Cloud Federation to Elastically Increase MapReduce Processing Resources

Alfonso Panarello, Maria Fazio, Antonio Celesti,
Antonio Puliafito, and Massimo Villari

DICIEAMA, University of Messina, Contrada di Dio, S. Agata, 98166 Messina, Italy
apanarello{mfazio,acelesti,apuliafito,mvillari}@unime.it

Abstract. MapReduce is a programming model that allows users the parallel processing of large data sets into a cluster. One of its major implementation is the Apache Hadoop framework that couples both big data storage and processing features. In this paper, we aim to make Hadoop Cloud-like and more resilient adding a further level of parallelization by means of cooperation of federated Clouds. Such an approach allows Cloud providers to elastically scale up/down the system used for parallel job processing. More specifically, we present a system prototype integrating the Hadoop framework and CLEVER, a Message Oriented Middleware supporting federated Cloud environments. In addition, in order to minimize overhead of data transmission among federated Clouds, we considered a shared memory system based on the Amazon S3 Cloud Storage Provider.Experimental results highlight the major factors involved for job deployment in a federated Cloud environment.

Keywords: Cloud Computing, Federation, Big data, MapReduce, Hadoop.

1 Introduction

MapReduce is a programming model for the parallel processing of large data sets. Hadoop MapReduce is one of the major implementation of the MapReduce paradigm developed and maintained by the Apache Hadoop project, that works in tandem with the parallel Hadoop File System (HDFS). Parallelization capabilities of a system strongly depends on available resources into the cluster. To fulfill several requests from many different users, an elastic approach for resource management is required. Cloud computing, offers such a feature. By means of virtualization resources can elastycally scale up/down. However for each Cloud Provider (CP) the number of available virtual resources depends on its own physical assets. In order to overcame such a limit, CPs can rent Virtual Machines (VMs) from big commercial provider or they can establish a federation relationship. The latter approch allows small-medium provider to cooperate in order borrow/lend resources according to particual agreements. In this manner, Cloud federation also offers to small/medium CPs new business opportunities, guaranteeing high flexibility in service provisioning in a trasparent way for end users. We chose to use CLEVER because, although it arises as middleware for the management of IAAS, it has been designed looking to the future and keeping an eye to the federation issues [1]. In fact, all communications, both inter-domain that intra-domain, use the technology XMPP, which in our opinion is a powerful solution to manage and to support the Cluod federation. So, our work aims to provide a Platform as a Service (PaaS) for a MapReduce processing of big data in a federated Cloud scenario. In particular, the solution we propose integrates the Hadoop functionalities into the the above mentioned CLEVER. Whenever a client submits a job to a Cloud, it, which may be not able to meet the client's request for computational tasks, processes the job exploiting resources distributed across different administrative domains. Each CP offers its processing resources according to the policies of the federation agreements and the provider that receives a commitment from the client manages the available pool of resources in the federation till the job processing ends. Our work, therefore, by exploiting the federated system potentialities, aims to add another parallelization layer to Hadoop Framework, thus making it elastic, scalable and cloud-like.

The rest of the paper is organized as follows. In Section 2, we provide a brief overview of current works on the topics dealt in the paper. Section 3 presents the proposed distributed processing service and one of many possible use cases. In Section 4, we introduce the technologies adopted in this work to arrange a real federated environment, i.e., Hadoop, CLEVER which is a Message Oriented Middleware (MOM) exploiting the Extensible Messaging and Presence Protocol (XMPP) technologies to handle the communication among the different administrative domains. Architectural details on how to integrate Hadoop and CLEVER are discussed in Section 4.3. In Section 5, we present experimental results highlighting the major factors involved for job deployment in a federated Cloud environment arranged by means of Hadoop and CLEVER. Section 6 concludes the paper.

2 Related Works

In the near future, the heavy penetration of sensing devices into Internet applications will cause the explosion of the amount of data to be stored and processed. This problem, well known as Big Data issue, is becoming the new buzzword in ICT world, involving both IoT and Cloud Computing, [2] technologies. Cloud Computing is already a consolidated technology useful for spreading massive computations on heterogeneous environments. In this perspective Cloud is becoming even more the basis for Big Data computation needs. At the Infrastructure as a Service (IaaS) level, Big Data can leverage the Computation capabilities of Clouds where the computation relying onVMs. Such an example is given in [3], where Hadoop is installed into VMs exploiting the Public Cloud as Amazon EC2. Here the authors re-modeled the resource provisioning of the VMs in public cloud platforms for big data applications. In particular the authors relied only on modifying the configuration of two types of EC2 VM instances that is Small instance and Extra Large (XLarge) instance for optimizing the processing of Big Data. Our work uses a similar approach of this ([3]), but we believe is much more challenging to setup a Hadoop environment in Federated Clouds. Cloud Federation [4] [5] represents a compelling opportunity in which IaaS Cloud Operators might achieve great business benefits, renting to others cloud operators the computation resources on-demand (see [6]). The well-know Hadoop platform can represent an appealing opportunity in this way because its architecture is well consolidated and widely used. Any Cloud Operator might offer Hadoop computation resources on-fly joining a federated cloud environment. Hadoop uses MapReduce paradigm, an high-level programming model for data-intensive applications using transparent fault detection and recovery, widely adopted in cloud datacenters such as Microsoft, Google, Yahoo, and Facebook. Hadoop is an opensource implementation firstly developed by Yahoo [7]. In our work it is possible to setup a high-level programming model even in Federated and Heterogeneous Clouds. Deploying VMs in federated scenarios with Hadoop nodes inside, is a challenge as shown in [8].Many works are trying to optimize Hadoop computations in heterogeneous environment like shown in LATE [9], TARAZU [10], Cross-Phase Optimization [11] and PIKACHU [12].These works look at the paradigm attempting to optimize all processing tasks, in particular the three main phases: *map, shuffle and reduce.* At the first stage of our solution, we are looking at the possibility to elastically increase the computation resources leveraging even more VMs. A step over we should also consider similar approach in which to organizer all MapReduces phases and tasks in a federated way, that is selecting federated providers and deploying suitable VMs. Another example of Big Data processing in the Cloud is presented in [13]. In this work the computation framework used is Sailfish, a new MapReduce environment similar to Hadoop. Sailfish was conceived for improving the disk performance for large scale MapReduce computations. Hence it is possible to make the selection of federated contributors based on types of MapReduce paradigms.

3 Distributed Processing Service in Cloud Federation

In a federated Cloud environment, a CP can benefit of the storage and computational resources other CPs acting on other administrative domains. To satisfy client's requests, each CP in the federation asks for available resources to the other members of the federation, which offer their unused resources at that time. Of course, the amount of resources offered for each request can be regulated by specific federation agreements, but such issue is out of the scope of this paper. A CP can require to establish a partnership with other CPs for multiple reasons: it has saturated its own resources and it needs external assets, it wants to perform software consolidation in order to save energy cost, it want to move part of processing into other providers for improving security or performance in order to respect particular Service Level Agreements (SLAs). In particular, in this paper, we focus on a federated Cloud scenario offering MapReduce processing service. MapReduce can take the advantages of data location, processing it on closer storage assets in order to reduce data trasmission delay. Thus, in our scenario,

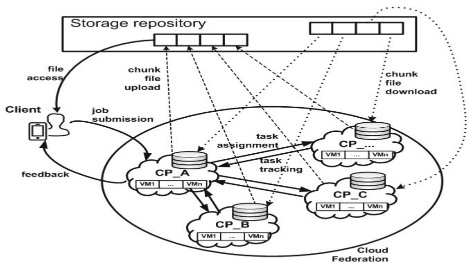


Fig. 1. Processing service management

CPs hold their internal storage system where deploying data sets they have to process. Since each CP stores a portion of data for local processing, we assume that big files that have to be processed are stored in an external Cloud Storage Providers (CSPs), such as Amazon S3, Google Drive, Dropbox, etc. The choise to rely on external public CSP was made to minimize the overhead associated to the data transmission between the federated domains. They, in this manner, have to exchange, each other, only coordination and sincronization messages. The idea behind such a service is shown in Figure 1. When a user requests to run a job, he contacts his Cloud Provider (CP_A in the example in Figure 1) and sends an input file (Xml file) containing the parameters nedeed to the job to be executed. CP_A involves all (or just a part of, depending on the job requirements) the CPs in the federation, giving them directives on the task they have to process. Supposing that the input data to process is memorized in a CSP that supports multipart download (i.e., the CSP splits the file in several blocks that clients are able to download), each involved CP to accomplish its task has to download only particular blocks of file. It is important to say that the system can scale both horizontally and vertically. It scales horizontally when a CP_A (Home CP) dynamically forwards the user task request to the federated domains (Foreign CP). But the system can also scales vertically when a chosen foreign CP, for some reason, cannnot longer fully meet the forwarded request by the home CP. So the foreign CP may in turn forward the sub-request to others available foreign CP. In this case, therefore, the CP plays both the role of foreign CP, towards the CP that initially sent the request, and the role of home cloud towards the new CPs to which it is forwarding the sub-request. Once each CP have download their respective blocks of data from CSP, it has to parallel process them by means of pieces of parallel processing middleware running on VMs. Each downloaded blocks is further divided in smaller chunks by the middleware used for parallel processing running on the Cloud domain. For simplicity, we assume that each CP in the scenario has an image of the VM including the piece of middleware for processing the task. However, additional mechanisms for VM image provisioning can be implemented to improve the flexibility of the offered service. At the end of the task, each CP uploads the results of the processing into the repository system and it notifies that to CP_A. As soon as all the CP end their work, CP_A

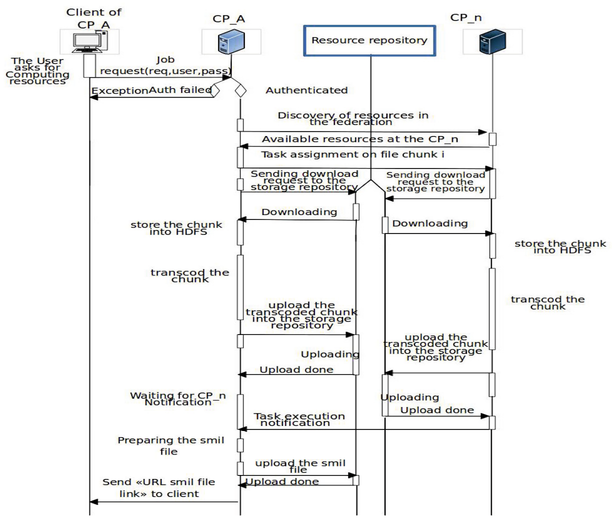

Client of CP_A
CP_A
Resource repository
CP_n
The User asks for Computing resources
Job request(req,user,pass)
ExceptionAuth failed
Authenticated
Discovery of resources in the federation
Available resources at the CP_n
Task assignment on file chunk i
Sending download request to the storage repository
Sending download request to the storage repository
Downloading
Downloading
store the chunk into HDFS
store the chunk into HDFS
transcod the chunk
transcod the chunk
upload the transcoded chunk into the storage repository
upload the transcoded chunk into the storage repository
Uploading
Upload done
Waiting for CP_n Notification
Uploading
Upload done
Task execution notification
Preparing the smil file
upload the smil file
Upload done
Send «URL smil file link» to client

Fig. 2. Processing service management

informs the client about the result of the processing. To better understand the benefits of this scenario, let us consider a video transcoding job as possible use case. A user would like to enjoy a movie that is available on a remote storage repository by using his mobile device. Unfortunately, the movie is stored as HD file and the user device is not able to play it. Thus, the user needs an on-fly video transcoding to convert the file to another format. The steps accomplished to obtain the transcoded video are shown in Figure 2. A client submits to his provider (CP_A) the job together with his credentials to access the service. If the authentication process has success, CP_A starts a resources discovery into the federated environment to look for available resources. The generic CP_n offers its storage and computational resources, if possible, and waits for instruction on the task to carry on and the chunks of file to process. The Hadoop framework at CP_A, exploiting the MapReduce features, parallelizes the transcoding process of the video file thus to involve as much resources as possible. As soon as CP_n receives the file localization information, it starts the download of the file chunks and put them (uploads) in its HDFS cluster for local processing. At the end of the processing step, CP_n stores the result of its processing in the CSP and sends to CP_A an end task notification. Once CP_A has received all the end task notifications from all the involved CPs, it generates a *SMIL* file, i.e., an XML file used to play the video avoiding to merge all the processed chunks. Also the SMIL file is uploaded into the CSP and it provides the base location of the video chunks and the necessary information for the client player to rebuild the whole video file. Finally, CP_A notifies its client about the end of job execution and provides him the location of the SMIL file.

4 Reference Scenario

In this Section, we describe our reference scenario including CLEVER, Hadoop, and Amazon S3.

4.1 Hadoop Overview

Hadoop MapReduce is a software framework to write and run applications in processing in parallel huge amounts of data (e.g. terabyte of datasets) on large clusters in a reliable, fault tolerant manner. A MapReduce job usually splits the input data set into independent chunks, which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Both the input and the output of the job are stored in a distributed file system, that is the Hadoop File System (HDFS).Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the HDFS are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster. The Hadoop framework has a Master/Slave architecture. MapReduce components consist of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master. The master node of the HDFS is called *NameNode*. It manages the namespace file system by maintaining a file metadata image that includes file name, location and replication state. DataNodes manage storage resources into the host they run on and allow read/write accesses. A typical Block size is 64 MB. Thus, a HDFS file is chopped up into 64 MB chunks, and, if possible, chunks are located at different DataNodes.

4.2 CLEVER Overview

The CLoud-Enabled Virtual EnviRonment (CLEVER) is a Message-Oriented Middleware for Cloud comptuting (MOM4C), able to support several Cloud-based services [14]. Each CLEVER Cloud includes several distributed hosts organized in a cluster. Each Phisical Machine (PM) is controlled by a management module, called Host Manager (HM), and only one host runs a cluster management module, called Cluster Manager (CM) that acts as interface between Cloud and clients. CM receives commands from clients, gives instructions to HMs, elaborates information and finally sends back results to clients. It also performs tasks for cluster orchestration. A CLEVER Cloud makes use of XMPP to exchange all communication messages and presence information in a near-real time fashion. A Jabber/XMPP server provides basic messaging, presence, and XML routing features within the Cloud. All the PMs in the Cloud are connected via a Multi User Chat (MUC) and cooperate according to the CM orchestration directives. In CLEVER, CM and HMs implements software *Agents* communicating through XMPP. Hence, it is easy to include new modules and functionalities to the CLEVER environment by adding new Agents and updating the CM and HMs configurations for the correct delivering of messages.

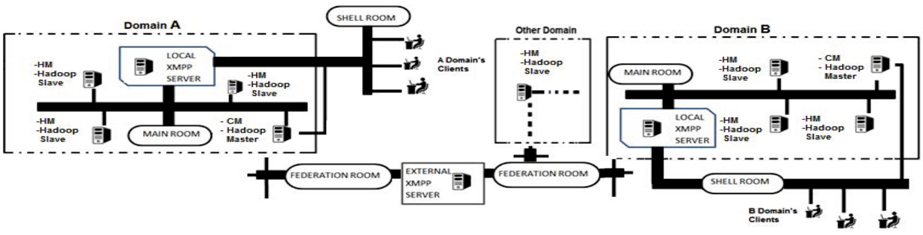
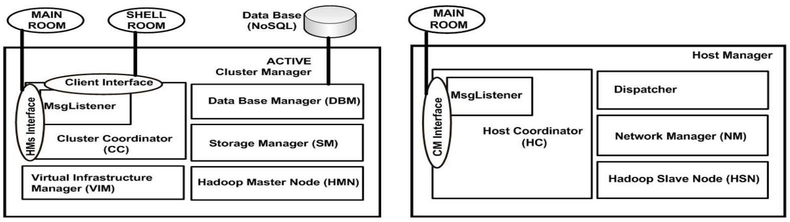


Fig. 3. CLEVER Federation Management

With CLEVER, each Cloud involved in the federation is identified by a *Jabber ID* (JID). As shown in Figure 3, in order to set up a federation, CMs belonging to different administrative domains exchange messages through the MUC with the unique room ID *Federation*, and only the authenticated, by the XMPP server itself or by external third party entities [15], ACTIVE CMs of federated Clouds can access it.

4.3 Integration of Hadoop in CLEVER

To make the Hadoop functionalities cloud like, we make use of a virtual infrastructure provided by CLEVER. VMs run on HMs and work as slaves of the Hadoop cluster. Virtual Hadoop slaves are coordinated by the Hadoop Master arranged at the CLEVER CM. The first advantage of the integration of Hadoop in CLEVER is that, typically, Hadoop uses the TCP/IP layer for communication, and it is a problem during the inter-domain comunication due to heavy usage of firewalls by each domain which take part to federation. Infact firewalls can block inter-domain communication. So, integrating Hadoop in CLEVER, federation messages can be sent on port 80 thanks to XMPP technology.The second one is that the system can automatically scale according to real time requirements. The two main software agents enabling CLEVER to integrate Hadoop are the *Hadoop Master Node (HMN) Agent* and *Hadoop Slave Node (HSN) Agent.* In the following, we discuss their activities and synchronization processes. Figure 4.a shows the software components at the CM. Through the *HMs interface*, the CC communicates with all the HMs in the cluster, exchanging information on available resources, running tasks, work specifications and offered services. The CC makes use of the *Client interface* to interact with Cloud clients, in order to receive client requests, and to give back inquired services. The Client interface allows service provisioning to clients exchanging XML messages into the



(a) Cluster Manager design. (b) Host Manager design.

Fig. 4. Integration of Hadoop in CLEVER

Shell Room. The VIM is the agent designed for managing virtual infrastructures. Moreover, the CM makes use of an internal NoSQL database for storing current system configurations, which is properly updated by the Data Base Manager (DBM). Figure 4.b shows the software components at the HM. The agent specifically designed to support the Hadoop activities in the Cloud is the *HMN Agent*. It provides the configuration settings to all the virtual nodes in the Hadoop cluster. The CLEVER HMN works as master for Hadoop cluster. Specifically, it implements the Hadoop functionalities to manage the hadoop system. At the startup, the *HMN Agent* reads the Hadoop configuration setup and then the CC subscribes this new Agent in the list of all the active agents of CLEVER, in order to make it reachable from the agents instantiated in the HMs. After the early registration, the *HMN Agent* can receive Notifications from the agents at the HMs.

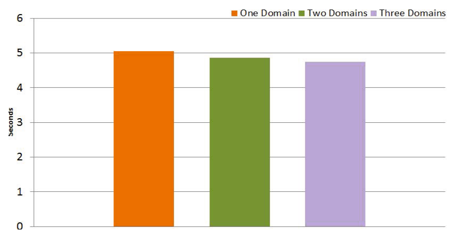
4.4 Amazon S3

Amazon S3 is Cloud storage service. It is designed to make web-scale computing easier for developers. Amazon S3 provides a simple web-services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

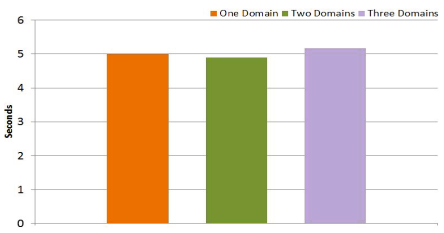
5 Experiments

In this Section, we discuss several experiments, we conducted on a real testbed involving four CLEVER/Hadoop administrative domains (i.e., A, B, C, and D) acting as federated Cloud providers and Amazon S3 acting as real Cloud storage provider. The objective of the experiments described in this Section consists to know what are the main factors needed for the job submission in a federated

Cloud environments and demonstrating, considering a real testbed, how Cloud federation can enable Cloud providers to take the advantages of parallel distributed processing. it is important to emphasize that integrating Hadoop in CLEVER we are adding a second livel of parallelization. In order to test the whole environment, we considered a parallel video transcoding use case involving several federated cloud providers. In particular, we arranged the testbed considering four physical servers (one per Cloud domain) running in total 10 VMs and Amazon S3. Experiments were conducted with the following hardware configuration: CPU: Intel(R) Core(TM)2 CPU 6300; 1.86GHz, 3GB RAM, running Linux Ubuntu 12.04 x86_64 OS and VirtualBox. Each experiment was repeated 50 times in order to consider mean values and a low confidence intervals.In the following, we summarize the main phases involved in our experiments. The process starts at time t_0 when a Cloud client sends a video transcoding request to a particular CLEVER domain. At time t_1 the CLEVER cloud that receives the request decides to establish a federation with the other CLEVER domains, retrieving domains information. For simplicity, in this paper, we have not treated how this process can be accomplished in autonomic fashion, but we a priori arranged the environment using the CLEVER commands. At t_2, the Cloud provider, that has initiated the federation establishment process, performs a task assignment involving the whole federated environment. Supposing that the Cloud that has started the process uses an external Cloud storage service provided by Amazon S3, each involved federated CLEVER Cloud will download only a particular number of video chunks for processing using the multipart download mechanism. In the end, t_4 indicates the time taken by each CLEVER Cloud to upload the previously downloaded video chunks in HDFS of the local domain, so that the Hadoop task tracker slave node, controlled by means of CLEVER, can process them. Figure 5.a shows the average time required for the accomplishment of phase 1 $(t_1 - t_0)$. It is possible to observe that, independently from the number of external administrative domains, the time for retrieving domain information remains constant taking roughly 5 seconds. We attribute this overhead at the access operation to the local database needed to CLEVER to

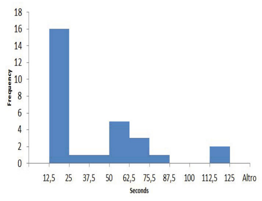


(a) Average time required to retrieve domain information on Clouds.

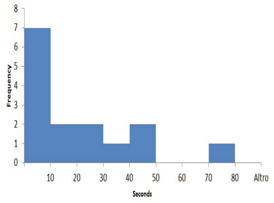


(b) Average time required to forward a request to federated Clouds.

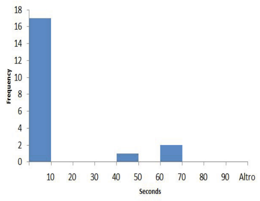
Fig. 5. Retrieving information and forward request times



(a) Download Time histogram for 20MB block size.



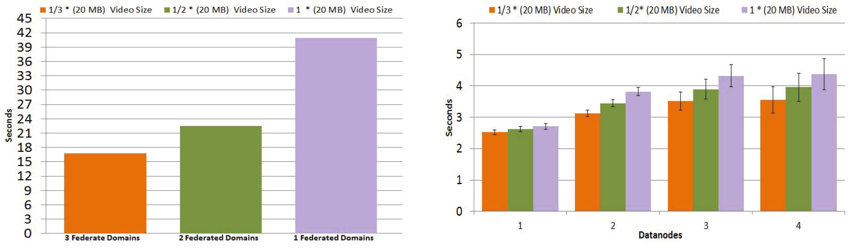
(b) Download Time histogram for 10MB block size.



(c) Download Time histogram for 7MB block size.

Fig. 6. Download time from Amazon S3

retrieve the network parameters of the other domains. At phase 2, $t_2 - t_1$ interval indicates the time required to forward the video transcoding request to the external federated CLEVER Clouds domains. These times are shown in the Figure 5.b. After that the CLEVER Cloud that has started the process, obtained the network information regarding external Cloud domains, runs a new thread for each of them, sending the requests in parallel. Thus, the average time does not change if the number of the foreign domains does. Figure 5.a, 5.b, and 5.c show respectively the distribution of the download times of 1/3, 1/2, and the whole video files from Amazon S3. In our timing diagram, this time is represented by the $t_3 - t_2$ interval. Observing the Figure 6.a, 6.b, and 6.c, we can notice that, if there is only one domain which takes part to the federation, it has to download the whole video file(20MB), instead when there are other domains into the federetion, each of them has to retrive only a block of the original file. So, when the number of the federated domains increases, the download-time decreases. In particular Figure6.a, shows the download time when a single external CLEVER Cloud administrative domain takes part to the federation, so that, it has to download the whole video file(20MB) from Amazon S3. Figures 6.b and 6.c, instead, respectively show the download times when two and three external CLEVER Cloud administration domains take part to the federation. In fact, each domain downloads only specific block of video file. Observing the graphs, we can note that the download time for the whole 20MB file takes roughly 40 seconds, while the times needed for downloading half file (10MB) and a third of file (7MB) take respectively roughly 22 and 15 seconds. Figure 5 summarizes the aforementioned results. Moreover, each download takes place in parallel, so we have a double benefit, the first one due to the smaller blocks size to be downloaded, the second one due to the parallelization of the download in these blocks. The average download time is depicted in Figure 7.a. Instead, Figure 7.b shows how the average upload time of blocks of file in the HDFS of each domain. This time changes according to the number of active DataNodes and video file sizes. Observing the graph depicted in Figure 7.b, we can notice that increasing the number of Hadoop Data Nodes the upload time increases too. We can motivate



(a) Average download time of file blocks from Amazon S3.

(b) Average Upload time of file blocks in Hadoop.

Fig. 7. Average download and upload

this trend remembering that the Hadoop has been configured with a redundancy parameter equal to 2. In fact with a single active DataNode, the upload time has a very low value, because the system does not have the need to replicate the file. Instead, due to Hadoop's data replication mechanisms, increasing the number of Data Nodes, we can notice a linear increase of the upload.

6 Conclusion

In this paper, we discussed how can be possible to apply the MapReduce paradigm in a federated Cloud environment. MapReduce allows to perform a parallel processing of large data set stored into a file system. The Hadoop framework couples the MapReduce algorithms with the HDFS storage system. The proposed solution integrates the Hadoop framework into CLEVER and uses Amazon S3 as external CSP. We deeply discussed the proposed processing service focusing on job submission.

References

1. Panarello, A., Celesti, A., Fazio, M., Villari, M., Puliafito, A.: A Requirements Analysis for IaaS Cloud Federation. In: 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain (2014)
2. Petruch, K., Stantchev, V., Tamm, G.: A survey on it-governance aspects of cloud computing. IJWGS 7(3), 268–303 (2011)
3. Yuan, Y., Wang, H., Wang, D., Liu, J.: On interference-aware provisioning for cloud-based big data processing. In: 2013 IEEE/ACM 21st International Symposium on Quality of Service (IWQoS), pp. 1–6 (June 2013)
4. Rochwerger, B., Breitgand, D., Epstein, A., Hadas, D., Loy, I., Nagin, K., Tordsson, J., Ragusa, C., Villari, M., Clayman, S., Levy, E., Maraschini, A., Massonet, P., Muñoz, H., Tofetti, G.: Reservoir - when one cloud is not enough. Computer 44, 44–51 (2011)

5. Kertesz, A., Kecskemeti, G., Marosi, A., Oriol, M., Franch, X., Marco, J.: Integrated monitoring approach for seamless service provisioning in federated clouds. In: 2012 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 567–574 (February 2012)
6. Toosi, A.N., Calheiros, R.N., Buyya, R.: Interconnected cloud computing environments: Challenges, taxonomy, and survey. ACM Comput. Surv. 47, 7:1–7:47 (2014)
7. The Apache Hadoop project: the open-source software for reliable, scalable, distributed computing, `http://hadoop.apache.org/`
8. Gahlawat, M., Sharma, P.: Survey of virtual machine placement in federated clouds. In: 2014 IEEE International Advance Computing Conference (IACC), pp. 735–738 (February 2014)
9. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R., Stoica, I.: Improving mapreduce performance in heterogeneous environments. In: 8th USENIX Conference on Operating Systems Design and Implementation, OSDI 2008, pp. 29–42. USENIX Association, Berkeley (2008)
10. Ahmad, F., Chakradhar, S.T., Raghunathan, A., Vijaykumar, T.N.: Tarazu: Optimizing mapreduce on heterogeneous clusters. SIGARCH Comput. Archit. News 40, 61–74 (2012)
11. Heintz, B., Wang, C., Chandra, A., Weissman, J.: Cross-phase optimization in mapreduce. In: Proceedings of the 2013 IEEE International Conference on Cloud Engineering, IC2E 2013, pp. 338–347. IEEE Computer Society, Washington, DC (2013)
12. Gandhi, R., Xie, D., Hu, Y.C.: Pikachu: How to rebalance load in optimizing mapreduce on heterogeneous clusters. In: USENIX Conference on Annual Technical Conference, USENIX ATC 2013, pp. 61–66. USENIX Association, Berkeley (2013)
13. Rao, S., Ramakrishnan, R., Silberstein, A., Ovsiannikov, M., Reeves, D.: Sailfish: A framework for large scale data processing. In: Proceedings of the Third ACM Symposium on Cloud Computing, SoCC 2012, pp. 4:1–4:14. ACM, New York (2012)
14. Fazio, M., Celesti, A., Puliafito, A., Villari, M.: A message oriented middleware for cloud computing to improve efficiency in risk management systems. Scalable Computing: Practice and Experience (SCPE) 14, 201–213 (2013)
15. Celesti, A., Fazio, M., Villari, M.: Se clever: A secure message oriented middleware for cloud federation. In: IEEE Symposium on Computers and Communications (ISCC), pp. 35–40 (July 2013)