Scheduling multiple virtual environments in cloud federations for distributed calculations

A. J. Rubio-Montero[a], E. Huedo[b], R. Mayo-García[a]

[a]*Centro de Investigaciones Energéticas Medioambientales y Tecnológicas (CIEMAT), 28040 Madrid, Spain.*
[b]*Departamento de Arquitectura Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain.*

Abstract

In the pool of cloud providers that are currently available there is a lack of standardised APIs and brokering tools to effectively distribute high throughput calculations among them. Moreover, the current middleware tools are not able to straightforwardly provision the ephemeral and specific environments that certain codes and simulations require. These facts prevent the massive portability of legacy applications to cloud environments. Such an issue can be overcome by effectively scheduling the distributed calculations using the basic capacities offered by cloud federations. In this work, a framework achieving such a goal is presented: a pilot system (GWpilot) that has been improved with cloud computing capabilities (GWcloud). This framework profits from the expertise acquired in grid federations and provides interesting features that make it more efficient, flexible and usable than other approaches. Thus, decentralisation, middleware independence, dynamic brokering, on-demand provisioning of specific virtual images, compatibility with legacy applications, and the efficient accomplishment of short tasks, among other features, are achieved. Not only this, the new framework is multi-user and multi-application, dynamically instantiating virtual machines depending on the available and demanded resources, i.e. allowing users to consolidate their own resource provisioning. Results presented in this work demonstrate these features by efficiently executing several legacy applications with very different requirements on the FedCloud infrastructure at the same time.

Keywords: Cloud scheduling, pilot jobs, cloud middleware, resource management, application compatibility.

1. Introduction

One of the main achievement of grid research was the establishment of a service oriented architecture (SOA) [1] for high-throughput computing (HTC) that was widely accepted by the whole community. This fact allowed the federation of large volume of resources across the world, but also implied a long process of testing and standardisation. One of the outcomes was the creation of a set of APIs for highly distributed computing [2], as well as the establishment of protocols to interface with the computational services offered.

However, grid has not properly addressed several problems. One of them is the rigidity of configurations that are present in the federations. Besides, one of the major issues in grid computing is still the efficiency of the submitted jobs. Such efficiency can be considered from different perspectives [3], but always bearing in mind that the final users want their calculations ended in the shortest possible time. For this purpose, it is mandatory to count on scheduling mechanisms that properly build and distribute these jobs among available providers. Several strategies have been followed, such as the dynamic allocation of jobs depending on the infrastructure status and capacity at any time [4], but pilot job systems provide low overheads and great flexibility [5].

Cloud paradigm covers much more computational needs than the ones required for an HTC platform, like resource elasticity, service consolidation, or cost reduction. In the case of distributed computations, cloud promises to be more simple, flexible, usable and available than grid. Nevertheless, the latter affirmation is far away from being a reality in many cases by now. In first place, the diverse sponsor institutions, funded projects, infrastructure providers, and manufactures have different views, and propose different models on how the cloud federation should be. A good introduction of this matter can be found in [6]. Due to this diversity, the result is an increased complexity of the current cloud platforms from the user point of view. On the other hand, although the flexibility is increased, users not always can run their applications exactly on the virtual environment that they require and resources are effectively limited for every user.

To reduce the final makespan of any calculation, the developer should take account of localisation and performance of the available resources, but it also requires some abstraction mechanisms. For this purpose, he should rely on *brokers* in the first instance. However, in cloud computing, these basic brokering capabilities are still far from being provided and they usually lack of APIs similar to the ones available for HTC or grid. As commented above, there is a strong heterogeneity of cloud providers that can be explored by means of a cloud brokering approach [7]. As a result, optimising the placement of virtual machines across multiple clouds, and also abstracting the deployment and management of components of the virtual infrastructure created are complex. Such plethora of solutions were surveyed and classified by a proposed taxonomy by Grozev and Buyya [8], but the creation of federations is the first step to reduce this complexity. The common services deployed in federations allow users to obtain a unified and abstracted view of the

cloud architecture as a whole, which subsequently gives them the opportunity to target specific providers for their needs.

However, calculating the best match between a set of computational requirements and resources in an effectively characterised infrastructure is, by definition, an NP-complete problem [9], and only sub-optimal schedules can be found. Furthermore, cloud federations will be more dynamic than grids because more parameters are taken into account for scheduling [10, 11]. Nevertheless, that information is far from being provided by the services currently deployed. In this sense, the experience with grid is clear: in a large collaborative federation, the deployment of those complex algorithms was quite limited in production due to this persistent lack of resource characterisation.

Therefore, cloud communities can benefit from the grid expertise following two approaches whose suitability was demonstrated in grid environments: (i) specific scheduling algorithms devoted to improve certain calculations, such as self-schedulers; and (ii) late-binding techniques, where pilot jobs consolidate the characterisation through the resource provisioning. Combining these approaches will be very useful. Nevertheless, most of pilot systems and similar approaches lack the needed adaptability and compatibility to achieve this goal.

To tackle these issues, a new general-purpose framework devoted to efficiently schedule distributed calculations in cloud federations is presented in this work. The approach is based on pilot jobs and inherits the long expertise acquired with grid computing through last years. Unlike other frameworks, the solution allows users the arbitrary characterisation of resources without modifying the pilot code or their legacy applications. This feature enables the customised provisioning guiding and the straightforwardly incorporation of their own scheduling algorithms devoted to their specific calculations. Thus, apart from a complete view of the architecture and the advantages provided by the new framework, the objective is to focus on the features which are not accomplished yet by other systems.

To demonstrate these features, real calculations have been performed on the EGI FedCloud using three applications with different requirements for their virtual environments.

2. IaaS cloud federations

2.1. Basic provisioning in clouds

The need of specific interfaces that abstract the common operations with VMs (creation, booting, stopping, halting, destruction, etc.) has driven the appearance of several proposals and implementations since 2006 (such as Globus VWS, Nimbus, Eucalyptus [12]). However, Amazon Web Services[1] (AWS) was the first large IaaS provider and many deployments were based on its interfaces, because it was considered as *defacto* standard for the industry. Lately, the Open Grid Forum (OGF) standardisation group proposed the Open Cloud Computing Interface (OCCI [13]), which is supported by a wide set of current virtual infrastructure managers (VIMs) such as OpenNebula[2], OpenStack[3], or Synnefo[4].

Other issue is the need of instantiating VMs with a certain configuration. The initial approach is to upload the customised disk images of the VM to the provider. Nevertheless transfers are too expensive due to the size of images. Therefore, it is more efficient the support of generic VM templates at every provider. Thus, the *contextualisation* is the procedure to preconfigure a VM at boot time. In this sense, several technologies have been developed, many of those tightly dependent on a concrete VIM. Finally, Cloud-Init[5] is prevailing around the current IaaS providers.

The approach presented in this work follows OCCI and Cloud-Init specifications, because of their widespread adoption. Nevertheless, the interfaces and contextualisation tools are not enough to completely manage a virtual environment. Other services and systems are needed, especially when multiple cloud providers are available.

2.2. Multiple providers versus federations

There is a conceptual differentiation between a simple group of providers and the ones making a federation that has important implications on the feasible scheduling to be performed. Following several definitions in the related work [8, 14], when a client (or service) uses multiple, but independent, and not related grid or cloud providers, it is working on a multi-grid or multi-cloud environment. Therefore, it is the client (or service) who must completely manage the compatibility among interfaces, monitor every provider, and handle its authorised accounts because it is working on different configuration and security domains. This entails a huge developing work that limits the scheduling capacity of the client system. In consequence, the distribution of the calculation among providers will scale on the order of few orders of magnitude, although these providers can supply a great amount of resources (e. g. AWS).

These difficulties are widely studied in the related work, and they are usually presented as an interoperation [2] issue among grid or cloud infrastructures. For example, the interoperation of grid islands [15] (or multi-grid [14] environments) has been managed through tools such as workflow managers, brokers and scientific gateways. Additionally, the multi-cloud [8] approach is currently too common due to the multiplicity of cloud conceptions and commercial providers. Interoperation among clouds can be faced with similar tools as grid, but entails the same drawbacks that result in poor scalability.

To enable scheduling systems for managing providers on the order of thousands, these systems should work on federated infrastructures. Federations voluntarily associate providers, which even share their resources among each other, but completely following the SOA model [16]. In this sense, the weaknesses of multi-grid and multi-cloud approaches are not related

[1] http://aws.amazon.com

[2] http://opennebula.org

[3] http://www.openstack.org

[4] http://www.synnefo.org

[5] http://cloudinit.readthedocs.org

to deal with the interoperability issues that SOA tackles. Thus, it is not enough to offer services as the ones described in previous Subsection 2.1. These services must accomplish common visibility, governance, security, orchestration and monitoring properties among others. Therefore, it does not simply imply the agreement to use certain protocols; it also includes the establishment of common services as:

- Information systems (IS): they are indexation locations where the rest of services are dynamically described. They constitute the starting point from which any client (i.e. the scheduler) can discover the resources belonging to the federation. The development efforts to characterise interesting aspects of the infrastructure should be focused on this service.

- Authentication and authorisation infrastructure (AAI): it stands for the group of services, authorities and procedures that enable the security governance in federated environments, which are usually based on encryption and temporal tokens. They work together as an overlapped and independent infrastructure that allows the management of users and projects within, or even crossing federations, i.e. it allows the establishment of virtual organisations (VOs). Through AAI, the clients or groups of clients are granted to use certain amount of resources by setting quotas for them in providers according to signed contracts.

- Accounting, monitoring and incident systems: they compile the performance, throughput and failures of every provider and every user through the time. Therefore, they offer a detailed measurement of the current QoS of the whole infrastructure that can be useful for scheduling, not only to check the compliance with the SLA contracts subscripted by providers. In this sense, the information should be also summarised in IS for further benefit. Additionally they include the procedures of notification and solving from the issues detected.

2.3. Federations established

During the last five years, several projects have dealt with the complexity of making up an IaaS cloud federation. The advances obtained are more related to the expertise acquired than the corresponding small-sized test-beds[17, 18] set up to demonstrate some achievements. For example the RESERVOIR project[19] advised to follow an architecture based on loose-coupled providers and the needed of counting on information and accounting services similar to the grid ones, but improved for the SLA accomplishment. In this sense, CONTRAIL[20] focuses efforts on the requirements to construct an AAI compatible with a set of proposed REST cloud interfaces, besides the verification of SLA adherences. On the other hand, the use of common middleware releases was extensively used in grid to tackle the inter-operation, authentication and deployment issues within a federation. StratusLab[6] offered a distributable Toolkit [21] to easily deploy IaaS providers with the support of the AAI established for grid. Although Helix-Nebula[7] project made use of this distribution, it just provided two centralised black-box interfaces (*the Blue-Box*)) [22] based on two *brokers* in order to profit from a small set of commercial providers [18].

Some public clouds, like AWS, support other kind of AAI for delegated access, allowing several external identities to use the same account. External identities can come from different identity providers supporting OpenID Connect (OIDC). This could be used to access different clouds if the identity is previously mapped in each provider. However, although the mechanism makes easier the management of the AAI, allowing the creation of private VOs on-demand, the multi-cloud heterogeneity is maintained.

One of the largest federation of resource providers is the European Grid Infrastructure[8] (EGI) and its association with Open Science Grid[9] (OSG), as well as with other infrastructures such as NorduGrid. Together, they build up a multi-propose platform counting on more than 530,000 processors and 500 PBs of storage, distributed among hundred providers, and opened to any scientific area. In this sense, the EGI FedCloud initiative is taking advantage of grid experience to deploy grid-style services to enable its cloud federation[10] [23]. The goal was achieved by deploying common services with the properties described in the previous subsection:

- The top Berkeley Database Information Indexes (BDIIs) as IS. They are LDAP servers that compile the characterisation of providers structured following the GLUE schema. Several can co-exist in an infrastructure. They can also be hierarchically organised, filtering the information of BDIIs bounding sub-infrastructures.

- The establishment of an AAI based on signed X.509 certificates, distributed certification authorities (CAs), temporal and delegated proxies, and the Virtual Organisation Membership Service (VOMS) as authorisation service.

- The deployment of monitoring systems to regionally and globally test the infrastructure, which trigger several notification and tracking actions. Currently, SAM tests are managed with Nagios, and incidents are tackled with the GGUS ticket system. Accounting is performed through APEL.

Therefore, as cloud sites fully support the EGI AAI based on X.509 and VOMS, sharing the same VOs already established for grid. Providers must be compatible, at least, with OCCI, but also expose their characteristics by LDAP to be compiled by top BDIIs. This last aspect is very important to establish a real federation. OCCI shows information about VM templates, allowed resources, etc. However, to perform the discovering

[6] http:// http://www.stratuslab.eu

[7] http:// www.helix-nebula.eu

[8] http://www.egi.eu

[9] http://www.opensciencegrid.org

[10] https://www.egi.eu/infrastructure/cloud/

of new cloud sites or to facilitate their monitoring, any system should have access to an IS. In the same way, the accounting, monitoring and incident tracking was managed with the same tools than EGI, but adapted to the cloud environment.

Other issue is the essence of the association in a federation, which determines the policy for resource provisioning. Public clouds are usually homogeneous, their configuration seldom change (they are composed of a fixed number of regions with identical instance types), and dynamic discovery would not be needed. Thus, clients of commercial providers are usually more concerned about their budget and their resource scheduling will then be focused on decreasing costs. Monitoring the performance obtained is mainly used to verify the compliance of the supply contract signed. On the other hand, SLAs are much less restrictive in infrastructures based on the voluntary sharing of resources, such as FedCloud. Providers can modify arbitrarily their amount and the performance of the resources offered. Consequently, the scheduling among providers should be oriented to guarantee a minimum QoS in order to make feasible some calculation.

2.4. Brokering approaches

To be considered a cloud *broker*, a system must perform at least: the dynamic discovering and monitoring of cloud providers; the selection of every provider based on a set of requirements; and, the automatic management of provisioned VMs. In this sense, based on the OCCI standard, several cloud brokering solutions have been developed [24, 25, 26, 27]. However, many of them work on a multi-cloud environment that does not allow the automatic discovery of providers. Additionally, it must be clarified when the IaaS providers are statically ranked or selected in a simpler way such as round-robin, without taking into account neither of requirements set for diverse applications nor of the changes in IaaS statuses, the selection performed cannot be considered really brokering or scheduling either. In this work, these tools are denoted as *suppliers*. They repeatedly appear through the related work [28, 29, 30, 31, 32, 33, 34, 35, 36, 37].

However, due to the limited number of IaaS providers, *brokers* were successfully fulfilled by *suppliers* through years. Additionally, VIMs can act as a kind of *broker* when making use of external providers to grow their own resources. To be stackable, VIMs must expose an IaaS interface such as OCCI or AWS. Nevertheless, these protocols are not directly usable by legacy HTC applications [2]. For this purpose, a hybrid API such as DRMAA-OCCI [25] was proposed to enable a direct execution of jobs on the VMs managed by a cloud *broker*.

Perhaps the earliest approach that schedule jobs among remote resources as they were cloud providers can be found in [38]. In this paper, a mechanism that makes use of the existent grid middleware to schedule VMs that contain some software in remote sites is proposed. GridWay takes account of the state and characteristics of sites to submit a wrapper encapsulated into a regular grid job that is able to boot a virtual machine. OS images were directly uploaded to sites or to storage elements through protocols such as GridFTP. Obviously, this approach does not exploit all the advantages that virtualisation offers, and only was justified by the absence of cloud interfaces and middleware.

With the appearing of first cloud federations, more specialised systems were deployed, but they cannot be considered as VIMs. For example, following the same idea of direct execution, the PMES *broker* [24] offers the OGSA-BES interface to allow instantiating a single VM per each computational task formatted with the JSDL specification. In any case, this type of approaches is limited to the execution of long jobs that compensate for the VM instantiating time. In explanation, the effective execution time has to be longer enough to make up for all generated overheads: scheduling the suitable provider, protocol operations, booting the VM, contextualising, install software, and transferring data. For example, if the minimum overhead achieved is five minutes, to maintain these overhead under the ten percent threshold, the duration of the job must be longer than forty five minutes.

However, the majority of new approaches are oriented to consolidate complex services on demand, not to offer compatibility with legacy applications. For example, SlipStream[11] was tested in the Helix-Nebula project, although it can work in a multi-cloud environment. It performs VM image management and contextualisation, with virtual cluster automated deployments. It makes use of CPU/Disk/Memory metrics in its scheduling, but it does not inspect the requirements of applications. In contrast, QBROKAGE [26] is an interesting solution recently proposed that focuses the brokering mechanism on satisfying the QoS requirements of applications. Other solutions overcome the absence of federated services building their own ones. For example, CompatibleOne[12] [27] is a complete platform, with its own user management, accounting and monitoring systems. Nevertheless, neither exposes an OCCI service, nor standard interfaces to execute jobs.

3. Network overlays

The issues set out through the last subsection suggest that cloud provisioning is not enough to enable the efficient execution of distributed applications, so supplementary tools are needed. The grid experience has demonstrated that pilot job technique is the most suitable approach to perform calculations in a highly distributed environment [39]. In addition, any pilot system can be used in cloud environments if their pilot factory is modified to instantiate VMs, directly accessing IaaS interfaces, using *suppliers* or *brokers*.

The utilisation of pilot systems implies the inheritance of their benefits but also their drawbacks. In general, for the most of those systems that demonstrated their suitability in grid [28, 35, 36, 37, 5], the use of cloud resources will provide the following advantages:

- Compatibility with applications previously adapted to these systems, at least, preserving the achieved performance or speedup.

[11] http://sixsq.com/products/slipstream.html
[12] http://www.compatibleone.org

Table 1: Comparative of pilot systems used in cloud

Design adaptation	System name	Communication	Customisable characterisation	Easy deployment	Legacy applications	Automatic scheduling	Guided provisioning	Compatible with grid services
LRMS based	Elastic Clusters	Push	☐	☐	☒	☒	☐	☐
	glideinWMS		☐	☐	☒	☒	☒	☒
Application oriented	Big-Job		☐	☒	☐	☐	☒	☒
Grid scheduler	ServiceSs		☐	☐	☐	☒	☒	☐
	GWpilot	Pull	☒	☒	☒	☒	☒	☒
LHC oriented	DIRAC		☐	☐	☐	☒	☒	☒
Volunteer Computing	3G-Bridge		☐	☐	☐	☒	☐	☒

- Maintaining the characterisation required by the previously adapted applications.
- Concurrent use of clouds and grids (and other infrastructures such as local clusters or volunteer computing if supported by the system).
- Keeping the expertise acquired from grid, reducing the user training-gap and the operational costs.
- Preserving the robustness of the system and the use of complementary tools such as web browsers, monitoring tools, etc.

The weaknesses are the consequence of their design, mainly if the system adopts a pushing or pulling behaviour. The former implies technical issues, while the latter strongly condition a feasible scheduling [5]. Thus, this aspect will be used to perform the first differentiation among systems.

3.1. Pushing based systems

Elastic virtual sites and clusters

Independently of how the IaaS cloud is accessed (with *suppliers*, *brokers* or directly), when the issue of the VM creation is solved, the following main problem is to offer the new provisioned cloud resources in a way compatible with the usual execution of HTC applications. Therefore, many earlier approaches were based on the set-up of virtual clusters with some specific services, or even a complete virtual grid site. The simplest mechanism is to create virtual nodes at the cloud provider to dynamically grow a private cluster [32, 40, 34, 33]. Other solution is to increase the nodes of a local grid site [30] or even completely place the site at the cloud provider [31]. In such a case, the computational tasks can be directly scheduled by the local resource management system (LRMS) of the cluster, or by any grid scheduler, respectively.

The placement of cluster nodes in remote locations has multiple drawbacks anyway. First, the high latency of WANs will prevent the normal behaviour of the cluster, or the achievement of some calculations. Besides the unfeasibility of these networks to achieve tightly-coupled parallel applications, the LRMS usually requires shared file systems that dramatically decrease their performance in these environments. Therefore, data and software locality is a great obstacle. Second, the necessity of bi-directional communication among nodes and master implies either the assignation of public IPs for every node, the use of Message Accumulators (MAs), or even setting up a VPN among nodes. IPv4 pubic addressing is limited in cloud providers and can have an associated cost. In addition, security issues must be considered. The use of MAs or VPNs increases the latency of communications.

On the other hand, to set several complete virtualised grid sites implies the configuration of much middleware and their subsequent management.

It is important to mention that several tools for automation and monitoring of deployments are currently available for clouds [41], as well as several cloud *brokers* perform similar procedures, and even VIMs such as OpenStack or OpenNebula can be stacked to grow their resources [42]. However, although the work needed to set up a virtual cluster or grid site has been reduced, the performance loss and the complexity acquired, respectively, cannot be justified when other solutions as pilot jobs are available.

The main drawback of the pushing mechanism is the need of direct access pilot services running in the VMs from the pilot server. In general, this implies the assignment of public IPs to every node, or the use of MAs. In this sense, the nodes that elastically grow private clusters [32, 40, 34, 33] can be considered as this type of pilot jobs. However, by following more strictly the definition of pilot system proposed in [5], a similar approach with Condor can be found in glideinWMS [28].

Other possibility is to directly manage VMs as pilots through SSH commands. For example, ServiceSs [24] is an extension of COMPSs that makes uses of the aforementioned PMES *broker* to provision this type of VMs. Other example is the Ser-

vice Manager [29], a cloud *supplier* that includes the monitored VMs in the GridWay Host Pool. Subsequently, GridWay can schedule tasks among these VMs and execute them when it makes use of its SSH driver. A mixed mechanism is the one used by Big-Job [35], where the Pilot-Agent is started through SSH. Although, it runs independently, the communications with Manager is performed through a Coordination Service that can be considered as a MA.

3.2. Pulling based systems

Nowadays, DIRAC (its VMDIRAC extension) is the best exponent of profiting cloud infrastructures in production with pilot jobs. To perform cloud provisioning, a new VM Director Agent is used instead of the Pilot Director. Thus, the architecture of DIRAC is maintained. Currently, the VM Director Agent is a cloud *supplier* that simply dispatches VMs to a pool of known providers [36]. However, it is expected that, following the DIRAC design, the double-matching mechanism among requirements of tasks and the discovery of cloud resources will be applied as they were used with grid sites. Thus, its functionality as grid *broker* will be preserved and users will be able to continue submitting jobs through gLite commands. Moreover, its functionalities are extended with the advantages of the different types of contextualisation that allow the specific configuration for physicists (HEPiX) or external monitoring (e.g. Ganglia).

The recent 3G-Bridge [37] implementations are able to maintain a BOINC server with Clients (the pilots) running in provisioned VMs. The work is focused on the customisation and the contextualisation needed to accomplish the common user-tasks in the Clients. The scheduling among cloud sites is relegated. As an example for grid users, a modified CREAM server can rely on 3G-Bridge to dispatch their tasks.

Therefore, the JDL specification of every task should be transferred to the scheduling mechanism of DIRAC or BOINC and subsequently, the tasks should be executed as on a grid node. Nevertheless, the constraints imposed by the Pilot Agent and the Client software make users not really being able to choice among virtual environments. Thus, every community should build their common VM images before any deployment.

3.3. Discussion

The described approaches have some problems already explained that can prevent their deployment for certain calculations. However, the orientation of these systems is the reason of going away from a general-purpose and user-oriented scheduler.

The approaches such as ServiseSs and DIRAC are proposing themselves as a kind of Platform-as-a-Service (PaaS) cloud [43]. The first one is focused on offering an IDE (an Eclipse plug-in) and a new API based on code directives for the automatic parallelization and orchestration of applications and services in cloud. The approach is powerful and extensible, but not standardised, and legacy applications will be incompatible with it. Moreover, tasks cannot directly access to the required data because these are usually stored in grid elements, which protocols are not considered. Additionally, the disadvantages of using an external scheduler with pilot jobs were explained in [5], being more feasible to manage provisioning and task scheduling in a box. On the other hand, VMDIRAC deals with supporting the scientific communities already consolidated in grid. In this sense, the scheduling policy of VMDIRAC will be clear: improve the throughput of large jobs. Therefore, it will not be a problem if the system sequentially provision one VM per each pending task, as it is performed in grid; or either if the VMs must be previously customised for every community. The objective is the extension to cloud of the scientific production maintaining the compatibility with previous grid services. It is expected that glideinWMS will pursue the same objectives and lack the same scheduling features. Other issues about their difficulty of installation or performance can be found in [5].

The technologies added with 3G-Bridge are oriented to improve the compatibility and interoperation among infrastructures, not to offer the necessary tools for customising scheduling capabilities. Unlike, Big-Job provides a complete framework to abstract the management of pilots and their provisioning as VMs. Nevertheless, developers must explicitly indicate the provider where to execute the pilot, and the pilot to execute tasks. The objective is to maintain the freedom to implement any scheduling policy at the application level. In this sense, algorithms such as MapReduce have been developed on cloud [44], but any legacy application should be rewritten to use them.

4. Proposed solution

The standardisation process that makes possible the establishment of cloud federations also opens the door to a grid meta-scheduler as GridWay [45] to directly make use of cloud resources. In consequence, the advanced scheduling features of GridWay, its usability and compatibility will really come into cloud. For this purpose it is necessary to implement two new Information and Execution drivers able to manage cloud providers: the GWcloud IM and ED, respectively. The solution [46] differentiates from other approaches based on GridWay cited through the related work [38, 30, 31, 29] in that it actually brokers customised workspaces for distributed applications on current clouds, i.e. it does not use either *suppliers* or outdated middleware. The GridWay Scheduler takes the decisions of where VMs are started. Benefits include:

a) Automatic discovering and monitoring of providers that belong to several cloud federations.

b) Scheduling capabilities based on constraints, ranking, fairsharing, deadlines, etc., to instantiate VMs at providers with certain characteristics, like:

   - specific VM image (e.g. by the *appdb.egi.eu* identifier);
   - available hardware, with advanced reservations;
   - associated costs and budgets, QoS, etc.

c) Grid security standards and protocols are preserved to enable compatibility with external grid services.

d) Direct distribution and execution of jobs in VMs.

e) Minimal contextualisation, fully customisable by the user.

f) Compatibility with legacy applications.

However, time spent in every VM instantiation can represent an excessive added overhead comparable to the waiting times in LRMS queues at free grid sites. Therefore, the solution is only suitable for long jobs such as bag of tasks (BoTs), ephemeral services, or pilot jobs for resource provisioning. This last approach is the one followed in this work due to its benefits.

To do so, this work profits from GWpilot, a system that includes the advantages from pilot systems listed in the beginning of Section 3, but also differentiates from other approaches used in cloud environments in:

a) General-purpose pulling pilot system, stackable with other scheduling tools.

b) Friendly interface (the GridWay CLI) and compatibility with legacy applications (DRMAA and OGSA-BES).

c) Independent and easy configuration, lower overheads that allow decentralised and local installations, even on the PC of the user.

d) Parallel accounting of associated costs.

e) Personalised user-level scheduling of tasks and VMs that allows:

- post-configuration of VMs on demand;
- customised monitoring of new configurations;
- personalised provisioning;
- efficient execution of very short jobs.

4.1. The GWcloud Information Driver (ID)

This new driver performs the discovery and monitoring of cloud providers in federations. It looks up for cloud providers in the IS (in top BDIIs for FedCloud) of one or multiple infrastructures). The user can configure the search to constraint the matches to certain characteristics published by providers. Subsequently, the driver filters the information to dynamically notify GridWay about the characteristic of every provider in which the user is authorised. Every provider found is included as an independent resource in the Host Pool. Thus, the information can be consulted by the user through the GridWay commands and it is shown as:

- The URI contact endpoint, the protocol, hypervisor and VIM release, the maximum number of available cores, etc.
- Every OS template name (the *os_tpl*) and its *appdb.egi.eu* image identifier are compiled in a list of pairs and included as a new tag.

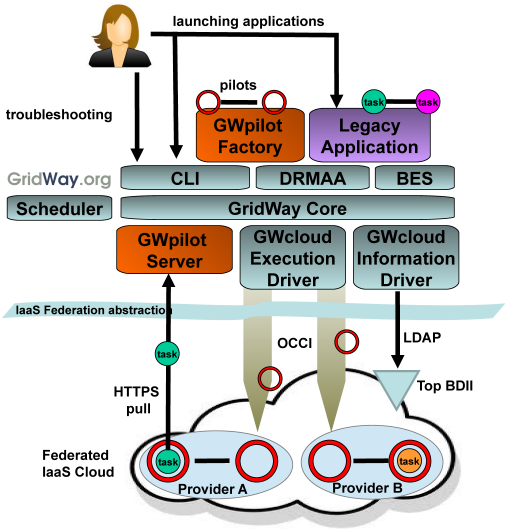


Figure 1: GridWay ecosystem architecture for cloud federations.

- Every resource template (*resource_tpl*) is shown as a different queue, with its own characterisation: number of cores, memory, etc.

To include resource templates as batch queues allows GridWay Scheduler to deal with cloud providers as they were grid sites to perform the VM brokering. For GridWay, a VM creation will be similar to a job execution.

Currently, the driver supports the EGI FedCloud IS, but it can be modified to directly use OCCI or AWS EC2 interfaces in order to work on a multi-cloud environment. The adaptation should be straightforwardly performed for the former protocol, because the characteristics taken into account by the driver are based on the description of resources made by this standard, as it was explained above. For EC2 the instance types are fixed, but similar to OCCI resource templates (*resource_tpl*), as well as AMI is to *os_tpl*. However, besides the proper translation, the driver must manage the availability zones and regions allowed to use, which should be addressed as different providers. When a commercial service is used (AWS itself), the spot price can be added as another characteristic of the queues.

4.2. The GWcloud Execution Driver (ED)

This driver enables the direct execution of a conventional grid job in a VM exclusively instantiated for this purpose. The driver can utilise the user's proxy credentials because it runs in the user-space mode. This allows using resources from federated clouds based on X.509 and VOMS. Additionally, the proxy is contextualised to be remotely used by jobs to access grid services. To preserve its integrity, the contextualisation file is encrypted, restricting the access only to secure OCCI services. On the other hand, the rOCCI-client [47] is used to perform the operations against the providers. Therefore, when the Scheduler chooses a cloud provider to execute the job, the driver performs the following steps:

1. It gets and stores the match, i.e. the description of the job and the URI of the OCCI service.

2. It interprets the job description to obtain the inputs, outputs and executable URIs, the *os_tpl*, and the *resource_tpl*.

3. Contextualisation: It makes a Cloud-Init file that includes:

   (a) creation of a new user with *sudo* privileges;

   (b) creation of a file with the temporal user proxy;

   (c) inclusion of the EUGridPMA repositories;

   (d) pre-installation of CA's certificates and minimal grid tools (*globus-url-copy*);

   (e) shell lines needed to download inputs, execute the job and store the outputs (i.e. through GridFTP or the Globus GASS protocols).

4. It builds and performs an OCCI *create* operation that includes the contextualisation file, the *resource_tpl*, the *os_tpl* and the URI of the provider. Subsequently, the job is considered in a *PENDING* state.

5. It waits for the VM starting to change the job state to *ACTIVE*. To make this periodically, it performs OCCI *describe* operations. If this circumstance does not happen during the timeout set in the job description, the job is considered as *FAILED*.

6. When the VM is running, the driver waits for the VM becoming into *inactive*; subsequently, the job is considered *DONE*. However, if other VM condition is reached, it returns *FAILED*.

7. Finally, it deletes the VM.

Note that a DONE state just only implies that the job was ended. It is the submitter (i.e. the user, some application or the pilot factory) who should interpret the exit status code or the outputs from the job.

The mechanism described is also feasible for AWS if the support of OIDC is implemented and OCCI operations are replaced by EC2 actions. However, the compatibility with grid services implies the use of X.509 proxies.

4.3. GWpilot and multi-level scheduling

The GWpilot framework counts on two main modules in addition to the pilots: the GWpilot Server and the Factory. The behaviour of both is fully described in [5, 48] and it is maintained in this work. On the other hand, the implementation of pilots is lightweight and without library dependencies, i.e. they can run on any kind of Linux OS. So, no especial configurations are needed to deploy the pilot overlay on cloud federations, allowing users to choose their virtual environment. This distinguishing feature has not been achieved by the current pilot systems that follow a pulling mechanism [36, 37].

According to the number and requirements of the tasks created by any user or application, the Factory automatically builds the necessary pilot jobs that will be executed in the VMs. Moreover, Factory takes account of the constraints and preferences related to the creation of the virtual environment that users set for these tasks. Those main ones are the:

- list of feasible image identifiers (e.g. *appdb.egi.eu* IDs for FedCloud);

- minimum memory, virtual CPUs, and local storage;

This is so because users do not have to know either the *os_tpl* or the *resource_tpl* of every provider, only what virtual environments can be used from the ones available at the cloud marketplace.

Factory only copies these requirements from task to pilots. The Scheduler is the one that matches them with a concrete *os_tpl* and *resource_tpl* at some provider. For this purpose, GridWay Scheduler will use the information that dynamically updates the GWcloud Information driver to select the most suitable cloud provider every time. Then, the management of the VM creation and the pilot execution is delegated to the GWcloud Execution driver. Consequently, the pilots executed will be enrolled to the GWpilot Server and the first level of scheduling (the resource provisioning) is successfully completed.

Therefore, users can run their legacy codes on GridWay as usual. The tasks created by these applications are scheduled among the pilots enrolled and subsequently within the virtual environment selected. This constitutes the second level of scheduling (the task scheduling). Potentially, the combination of levels allows advanced scheduling techniques [5], although, this work will be focused on how the user can guide the provisioning into a federated cloud.

To better understand how the task scheduling and resource provisioning are coordinated in a cloud federation, the sequence of steps is explained as follows and in Figure 2:

1. GWcloud ID periodically searches for cloud provider updates in the IS (top-BDIIs in FedCloud), which are included into the Host Pool.

2. The GridWay Scheduler notices that some cloud provider is *free* and fulfils the requirements of certain pilot waiting in the Job Pool. Consequently, the *SUBMIT* operation is sent to the GWcloud ED.

3. The GWcloud ED processes the operation as in Subsection 4.2, sending back a *CALLBACK* operation and performing the *create* operation against the OCCI service of the provider.

4. The provider creates the VM following the provided *os_tpl* and *resource_tpl*. Through the booting process, the Cloud-Init contextualisation starts the pilot job.

5. The pilot advertises the PiS and updates their characteristics. Therefore the provisioning phase is completed and begins the task scheduling as usual. GridWay Scheduler continuously dispatches tasks to the pilot if they are waiting in the Job Pool.

6. When the pilot is *idle* during certain interval, it *ends*, and as any other job, the VM automatically shuts down.

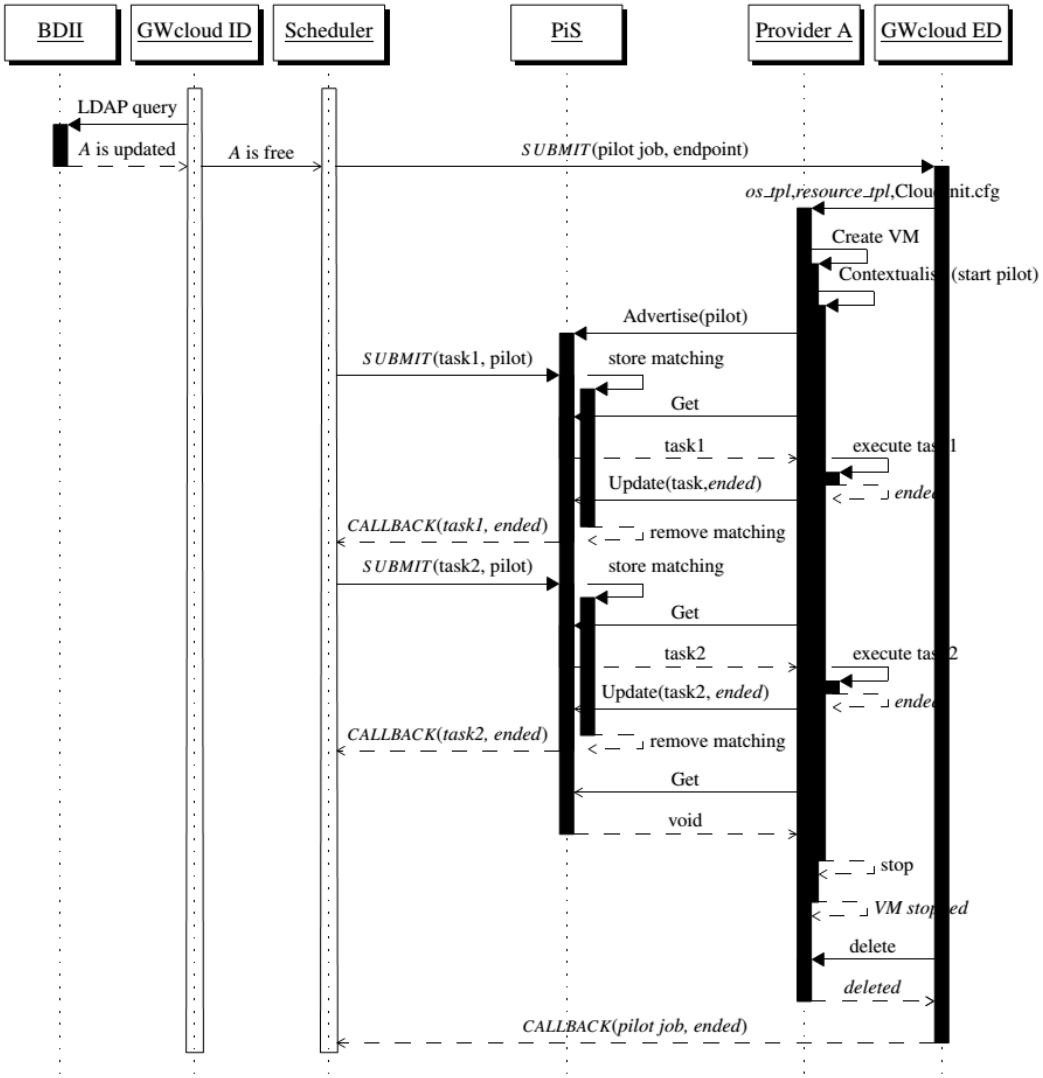


Figure 2: Sequence of activities performed by the actors of the framework to accomplish any task. They correspond to the steps 1-7 described in Subsection 4.3.

7. The GWcloud ED periodically tests the VM state through OCCI operations. If the VM shuts down, the driver performs the deletion of the virtual workspace because the pilot has *ended* its execution.

The basic configuration of the framework is similar to the ones described in [5, 48] but with two exceptions. The existence of a contextualisation file, which is not needed to be modified for a straightforward adaptation of any application, and the meaning of the $-w$ option, that is now related to the maximum allowed time to boot the VM. With relation to the later, the banning feature is the only way to currently know the quotas set at cloud providers that share their resources, as will be demonstrated through the experiments performed in the following section. This circumstance should not happen with commercial providers, for which the first step should be focused on detecting price changes rather than on their availability.

5. Experiments

The objective of this section is to show real use cases of profiting from federated clouds. The proposed experiments will demonstrate the capabilities of the solution that differentiates from others approaches, in particular:

a) the decentralisation and on-demand provisioning: calculations and VMs will be completely managed in the personal workstation of the user, or in a shared server of the institution for multiple corporate users;

b) customisation of cloud provision: any user in the system will deploy VMs with arbitrary configurations and posteriorly he will perform post-configuration tasks on these VMs;

c) multi-environment and fair-share preservation: several users can simultaneously run multiple applications that require different types of VMs;

Table 2: Requirements of the applications used in this work.

	XMM-Newton SAS	**GAMOS**	**FLUKA-64**
Scope	X-ray exploration	radiotherapy	matter interactions
Release	14.0.0	5.0.0	2011.2c
SO Requirements	Ubuntu 14.04	Centos 6.x	Linux 64-bits
Post-contextualisaton	HEAsoft >= 6.16	GEANT = 4.9.6, ROOT = 5.34.10	gfortran >= 4.4
requirements	xml/perl/curl/Xorg dev. libs.	gcc-c++	gfal2
	CDF files	Xorg dev. libs.	
Total distributable size (compressed)	1.5 GB	481 MB	148 MB
Installation size	2.6 GB	2.2 GB	605 MB
Input size (compressed)	173 MB	236 B	3.2 KB
Output size (compressed)	200 MB	~1 MB	26 MB
Execution time 1 task (Xeon X5365)	31 m	1 h 3 m	289 s
Number of tasks	200	100	1000
Serial execution time (Xeon X5365)	4 d 7 h 20 m	4 d 9 h 1 m	3 d 8 h 16 m

d) dynamic and configurable IaaS brokering across multiple providers: users will guide the provisioning and not only the workload scheduling;

e) performance: calculations will be compound by short tasks to show the low overheads that system introduces;

f) middleware independence: legacy grid services will be used to demonstrate that the system does not require especial services at IaaS providers, nor special VM images to be compatible with the established grid infrastructures;

g) compatibility with legacy applications: the codes used had been adapted following the DRMAA standard.

For this purpose, three real applications were distributed and executed on-demand across the FedCloud resources following the several fair-sharing criteria. The framework worked on customary hardware equivalent to a up-to-date personal computer. Additionally, the idea is not to deploy pre-configured VMs or VMs similar to the worker nodes used in the EGI grid infrastructure, i.e. Scientific Linux with gLite/UMD middleware. Therefore, a clean Debian-based and Red Hat-based template images will be chosen from the *appdb.egi.eu* marketplace repositories. Moreover, to demonstrate the compatibility with legacy codes, the distribution of tasks is managed through DRMAA.

5.1. Applications, post-configuration and real calculations

The objective of the experiments performed is to evaluate the suitability of the framework for legacy HTC applications. Moreover, it is of interest to demonstrate that the advantage of creating a personal virtualised environment is preserved. For this reason, applications which excessive dependencies and installation size that are difficult to deploy in grid are the ones selected for the experiments in cloud. The intention is to show how users can easily set up the virtual environments needed by these applications without dealing with contextualisation or uploading pre-configured virtual images. Furthermore, users only should write a script that configures the application, which will be submitted as a simple task to provisioned pilots.

Therefore, this subsection is focused on the requirements of the selected applications as well as the description of the real calculations performed to assure reproducibility. In this sense, the needs and calculus are completely different, as are summarised in Table 2.

5.1.1. FLUKA

FLUKA [49] is a general purpose tool for calculations of particle transport and interactions with matter. FLUKA[13] can simulate with high accuracy the interaction and propagation in matter of about 60 different particles and all the corresponding antiparticles. Materials can be simple elements, compounds or alloys.

In this work, up to a million particles have been simulated, each taking less than a second, for making studies on radiation interaction with matter. To be distributable, the division of workload was fixed to 1,000 particles per task, i.e. 1,000 tasks that will spend less than 5 minutes in current processors. On the other hand, FLUKA has not special requirements with the exception of being executed on Linux 64-bits platforms, for which it should be compiled before any calculation.

Therefore, this application will act as wild-card through performed experiments, demonstrating that very short tasks can be efficiently distributed in different customised virtual environments, following several scheduling policies.

5.1.2. GAMOS

The GEANT4-based Architecture for Medicine-Oriented Simulations (GAMOS[14]) [50] is a framework based on GEANT4 [51] specialized for the simulation of the radiation with the body, i.e. medical applications in both fields of medical image (PET/SPECT) and radiation therapy (teletherapy and brachytherapy). It also can simulate the needed doses to calibrate medial apparatus. The software can run on diverse Linux

[13] http://www.fluka.org

[14] http://fismed.ciemat.es/GAMOS/

distributions but it is available only as source code and requires certain releases of GEANT4 and ROOT [52]. The compilation of them lasts more than one hour on current processors. Consequently, unlike FLUKA, the complete framework has been pre-compiled for CentOS 6.7 to be transferred during the configuration phase of the virtual environment.

In every experiment performed a real calibration of the commercial LINAC VARIAN 2100 C/D [53] was performed. The calculation is composed by 100 tasks that calculate 34 millions of histories of particles [54] each. Everyone is in charge of calculating the particles' phase-space out of the LINAC head and posteriorly, of obtaining a dose deposition in a phantom by recycling the particles previously obtained on the phase-space.

5.1.3. XMM-Newton Science Analysis System (SAS) software

XMM-Newton is the most sensitive X-ray satellite ever built and the largest satellite ever launched by ESA. It has been operating as an open observatory [55] since the beginning of 2000. The large amount of data collected by XMM-Newton is due to its unprecedented effective area in the X-ray domain in combination with the simultaneous operation of all its instruments. All the data taken by this satellite are kept in the XMM-Newton Science Archive (XSA). The Scientific Analysis System (SAS) [56] is a software suite for the analysis of all these data. The execution of the SAS software on a grid has been successfully studied in [57]. However, the deployment of the new versions of the SAS software in grid infrastructures is not trivial, and requires an important effort from the VO administrators. This is so because developers of SAS only create compiled versions for few OS releases.

The current SAS[15] release (v14.0.0) has an installation size of 1.8 GB, and additionally requires external software such as HEAsoft[16](v6.17, 202 MB). The only Linux appliance currently suitable in FedCloud is an Ubuntu 14. On the other hand, besides to the observation data files (ODF, ~600 MB), SAS needs the current calibration files (CCF, another ~ 600 MB).

The calculation considered in this work is composed of 200 tasks, every one of which contains a script that installs and configures the SAS and HEAsoft software and then performs a default complete analysis (camera, spectrometers and optical monitor) based on the ODF provided. Obviously, ODF can be directly downloaded from the XSA repository by the script, but to preserve the comparison with other experiments and to show the compatibility with grid protocols, all the files were transferred through GASS from the *client host* as it were any grid storage element. SAS, HEAsoft and the CCF are downloaded by the script only when the provisioned VM has not been previously configured, they take 1.5 GB compressed.

On the other hand, different ODF are the input of different calculations and subsequently are always transferred in every execution. (As the experiments are an example, only the observation number 0144090201 was used, but always is transferred as if were different inputs). A task last 31 m on an X5365 processor, and generates an output of 339MB. As ODF and output are transferred compressed, every tasks moves 373MB in staging processes.

5.2. Deployment with fair-share rules

The objective is to show how the framework schedules different virtual environments following the application requirements and fair-share policies. For this purpose, the permeability from task scheduling and provisioning levels must be automatically performed, taking into account the possible limits/thresholds set by a hypothetical administrator.

5.2.1. Task scheduling

Users should have to worry about in what environment the applications will run. Thus, virtual SO images suitable for the applications were selected among the available ones at the *appdb.egi.eu* marketplace. The Basic Ubuntu Server 14.04 LTS[17] and the Centos 6[18] virtual appliances were used for the execution of XMM-Newton SAS and GAMOS respectively. This election was performed considering that these appliances are the most offered by cloud providers and they do not include other software than the basic installation with exception to Cloud-Init. Therefore, both applications included in their tasks the corresponding *appdb.egi.eu* ID as requirement of execution. On the other hand, FLUKA did not constraint the execution to one of these appliances. It worked as an opportunistic application that can use any free pilot at once.

Different initial priorities can be set for any application by different ways, for example, setting certain one by every user launching any application in the GridWay configuration. Additionally, initial priorities can be specified by including different deadlines in the description of the tasks, i.e the longer deadline, the smallerpriority. With their combination, several scheduling policies have been implemented through the experiments.

Moreover, the hardware templates (*resource_tpl*) are constrained to the ones offering one core and a minimum of 1GB of RAM. Note that driver automatically filters providers supporting the 1.1 release of OCCI offered through an encrypted endpoint. Then, it subsequently includes their hardware templates as available batch queues into the resource pool of GridWay. Therefore, users only have to specify these queues as any other requirement during the task submission.

5.2.2. Guided provisioning and configuration limits

The scheduling parameters concerned to pilots have been set as those of the first experiments in [5], except those related to cloud provisioning. In this sense, the Factory is allowed for managing a maximum of 50 pilots (running on VMs), but the Scheduler only will wait 600 s for the creation of every VM. On the other hand, the dispatching chunk, i.e. the number of tasks and VMs managed during a scheduling cycle of 10 s, is set to 20. However, the submission is also limited to dispatch

[15] http://xmm.esac.esa.int/sas/

[16] http://heasarc.gsfc.nasa.gov/docs/software/lheasoft/

[17] https://appdb.egi.eu/store/vo/image/de355bfb-5781-5b0c-9ccd-9bd3d0d2be06

[18] https://appdb.egi.eu/store/vo/image/e009209f-b62b-552e-b26c-eef351264f58

Table 3: FedCloud IaaS providers used in experiments. Sites that do not accomplish minimal requirements (image, OCCI 1.1, encrypted endpoint) are omitted. Additionally, the technology used in every resource is also shown.

Provider	*os_tpl#*		*resource_tpl#*		Max.	
(OCCI endpoint)	Ubuntu	CentOS	*ID*	GB	Cores	VIM
https://carach5.ics.muni.cz:11443	☒	☒	small	2	715	OpenNebula
https://cloud.cesga.es:3202	☒	☐	small	2	432	OpenNebula
https://controller.ceta-ciemat.es:8787	☒	☒	m1-small	2	224	OpenStack
https://egi-cloud.pd.infn.it:8787	☒	☒	m1-small	2	285	OpenStack
https://fc-one.i3m.upv.es:11443	☒	☒	small	1	16	OpenNebula
https://fsd-cloud.zam.kfa-juelich.de:8707	☒	☐	small	1	447	OpenStack
https://occi.nebula.finki.ukim.mk:443	☒	☐	small	1	360	OpenNebula
https://prisma-cloud.ba.infn.it:8787	☒	☒	small	1	600	OpenStack
https://sbgcloud.in2p3.fr:8787	☒	☒	m1-small	2	384	OpenStack
https://stack-server-01.ct.infn.it:8787	☒	☐	small	1	66	OpenStack

one VM per suitable provider in every cycle. In addition, the resource banning feature of GridWay is enabled, so whenever a resource fails it is banned for a variable period of time. This last option will be of importance in experiments, because, currently the only real accurate way to know the quotas established at providers belonging to FedCloud is by continuously testing the creation of VMs [46].

However, to take into account the virtual appliances needed by every application is essential for cloud brokering. In this sense, the main interest in GWpilot Factory is its capacity to perform a guided provisioning based on the description and status of every task. In general, the requirements of pending tasks with biggest priorities are used first in provisioning. Additionally, the running tasks increase the weight of certain requirements to avoid starvation if the remote resource fails. Consequently, Factory automatically manages the creation of certain types of VMs following the fair-share permeated from task scheduling.

Moreover, Factory allows forcing to break this fair-share by configuring a default weight to certain characteristics of providers. With this feature, administrators usually prioritise some resources over others, and improve some types of calculations.

In any case, any unskilled user can easily reproduce this configuration by modifying *gwd.conf* and *sched.conf* files.

5.2.3. Test proposed

Priorities allow task scheduling policies such as shortest task first (STF) and longest task first (LTF). Moreover, their influence on the guided provisioning can be controlled by configuring the GWpilot Factory to force the selection of some resources over others. In this sense, five experiments were performed in this work:

(1) Fair-sharing based on avoiding the task starvation: tasks only get more priority when their waiting time **WT** increases.

(2) The longest tasks are prioritised (LTF): the priority order is GAMOS, XMM-Newton SAS and FLUKA (e.g. the **GXM** policy).

(3) Shortest tasks are executed first (STF): the priority order is FLUKA, XMM-Newton and GAMOS (**FXG**).

(4) An arbitrary priority is set to provisioning: despite precedence in task scheduling (XGF), the environments suitable for running XMM-Newton are always prioritised, i.e. the creation of VM based on Ubuntu (**XGF.p**).

(5) Other arbitrary priority is set to provisioning: the opposite of last experiment, GAMOS is prioritised in task scheduling and its suitable environment (CentOS) in provisioning (**GFX.p**).

Every application only managed 50 tasks at same time and the three were simultaneously started in every experiment. The intention is to simulate a usual situation in an institution, where the resources are limited and fair-sharing rules are a must.

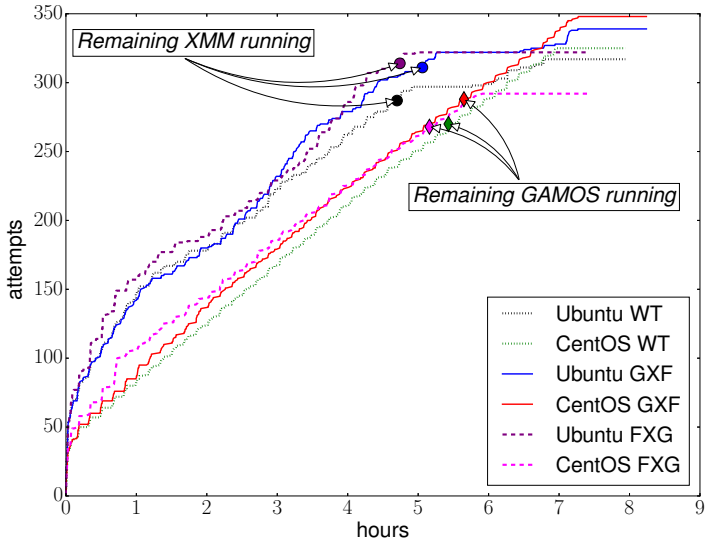
5.3. Test-bed

Experiments should be managed on the current hardware usually available for the corporate workplaces. Therefore, a machine with one i3-530 (2 cores, 2.93GHz) and 4GB of RAM was configured with GridWay, GWpilot and GWcloud drivers. Additionally, to avoid the necessity of host certificates, the system will use the Global Access to Secondary Storage (GASS) from Globus middleware as transference protocol.

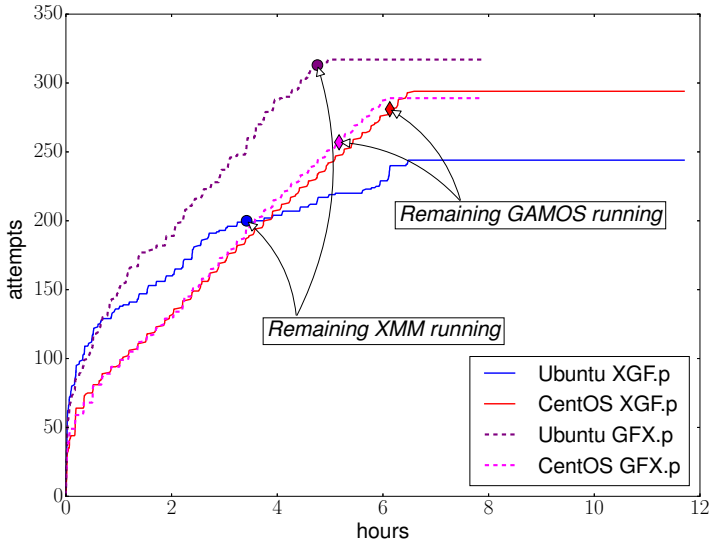
The EGI FedCloud infrastructure will be used to perform the tests. As that of July 2015, the infrastructure is considered in production and it counts on more than 40 providers, which offer more than 15,000 cores under the *fedcloud.egi.eu* virtual organisation.

5.4. Results

First impression is the noteworthy limitation of the amount of available resources (see Table 3). The number of theoretically suitable providers (10) was around the half of the available ones (24) in FedCloud. This is mainly due to the fact that the required *os_tpl* templates were not deployed in every site. Moreover, the number of reliable providers is really smaller (4), as it is shown in Tables 4 and 5. These results demonstrate the two affirmations done in this work: as grid sites, federated cloud providers are not immune from errors produced by the middleware, network outages, or misconfigurations; the other issue is the illusion of the fully availability of the resources, because user quotas are not currently shown in information systems. Therefore, experiments demonstrate the suitability of the



(a) Provisioning only guided by the priority and requirements of tasks.



(b) Certain OS is weighted in provisioning.

Figure 3: Accumulated VM creation attempts through the experiments with different task scheduling and provisioning policy.

proposed provisioning based on continuously checking the real availability and reliability of every resource. Scheduler always dispatches a VM to providers until it gets a failure from the GWcloud Execution driver. Consequently it bans the provider during $(1 - e^{\Delta t/260}) \cdot 600$ s, i.e. according to the elapsed time Δt from last failure, the banning time can be set to a maximum of one hour. These are the reasons for the number of failed creations in every reliable resource as they stand for the times that the driver has failed to create a VM because any quota has been reached. This circumstance does not necessarily imply that SLA contracts are being broken, but simply that these providers are full or they already supplied the maximum amount of resources guaranteed to this VO. However, in the case of unreliable resources, this number really means the number of failed VM creations and OCCI errors. This behaviour is usual in infrastructures focused on sharing resources that recently become in production [58], but it completely unacceptable with commercial providers.

The jagged shape of lines in Figures 3-(a,b) are also consequence of the provisioning procedure. Both describe the evolution of the accumulated creation attempts of the two kinds of VMs considered: Ubuntu and CentOS. Due to the effective limitation of resources by configuration (50 pilots) and by the reliability of the infrastructure, this kind of figure is more suitable

Table 4: Number of VM instances successfully set up and failed during the experiments in which only the requirement of certain virtual appliance is considered for provisioning.

	WT				GXF(=LTF)				FXG(=STF)				
	Ubuntu		CentOS		Ubuntu		CentOS		Ubuntu		CentOS		
Provider	up	fail	up	fail	up	fail	up	fail	up	fail	up	fail	
https://carach5.ics.muni.cz:11443	10	41	10	54	10	42	10	55	10	39	10	47	
https://cloud.cesga.es:3202	0	34	-	-	0	37	-	-	0	34	-	-	
https://controller.ceta-ciemat.es:8787	9	30	9	53	8	31	10	55	8	33	10	51	
https://egi-cloud.pd.infn.it:8787	1	19	0	47	1	21	0	50	0	21	0	44	
https://fc-one.i3m.upv.es:11443	0	25	0	70	0	28	0	73	4	27	9	56	
https://fsd-cloud.zam.kfa-juelich.de:8707	0	41	-	-	0	49	-	-	0	40	-	-	
https://occi.nebula.finki.ukim.mk:443	0	34	-	-	0	35	-	-	0	33	-	-	
https://prisma-cloud.ba.infn.it:8787	0	35	0	52	0	37	0	66	0	34	0	49	
https://sbgcloud.in2p3.fr:8787	0	30	0	49	0	32	0	49	0	33	0	45	
https://stack-server-01.ct.infn.it:8787	0	28	-	-	0	27	-	-	0	28	-	-	
Failures due to exceeding quotas		90		154		94		160		99		154	
Failures due to unreliable providers		227		171		245		188		223		138	
FLUKA		07 h 56 m 23 s				08 h 14 m 29 s				07 h 25 m 20 s			
GAMOS		07 h 13 m 24 s				07 h 14 m 45 s				07 h 03 m 54 s			
XMM-Newton SAS		05 h 12 m 49 s				05 h 39 m 59 s				06 h 05 m 02 s			

VM provisioning

Makespan

Table 5: Number of VM instances successfully set up and failed in the experiments where the priority was also considered for provisioning.

	XGF.p				GFX.p			
	Ubuntu		CentOS		Ubuntu		CentOS	
Provider	up	fail	up	fail	up	fail	up	fail
https://carach5.ics.muni.cz:11443	12	38	9	54	10	38	10	50
https://cloud.cesga.es:3202	0	33	-	-	0	32	-	-
https://controller.ceta-ciemat.es:8787	10	23	8	52	8	32	10	50
https://egi-cloud.pd.infn.it:8787	2	15	0	48	0	23	0	47
https://fc-one.i3m.upv.es:11443	7	21	6	46	4	28	8	48
https://fsd-cloud.zam.kfa-juelich.de:8707	0	27	-	-	0	42	-	-
https://occi.nebula.finki.ukim.mk:443	0	28	-	-	0	30	-	-
https://prisma-cloud.ba.infn.it:8787	0	18	0	47	0	35	0	48
https://sbgcloud.in2p3.fr:8787	0	22	0	47	0	26	0	46
https://stack-server-01.ct.infn.it:8787	0	19	-	-	0	31	-	-
Failures due to exceeding quotas	-	97	-	200	-	98	-	148
Failures due to unreliable providers	-	147	-	94	-	219	-	141
FLUKA		06 h 40 m 11 s				07 h 14 m 36 s		
GAMOS		11 h 42 m 17 s				07 h 52 m 10 s		
XMM-Newton SAS		05 h 22 m 57 s				06 h 10 m 19 s		

VM provisioning

Makespan

for evaluating the policy implemented in every experiment than to simply show the VMs provisioned through the time [46].

In the case where no prioritisation is forced in provisioning (Figure 3-(a)), it can be seen that experiments initially require more VMs with Ubuntu than with CentOS, which is a consequence of the higher number of providers offering the former OS. From one hour on, Factory performs a similar number of creation attempts for Ubuntu and CentOS as time goes by. Correspondingly, the small number of providers supporting CentOS causes a straight shape. When the XMM-Newton code does not require more VMs (marked with big dots) due to all remaining tasks are already running, it can be seen how the Factory performs some creation attempts for Ubuntu because these tasks retain some influence on provisioning as well as FLUKA can also benefit from these resources. The behaviour is repeated when GAMOS is close to end (diamonds). As commented before, this scheduling is performed to avoid starvation of XMM-Newton or GAMOS last tasks if some VM fail, which can increase dramatically their makespan. Independently from that, whenever the VMs creation attempts by XMM (dots) and by GAMOS (diamonds) are reached, the new attempts are only due to FLUKA as it is the one that can be executed on both Ubuntu and Centos. For this reason, provisioning is balanced between both OSs at the end of the WT and GXF experiments. In the case of FXG, where FLUKA are always prioritised, its shorter makespan avoid seeing this behaviour. On the other hand, Factory stops doing attempts for Ubuntu till the GAMOS code also stops requiring more CentOS (see again the WT and GXF experiments). This is due to Factory tries to progressively replace Ubuntu VMs by CentOS only if possible.

Regarding the experiment in which weighted provisioning is done (Figure 3-(b)), again there are more Ubuntu VMs creation attempts during the first hour due to the higher number of cloud providers supporting this OS. However, it can be also seen how the weighted provisioning influences the behaviour of the framework when the creation of VMs based on Ubuntu is always prioritised. In this sense, for the pair of solid lines it can been seen that the Factory with less attempts can boot more Ubuntu VMs (see XGF.p experiment in Table 5). As XMM-Newton has a higher weight, the framework processes XMM-Newton tasks prior to FLUKA ones, but both calculations end earlier. Obviously, this policy is detrimental to GAMOS ex-

ecution. On the opposite, for the dash lines, where GAMOS running only in CentOS has a higher weight, a much higher value of Ubuntu creation attempts is performed by the Factory as reliable providers become saturated of CentOS and reached their quotas.

Last but not least, looking to Table 5 it is demonstrated that weighted provisioning is worth performed for the application with longer task execution time. Being FLUKA the application with shorter task execution time, its makespan varies an 8% between being XMM or GAMOS the application with a higher weighted provisioning, so its influence is low. Nonetheless, GAMOS makespan is reduced to a 0.67 ratio if it has a higher weighted provisioning instead of XMM as the latter only increases its makespan to a 1.15 ratio.

The experiments performed demonstrate the points a)-b) stated at the beginning of this section. However, the scheduling implemented is focused on exploiting the behaviour of a federation based on the resource sharing. However, the mechanism to constraint resources and the fair-share policies are of importance to properly select commercial providers. Users can force some of their applications to use only the cheaper resources and allow others to progressively request VMs more expensive. For this purpose, they only need to set the corresponding requirement or ranking statement in every task description. Moreover, administrators can configure the Factory to force the use of certain cloud providers with favourable agreements. Combinations of the features provided by the framework are suitable for implementing a wide range of scheduling suitable for any kind of federation or multi-cloud environment.

6. Conclusions and future work

A generic framework for performing massive distributed calculations in federated clouds has been presented in this work. Unlike other approaches, the system supports multi-environments and fair-sharing, so several users can simultaneously run multiple legacy applications that require different types of VMs. It is able to perform a dynamic, configurable, automatic, and efficient IaaS brokering based on the current status of the cloud federations and also allows the decentralisation and on-demand provisioning of customised virtual environments in cloud. All the previous capabilities clearly differentiate it from other approaches.

The suitability of the framework has been successfully demonstrated by effectively running three applications on the European FedCloud at the same time. Applications require different customised virtual environments, which have been deployed following several scheduling policies based on permeating the requirements of tasks.

The design of the framework also allows preserving these features in a multi-cloud environment. However, the use of public providers implies the development of strategies based on costs and budgets, different to the ones suitable for a federation of resources. The explanations, implementations and subsequent tests give room to perform new contributions of this research within future works.

Acknowledgements

This work was supported by the COST Actions BETTY (IC 1201) and NESUS (IC 1305), as well as partially funded by the Spanish Ministry of Economy and Competitiveness projects CODEC (TIN2013-46009-P) and FedCloudNet (TIN2015-65469-P). Additionally, part of this work is based on observations obtained with XMM-Newton, an ESA science mission with instruments and contributions directly funded by ESA Member States and the USA (NASA).

References

[1] I. Foster, C. Kesselman, J. Nick, S. Tuecke, Grid services for distributed system integration, Computer 35 (6) (2002) 37–46. doi:10.1109/MC.2002.1009167.

[2] M. Riedel, E. Laure, T. Soddemann, L. Field, et al., Interoperation of world-wide production e-Science infrastructures, Concurrency Computat.: Pract. Exper. 21 (8) (2009) 961–990. doi:10.1002/cpe.1402.

[3] R. Jain, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling, Wiley- Interscience, 1991.

[4] A. J. Rubio-Montero, M. A. Rodríguez-Pascual, R. Mayo-García, Evaluation of an adaptive framework for resilient Monte Carlo executions, in: 30th ACM/SIGAPP Symposium On Applied Computing (SAC'15), ACM New York, Salamanca, Spain, 2015, pp. 448–455. doi:10.1145/2695664.2695890.

[5] A. J. Rubio-Montero, E. Huedo, F. Castejón, R. Mayo-García, GWpilot: Enabling multi-level scheduling in distributed infrastructures with GridWay and pilot jobs, Future Generation Computer Systems 45 (2015) 25–52. doi:10.1016/j.future.2014.10.003.

[6] A. Kertesz, Characterizing Cloud Federation Approaches, Computer Communications and Networks, Springer, 2014, Ch. 12, pp. 277–296. doi:10.1007/978-3-319-10530-7_12.

[7] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, I. M. Llorente, Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, Future Generation Computer Systems 28 (2) (2012) 358–367. doi:10.1016/j.future.2011.07.003.

[8] N. Grozev, R. Buyya, Inter-Cloud architectures and application brokering: taxonomy and survey, Softw: Pract. Exper. 44 (2014) 369–390. doi:10.1002/spe.2168.

[9] M. Garey, D. Johnson, Computers and Intractibility: A guide to the Theory of NP-completeness, W. H. Freeman&Co, New York, 1979.

[10] G. Aceto, A. Botta, W. de Donato, A. Pescapè, Cloud monitoring: A survey, Computer Networks 57 (9) (2013) 2093–2115. doi:10.1016/j.comnet.2013.04.001.

[11] M. Sheikhalishahi, R. Wallace, L. Grandinetti, J. L. Vázquez-Poletti, F. Guerriero, A multi-dimensional job scheduling, Future Generation Computer SystemsAvailable Online. doi:10.1016/j.future.2015.03.014.

[12] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, W. Karl, Scientific Cloud Computing: Early Definition and Experience, in: 10th IEEE International Conference on High Performance Computing and Communications (HPCC'08), Dalian, China, 2008, pp. 825–830. doi:10.1109/HPCC.2008.38.

[13] A. Edmonds, T. Metsch, A. Papaspyrou, A. Richardson, Toward an open cloud standard, IEEE Internet Computing 16 (4) (2012) 15–25. doi:10.1109/MIC.2012.65.

[14] G. S. P. Kacsuk, Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal, Journal of Grid Computing 3 (3–4) (2005) 221–238. doi:10.1007/s10723-005-9012-6.

[15] M. A. ao, R. Buyya, S. Venugopal, InterGrid: A Case for Internetworking Islands of Grids, Concurrency and Computation: Practice & Experience 20 (8) (2008) 997–1024. doi:10.1002/cpe.1249.

[16] K. Laskey, P. Brown, J. A. Estefan, F. G. McCabe, D. Thornton, Reference Architecture Foundation for Service Oriented Architecture 1.0, soa-ra-v1.0-cs0 (04 December 2012).
URL `http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html`

[17] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Muñoz, G. Toffetti, Reservoir - When One Cloud Is Not Enough, Computer 44 (3) (2011) 44–51. doi:10.1109/MC.2011.64.

[18] Helix-Nebula, D5.4 Final Flagship Deployment Report, Deliverate (30 May 2014).
URL http://helix-nebula.eu/publications/deliverables/d54-final-flagship-deployment-report

[19] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, F. Galan, The Reservoir model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 1–11. doi:10.1147/JRD.2009.5429058.

[20] P. Harsh, Y. Jegou, R. G. Cascella, C. Morin, Contrail Virtual Execution Platform Challenges in Being Part of a Cloud Federation, in: 4th European Conference, (ServiceWave 2011), Vol. 6994 of Towards a Service-Based Internet. Lecture Notes in Computer Science, Poznan, Poland, 2011, pp. 50–61. doi:10.1007/978-3-642-24755-2_5.

[21] C. Loomis, M. Airaj, M.-E. Bégin, E. Floros, S. Kenny, D. O'Callaghan, StratusLab Cloud Distribution, in: European Research Activities in Cloud Computing, Cambridge Scholars Publishing, pp. 260–282, 2012. ISBN(10) 1-4438-3507-2, ISBN(13) 978-1-4438-3507-7.

[22] F. H. B. Megino, R. Jones, K. Kucharczyk, R. M. Llamas, D. van der Ster, Helix Nebula and CERN: A Symbiotic approach to exploiting commercial clouds, in: 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013), Vol. 513 of Journal of Physics: Conference Series, 2014, p. 032067. doi:10.1088/1742-6596/513/3/032067.

[23] A. Simón, E. Freire, R. Rosende, I. Díaz, A. Feijóo, P. Rey, J. López-Cacheiro, C. Fernández, EGI FedCloud Task Force, in: 6th Grid Iberian Infrastructure Conference (IBERGRID'12), Lisbon, Portugal, 2012, pp. 183–194.

[24] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, R. M. Badia, ServiceSs: An Interoperable Programming Framework for the Cloud, J. Grid Computing 12 (1) (2014) 67–91. doi:10.1007/s10723-013-9272-5.

[25] P. Tröger, A. Merzky, Towards Standardized Job Submission and Control in Infrastructure Clouds, J. Grid Computing 12 (2014) 111–125. doi:10.1007/s10723-013-9275-2.

[26] G. F. Anastasi, E. Carlini, M. Coppola, P. Dazzi, BROKAGE: A Genetic Approach for QoS Cloud Brokering, in: 7th IEEE International Conference on Cloud Computing (IEEE CLOUD 2014), Alaska. USA, 2014, pp. 304–311. doi:10.1109/CLOUD.2014.49.

[27] S. Yangui, I.-J. Marshall, J.-P. Laisne, S. Tata, CompatibleOne: The Open Source Cloud Broker, J. Grid Computing 12 (1) (2014) 93–109. doi:10.1007/s10723-013-9285-0.

[28] P. Mhashilkar, A. Tiradani, B. Holzman, K. Larson, I. Sfiligoi, M. Rynge, Cloud Bursting with GlideinWMS: Means to satisfy ever increasing computing needs for Scientific Workflows, in: 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013), Vol. 513 of Journal of Physics: Conference Series, IOP Publishing, 2014, p. 032069. doi:10.1088/1742-6596/513/3/032069.

[29] A. Lorca, J. Martín-Caro, R. Nún'ez-Ramírez, J. Martínez-Salazar, Merging on-demand HPC resources from Amazon EC2 with the grid: a case study of a Xmipp application, Computing and Informatics 31 (1) (2012) 17–30.

[30] M. Rodríguez, D. Tapiador, J. Fontan, E. Huedo, R. Montero, I. Llorente, Dynamic provisioning of virtual clusters for grid computing, in: 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC08), Vol. 5415 of Euro-Par 2008 Workshops - Parallel Processing. Lecture Notes in Computer Science, 2009, pp. 23–32. doi:10.1007/978-3-642-00955-6_4.

[31] C. Vázquez, E. Huedo, R. S. Montero, I. M. Llorente, On the use of clouds for Grid resource provisioning, Future Generation Computer Systems 27 (5) (2011) 600–605. doi:10.1016/j.future.2010.10.003.

[32] G. Juve, E. Deelman, Automating Application Deployment in Infrastructure Clouds, in: Third International Conference on Cloud Computing Technology and Science (CloudCom), 2011, pp. 658–665. doi:10.1109/CloudCom.2011.102.

[33] E. Walker, J. P. Gardner, V. Litvin, E. L. Turner, Personal adaptive clusters as containers for scientific jobs, Cluster Computing 10 (3) (2007) 339–350. doi:10.1007/s10586-007-0028-5.

[34] R. Moreno-Vozmediano, R. S. Montero, I. M. Llorente, Multi-Cloud Deployment of Computing Clusters for Loosely-Coupled MTC Applications, IEEE Transactions on Parallel and Distributed Systems 22 (6) (2011) 924–930. doi:10.1109/TPDS.2010.186.

[35] A. Luckow, M. Santcroos, A. Zebrowski, S. Jha, Pilot-Data: An abstraction for distributed data, Journal of Parallel and Distributed ComputingAvailable online. doi:10.1016/j.jpdc.2014.09.009.

[36] R. Graciani, A. Casajús, A. Carmona, T. Fifield, M. Sevior, Belle-DIRAC Setup for Using Amazon Elastic Compute Cloud, Journal of Grid Computing 9 (1) (2011) 65–79. doi:10.1007/s10723-010-9175-7.

[37] J. Kovács, A. C. Marosi, A. Visegrádi, Z. Farkas, P. Kacsuk, R. Lovas, Boosting gLite with cloud augmented volunteer computing, Future Generation Computer Systems 43–44 (2015) 12–23. doi:10.1016/j.future.2014.10.005.

[38] A. J. Rubio-Montero, R. S. Montero, E. Huedo, I. M. Llorente, Management of Virtual Machines on Globus Grids using GridWay, in: 21st IEEE Int. Parallel and Distributed Processing Symposium (IPDPS 2007), IEEE CS Press, Long Beach, USA, 2007, pp. 1–7. doi:10.1109/IPDPS.2007.370548.

[39] J. T. Mościcki, M. Lamannaa, M. Bubak, P. M. A. Sloot, Processing moldable tasks on the Grid: Late job binding with lightweight user-level overlay, Future Generation Computer Systems 27 (6) (2011) 725–736. doi:10.1016/j.future.2011.02.002.

[40] E. Michon, J. Gossa, S. Genaud, M. Frincu, A. Burel, Porting Grid Applications to the Cloud with Schlouder, in: IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), Bristol, United Kingdom, 2013, pp. 505–512. doi:10.1109/CloudCom.2013.73.

[41] K. Torberntsson, Y. Rydin, A Study of Configuration Management Systems. Solutions for Deployment and Configuration of Software in a Cloud Environment, B.S. Thesis. Uppsala University. Sweden (June 2014).

[42] R. S. Montero, R. Moreno-Vozmediano, I.M.Llorente, An elasticity model for High Throughput Computing clusters, J. Parallel Distrib. Comput. 71 (6) (2011) 750–757. doi:10.1016/j.jpdc.2010.05.005.

[43] V. Méndez, A. Casajús, V. Fernández, R. Graciani, G. Merino, Rafhyc: an Architecture for Constructing Resilient Services on Federated Hybrid Clouds, J. Grid Computing 11 (2013) 753–770. doi:10.1007/s10723-013-9279-y.

[44] S. Sehgal, M. Erdelyi, A. Merzky, S. Jha, Understanding application-level interoperability: Scaling-out MapReduce over high-performance grids and clouds, Future Generation Computer Systems 27 (5) (2011) 590–599. doi:10.1016/j.future.2010.11.001.

[45] E. Huedo, R. S. Montero, I. M. Llorente, A modular meta-scheduling architecture for interfacing with pre-WS and WS Grid resource management services, Future Generation Computer Systems 23 (2) (2007) 252–261. doi:10.1016/j.future.2006.07.013.

[46] A. J. Rubio-Montero, E. Huedo, R. Mayo-García, User-guided provisioning in federated clouds for distributed calculations, in: Workshop on Adaptive Resource Management and Scheduling for Cloud Computing, Lecture Notes in Computer Science, Springer, San Sebastián, Spain, 2015, p. (In print).

[47] B. Parák, Z. Šustr, F. Feldhaus, P. Kasprzakc, M. Srbac, The rOCCI Project: Providing Cloud Interoperability with OCCI 1.1, in: International Symposium on Grids and Clouds (ISGC), SISA PoS, Taipei, Taiwan, 2014, pp. 1–15.

[48] A. J. Rubio-Montero, F. Castejón, E. Huedo, R. Mayo-García, A novel pilot job approach for improving the execution of distributed codes: application to the study of ordering in collisional transport in fusion plasmas, Concurrency and Computation: Practice & Experience 27 (13) (2015) 3220–3244. doi:10.1002/cpe.3301.

[49] T. Böhlen, F. Cerutti, M. Chin, A. Fassò, A. Ferrari, P. Ortega, A. Mairani, P. Sala, G. Smirnov, V. Vlachoudis, The FLUKA Code: Developments and Challenges for High Energy and Medical Applications, Nuclear Data Sheets 120 (2014) 211–214. doi:10.1016/j.nds.2014.07.049.

[50] P. Arce, J. I. Lagares, L. Harkness, D. Pérez-Astudillo, M. Ca'nadas, P. Rato, M. de Prado, Y. Abreu, G. de Lorenzo, M. Kolstein, A. Díaz, GAMOS: A framework to do Geant4 simulations in different physics fields with an user-friendly interface, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 735 (2014) 304–313. doi:10.1016/j.nima.2013.09.036.

[51] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, F. Behner, L. Bellagamba, J. Boudreau, L. Broglia, A. Brunengo, H. Burkhardt, S. Chauvie, J. Chuma, R. Chytracek, et al., Geant4 - a simulation toolkit, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 506 (3) (2003) 250 – 303. doi:10.1016/S0168-9002(03)01368-8.

[52] I. Antcheva, M. Ballintijn, B. Bellenot, M. Biskup, R. Brun, N. Buncic, P. Canal, D. Casadei, O. Couet, V. Fine, L. Franco, G. Ganis, A. Gheata, D. G. Maline, M. Goto, J. Iwaszkiewicz, A. Kreshuk, D. M. Segura, R. Maunder, L. Moneta, A. Naumann, E. Offermann, V. Onuchin, S. Panacek, F. Rademakers, P. Russo, M. Tadel, ROOT - A C++ framework for petabyte data storage, statistical analysis and visualization, Computer Physics Communications 180 (12) (2009) 2499 – 2512. doi:10.1016/j.cpc.2009.08.005.

[53] M. Quintero, P. Arce, J. Lagares, Caracterización del haz de aceleradores de fotones de uso terapéutico para su simulación con GAMOS/GEANT4 y BEAMnrc/EGSnrc (2-5 de June 2009).

[54] A. J. Rubio-Montero, P. Arce, J. I. Lagares, Y. P. Ivanov, D. A. Burbano, G. Díaz, R. Mayo, Performance Tests of GAMOS Software on EELA-2 Infrastructure, in: Proceedings of the Second EELA-2 Conference, Editorial CIEMAT (Madrid, Spain), Choroní, Venezuela, 2009, pp. 379–385.

[55] F. Jansen, D. Lumb, B. Altieri, J. Clavel, M. Ehle, C. Erd, C. Gabriel, M. Guainazzi, P. Gondoin, R. Much, R. Muñoz, M. Santos, N. Schartel, D. Texier, G. Vacanti, XMM-Newton observatory (I. The spacecraft and operations), Astronomy & Astrophysics 365 (1).

[56] C. Gabriel, M. Denby, D. J. Fyfe, J. Hoar, A. Ibarra, E. Ojero, The XMM-Newton SAS - Distributed Development and Maintenance of a Large Science Analysis System: A Critical Analysis, in: Astronomical Data Analysis Software and Systems (ADASS) XIII, Vol. 314 of ASP Conference Proceedings, Strasbourg, France, 2004, pp. 759–763.

[57] A. Ibarra, D. Tapiador, E. Huedo, R. Montero, C. Gabriel, C. Arviset, I. Llorente, On-the-fly XMM-Newton Spacecraft Data Reduction on the Grid, Scientific Programming 14 (2) (2006) 141–150. doi:10.1155/2006/739583.

[58] A. J. Rubio-Montero, L. Flores, F. Castejón, E. Montes, M. Rodríguez-Pascual, R. Mayo, Executions of a Drift Kinetic Ecuation solver on Grid, in: 18th Euromicro Int. Conf. on Parallel, Distributed and Network-Based Processing (PDP 2010), IEEE CS Press, Pisa, Italy, 2010, pp. 454–459. doi:10.1109/PDP.2010.40.

***Biographies (Text)**

Antonio Juan Rubio Montero received the M.Sc. degree in computer science from the Universidad Complutense de Madrid (UCM), Madrid, Spain, in 2003. He was a Programmer and Systems Architect with several private companies. Since 2006, he has been a Researcher on Grid technologies at CIEMAT. He is the author of more than 30 publications focused on Grid application porting and Cloud computing, which were the results from his participation in several international and national Grid initiatives. He is also in charge of maintaining the Grid sites and participates in the administration of the computational resources devoted to scientific research belonging to the ICT Division, CIEMAT.

Eduardo Huedo Cuesta received his M.E. in computer science (1999) and Ph.D. in Computer Architecture (2004) from the Universidad Complutense de Madrid (UCM). He is an Associate Professor of Computer Architecture and Technology at UCM. Previously, he was a Postdoctoral Researcher at the Advanced Computing Laboratory at Centro de Astrobiolog\'ia (CSIC-INTA), associated to the NASA Astrobiology Institute. He has published more than 60 scientific papers in the field of High-Performance, Distributed, Grid and Cloud Computing, and contributed to more than 20 research and development programmes.

Rafael Mayo Garcia received the Ph.D. degree in physics from the Universidad Complutense de Madrid (UCM). He was with UCM as a Researcher in experimental and computational plasma physics, and as Adjunct Faculty. He was also with Data Networks. Since 2005, he is with the CIEMAT Supercomputation and Grid Developments Unit. He is the author of 23 articles and more than 60 papers and has been involved in more than 30 international R&D projects where he has also organized managerial and coordinating activities. Dr. Mayo-Garc\'a has obtained several postdoctoral fellowships such as Marie Curie and Juan de la Cierva grants.

***Biographies (Photograph)**

Click here to download high resolution image