Cloud Governance by a Credit Model with DIRAC

Víctor Méndez Muñoz[1], Adrian Casajús Ramo[2], Ricardo Graciani Diaz[2] and Andrei Tsaregorodtsev[3]
[1]*Computer Architecture and Operating Systems Department, Universitat Autònoma de Barcelona (UAB), Edifici Q, Campus UAB, ES-08193 Bellaterra, Spain*
[2]*Department of Structure and Constituents of Matter, University of Barcelona, Diagonal 647, ES-08028 Barcelona, Spain*
[3]*Centre de Physique des Particules de Marseille, 163 Av de Luminy Case 902 13288 Marseille, France*

Keywords: Cloud Computing, Federated Cloud, Cloud Governance.

Abstract: Nowadays, different eScience actors are assuming the Federated Cloud as a model for the aggregation of distributed cloud resources. In this complex environment, service delivery and resource usage is an open problem, where multiple users and communities are committed to particular policies while using federated resources. In the other hand, DIRAC has become a multi-community middleware fully interoperable in Distributed Computing Infrastructures (DCI), including several cloud managers. Furthermore, DIRAC is able to federate Infrastructure as a Service (IaaS) to provide Software as a Service (SaaS) in a transparent manner to the final user. This paper defines a credit model for the resource usage providing automatic management in federated cloud. It is presented a prototype of this model, which is implemented with DIRAC, describing a test for the model assessment and drawing up conclusions for a production level federated cloud governance.

1 INTRODUCTION

The benefits of cloud computing are related with a set of key features. Cloud uses Internet access to remote services or resources supported by virtual storage and computing. It is provided on-demand and depending on user workload it can scale up or down without affecting the service level. A main asset of these features is a virtual running environment deployed for particular user requirements, instead of a bare-metal platform under provider specifications. By this mean Cloud is saving operational costs and simplifying software maintenance. Moreover, the cloud scaling features are of great importance for an efficient use of the resources.

The cloud key features can be implemented in different ways, depending on the particular physiognomy of the cloud deployment model. In this sense, federated cloud model is the evolution of private clouds, aggregating not only IaaS, but also additional services for eScience communities such as monitoring, authentication, authorization, marketplace and accounting (Simon et al., 2012).

DIRAC (Tsaregorodtsev et al., 2008) has evolved from single community to multi-community framework. The cloud management of DIRAC follows Rafhyc architecture (Méndez et al., 2013b) for clouds of hybrid nature. DIRAC is using Cloud resources in the LHCb production (Ubeda et al., 2014), federating private clouds supported by OpenStack (OpenStack, 2014) and OpenNebula (Llorente et al., 2011). Another production level use of cloud was in Belle Monte Carlo simulation campaign with AWS (Lerner, 2006) using Amazon EC2 interface (Graciani Diaz et al., 2011).

This paper is focused in the next challenge of providing federating cloud in multi-community environments. For this purpose it is necessary to leverage the use of the resources between different communities. This is achieved by a governance between the policies of IaaS and the policies of communities. The main idea is to assign resource credits in terms of CPU, memory or IO usage (temporal and permanent storage as well as network resources), to the running campaigns of the communities. Such credits are consumed on-demand, with a pay-per-use approach, but instead of legal tender resource credits are used. This pay-per-use concept can be transparently adopted in the case of commercial clouds prices, where the credits are simply pre-payment. Main cloud managers are providing billing systems which can be enabled for pricing. However, non commercial clouds are coming from capital expenditure (Capex) model in the use of the resources. In this model user invests capital

Méndez Muñoz V., Casajús Ramo A., Granciani Diaz R. and Tsaregorodtsev A..
Cloud Governance by a Credit Model with DIRAC.
DOI: 10.5220/0004977806790686
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (FedCloudGov-2014), pages 679-686
ISBN: 978-989-758-019-2

in ownership infrastructures, or delegates invest to a computing hosting institution. Then, resources are allocated without a pay-per-use on-demand scheme, just relating investment negotiations with available power and storage capacity. A first step to provide an on-demand price is to have a total cost breakdown of the resources in an updated system, then to apply a commercial margin considering the market competence. Significant advances in the last years (Cohen et al., 2013) (Konstantinou et al., 2012) are targeting a costs management level. In the meantime, the presented model based in resource credit metrics is an affordable approach to offer on-demand resources in non commercial IaaS sites. Furthermore, it is compatible with future metrics, when price management will reach maturity in non commercial clouds.

An additional important point of the provided governance is about keeping the key features of cloud computing, with elastic allocation of resources as well as Virtual Machine (VM) configuration to run any scientific software.

Section 2 describes a previous approach of federated cloud with DIRAC. The contribution of the paper is a credit model for cloud governance in Section 3. Additional contribution in Section 4, describes a prototype implementation with DIRAC and test results for model assessment is shown in Section 5. Conclusions and future work can be found in Section 6.

2 PREVIOUS APPROACH OF FEDERATED CLOUD WITH DIRAC

DIRAC has a cloud extension integrating different IaaS providers which need to fulfil some requisites (Méndez et al., 2013a). In addition to the mentioned Amazon, OpenNebula and OpenStack clouds, CloudStack (CloudStack, 2014) has also been integrated with DIRAC (Méndez et al., 2012) (Albor et al., 2011). Beyond providing the aggregation of different cloud managers, DIRAC includes federated services for eScience communities like monitoring, authorization or image metadata catalog.

Regarding the key features of cloud computing, DIRAC cloud extension allows VM scheduling policies for different possibilities of VM horizontal auto scaling set-up. The scale up can be more aggressive and scale down softer, to reach better job response times. On the other hand, soft scale up and hard scale down can be set-up to reduce VM overheads and therefore a better efficiency, when job response time is not a constrain.

Another key feature is the image management by generic contextualization to configure the VMs depending on the SaaS requirements and the IaaS endpoint within to be deployed. I.e. DIRAC can use a golden image which is configured for different purposes and IaaS environments. This image management saves operational costs and minimizes complexity. Two contextualization approaches are contemplated, HEPiX contextualization for *cernvm* images (Bunic et al., 2011), and *ssh* contextualization for generic images, which require an available public IP and *sshd* service running on boot. Both of them can be combined with *cvmfs* software repository (Jakob Blomer, 2010), simplifying the software distribution in the VMs.

Fig. 1 shows the main part of VM allocation and job brokering. User submits jobs to the DIRAC Workload Management System (WMS), which are initially stored in the Task Queue (TQ), with an associated SaaS workload. Then, VM Scheduler is checking each of the queues and depending on workload it is instantiating new VMs with specific requirements for the queued jobs. Latter, when the VM is created, contextualized to run DIRAC VM agents and configured with SaaS requirements, then, Job Agent starts and secured connects to the Matcher to get the corresponding SaaS payload.

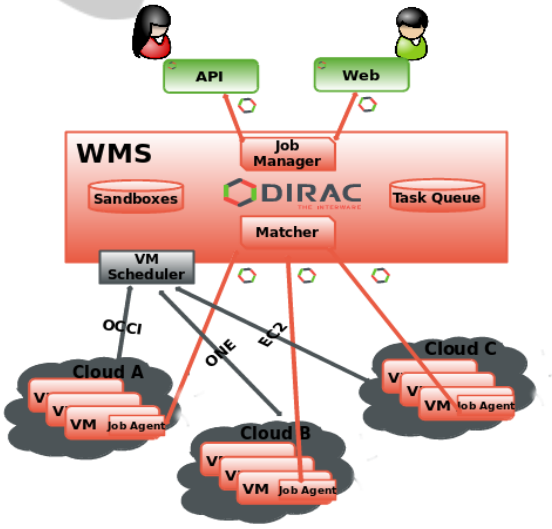

Figure 1: DIRAC VM schedulig with job brokering.

For simplicity, further VM features are not drawn in Fig. 1. It is worth to mention the VM stoppage and status management. A VM runs many jobs until some stoppage conditions are reached. VM Monitor agent is in charge of VM status update, reporting statistics to the VM Manager, as well as VM stoppage control. VM Monitor is a client of the VM Manager

service at the DIRAC server side. Stop command can be launched from VM when no jobs are running in the VM for a certain period, or it can also be requested to stop by an authorized external request from VM Manager, then, VM Monitor is doing a graceful stoppage after current job is finished.

3 CREDIT MODEL FOR CLOUD GOVERNANCE

This section presents a credit model for cloud governance of multi-community environments for the use of federated cloud resources. Fig. 2 shows such model scheme. In the bottom of the diagram supply is supported by Resource Providers (RP). In the top of the diagram User Communities (UC) demand services. Cloud Governance Actors (CGA) are found in the mediation between RP and UC.

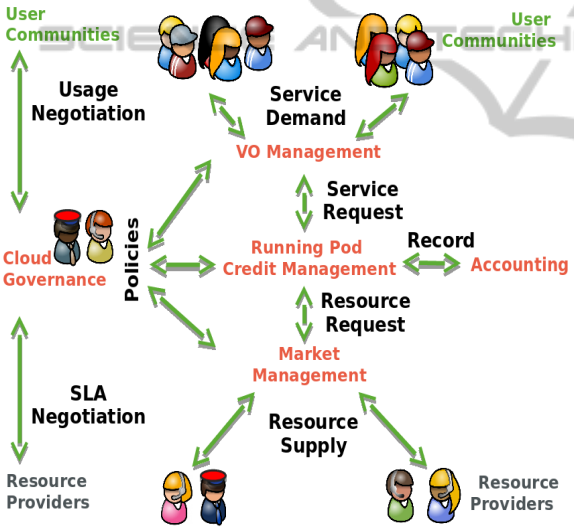

Figure 2: Credit model for cloud governance.

Previously to matching resources and services, it is necessary to negotiate the governance conditions. For this purpose each actor cover basic actions. GCA takes the central part of the governance management. To this concern it is necessary to have an overview of the hold production, providing scientific directions for the UC, including technical guidance of how to run scientific software in the Cloud. GCA also require to have an overview of the different RP technologies as well as a know-how in federated cloud management technologies. Cloud governance process is starting in the left of the diagram, from bottom-up flow, Service Level Agreements are continuously negotiated with every RP to obtain an overall resource supply power. Considering this power as a constrain of the model, usage is negotiated with every UC attending to scientific directions for the different use cases, as well as the UC financing contribution to the overall system.

Governance specifications are updated through a continuous negotiation process. Once supply and demand are identify, it is possible to define governance policies to be apply in three levels of the credit model, which are shown in the central part of Fig. 2. Usually, a user community is a Virtual Organization (VO) in terms of authorization in the use of resources. Eventually, if a single VO is composed of many UCs, it can be broken down into virtual groups. VO policies describe which *running pods* are assigned for each UC or virtual group, to be able to run their SaaS request. The running pod is an abstraction of the requisites to be included in a VM which can be submitted to certain IaaS providers. Secondary, VO policies can define external links to third party VO authorization for user and group management. Therefore, user service requests can be matched with specific running pods.

Running pod policies define how many overall resource credits are assigned for an specific period of usage. Furthermore, policies also describe which IaaS RP can be used in the running pod, since UC can use a subset of the federated resources to meet with particular SLA running requisites. Image requirements for a SaaS running pod are also defined, including the contextualization parameters for the image configuration to deploy the necessary VMs to provide the requested SaaS. Thus, a SaaS request produces several resource request attending to the workload.

Market management policies define maximum available resources in each RP, including total computing power, as well as the maximum VCPU, memory and disk per VM. It is a market with trades between resources and credit. In addition, it is possible to define some opportunistic use of the resources which can be allocated for those running pods without credit left. When the overall power demand is larger than the power supply, then, it is required to define overflow policies for the running pods, between several options as follows:

- To increase the power supply with third party RP, for example using cloud bursting. Another option in hybrid clouds is to include in the federation the use of commercial providers, delegating to such IaaS providers the responsibility of providing as much power as needed. This is the ideal solution in terms of elasticity features, but it requires flexible budget to acquire public cloud resources on-demand and imposes additional financial issues.
- To enable priorities between running pods so that whether power demand is larger than supply, run-

ning pods with less priority will fall into larger response times. In this manner, higher priority running pods are less affected when no more computing power is available. This option can be applied simultaneously with the first option.

- A variant of the priority approach is the option of stopping VMs when demand is larger than supply. This feature can be configured for the running pods which have non elastic requisites. This does not resolve all the possible power requirements, but it can be applied to first and second option for a sensible use of the resources.

The second and third options are constraints to the elasticity which is one of the assets of cloud computing. Furthermore, it is necessary to consider additional stoppage policies to avoid resource blocking of some running pods with others of the same priority. Grateful stoppage avoids the loses of the current payload when stopping a VM to free the resource for other running pod.

The right part of the Fig 2 is the necessary accounting of the resource and credits usage. An example of such federated cloud accounting system could the one promoted by EGI Fedcloud (EGI-FedCloud, 2014) using APEL (APEL-EMI, 2014) system. APEL was originally developed for grid infrastructures, and currently, it has available plug-ins for OpenStack and OpenNebula, while more cloud managers are coming. APEL is considering the resource usage per VO basis, but a direct metrics on credit accounting. Credit accounting could be separated system, but will fall into inconsistencies between the two systems. For this reason double accounting is something to avoid, so it would be mandatory to integrate RP accounting per VO basis with credit model accounting which is per running pod basis.

4 A PROTOTYPE OF CLOUD GOVERNANCE WITH DIRAC

Cloud governance prototype is a modification of previous DIRAC cloud extension, including an implementation of the credit model described in Section 3. Only VCPU hours are considered for the prototype metric leaving aside other metrics like memory or IO usage. New configuration sections have been included in the DIRAC configuration service (CS) to support governance parameters. Core functionalities has been included in the VM Manager to attend credit status request of the VM Scheduler and credit updates from the VM Monitor of each VM.

Governance process starts defining CS parameters to associate a running pod with a particular group of users. A user is part of many groups corresponding to the running pods which is able to use. The prototype adds the running pod governance parameters:

- Maximum number of deployed VCPUs
- Running pod campaign start and end dates
- Running pod total VCPU hours of credit for the campaign.

Deployed VCPUs are the VM which have been submitted even if they have not reached a running status jet. They are corresponding to VMs which are in status of booting, contextualizing or running. So maximum running VMs can be lower than maximum deployed, or equal, but never greater. For governance purposes two new parameters are included at each IaaS configuration section:

- Maximum number of deployed VCPUs
- Maximum number of opportunistic deployed VCPUs

VM Scheduler has a submission logic based in previous features as well as in the credit model. For each active running pod, VM Scheduler checks Task Queue jobs and VM horizontal auto scaling policy. Whether TQ and VM horizontal auto scaling policy indicate the possibility of a new VM creation, further governance conditions have to be checked with the following algorithm:

```
If runningPod(deployed VCPUs < max VCPUs):
  If there is credit left:
    list = IaaS(deployed VCPUs < max VCPUs)
  else:
    list = IaaS(deployed VCPUs <
                  max oportunistic VCPUs)
  if list not null:
    candidate = random(list)
    submit(VM, candidate)
```

- Algorithm: Credit checking -

Algorithm checks first if the maximum number of deployed VCPUs for a running pod in a particular moment has been reached. It the limit has not been reached, then it checks if there is credit in VCPU hours for the running pod. If there is credit left an IaaS site candidate list is created with those sites which have not reached the maximum number of deployed VCPUs of each site. If there is no credit left, then the IaaS site candidate list is created with those sites which have not reached the maximum number of deployed VCPU for opportunistic purposes of each site. Finally, a random selection from the candidate list is chosen to submit a new VM.

In parallel to the VM Scheduler process there is a continuous credit update. Each VM is running a VM Monitoring agent, which is reporting periodical heart beats with statistical information to the VM Manager. For governance purposes, heart beat includes a parameter of the number of VCPUs of the VM. On the server side, the VM Manger sets instance history record in a database, calculates the gap between the last heart beat of the instance and the current one, and adds the VCPU hours to the used VCPU hours in the running pod database. The VCPU is wall-time since the resource is allocated even if not real CPU is used, same as commercial clouds scheme.

5 PROTOTYPE TEST

This section presents a test designed in order to evaluate elasticity key feature, including scale-up and down. It also evaluates the overall system response when some of the running pod is active but without credit left, so it has to run the payload using opportunistic resources. The test is not aiming a stage with a power demand larger than power supply. In this sense it is a test with no job response time constrains, the target is to validate the core features of the credit model for cloud governance, particularly about elasticity.

5.1 Test Set-up

Test set-up top down description defines two user groups: A and B. Each group has two running pod because it is uses different instance types, *m1.small* flavour with 1 VCPU, 2GB of memory and 20 GB of disk, and *small* instance type with 1VCPU, 1GB of memory and 10 GB of disk. The job submissions of a group can run in both instance type running pods, but these jobs can not match with the VMs submitted by the other running pod.

The RP of the test are part of the EGI Fedcloud. CESGA cloud with OpenNebula and BIFI cloud with OpenStack. Maximum number of VCPUs by each IaaS provider is 25 VCPUs, with *small* instances for CESGA and *m1.small* for BIFI. Both of them have a maximum of 5 VCPUs for opportunistic use.

Submission pattern is composed by 10 series of 100 jobs each. The jobs of a single series are submitted about the same moment using DIRAC parametric jobs. The sequence is to start to submit the 100 jobs series of a group, then the other group series, and repeating the period up to 10 series. Using this pattern it is possible to evaluate the adaptability of the resources for VMs which can be used by a particular group.

Job workload is running a fractal plotting software (*mandelbrot*), with high *CPU/IO* ratio. The binary is allocated in the VMs using the job input sandbox and the fractal plot is put in the output sandbox. A single job process has been previously tested in the clouds, with very similar execution time between different providers. Group B has been configured with a maximum VCPU hours to run 3 series of 100 jobs, then opportunistic use of resources will apply.

VM horizontal auto scale policy is configured for a compromise between VM efficiency and total wall time. The VM is configured with 5 minutes of margin time before halt. I.e. previously to stop a VM is necessary a margin time without any workload running. If more workload of the next series is matched within this halting margin time, then the VM is not stopped. Only jobs of the same group can match the payload of a VM. Series pattern time gaps are taken as reference a normal distribution with average in the half of the time of the larger job response time, obtained from a previous processing of 100 jobs from cold system. The idea is not to simulate real user behaviour, but to test the elastic scale up and down of the system, producing series which uses previous VMs of the same group, as well as new VMs, depending on the workload gap times and the available power for a group in a specific moment.

The contextualization is done by ssh with an agent is polling the VMs in *waiting for context* status. This method is needing synchronization between submission time and contextualization time, which it is a disadvantage compared with context methods integrated in the cloud managers, like *cloudinit* or *prolog and epilog*. In the other hand *ssh* contextualization is not requiring additional libraries in the images neither particular implementations of the cloud managers. With a public IP and a *sshd* running in the VM, the same context scripts can be launched independently of the IaaS site.

5.2 Test Result

This section presents different results of the test, starting from general plots then breaking down in more detailed metrics.

Fig. 3 shows the running VMs, those VMs which has been submitted and booted, then contextualized and finally declared running by the VM Monitor when Job Agent starts to match jobs. During the VM running, the VM Monitor is sending periodical heart beats. If a VM is not changing status or sending running heart beat in 30 minutes then it is declared stalled. Two VMs were stalled in the test, without reaching the running status the VM were in error sta-

Figure 3: Overall Running VMs.

tus in the BIFI OpenStack dashboard. Additionally, three VMs at CESGA an other three VMs at BIFI were booted without problem, but they were not contextualized because they reached the 30 minutes limit and declared stalled by DIRAC, when actually they were not stalled.

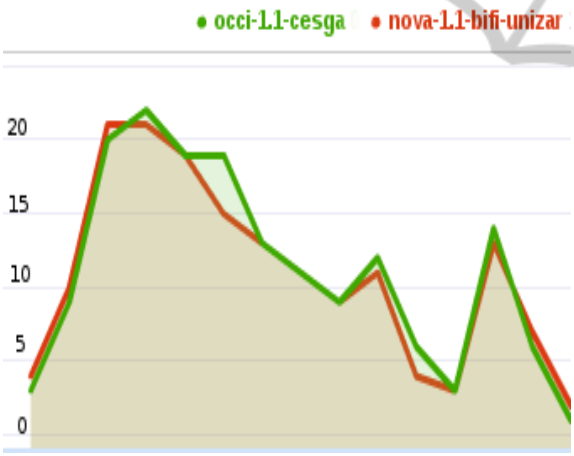

Figure 4: Running VMs by IaaS endpoint.

Peaks and valleys in Figs. 3, 4 and 5 are demonstrating the ability of DIRAC to scale up and down adapting to different workloads patterns in multi-community environment. Let see next plots for further details about scalability.

Running VMs by IaaS endpoint is shown in Fig. 4. The running VMs shape is about the same than overall running VMs of previous plot. CESGA and BIFI clouds, which are supported by different cloud managers, with the same DIRAC configuration are obtaining very close results in terms of running VMs. However, a workload analysis by endpoint shows that 63.5% of the workload has run in CESGA cloud, while BIFI has run 36.5%. This is quite significant in terms of power features. CESGA was providing for this test instance types with 1VCPU, 1GB of memory and 10GB of disk, while BIFI was giving instances of 1VCPU, 2GB of memory and 20GB of disk. With the same VCPU number per instance type, BIFI instances provide more memory and disk, but it is able to process about the half of the workload with near the same running hours of VMs. Thefore VCPU provided by CESGA have more power by near a factor two. It is clear that VCPU/h is not an appropriate metric for power, and therefore neither for credit.

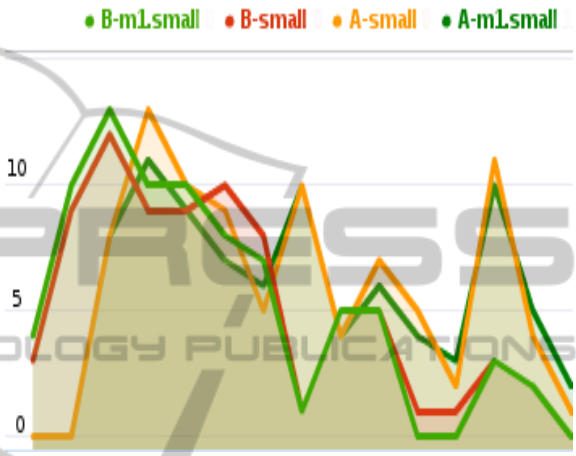

Figure 5: Running VMs by running pod.

Fig. 5 shows running VMs by running pod. As it was mentioned above in the test set-up, the jobs of group A can run in *A-small* or *A-m1.small*, and equivalent with group B. A group is divided in two running pods to evaluate instance types differences and therefore IaaS provider differences in multi-community stage. However, the response is equivalent in the level of running pods of group A, and also equivalent for group B running pods. This is demonstrating that high level governance policies are working transparently in multiple IaaS providers. At the same time, the scalability is independent between groups, following separated workload patterns.

Fig. 5 also shows opportunistic features. When there is no credit is left for group B, running VMs reach a plateau without trespassing the maximum number of opportunistic resources per IaaS endpoint.

Fig. 6 is a job histogram which provides some information. The first series of each group are in a cold system, therefore, they have the first jobs response time near 1000 seconds, while in the rest of series the first jobs are below 500 seconds. The opportunistic features are producing larger serie 4 and even larger in serie 5 of the group B.

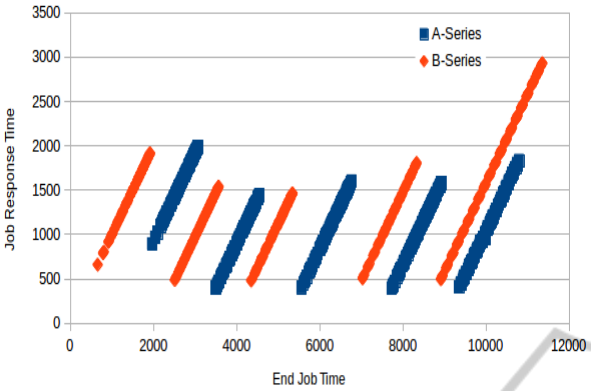

Figure 6: Jobs histogram.

6 CONCLUSIONS

The first conclusions are about elasticity. The prototype result shows a good scaling features for multi-community workloads which are using different VMs. The credit model has been demonstrated in the test, restricting the scale up to opportunistic resources when no more credit is left, but it is not affecting the scale down. Thus, the model is responding as expected by the larger job response time for series of the opportunistic cases.

The prototype has shown a secondary constrain about scalability, which could be resolved in future work. Complex scale up and down environments, with multi-community support, are producing high rates of creation and deletion. In this environment, a lack of synchronization has been detected between the submission rate and the *ssh* contextualization rate, which has to be tuned in production stages to avoid false positives in stalled machines. At the end of the day, booting time is constrained by cloud manager capabilities so there is no definitive solution from DIRAC side. Furthermore, error status VMs are also going to stalled as expected. Anyway, it is necessary an additional DIRAC agent trying to halt the VMs registered in stalled status, not only to clarify false positives, but also to free resources when possible.

Test results have shown that VCPU is not a proper credit metric, because it is not a power metric. For a governance based in credit model it is important to uses credits clearly related with the workloads. In this sense CPU power can be calculated by benchmarking the VMs power, as suggested in Rafhyc architecture (Méndez et al., 2013b). At the same time not only CPU power is consuming resources, memory footprint requisites, permanent and temporal disk requirements as well as network resources should also be considered in the credit metric of cloud computing.

There are some conclusions about governance policies. Notice that resource blocking stages have not been considered in the prototype. Future work shall consider to include a method to avoid resource blocking for the cases of power supply lower than demand. An tool for this method implementation can be the VM graceful stoppage option of DIRAC. Moreover, test has demonstrated that high level governance policies are working transparently in multiple IaaS providers. Further, each group workload is corresponding to separated elasticity pattern. Simultaneously, opportunistic policies are responding as expected in the credit model, not only terms of resource use limits, but also in job response times.

ACKNOWLEDGEMENTS

This work was supported by projects FPA2007-66437-C02-01/02 and FPA2010-21885-C02-01/02, assigned to UB. We are greatly in debt with EGI Fedcloud, in particular with BIFI and CESGA for providing the cloud resources for the test.

REFERENCES

Albor, V. F., Silva, J. J. S., Gómez-Folgar, F., López-Cacheiro, J., and Diaz, R. G. (2011). Dirac integration with cloudstack. In *Proceedings of 3rd IEEE International Conference on Cloud Computing Technology and Science (IEEE CloudCom 2011)*.

APEL-EMI (2014). http://www.eu-emi.eu.

Bunic, P., Aguado-Sanches, C., Blomer, J., and Harutynunyan, A. (2011). Cernvm: Minimal maintenance approach to the virtualization. *Journal of Physics Conference Serires*.

CloudStack (2014). http://www.cloud.com.

Cohen, S., Karagiannis, F., Courcoubetis, C., Iqbal, K., Andreozzi, S., and Heikkurinen, M. (2013). Computing einfrastructure cost estimation and analysis. pricing and business models. Technical Report Deliverable D2.3, e FISCAL : Financial Study for Sustainable Computing e Infrastructures.

EGI-FedCloud (2014). https://wiki.egi.eu/wiki/Fedcloud-tf:Blueprint.

Graciani Diaz, R., Casajus Ramo, A., Carmona Aguero, A., Fifield, T., and Sevior, M. (2011). Belle-dirac setup for using amazon elastic compute cloud. *Journal of Grid Computing*, 9:65–79. 10.1007/s10723-010-9175-7.

Jakob Blomer, T. F. (2010). A fully decentralized file system cache for the cernvm-fs. In *Proceedings of 19th International Conference*, pages 1–6.

Konstantinou, I., Floros, E., and Koziris, N. (2012). Public vs private cloud usage costs: the stratuslab case. In *Proceedings of the 2nd International WorkShop on Cloud Computing Platforms*.

Lerner, R. (2006). Amazon web services. *Linux journal*, 143:20.

Llorente, I. M., Montero, R. S., and Milojicic, D. (2011). Opennebula: A cloud management tool. *IEEE Internet Computing*, 15:14.

Méndez, V., Albor, V. F., Diaz, R. G., Ramo, A. C., a, T. F. P., Arévalo, G. M., and Silva, J. J. S. (2012). The integration of cloudstack and occi/opennebula with dirac. *Journal of Physics Conference Serires*.

Méndez, V., Ramo, A. C., Albor, V. F., and Diaz, R. G. (2013a). How to run scientific applications with dirac in federated hybrid clouds. In *ADVCOMP 2013 : The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences*, pages 73–78. IARIA. ISBN: 978-1-61208-290-5.

Méndez, V., Ramo, A. C., Albor, V. F., Diaz, R. G., and Arévalo, G. M. (2013b). Rafhyc: an architecture for constructing resilient services on federated hybrid clouds. *Journal of Grid Computing*, 11(4):753–770.

OpenStack (2014). http://openstack.org/.

Simon, A., Freire, E., Rosende, R., Diaz, I., Rey, P., Lopez-Cacheiro, J., and C.Fernandez (2012). Egi fedcloud task force. In *IBERGRID, 6th Iberian Grid Infrastructure Conference Proceedings*.

Tsaregorodtsev, A., Bargiotti, M., Brook, N., Casajus Ramo, A., Castellani, G., Charpentier, P., Cioffi, C., Closier, J., Graciani Diaz, R., Kuznetsov, G., Li, Y. Y., Nandakumar, R., Paterson, S., Santinelli, R., Smith, A. C., Miguelez, M. S., and Jimenez, S. G. (2008). Dirac: a community grid solution. *Journal of Physics: Conference Series*, 119(6):062048.

Ubeda, M., Méndez, V., andlippe Charpentier, F. S., and closier, J. (2014). Integration of cloud resources in the lhcb distributed computing. *Journal of Physics Conference Serires*.