Running Big Data on the EGI Federated Cloud

J. López Cacheiro[1], A. Simón[1], J. Villasuso[1], E. Freire[1], R. Rosende[1], I. Díaz[1], A. Feijóo[1], P. Rey[1], C. Fernández[1] and B. Parak[2]

[1] Fundación Centro de Supercomputación de Galicia, Spain
`grid-admin@cesga.es`
[2] CESNET, Czech Republic
`boris.parak@cesnet.cz`

Abstract. Big Data is an emerging technology that could benefit from existing developments in cloud technologies. In this paper we evaluate the suitability of FedCloud–the federated cloud infrastructure developed inside EGI–to perform Big Data analytics using Hadoop. Due to the size of the tests performed, they represent also a real benchmark of the performance of FedCloud under heavy usage conditions, providing the first results about the scalability of the system.

1 Introduction

Nowadays, the possibility to run Big Data calculations over federated clouds is a topic that is attracting the interest of the research community. Federation of resources poses serious challenges both from the cloud and Big Data perspectives. The aim of this paper is to evaluate the suitability of the FedCloud federated cloud infrastructure that it is being developed inside EGI to run typical Big Data analytics using Hadoop.

Big Data is one of the buzzwords that sounds everywhere, the Big Data community is working to extend the scalability of traditional databases (RDMS) using new technologies based on a share-nothing architecture. In this arena, one of the most well-known solutions is Hadoop [1], an open-source framework that implements the MapReduce computational paradigm on top of a parallel HDFS filesystem.

The EGI Federated Cloud Task Force [2] started its activity in September 2011 with the aim of creating a federated cloud testbed–named FedCloud–to evaluate the utilization of virtualized resources inside the EGI production infrastructure. Even if the development is still in progress, the testbed is already available for experimentation by the different user communities. A more detailed description of the EGI FedCloud infrastructure is given in [3].

The aim of our work is to do an initial assessment of the suitability of FedCloud to run Hadoop through a series of real-world benchmarks. The paper is organized as follows: in Section 2 we describe the methodology used to deploy Hadoop inside the FedCloud federated cloud infrastructure; in Section 3 we show the results obtained after running several deployment and execution benchmarks based on the analysis of two common data sources: Encyclopædia Britannica and Wikipedia; finally in Section 4 we present the main conclusions of this work, summarizing the main problems encountered as well as outlining the future steps that could be taken to provide Hadoop as a on-demand service in FedCloud.

2 Methodology

In this Section we describe the methodology that we have followed to deploy a Hadoop cluster instance inside the FedCloud federated cloud infrastructure. Basically, three steps were involved in the process:

1. *Cluster startup*: obtaining the resources necessary to form the cluster from each of the sites in FedCloud
2. *Hadoop configuration*: configuring the Hadoop cluster
3. *Benchmark execution*: loading the data sets and running various MapReduce jobs

Each of these steps is described in more detail in the following subsections.

2.1 Cluster startup

A general overview of the overall cluster startup process is given in Figure [**?**]. In order to simplify the later configuration, we created a customized virtual machine (VM) image template that includes all the software necessary to run Hadoop. This image is based on Scientific Linux 6.4 and contains the following customizations:

– *Firewall:* iptables is configured to allow inter-cluster communication.
– *SSH:* enable root access through ssh public key authentication (*authorized_keys*).
– *Modules:* we make use of the Modules package to simplify the deployment of the software needed to run Hadoop.
– *Java:* we use Oracle Java JDK 1.6.0_33 because it offers better performance than the OpenJDK version. It is also configured through Modules to allow simple migration to other Java versions.
– *Hadoop:* both Hadoop-1.0.3 and Hadoop-1.0.4 are included. The configuration using Modules makes easy to include additional versions and decide the one we want to use when starting the Hadoop cluster.

This customized image was registered in the EGI Marketplace [4] which provides a common place where where we can can store images. Unfortunately, the Marketplace does not store the VM images into a common repository, it just shows which resource providers (RPs) have the VM image available at their endpoint. This means that each RP has to manually download the image to his local site in order to make it available at his endpoint. Currently there is no automated way to perform this step so the distribution of the VM template to all the sites is a time consuming effort that involves contacting the FedCloud administrators of each each site. It should be noted that a vanilla SL6 or Debian image from the Marketplace could have also used instead of using our customized image, which main purpose was just to allow us to pre-install software.

The next step is to instantiate the virtual machines that will form the Hadoop cluster, for this purpose we make use of the new implementation of the OCCI 1.1 specification under rOCCI [5] as well as our VOMS proxy credentials. The new rOCCI implementation provides several advantages like the fact that we are able to avoid the previous issue that all user certificates request are mapped to a single ONE_USER account. Additionally, the usage of VOMS over plain X509 auth greatly simplifies the access to the different sites that form FedCloud without the need to request each site to give access to our X509 certificate's DN.

Unfortunately FedCloud does not count with a workload management system, so each VM creation request must be sent to the appropriate endpoint. This means that we have to take care ourselves of the partitioning of the cluster between the different FedCloud sites.

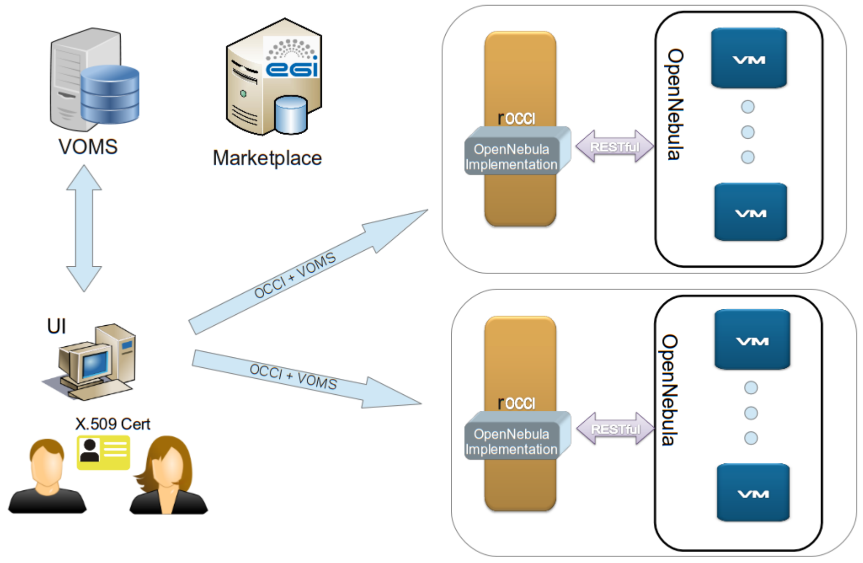

Fig. 1. Cluster startup process using rOCCI and VOMS from a single UI.

About the resources used by each of the instances, in order to increase the concurrent number of VMs, we used small size VM instances with 1GB of RAM and 1 CPU.

2.2 Hadoop configuration

After having the cluster running, the next step is to prepare the nodes to run Hadoop. As it has been explained in 2.1, the software necessary is already pre-deployed in the nodes so we only have to configure Hadoop.

Our Hadoop cluster will consist of one *master* node and a variable number of *slave* nodes depending on the size of the cluster. For example in the case of a 101 node cluster, one node will be configured as master and the remaining 100 as slaves. The master node will run the *namenode*, *secondarynamenode* and *jobtracker* services and each of the slaves will run just one *tasktracker* and one *datanode* services due to the fact that we are using small VM instances with just 1GB of RAM and 1 CPU we decide to run just one tasktracker and one datanode per slave.

The tuned configuration parameters are given in Table 1. The column type identifies the configuration file where the parameter is set, the complete configuration files can be found at [6].

These parameters have been configured to adapt the Hadoop cluster to resources available–1GB of RAM and 1 CPU per node–. The *dfs.replication* parameter indicates how many copies we want to store of each block, in this case we use three replicas that will allow different nodes to execute the same mapreduce tasks–speculative execution–mitigating the performance degradation that could be experienced in a heterogeneous cluster with VMs running under hypervisors with different load conditions. The *mapred.tasktracker.map.tasks.maximum* and *reduce.tasks.maximum* are set to 1 because we only have 1 CPU available to run both services and the HADOOP_HEAPSIZE is set to 512MB to allow both running at the same time without exhausting the node's memory. The number of reduce tasks, *mapred.reduce.tasks*, is set to 95% of the number of slaves in the Hadoop cluster.

As we will see in Section 2.3 our Hadoop installation will cope with two very different use cases that influence our decission about the *dfs.block.size* values to use. In the case of the Encyclopædia Britannica's use case the block size is set to 1MB and in the Wikipedia's use case, that uses a much larger data set, to 64MB. Under this conditions both use cases generate less than 700 total mapreduce tasks which can be handled well by our hadoop master node which has only 1GB of RAM and 1 CPU.

2.3 Benchmark execution

In order to evaluate the suitability of FedCloud to perform Big Data analytics using Hadoop we selected two use cases which are quite representative of a broad range of jobs that could be performed in a Hadoop cluster. The benchmarks were run both in a federated cluster deployment and in a one-site-only cluster deployment, allowing us to compare the results and analyse the expected degradation when using remotely distributed resources.

The first use case is based in the Encyclopædia Britannica[7] data set–the 1911 version that is already in the public domain–. This represents a rather small data set of just 176MB, so that this use case can be considered as a small scale

Table 1. Tuned Hadoop configuration for small VM instances. It includes both the HDFS and MapReduce parameters used.

Parameter	Type	Value
$fs.inmemory.size.mb$	core-site	200MB
$io.file.buffer.size$	core-site	128KB
$mapreduce.task.io.sort.factor$	core-site	100
$mapreduce.task.io.sort.mb$	core-site	100
$mapred.tasktracker.map.tasks.maximum$	mapred-site	1
$mapred.tasktracker.reduce.tasks.maximum$	mapred-site	1
$mapred.reduce.tasks$	mapred-site	$0.95\times$ *num. slaves*
$dfs.datanode.du.reserved$	hdfs-site	1GB
$dfs.block.size$	hdfs-site	1MB / 64MB
$dfs.replication$	hdfs-site	3
HADOOP_HEAPSIZE	hadoop-env	512MB

benchmark that will serve to evaluate the expected degradation in performance due to fact that we will be using federated resources located at different sites. The data set is downloaded directly from the master node and later on it is distributed from there to the slaves, what is called a *put* operation in Hadoop argon. This use case has a great overload due to the creation of rather small mapreduce tasks that will stress the communication system because it uses a rather small block size of just 1MB, so each map operation has little to process.

The second use case is based in the Wikipedia[8] data set. This represents a much larger data set comprising 41GB, so it can be considered a more realistic benchmark of a Big Data application. Due to the size of the data set the files could not be downloaded directly in the master node so the *put* operation is done directly from our local Hadoop client located at CESGA. This use case will incur in a lower task creation penalization because it uses a larger block size of 64MB, so each map task will take much longer than the time needed for its creation.

The benchmarks are run for different cluster sizes that range from 10 to 101. In all the benchmarks we measure two parameters:

1. *The put time:* the time to load the files into the Hadoop HDFS filesystem
2. *The wordcount time:* the time to perform a simple wordcount mapreduce job that computes the number of occurrences of each word over the complete data set.

All the measurements where repeated at least 5 times in order to evaluate the variability of the results, the only exception were the *put* times for the Wikipedia that due to its duration could only be run once.

3 Results

Following the methodology described in the previous Section we deployed Hadoop inside FedCloud and executed the benchmarks mentioned in subsection 2.3. In this

Section we present the benchmark results obtained for the two selected use cases: Encyclopædia Britannica and Wikipedia.

The first results to consider are those related to the cluster startup process described in Section 2.1. In this case we are dealing with the instantiation of a number of VMs that ranges from 10 to 101 so it can also serve as a stress test of the FedCloud rOCCI framework as well as the underlying cloud management platform–currently only OpenNebula is supported by rOCCI.

After some initial VM deployment tests we discovered that most of the sites involved in FedCloud were misconfigured or did not support rOCCI, and only two sites were providing a working rOCCI endpoint: CESGA and CESNET. So we deployed our federated infrastructure using these two sites.

The startup times for each cluster deployment varied and the most representative ones are those obtained for the larger deployment, the 101 node cluster. In this case, the time needed by the rOCCI client to return all the resource endpoints corresponding to each VM ranged from 71 to 86 minutes. This is just the time to get the identifier corresponding to each VM instance, actually to have all the VMs running took even longer, around 80 minutes more. So the total cluster startup time ranged from 2.5 to 3 hours. But this was not the main issue encountered, the main problem was that some of the rOCCI requests failed with an *execution expired* error and some of the VMs did not start–in the worst case 21 out of 101 failed. We started to see this type of errors as soon as we tried to deploy, in a sequential way, more than 20 VMs at CESGA. In CESNET we could not do this analysis because there were only 10 VM slots available due to a restriction in the amount of public IPs that they had available.

Trying to understand the source of this issue we tracked the performance of the OpenNebula frontend during the cluster startup. As it can be seen in Figure 2 the frontend experiences a high load several minutes after the start of the cluster deployment, this causes it to respond slower, probably producing the rOCCI execution expiration errors mentioned above. Looking at the processes in the OpenNebula frontend, we could see that this increase in the load is mainly due to the scp processes launched to copy the image template to the OpenNebula nodes, more than 20 simultaneous scp processes seem to affect considerably the system performance, reducing its response times considerably.

Another problem we encountered was that FedCloud does not count with a workload management system, so each VM creation request must be sent to the appropriate endpoint. This means that we have to take care in advance of the partitioning of the cluster between the different FedCloud sites. In our case we choose to run 10 VM at CESNET–the maximum possible at the time of the benchmarks–and the rest of them at CESGA–ranging from 10 to 91.

Once the cluster is deployed we proceed to configure Hadoop. In this case the only problem encountered was due to the global firewall rules at each site that blocked Hadoop communications. Once this issue was addressed the Hadoop cluster started flawlessly. The time involved to configure and start the Hadoop cluster was really short, taking less than 1 minute in the larger setup with 101 nodes, so it is almost irrelevant in the total cluster startup time when considering the almost 3 hours needed for VM instantiation.

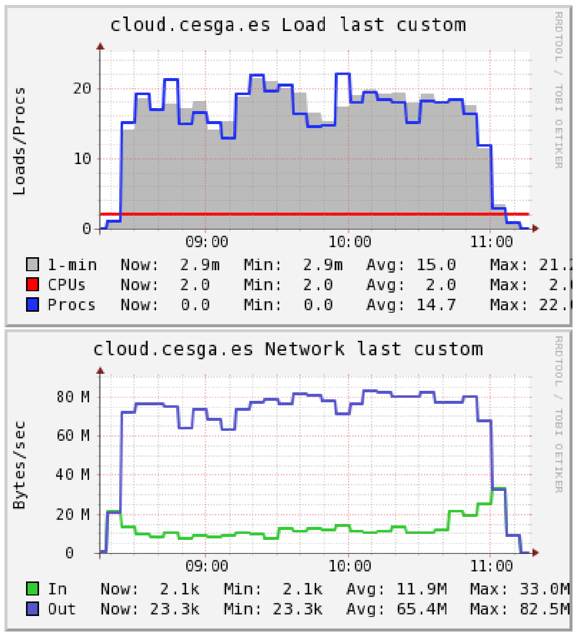

Fig. 2. OpenNebula frontend load and network usage graphs taken from Ganglia during the deployment of the 101 nodes cluster.

In Table 2 we show the results obtained for the Encyclopædia Britannica use case. All measurements are repeated 5 times and the average and standard deviation are displayed, allowing to have an estimation of the variability involved in the measurement. As it can be seen even if the block size used in this case is very small (1MB), the wordcount map reduce job scales really well and the overhead introduced by the federated cluster is small, being in general in order of 10-15%, and 28% in the worst case that corresponds to the 101 cluster. This is mainly due to the large overhead that mapreduce task creation and distribution imposes in this use case, especially in the larger cluster sizes that involve more reduce tasks–we are using a number of reduce tasks equal to the 95% of the total number of slaves.

As expected the put times are much lower in one-site-only scenario because in this case the connection between all the VMs is much faster because all of them are generally located in the same data centre.

In Table 3 we show the results obtained for the Wikipedia use case. It should be noted that due to the high duration of the transfers, we did not repeat the put measurements 5 times, and only the wordcount measurements were repeated. Also due to the longer duration of the benchmarks we only used the two larger cluster sizes: 51 and 101.

In this case the wordcount mapreduce results are quite surprising because the federated cluster instance is able to outperform the one-site-only instance in both scenarios. This is attributed mainly to the faster execution of the benchmark in CESNET nodes compared to CESGA nodes–CESNET VMs use Xen paravirtual-

Table 2. Benchmark times obtained for the Encyclopædia Britannica use case. Both the average time and standard deviation are reported. Federated times where measured in a federated cluster that included resources both from CESGA and CESNET; all one-site-only times where obtained in a one site deployment at CESGA except the 10 node cluster indicated in the second row that run at CESNET.

	Federated		One-site-only	
Cluster size	put (s)	wordcount (s)	put (s)	wordcount (s)
10 (CESGA)			47 ± 2	169 ± 1
10 (CESNET)			9.5 ± 0.2	160 ± 1
21	235 ± 12	108 ± 2	37 ± 1	97 ± 1
31	189 ± 4	90 ± 2	36 ± 2	78 ± 2
41	190 ± 11	81 ± 4	33 ± 2	71 ± 3
51	133 ± 7	73 ± 3	33 ± 1	66 ± 2
101	94 ± 7	67 ± 22	33 ± 4	52 ± 5

ization and a faster CPU whereas CESGA VMs use KVM and QEMU–. In this case the relative overhead due to mapreduce task creation is much lower than in the previous use case because we are using a much larger block size of 64MB.

The put times are much larger in the federated cluster due to the fact that the amount of data to transfer is much larger–HDFS stores three copies of the Wikipedia, 42GB each, so that the available bandwidth between the VMs and the UI located at CESGA plays a key role, being much lower in the case of CESNET VMs.

Table 3. Benchmark times obtained for the Wikipedia use case. Both the average time and standard deviation are reported except for the put times that where only measured once for each deployment. Federated times where measured in a federated cluster that included resources both from CESGA and CESNET; all one-site-only times where obtained in a one site deployment at CESGA.

	Federated		One-site-only	
Cluster size	put (s)	wordcount (s)	put (s)	wordcount (s)
51	19190	1001 ± 39	6705	1347 ± 117
101	13208	705 ± 14	5665	725 ± 18

4 Conclusions and future work

The results presented in this paper show the suitability of a federated cloud infrastructure like FedCloud to run Big Data analytics, especially if the data set is already pre-deployed in the Hadoop HDFS filesystem. In some cases we see that the federated cluster is even able to outperform the local one if the remote nodes are faster than the ones available locally.

We have shown that small VM instances are good enough to run Hadoop assuming you do an appropriate tuning of the configuration, of course using larger instances would simplify the configuration and it could improve the performance. In our benchmarks the *tasktrackers* run out of memory when running the Wikipedia's use case if we used a small number of reduce tasks, additionally when using a small *dfs.block.size* it was the *jobtracker* the service that run out of memory because it was not able to cope with the large amount of tasks.

The benchmarks performed allowed also to test the FedCloud infrastructure and the new rOCCI framework. There were some disappointing results about cluster startup times that were in the order of 3 hours for the 101 node cluster, additionally failures started to appear when trying to start more than 20 VMs. Even if we found that these failures should be mainly attributed to limitations in the RP cloud management backend, improvements in this area would be needed to create a really scalable federated infrastructure.

There are some other aspects of the FedCloud infrastructure that cloud be improved like the fact that there is no automated way in the EGI Marketplace to distribute of the image template to all the sites. There is also a lack of a workload management system to automatically distribute the VM instances between the available sites. A more strict testing of the sites would also help because at the time of running the benchmarks only two sites had operational rOCCI interfaces. A mechanism to query the resources available at a given site would be also useful. Another aspect to improve is related to network access between the nodes, the global firewalls of the sites can impede certain communications and then hinder the deployment of the Hadoop service.

About the future work, we will analyse the possibility of setting up a Hadoop service on top of FedCloud in order to offer Hadoop clusters on-demand through a simple web frontend where the user could select some basic parameters. This would allow other users to easily run their Hadoop applications in FedCloud.

The set of scripts created to deploy and execute the Hadoop service have been uploaded to github[6] so they can be used by other users interested in running Hadoop in FedCloud.

Acknowledgements

This work is partially funded by the EGI-InSPIRE (European Grid Initiative: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe) is a project co-funded by the European Commission (contract number INFSO-RI-261323) as an Integrated Infrastructure Initiative within the 7th Framework

Programme. EGI-InSPIRE began in May 2010 and will run for 4 years. Full information is available at: `http://www.egi.eu/`.

References

[1] Hadoop: Apache hadoop. `http://hadoop.apache.org/` (2013)
[2] FedCloud: EGI Federated Cloud. `https://wiki.egi.eu/wiki/Fedcloud-tf:FederatedCloudsTaskForce` (2013)
[3] Simón, A., Freire, E., Rosende, R., Díaz, I., Feijóo, A., Rey, P., López-Cacheiro, J., Fernández, C.: Egi fedcloud task force. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. (2012) –
[4] EGI: EGI Marketplace. `http://marketplace.egi.eu` (2013)
[5] EGI: rOCCI API. `https://github.com/gwdg/rOCCI` (2013)
[6] EGI: Hadoop on fedcloud deployment tools. `https://github.com/grid-admin/hadoop` (2013)
[7] Britannica: Encyclopædia Britannica. `http://www.britannica.com` (2013)
[8] Wikipedia: Wikipedia. `http://www.wikipedia.org` (2013)