image is from the AIA 193 Å channel.

Requirements and functionality

In the following section we describe some of the key requirements for ELVIRA and show how they have been met for the case of the Astrophysics application using the ELVIRA tools, the functionality provided by cloud infrastructures and a number of additional off-the-shelf components. Figure 2 provides an overview of the architecture of the system.

Configuration management

Functionality is required to ensure that instances are equipped to meet the requirements of the research application. We can distinguish between configuration at the cloud level such as choice of an appropriate instance type, assignment of virtual storage volumes and configuration within the instance itself such as configuration of the operating system, deployment of dependencies such as libraries or creation of user accounts. In the case of the Astrophysics application, we need to ensure that a 64bit instance with sufficient memory or swap space is created and that the Portland Group Fortran libraries are

Table 2 Running times of a short test run (with *t*=100) on a physical machine with 32 cores (AMD Opteron™ 6272, Bulldozer architecture, 2.1GHz)

Threads	1	2	4	8	10	12	14	16	20	24	32
Time (s)	1379	862	610	516	509	491	485	496	576	592	1350

installed. In order to support changing requirements we configure instances at runtime using the ELVIRA kickstart process rather than working with static images. Configuration scripts are managed in a Subversion repository and configuration data is passed into instances through the user-data mechanism provided by the EC2 API, allowing the instance configuration to be further contextualised. Bootstrap code in `/etc/rc.local` retrieves this code and executes it. Alternatively, Ubuntu's cloud-init system [20] can be used. The whole process is designed to be simple but to allow for expansion. In most cases where instances have a limited lifetime, the simple mechanism provided will be sufficient but it is possible to invoke other configuration management tools such as Chef [21] or Puppet [22] if required. The details of the process of providing configured instances are embedded in the ELVIRA tools so that users do not need to be concerned with them but can simply provide a list of required configuration options.

Job management

The execution of the actual workload requires job management functionality. In the simplest of cases there can be a one-to-one mapping between jobs and instances but this makes sense only if the overhead of creating an instance is negligible compared to the job running time and if the instance resources are fully utilised by the workload. The latter is particularly important when using a public cloud and quirks in the charging models of cloud service providers can make the question of efficient usage and costing more complex than it would appear at first sight [11]. As a consequence, a common approach taken is to deploy a traditional batch job submission system (e.g., [23-25]) or a workflow management system (e.g., [26,27]) into the cloud or to use cloud resources to extend an existing cluster or grid into the cloud through 'cloud-bursting' (e.g., [28]).

A number of clustering and grid environments have been ported to cloud infrastructures, often provided as a set of instance images that can be deployed relatively easily. An example is StarCluster [29], a cluster computing toolkit that supports the deployment of a batch job submission system based on Sun's Grid Engine into Amazon's EC2 cloud. This approach is suitable for a multitude of workloads and user communities where researchers are already using batch job submission systems and are thus familiar with their interfaces.

Our aim was to explore an alternative approach that would utilise the fact that cloud computing infrastructures already support multi-tenancy. Instead of deploying a traditional middleware into the cloud or adapting existing applications to cloud computing, we looked for an alternative that would fit into the philosophy of ELVIRA of building virtual private research environments, of providing simple solutions by leveraging the affordances of cloud computing technologies and of reusing and adapting existing code where possible. We did not wish to re-write the scientific application code to adapt them to the cloud (e.g., to use a MapReduce pattern, cf. [30]) as this would have made development and maintenance more difficult for the code owners.

We decided to instead explore the possibility of adapting the continuous integration system Jenkins [31,32]) to the needs of the project. It has the advantage of providing rich functionality for the management of jobs, their execution history, their input and output data as well as supporting distributed job execution through slave nodes. Its web-based user interface is highly configurable yet relatively easy to use and the system provides a robust plugin mechanism as well as a RESTful API to support extensions.

Table 3 Running times for a test job on different cloud infrastructures and instance types

Cloud	Type	Cores	Threads	Time(s)	Cost($/h)	Cost per job ($)
StACC	m1.large	1	1	12734	N/A	N/A
	m1.xlarge	2	2	7549	N/A	N/A
	c1.xlarge	4	4	4310	N/A	N/A
EC2	c1.xlarge	8	8	3602	0.58	0.58
	cc2.8xlarge	16	16	1303	2.40	0.87
Local		32	14	1863	N/A	

Experiments were run to determine the optimum number of threads and best results are reported. StACC instances are running on Dell PowerEdge R610 nodes with 2xQuadCore Intel Xeon E5504 2GHz and 16GB RAM. The local machine is a dual Opteron™ 6272 2.1GHz. Costs are EC2 Linux US East on-demand prices as of 2nd June 2013, prices per job are for pure running-time, excluding instance creation and configuration overheads.

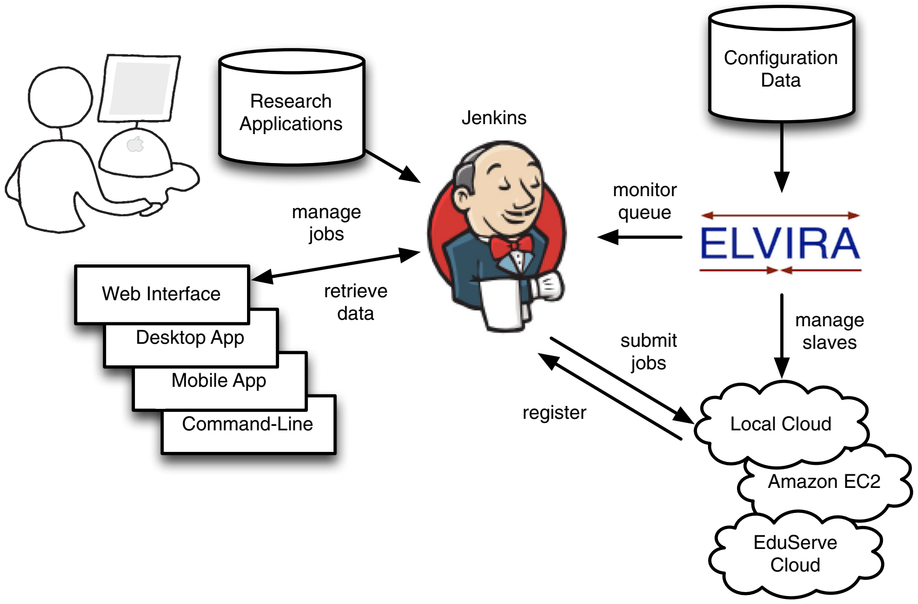


Figure 2 The ELastic Virtual Research Environment for Research Applications.

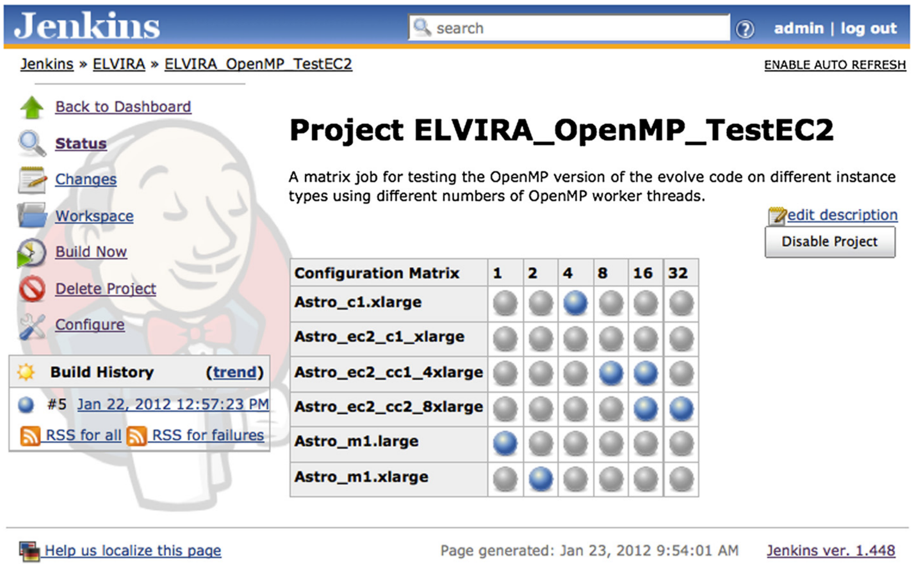
Jenkins comes with a wide range of plugins and additional tools such as mobile apps or integrations with integrated development environments are available.

Jenkins allows jobs to be configured through the web-based interface and it manages the relationship between these jobs and their execution histories ('builds'), capturing provenance information as well as input and output data. Jobs can be connected to each other so that they run in sequence and matrix jobs can be configured to produce parameter sweeps along a number of dimensions (cf. Figure 3). Matrices can be made sparse by supplying an expression that defines which elements of the matrix should be executed. Jobs can be configured to require a set of parameters including the option to upload input files. To control the environment in which a job executes, it can also be mapped to a specific worker or a set of workers that provide the specific runtime environment required by the job. Builds can be triggered manually by the user, through an external event such as a commit to a version control system or an update of a file or regularly through a `cron`-like scheduler.

Source code files for the application are hosted in a private `git` repository and a separate job exists to compile the sources for a range of processor architectures using the Portland Group Fortran [33] compiler. This step needs to run on a server at St Andrews that has a node-locked license for the compiler, so this job is locked to run on that server. The job execution script matches the CPU type used in the instance to the appropriate code. This is important as running times can differ signficantly between unoptimised code and code optimised for the specific CPU architecture used in the cloud instance.

At any point in time, users can monitor what jobs are being executed on the worker nodes as well as what jobs are scheduled for execution. A real-time view of the job log files is available that includes the output the executing program produces on its standard output and error channels. This allows errors to be spotted during runtime and jobs to be terminated if necessary. The log output is archived after a job has executed together with the job's output files, which can be retrieved through the web front-end. The web-based front-end allows multiple users to collaborate in managing jobs and builds as well as giving them shared access to output and provenance data.

The input data is usually of negligible size (fieldline F6 being the largest at ca. 11MB) but the output data can be large (2GB in the case of F6). The output also does not compress well, so it is transfered to Jenkins as-is. While this does not normally take much time in a local environment, the transfer times from a public cloud can be significant and highly variable. The faster the computation the more this matters, especially since faster instances will be more expensive. By default, Jenkins stages data back synchronously at the end of a job and at the moment this is how data is staged back to the server at St Andrews. This problem can be avoided if staging the data back is relegated to a subsequent job that can run in parallel to the next compute job. Alternatively, output data can be uploaded to a storage service such as S3 instead of being sent back to the Jenkins node. From there it



Project ELVIRA_OpenMP_TestEC2

A matrix job for testing the OpenMP version of the evolve code on different instance types using different numbers of OpenMP worker threads.

Configuration Matrix	1	2	4	8	16	32
Astro_c1.xlarge	●	●	🔵	●	●	●
Astro_ec2_c1_xlarge	●	●	●	●	●	●
Astro_ec2_cc1_4xlarge	●	●	●	🔵	🔵	●
Astro_ec2_cc2_8xlarge	●	●	●	●	🔵	🔵
Astro_m1.large	🔵	●	●	●	●	●
Astro_m1.xlarge	●	🔵	●	●	●	●

Build History (trend)
#5 Jan 22, 2012 12:57:23 PM
RSS for all RSS for failures

Page generated: Jan 23, 2012 9:54:01 AM Jenkins ver. 1.448

Figure 3 A project configured in Jenkins to run a set of jobs defined through a matrix of parameters (here, the worker node and the number of threads to use, cf. Table 3). A filter condition has been applied to make this a sparse matrix.

can be retrieved at a later point for analysis. Using S3 has other advantages such as providing an off-site, highly dependable archive for job output data.

Elastic scaling of resources

To allow resources to scale dynamically in response to the workload we have extended the Jenkins system by adding an external monitoring daemon that starts instances in response to changes in the build queue. Instance configurations are defined in the a central configuration file and assigned labels matching the labels defined in the job configurations in Jenkins. This allows a 1:n mapping between instance configurations and job definitions to be created so that different jobs can share instance configurations and instances.

The decision to create a new worker node is controlled by the formula given in Equation 1, which combines q, the number of jobs in the queue, w, the number of workers already running and i, the number of workers that are currently idle or still deploying. A factor f is used to adjust this relationship and a maximum m is defined for the number of instances that are turned on at each step. Figure 4 shows that if no instances are running and $n > m$ jobs are submitted, then m worker nodes will be started initially. The growth of the number of workers is limited but there is no maximum that is imposed although this would be easy to add. We assume cost associativity, i.e., that running n jobs in n instances in parallel costs as much as running n jobs sequentially in a single instance. This is roughly true for our jobs, which run for about 3 hours on a cc2.8xlarge instance and significantly longer on other resources. For workloads with shorter running times, a parameter $p > 1$ can be introduced to represent the number of jobs to run per instance.

$$min(m,q)\,\text{if}\,w = 0\,\text{else}\,min\left(q, m, \left\lfloor \frac{q}{f \times (w+i)} \right\rfloor\right) \quad (1)$$

Once created, instances register with the Jenkins server using the Swarm plugin [34] and start servicing the build queue. The Swarm plugin provides the necessary functionality for a client to register with Jenkins but it is the ELVIRA tools that manage the set of cloud instances, instantiate and terminate them, configure them to act as Swarm nodes and contextualise them to support the specific scientific application.

In contrast to the solution implemented by Strijkers et al. [23], our model does not take into account core counts. This is because we allow different configurations to match different workloads rather than trying to provide a generic clustering environment. Our model is a-historical, taking into account only the current state of the queue. This model is sufficient for us as our workloads are relatively long-running. For more general mixtures of workloads that include jobs with running times that are short compared to the queue poll frequency a history would probably need to be implemented as suggested in [23]. An additional extension that we are looking to implement is a scheme where we can automatically burst out from StACC into Amazon's EC2 service or a similar public cloud infrastructure.

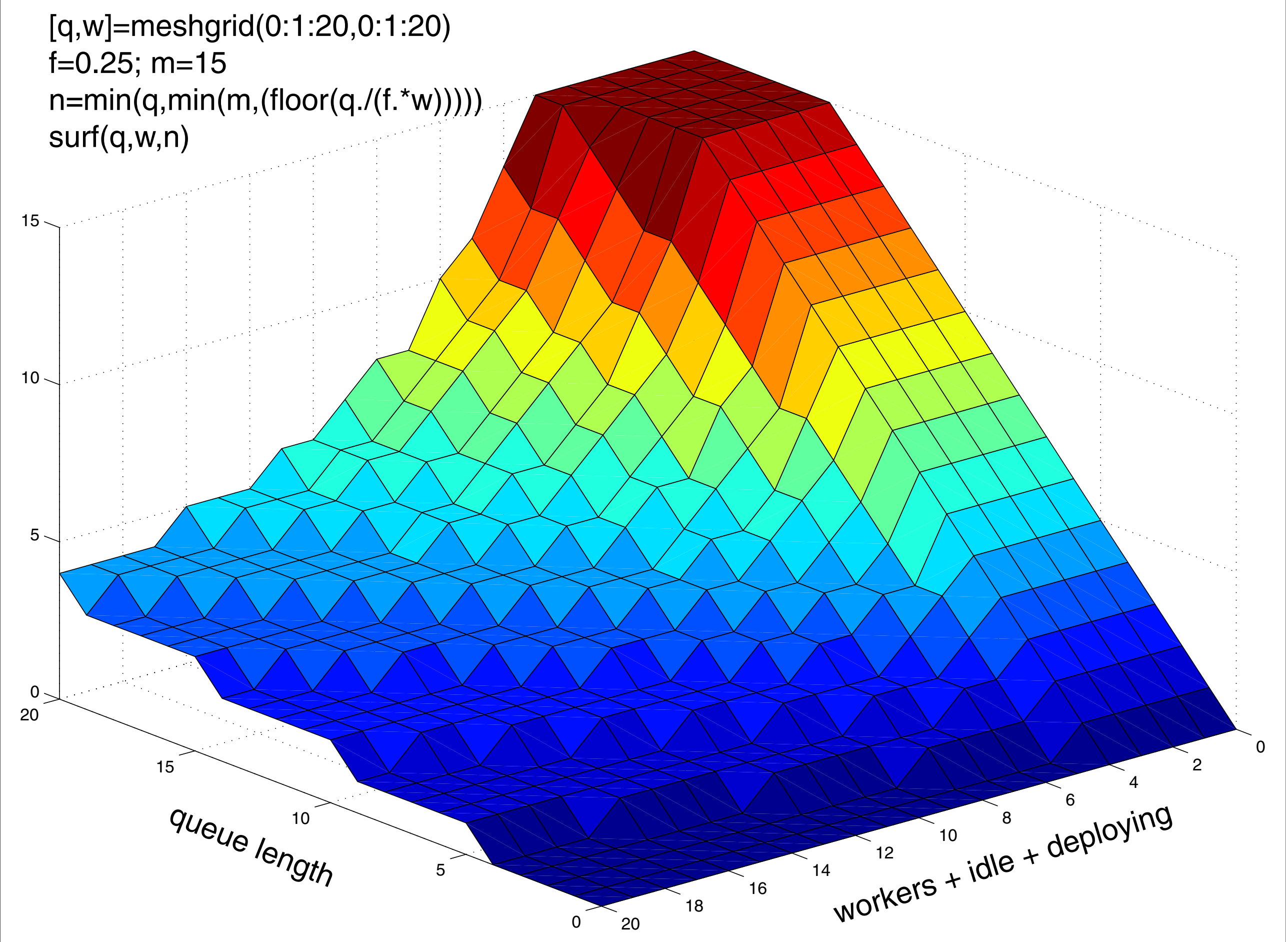


Figure 4 Matlab plot of equation 1. The growth of the number of worker nodes is limited by the number of workers already running, idle or deploying and there is an overall limit on the growth rate but no limit on the total number of workers.

Instance startup and configuration times

As both public and private clouds are shared infrastructures, instance startup times can be unpredictable. A recent study [10] has investigated this for three major public clouds. The findings show that instance creation time varies significantly between different instance launches. The mean launch time depends mainly on the instance image size and instance type. Launching multiple instances on Amazon EC2 does not increase the instance creation time on average [10].

In addition to the time it takes for the cloud infrastructure to provision an instance, configuration scripts that need to run also determine the time it takes for the instance to become available. To some extent, this time can be reduced by installing frequently required packages in the instance image used. For example, in the case of ELVIRA we installed OpenJDK in the instance image as well as the Swarm client .jar file to avoid having to download them every time an instance is created. The Portland Group Fortran libraries, in contrast, are staged in each time an instance is created since we wanted to keep the instance image generic.

To mitigate the time required for instance startup and configuration, it is important to ensure that instances are used to run a number of jobs, especially if the running time of the code is relatively short compared to the time it takes to provision a fully configured instance and register it with Jenkins. This ensures that the overheads are amortised across a larger workload. In contrast to the overheads of instance creation and configuration, the overhead of starting a job are negligible but the staging of output data back to the Jenkins server is not (see above).

Authentication and authorisation

Our virtual private research environment currently uses basic authentication based on a user database but Jenkins supports a number of different authentication options that allow it to be integrated with a number of different authentication options. Authentication between the Jenkins server and the worker nodes is via `ssh` keys, which are injected automatically by the ELVIRA configuration process. This means there is no need to set up a complete public key infrastructure (PKI) as in the case of Grid computing with all the key-management issues that go along with PKI such as the need for global trust, verification of identities and limited key lifetime. The comparative ease with which cloud-based resources can be established and shared within groups of collaborating researchers has also been pointed out by Wu et al. [35], who have developed a Science Gateway for Life Science applications.

Authorisation options are provided in Jenkins through an access control matrix per project, which allows fine-grained policies to be implemented that restrict the operations a user is allowed to invoke with respect to that project, the build history and the archived job data. We have used a single installation for a number of different projects using suitable authorisation settings. However, it is important to note that in environments where users cannot necessarily be trusted and are given permission to configure jobs it is necessary to deploy multiple instances of Jenkins using different Unix user accounts. In particular, it is important to ensure that the system is set up so that no jobs can be run under the Jenkins user account on the main server as this would give these jobs access to the Jenkins configuration information.

Conclusions

We have introduced the ELVIRA project and the concept of a *virtual private research environment*. The ELVIRA tools provide the necessary functionality to manage configurations of cloud resources that are tailored to researchers' needs and manage the elastic provision of these resources in response to workload demands. The researchers we have collaborated with are now able to utilise a virtual private research environment that allows them to manage the execution of their jobs, the resulting output data as well as provenance data through a web-based interface that is tried and tested. The ELVIRA tools provide the necessary extensions for the use of cloud resources but researchers do not need to worry about whether the resources used are provided through local physical resources, a local private cloud infrastructure or a public cloud like Amazon EC2.

Ease of use is important for the uptake of cloud computing for research applications. Ultimately, the success of cloud computing in research applications will depend on the creation of task-centric user interfaces to deal with the significant complexity involved in realising non-trivial applications. The approach we have taken has a number of advantages over alternatives:

- Direct control over the runtime environment used to run jobs allows users to configure it to their specific needs. Configuration here means configuration of the instances and other resources as well as the configuration of the operating system and any software installed. The ELVIRA tools use a central configuration file that can contain definitions for different clouds as well as ensembles of cloud resources to be used for different purposes. The repository of instance configuration scripts allows instances to be configured through declaring desired options rather than scripting.
- The elastic scaling functionality provided by ELVIRA adjusts the number of worker nodes to the current workload and through a set of parameters allows this scaling feature to be adjusted to fit the nature of the workload as well as the characteristics of the cloud environment used (e.g., a maximum number of instances). Different types of worker nodes can be configured for different workloads and jobs can be tied to these worker types.
- The fine-grained access control mechanisms provided by Jenkins allow research environments to be configured that allow researchers to collaborate by sharing access to builds. An annotation feature and the rigorous way that Jenkins uses to manage builds means that it is easy to refer to specific builds and their outputs, which is essential for collaborative research.
- The browser-based user-interface is relatively easy to use but provides access to rich functionality, allowing even complex jobs to be defined through the matrix build configuration option and workflows to be created through chaining jobs together.
- A seemingly trivial feature that is immensely important in practice is the log file viewer that allows the output of builds to be monitored in real-time through the web browser, allowing researchers to monitor build progress and health.
- The wealth of plugins and other extension of the Jenkins system allows the virtual private research environment to be tailored to the specific needs of a project or research group.
- Deployment of a Jenkins instance is very easy and well documented (e.g., in [31]). A basic installation that can be further configured and extended can be created in a cloud instance through the use of the ELVIRA tools, providing a starting point very rapidly. Maintenance is simplified through an integrated update mechanism and plugins to take care of tasks such as backups.
- The ELVIRA tools provide an interface to cloud instances that reduces the complexity by providing configurable defaults for a range of options that need to be provided for each operation with other cloud user interfaces. They make working with multiple clouds easier by providing these options per configured cloud infrastructure.

Our experience from previous projects shows that successul applications of e-Research infrastructures depend not just on more usable technologies but also crucially on the existence of a working division of labour that supports researchers and allows technological offerings to be adopted, appropriated and tailored to the specific needs of a research project [6]. We envisage the ELVIRA tools being used in a division of labour involving the researchers and system administrators. After all, providing a working runtime environment for research codes is non-trivial and there is no way to provide a generic solution to this, although trading configuration scripts and making them available through repositories may help. However, and more importantly, there are routine tasks to be done such as ensuring backups and generally ensuring secure operation that researchers would probably not want to be concerned with. At St Andrews, we run Jenkins on a server managed by our School's system administrators, who deal with day-to-day maintenance and monitoring while the researchers are free to build and manage the virtual private research environments they require and draw on cloud resources as needed. Our professional system administrators have also been crucial in initially setting up aspects of the runtime environment such as configuring server certificates (for https and authenticated SMTP for sending notification emails) and configuring Apache and Tomcat.

Finally, we wish to briefly touch on the issue of cost. While it is certainly possible to host an entire virtual private research environment in a cloud, we believe that given the current state of the art, in many cases it will be advantageous to host the Jenkins server, the repository for instance configurations and the ELVIRA daemon on a physical machine. Perhaps where a stable and well-managed private cloud exists, there will be a case for deploying onto virtual resources but the respective likelihood of downtimes needs to be assessed. The cloud's capacity for rapid recovery in the case of a resource failure can also be utilised in cases where a physical server is normally used. Hosting the parts of the system that are long-running in a public cloud is likely to incur significant costs and in a research environment it is likely that the full economic costs of hosting physical hardware will not be known. The case for deploying into a public cloud is therefore more likely to stem from the costs associated with transfering data across the boundaries of the cloud. If a significant proportion of the jobs are run in a public cloud then it will make sense to locate the whole research environment in that cloud and run all communication through the local IP addresses to avoid networking costs. It is certainly worthwhile modeling costs before making decisions about deployment [36] and it is important to take the cost of network traffic into account in addition to the cost of the cloud instances as this may be significant. In the UK, through funding from the University Modernisation Fund, EduServ is offering a cloud computing service [37] that may well compare favourably to Amazon's EC2 service as it does not charge for traffic into the Janet network.

Future work

The work reported here is the result of a short eight month project. We are continuing to evaluate and use the resulting tools and hope to thereby add to the growing body of experience with the use of cloud computing for research applications. Our aim in this paper was to enable concrete production use of the cloud rather than providing benchmarks that are often the focus in other publications (e.g., [25]). An important factor in the use of cloud computing is the utilisation of multi-core and heterogeneous architectures. We are currently investigating the possibility of porting the MHD code to GPGPU resources such as Amazon's Cluster GPU instances. Utilising such resources in combination with the multiple cores that the CPU provides promises a further reduction in running times but it remains to be seen whether the speedup achieved will justify the higher cost of these instance types.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

IS was the ELVIRA project lead. AB was responsible for technical project management. AV was responsible for the implementation of the ELVIRA platform and liaison with the researchers using it. AvB and MAT were responsible for the substantive research on solar flares and for providing input to the requirements elicitation. All authors contributed to the writing of this article and all authors read and approved the final manuscript.

Acknowledgements

The work presented in this paper was funded by the UK's Engineering and Physical Sciences Research Council and JISC (EP/I034327/1). We would also like to thank Derek Wang, who has provided support for our use of the St Andrews Cloud Collaboratory. Our reviewers have provided useful comments that have helped to improve the paper significantly and we would like to thank them for this.

Author details

[1]School of Computer Science, University of St Andrews, St Andrews, UK.
[2]Harvard-Smithsonian Center for Astrophysics, Cambridge, MA, USA.

Received: 5 March 2013 Accepted: 25 June 2013
Published: 25 November 2013

References

1. Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud Computing and grid computing 360-degree compared. In: Grid computing environments workshop, 2008. GCE '08. IEEE, New York, pp 1–10
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M (2009) Above the clouds: a Berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html
3. Badger L, Grance T, Patt-Corner R, Voas J (2011) DRAFT cloud computing synopsis and recommendations: recommendations of the National Institute of Standards and Technology. Tech. Rep. Special Publication 800-146, National Institute of Standards and Technology
4. Cloud Computing for research and science: a holistic overview, policy, and recommendations. Tech. rep., e-Infrastructure Reflection Group 2012.

http://www.e-irg.eu/images/stories/dissemination/e-irg_cloud_computing_paper_v.final.pdf
5. Bradshaw D, Folco G, Cattaneo G, Kolding M (2012) Quantitative Estimates of the Demand for Cloud Computing in Europe and the Likely Barriers to Uptake. Tech. Rep. SMART 2011/0045, D4 – Final Report, IDC. http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf
6. Voss A, Asgari-Targhi M, Procter R, Fergusson D (2010) Adoption of e-Infrastructure services: configurations of practice. Philos Trans A R Soc (368): 4161–4176
7. Top 500 Supercomputer Sites. http://www.top500.org
8. Iosup A, Ostermann S, Yigibasi MN, Prodan R, Fahringer T, Epema DH (2011) Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans Parallel Distributed Syst 22(6): 931–945
9. Amazon EC2 GPU Instances. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using_cluster_computing.html
10. Mao M, Humphrey M (2012) A performance study on the VM startup time in the cloud. 2012 IEEE Fifth Int Conf Cloud Comput 0: 423–430
11. Gillam L, Li B, O'Loughlin J, Tomar AP (2013) Fair Benchmarking for cloud computing systems. J Cloud Comput: Adv, Syst Appl 2(6). Springer, doi:10.1186/2192-113X-2-6
12. Amazon EC2 API Tools. http://aws.amazon.com/developertools/351
13. Eucalyptus 3.2.2 Command Line Interface Reference Guide. http://www.eucalyptus.com/docs/3.2/cli/
14. AWS Management Console. http://aws.amazon.com/console/
15. Firefox Extension for Amazon EC2. http://sourceforge.net/projects/elasticfox
16. van Ballegooijen AA, Asgari-Targhi M, Cranmer SR, DeLuca EE (2011) Heating of the solar chromosphere and corona by Alfvén wave turbulence. Astrophysical J 736: 3. doi:10.1088/0004-637X/736/1/3
17. Asgari-Targhi M, van Ballegooijen AA (2012) Model for Alfvén wave turbulence in solar coronal loops: heating rate profiles and temperature fluctuations. Astrophysical J 746: 81. doi:10.1088/0004-637X/746/1/81
18. UK National Grid Service (NGS). http://www.ngs.ac.uk
19. The OpenMP® API specification for parallel programming. http://openmp.org
20. Ubuntu CloudInit. https://help.ubuntu.com/community/CloudInit
21. Opscode Chef. http://www.opscode.com/chef/
22. Puppet. http://puppetlabs.com
23. Strijkers R, Toorop W, Av Hoof, Grosso P, Belloum A, Vasuining D, Laat Cd, Meijer R (2010) AMOS: using the cloud for on-demand execution of e-science applications. In: Proceedings of the 6th IEEE International Conference on e-Science. IEEE Computer Society, Washington, pp 331–338. http://dx.doi.org/10.1109/eScience.2010.15
24. Glenis V, McGough AS, Kutija V, Kilsby C, Woodman S (2013) Flood modelling for cities using cloud computing. J Cloud Comput: Adv, Syst Appl 2(7). Springer, doi:10.1186/2192-113X-2-7
25. Cohen J, Filippis I, Woodbridge M, Bauer D, Hong NC, Jackson M, Butcher S, Colling D, Darlington J, Fuchs B, Harvey M (2013) RAPPORT: running scientific high-performance computing applications on the cloud. Philos Trans A R Soc 371(1983): 1471–2962
26. Juve G, Deelman E, Vahi K, Mehta G, Berriman B, Berman B, Maechling P (2009) Scientific workflow applications on Amazon EC2. In: 5th IEEE International Conference on e-Science: Workshops IEEE, pp 59–66
27. Nagavaram A, Agrawal G, Freitas M, Telu K, Mehta G, Mayani R, Deelman E (2011) A cloud-based dynamic workflow for mass spectrometry data analysis. In: Proceedings of the 7th IEEE international conference on e-science. IEEE, New York, pp 47–54
28. Humphrey M, Hill Z, van Ingen C, Jackson K, Ryu Y (2011) Assessing the value of cloudbursting: a case study of satellite image processing in windows azure. In: Proceedings of the 7th, IEEE international conference on e-Science. IEEE, pp 126–133
29. StarCluster. http://web.mit.edu/star/cluster
30. Dalman T, Dörnemann T, Juhnke E, Weitzel M, Smith M, Wiechert W, Nöh K, Freisleben B Metabolic Flux Analysis in the Cloud In: Proceedings of the 6th IEEE conference on e-science. IEEE, pp 57–64
31. Smart J (2011) Jenkins: the definitive guide. 1st edition. O'Reilly
32. Jenkins Continuous Integration Server. http://www.jenkins-ci.org
33. The Portland Group. http://www.pgroup.com/
34. Jenkins Swarm Plugin. https://wiki.jenkins-ci.org/display/JENKINS/Swarm+Plugin
35. Wu W, Zhang H, Li Z, Mao Y (2011) Creating a cloud-based life science gateway In: Proceedings of the 7th IEEE international conference on e-science. IEEE, New York, pp 55–61
36. Khajeh-Hosseini A, Greenwood D, Smith J, Sommerville I (2011) The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. Software: Practice and Experience: 447–465
37. Eduserv Cloud IaaS. http://www.eduserv.org.uk/services/cloud

doi:10.1186/2192-113X-2-20
Cite this article as: Voss *et al.*: An elastic virtual infrastructure for research applications (ELVIRA). *Journal of Cloud Computing: Advances, Systems and Applications* 2013 **2**:20.