CLOUD INTEGRATION WITHIN THE DIRAC INTERWARE

N.A. Balashov[2], R.I. Kuchumov[4], N.A. Kutovskiy[2], I.S. Pelevanyuk[2,3], V.N. Petrunin[4], A.Yu. Tsaregorodtsev[1,3]

[1]*Aix Marseille Univ, CNRS/IN2P3, CPPM, Marseille, France*

[2]*Joint Institute for Nuclear Research, 6 Joliot-Curie, Dubna, 141980, Russia*

[3] *Plekhanov Russian University of Economics, 36 Stremyanny per., Moscow, 117997, Russia*

[4] *Saint Petersburg State University, 7/9 Universitetskaya Emb, St Petersburg 199034, Russia*

E-mail: pelevanyuk@jinr.ru

Computing clouds are widely used by many organizations in science, business, and industry. They provide flexible access to various physical computing resources. In addition, computing clouds allow better resource utilization. Today many scientific organizations have their own private cloud used for both hosting services and performing computations. In many cases, private clouds are not 100% loaded and may be used as work nodes for distributed computations. If one connects clouds together, it will be possible to use them together for computational tasks. Therefore, the idea to integrate several private clouds appeared. A cloud bursting approach can be used for the integration of resources. However, in order to provide access to the united cloud for all participants, an extensive configuration will be required in all clouds. We studied the possibility to combine clouds by integrating them using a distributed workload management system – DIRAC Interware. Two approaches to spawning virtual machines were evaluated: the use of the OCCI interface and the native OpenNebula XML-RPC interface. They were tested, and both approaches allowed performing computing jobs in various clouds based on the OpenNebula software.

Keywords: Cloud, Grid, DIRAC

Nikita Balashov, Ruslan Kuchumov, Nikolay Kutovskiy, Igor Pelevanyuk, Vadim Petrunin, Andrei Tsaregorodtsev

1. Introduction

A computing cloud is a flexible tool for a wide range of tasks: services hosting, DevOps support, and intensive computational tasks are just some of them. It also increases the efficiency of hardware resource utilization by sharing them among different groups and tasks. The number of different computing clouds that a certain user has access to is growing, which raises a question: how to use several clouds together for either solving big tasks or distributing the load across different clouds more efficiently.

A cloud infrastructure deployed at the Joint Institute for Nuclear Research (JINR) was created in 2013 to manage LIT IT services and servers more efficiently using modern technologies, increase the efficiency of hardware utilization and service reliability, simplify access to application software and optimize the use of proprietary software as well as provide a modern computing facility for JINR users [1]. JINR helped organizations of its Member States to deploy their own local clouds, most of which are based on OpenNebula.

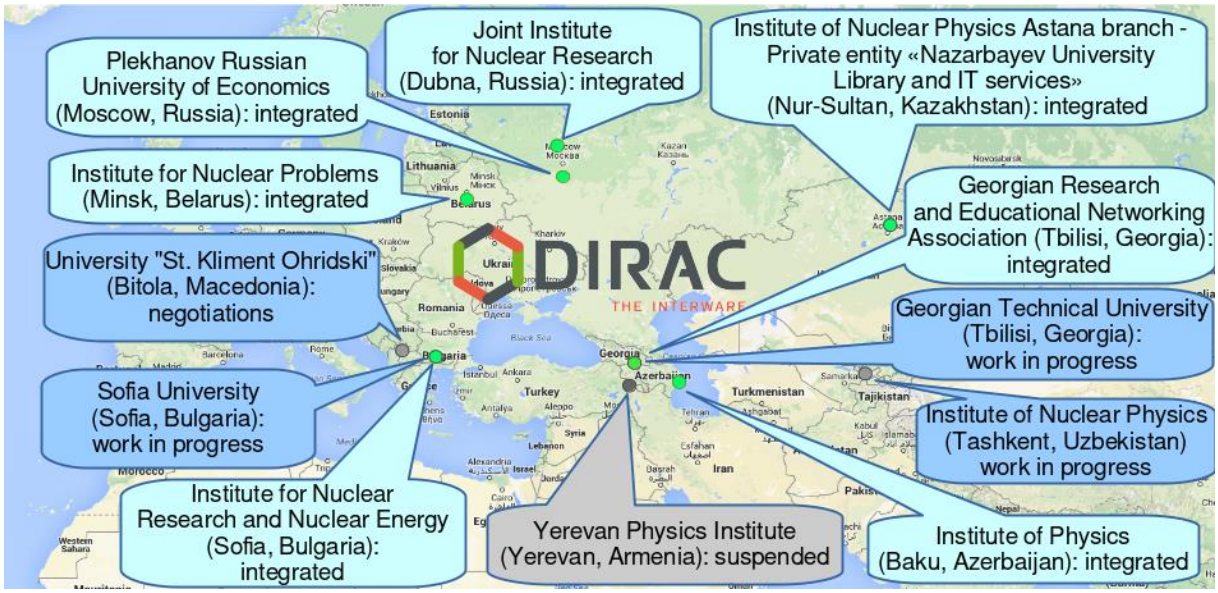

Figure 1. List of organizations possessing clouds

The first approach to cloud integration was based on the built-in mechanism of the OpenNebula cloud platform, called "cloud bursting" [2]. It works well for a small number of clouds joined together. However, it sufficiently increases the complexity of such infrastructure maintenance with a growing number of participating clouds.

2. Integration of clouds using DIRAC

DIRAC is a software framework for distributed computing providing a complete solution to one or several user communities requiring access to distributed resources. The DIRAC software offers a common interface to a number of heterogeneous computing and storage resources [3]. From the user's perspective, DIRAC is a system that accepts their computing jobs and uploads results to storage.

DIRAC uses a pilot mechanism to run jobs on heterogeneous resources. The general idea of the pilot is the following: jobs do not run on computing resources directly but through a special program called "Pilot". DIRAC can send Pilot jobs to different computing resources, so the integration of a computing resource means an ability to send the Pilot to a resource and successfully run it there. Once the Pilot is running, it provides its parameters to the DIRAC queue and requests a job that will match the resource. Therefore, if there are no jobs available, it will finish the execution and free the resource. However, if there is a job for the resource, it will be submitted to the Pilot. The job will be started

inside as a child process of the Pilot. It means that the Pilot can track the job execution and report such information. After completing the job, the Pilot may request another job or finish the execution if it has already occupied resources for a long time or there are no jobs left.

Using pilots may look like a big overhead, but they are very useful. For end-users, pilots eliminate all errors related to the inability of the resource to run a job, such as a local disk occupancy, some network connection errors, or wrong credentials. All these problems are surely serious and generally should be fixed by administrators, but they should not disturb users. Pilots also provide basic utilities for data operations, which means that the job itself may download or upload data from or to the storage.

With this approach, one may say that resources are integrated if users' computing needs are satisfied without worrying where our job is running exactly, or, in the context of this article, in which cloud it was executed. In terms of running a job in the cloud, the most difficult part is to launch a Pilot on these types of resources. In our case, it means that the dedicated virtual machine (VM) should be spawned in order to start the Pilot. However, there is no reliable and simple way to initiate the Pilot on the arbitrary VM, so the Pilot should be initiated automatically after the VM boot.

The virtual machine should also provide the necessary software for jobs. There are several ways to do this. The software can be preinstalled on the VM image. It will require changing the VM image if the software has changed. Another option is to download software from the storage element and then configure it inside the job. This may work if the effort required to keep the software up to date is not substantial. The third option is to use CernVM Filesystem (CVMFS) for software distribution across cloud VMs. This option is preferred since it allows transparent software updates and local software caching, which is important if the size of the software is large.

It leads us to the fact that we need to prepare VM images to run Pilots and jobs on them.

3. Image preparation

To launch the Pilot on cloud VMs, a special disk image was created: it has the CentOS 7 operating system installed with the cloud-init app set up. The image can be used with almost all major cloud platforms that implement the Infrastructure-as-a-Service model, which is achieved mainly by the ability of the cloud-init to handle all different contextualization mechanisms of modern cloud platforms.

The pre-installed CernVM-FS client simplifies and optimizes software delivery to the job processing nodes: this makes it possible to access remote software repositories containing all end-user applications. Since CernVM-FS is a POSIX-compatible read-only file system, user jobs do not need to be set up in any special way to interact with it. Although CernVM-FS is a recommended software distribution solution, it is optional, and other methods can be used to transfer software.

The image needs to be manually uploaded by system administrators to all clouds in the integrated infrastructure and then registered in the DIRAC configuration along with additional VM parameters specific to the cloud. Thus, a single VM image can be uniformly used in clouds built on different platforms.

4. OCCI approach

The Open Cloud Computing Interface (OCCI) [4] is a set of open community-lead specifications developed to create a remote management API for IaaS model-based services that enable the development of interoperable tools for common tasks including deployment, autonomic scaling, and monitoring.

With the OCCI approach, it is important to run special services that will receive OCCI commands and translate them to OpenNebula. It may lead to additional administrating efforts and create additional points of failure.

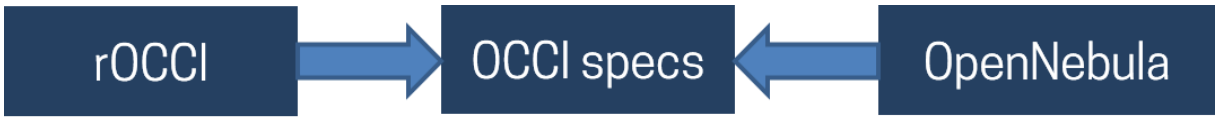

Figure 2. Dependencies between the cloud, the OCCI specification, and the rOCCI tool

Originally DIRAC supported the approach of creating new virtual machines using the rOCCI (ruby implementation of OCCI) command-line tool. It means that in order to spawn a virtual machine it was required to form the right rOCCI command and execute it in the shell. Therefore, the rOCCI tool is an additional point of failure. The main issue with this approach is that there are two points where errors may happen: the OpenNebula realization of the OCCI specification, the rOCCI realization of OCCI and the OCCI specification itself.

That is why we decided to create a module to use the OCCI interface directly by DIRAC. These types of modules are called Endpoints in DIRAC and in our case the OCCI Endpoint. To create an Endpoint the following methods should be available: create a virtual machine, delete a virtual machine. These are basic methods to start and stop virtual machines. There may be additional methods available to improve usability, monitoring and debugging: get a virtual machine status, assign a public IP address to a virtual machine, detach a public IP address. They are not mandatory and the cloud can run jobs even without them.

The OCCI endpoint was developed and tested. The possibility of running jobs in different clouds was demonstrated. With this module, the dependency on the rOCCI command is eliminated. Still, administrators of clouds have to support OCCI servers. We decided to create an Endpoint directly for OpenNebula clouds.

5. OpenNebula XML-RPC approach

After the experience with the OCCI Endpoint creation, the development of the XML-RPC Endpoint was rather straightforward. Basic methods required for the VM creation, VM deletion and getting the VM status were implemented using the OpenNebula XML-RPC interface. After that, it became possible to send jobs to the integrated cloud infrastructure. This approach eliminated the need to use OCCI services and rOCCI freeing administrators from additional work to support it.

One clear use case of the integrated cloud infrastructure is running Monte Carlo simulation jobs for experiments in High Energy Physics. Tests performed in clouds showed that jobs were successfully executed, and result data were uploaded to storage elements. This makes clouds a valuable resource for some tasks in scientific computing.

Source codes are included in the master version of the DIRAC Interware and now available to other user communities that use DIRAC [5].

6. Conclusion

Computing clouds are a valuable resource. Their flexibility allows using them in various use cases. The DIRAC Interware supported cloud resources but only through intermediate interfaces. That approach was not perfect, as all these interfaces (like OCCI) required support from both DIRAC and resource sides. The first tests showed that clouds might have their own issues, which did not appear on traditional grid resources. Most of the problems are related to overly strict firewall rules and limited network bandwidth. All these problems can be solved on working clouds with the help of local administrators.

The implementation of the OpenNebula XML-RPC Endpoint eliminated a large amount of additional work related to OCCI or rOCCI support without any loss in functionality. It simplified the integration and support of new OpenNebula cloud resources in DIRAC making it a useful resource for scientific computing at JINR and its Member States. The developed module is now publicly available, and other user communities may benefit from it.

References

[1] A.V. Baranov, N.A. Balashov, A. N. Makhalkin, Ye. M. Mazhitova, N.A. Kutovskiy, R.N. Semenov New features of the JINR cloud // The 8th International Conference «Distributed Computing and Grid-technologies in Science and Education (GRID'2018)», CEUR Workshop Proceedings, ISSN:1613-0073, vol. 2267, 2018, P. 257-261.

[2] Baranov A.V., Korenkov V.V., Yurchenko V.V., Balashov N.A., Kutovskiy N.A., Semenov R.N., Svistunov S.Y. Approaches to cloud infrastructures integration // Computer Research and Modeling, 2016, vol. 8, no. 3, pp. 583-590

[3] Gergel V., Korenkov V., Pelevanyuk I., Sapunov M., Tsaregorodtsev A., Zrelov P. (2017) Hybrid Distributed Computing Service Based on the DIRAC Interware // DAMDID/RCDL 2016. Communications in Computer and Information Science, vol 706. Springer, Cham. DOI: 10.1007/978-3-319-57135-5_8

[4] Michael Behrens, Mark Carlson, Andy Edmonds, Sam Johnston, Gary Mazzafero, Thijs Metsch, Ralf Nyrén, Alexander Papaspyrou, Alexis Richardson, Shlomo Swidler. (2011). Open Cloud Computing Interface — Infrastructure.

[5] Federico Stagni, Andrei Tsaregorodtsev, ubeda, Philippe Charpentier, Krzysztof Daniel Ciba, Zoltan Mathe, … Luisa Arrabito. (2018, October 8). DIRACGrid/DIRAC: v6r20p15 (Version v6r20p15). Zenodo. http://doi.org/10.5281/zenodo.1451647.