Distributed Storage Support in Private Clouds Based on Static Scheduling Algorithms

Dariusz Król
Academic Computer Center CYFRONET AGH
Cracow, Poland
dkrol@agh.edu.pl

Jacek Kitowski
Academic Computer Center CYFRONET AGH, and
Institute of Computer Science, AGH
Cracow, Poland
kito@agh.edu.pl

Abstract—**This paper is focused on an extension to an open source Infrastructure as a Service Cloud called Eucalytpus for supporting distributed storage according to a defined storage strategy. As a proof of concept, three algorithms known from the scheduling theory were implemented, namely MonteCarlo, Round Robin and Weighted Queuing. To evaluate the extension, a set of tests were performed on a sample Cloud installation using a modified version of the Eucalyptus cloud. The paper ends up with a discussion on choosing the most eficient static algorithm for data storing based on the obtained results.**

Keywords - cloud computing; storage management; Eucalyptus; scheduling.

I. INTRODUCTION

Gartner has identified the Cloud computing as one of the top 10 strategic technologies in 2011 [1]. Today, most of the big Information Technology (IT) companies offer some of their products within public clouds already. These suppliers applied the Cloud paradigm to provide a wide set of applications in an easily accessible manner, starting with e-mail clients, through office suites to content resource management systems. Although, each of those applications provides different functionality, they have a few things in common, e.g., they can be accessed via a web browser, and they are provided using the pay-as-you-go manner.

Besides examples in the industry, many scientific facilities started adapting the Cloud computing. This is possible due to the existence of several open-source projects which implement the Cloud computing paradigm with open standards. While the adaption of clouds in the industry is often focused on applications, the scientific centers rather aims at providing infrastructure-level services which facilitate access to compute and storage resources.

A similar approach to resource provisioning is well known from many previous works concerning Grid environments [2]. While Clouds are business-oriented from the beginning, Grids are science oriented. From the user point of view, the main difference is the orientation on different usage modes [3]. While Clouds expose a small but well-defined interface set, Grids provides a wide-set of functions regarding similar functionality.

Existing clouds can be divided into three different groups with regard to the visibility and availability of a cloud from the users point of view. The most available are public clouds that can be used by everyone without any constraints. This category includes Amazon Elastic Compute Cloud (Amazon EC2) [4], Microsoft Azure [5], Google AppEngine [6] and many others. The opposite of public clouds are private clouds. In most cases, they are limited to the resources of a single organization and can be accessed only from within the organization's network and by an organization member. The third group concerns private clouds whose computation power and storage capacity can be extended by resources of public clouds. This group includes also hybrid clouds.

Another taxonomy of clouds concerns styles in which the customer uses Cloud. This taxonomy includes:

- Infrastructure as a Service (IaaS) Clouds which provide access to virtualized pool of resources using which customers assemble virtual machines,
- Platform as a Service (PaaS) Clouds which provide access to a well defined runtime environments and programming services which are used to develop applications without troubling with virtual machines,
- Software as a Service (SaaS) Clouds which deliver concrete applications which are deployed at the providers infrastructure.

Finally, clouds can be divided base on the type of resources which are provided. Today, this taxonomy includes two elements: compute clouds and storage clouds. The first group comprises clouds which provide access to computational power by running virtual machines or applications on a specified virtualized hardware, e.g., a virtual machine with a single, normalized, virtual CPU, 512 MB of RAM and 10 GB of hard drive capacity. On the other hand, the storage clouds enable users to store data sets in a number of ways, i.e., in files, (non-)relational databases or block devices. In theory, the storage clouds can provide an infinity storage capacity on demand.

In this paper, we focus on private, storage clouds. They can be used as an convenient way for storing users data, e.g., application results on storage resources of a single organization by organization members. It can be also used to virtualize different types of storage systems, e.g., disk arrays, local disks etc., to be visible as a single storage system from the end user point of view, thus it can increase the simplicity of sharing data between different users and applications. A storage cloud can be used to store different types of data, starting with text files and ending with binary files. As long as data can be written to a file, they can be stored in the Cloud.

To build a private, storage cloud in an effective way, a cloud implementation has to provide support for heterogeneous storage resources and different data distribution algorithms.

 ISBN: 978-1-61208-153-3

The former functionality provides a capability of connecting existing storage devices into a single system. The latter functionality is used to increase the performance of the cloud, e.g., read and write transfer rate.

In this paper, we intend to describe an extension for an existing, open-source Cloud which aims at providing a data distribution functionality based on static scheduling algorithms. Although, the developed extension is independent from a concrete data distribution algorithm, this paper focuses only on a few popular algorithms known from the queuing theory.

The rest of the paper is organized as follows. In Section 2, we describe a number of existing Cloud solutions as well as a few data management systems. Then, in Section 3, the Eucalyptus project is described in more details. In Section 4, a design of the extension of the Eucalyptus system which provides support for distributed storage is presented. Next, in Section 5 an implementation of our extension is presented. In Section 6, we come to an experimental evaluation of the presented extension. The paper is concluded in Section 7.

II. RELATED WORKS

OpenNebula [7] is an open-source toolkit for building compute-oriented private, public or hybrid clouds. The toolkit provides an abstraction layer on top of physical resources of a data center using the virtualization mechanism. It is oriented on deploying multitier services as virtual machines on distributed infrastructure. OpenNebula aims to overcome shortcomings of existing virtual infrastructure solutions, i.e., inability to scale to external clouds, a limited choice of interfaces with the existing storage and network management solutions, few preconfigured placement policies or the lack of support for scheduling, deploying and configuring groups of virtual machines. OpenNebula is fully open source and its source code can be freely checkout from a public repository. It supports different hypervisors, i.e., Xen [8], Kernel-based Virtual Machine (KVM) [9], VMware [10], for running virtual machines. In terms of storage mechanisms, it is limited to a repository of Virtual Machine (VM) images only. The repository can be shared between available nodes with the Network File System (NFS). It is also possible to take advantage of block devices, e.g., Logical Volume Manager (LVM) to create snapshots of images in order to decrease time needed to run a new instance of image. Due to this limitation, it is not a suitable tool for building storage clouds.

Another open-source solution for building different types of clouds is OpenStack. It is a joint effort of NASA and RackSpace. NASA contributed to the project by releasing its middleware, called Nebula [11], for managing virtual machines at physical infrastructure. RackSpace contributed with its storage solution known as Cloud Files [12]. OpenStack [13] is a collection of tools for managing data centers resources to build a virtual infrastructure. In terms of computations, OpenStack provides OpenStack Compute (Nova) solution which is responsible for managing instances of virtual machines. In terms of storage, OpenStack provides OpenStack Object Storage (Swift) which is an object storage solution with built-in redundancy and failover mechanisms. There is also a separate subsystem, called OpenStack Imaging Service, which can be used to lookup and retrieving virtual machine images. Since the first release of OpenStack was in October 2010, there are no articles about production deployments of the toolkit in either industry or scientific area yet. Thus, there is no information about the performance and stability of OpenStack. Also, OpenStack lacks of an interface that would be compatible with the Amazon clouds which is a *de facto* standard in the Cloud ecosystem.

Eucalyptus system [14] is an example of an open source project which became very popular outside the scientific community and is exploited by many commercial companies to create their own private clouds. It was started as a research project in the Computer Science Department at the University of California, Santa Barbara in 2007 and today is often treated as a model solution for providing infrastructure as a service. Eucalyptus aims at providing an open source counterpart of the Amazon EC2 and Simple Storage Service (Amazon S3) [15] clouds in terms of interfaces and available functionality.

There are two versions of the Eucalyptus Cloud: Community and Enterprise. The Community edition will be described in the next section in more details. The Enterprise Eucalyptus provides direct integration with Storage Area Networks (SANs) [16], e.g., Dell Equallogic or NetApp. However, to our best knowledge, this integration does not allow to combine different types of storage systems within a single Cloud installation. Also, a Cloud administrator can`t provide policy for data distribution among available storage resources.

Another commercial product is EMC2 Atmos which is a complete Cloud Storage-as-a-Service solution [17]. It provides massive scalability by allowing to manage and attach new storage resources from a single control center. Atmos delivers policy-based information management feature which allows to define bussiness level policies how the stored information should be distributed between available resources. It also reduces required effort for administration by implementing auto-configuring, auto-managing and auto-healing capabilities. Although, Atmos provides many interesting features and capabilities, it does not provide integrations with existing Clouds to our best knowledge. It is rather a separate solution oriented on the storage only which operates besides a computing Cloud.

DCache [18] is a data management system which implements all the requirements for a Storage Element in the Grid. It was developed at CERN to fulfil the requirements of the Large Hadron Collider for data storage. One of its main features is the separation of the logical namespace of its data repository from the actual physical location of the data. DCache exposes a coherent namespace built from files stored on different physical devices. Moreover, dCache autonomously distributes data among available devices according to the currently available space on devices, workload and the Least Recently Used algorithms to free space for the incoming data. Although dCache distributes data in an autonomic way, there are settings which can be configured to tune the dCache installation to specific requirements of a concrete user. This parameter set contains rules which can take as an input a directory location within the dCache file system and storage information of the connected Storage Systems as well as the IP address of the client and as an output such a rule returns a destination

 ISBN: 978-1-61208-153-3

where the data should be sent. DCache is a Grid-oriented tool by design, thus it is not compatible with existing Cloud solutions. DCache provides a programming interface similar to a filesystem interface which is at a lower level of abstraction comparing to the storage cloud interface. However, dCache could be used as a storage system which is used by a storage cloud rather than being a complete storage cloud solution.

III. EUCALYPTUS – OPEN SOURCE PRIVATE CLOUD

As describe above, Eucalyptus is one of the existing solutions for building private, public or hybrid clouds. It supports both compute and storage clouds. The most characteristic feature of Eucalyptus is the fact that it is fully compatible with the Amazon EC2 and S3 clouds at the interface layer. Therefore, it can be used interchangeably with the Amazon clouds without any modification of the users application.

Every Eucalyptus installation consists of a few loosely coupled components, each being able to run on a separate physical machine to increase scalability. The front end of such a cloud is "Cloud controller" which is an access point to the features related to virtual machines management. While "Cloud controller" is responsible for computation, the "Walrus" component is responsible for data storage. Each virtual machine runs on a physical host which is controlled by the "Node controller" element. A group of nodes can be gathered into a cluster which exposes a single access point, namely "Cluster controller" from the virtual machine management side and "Storage controller" from the virtual machine images repository side.

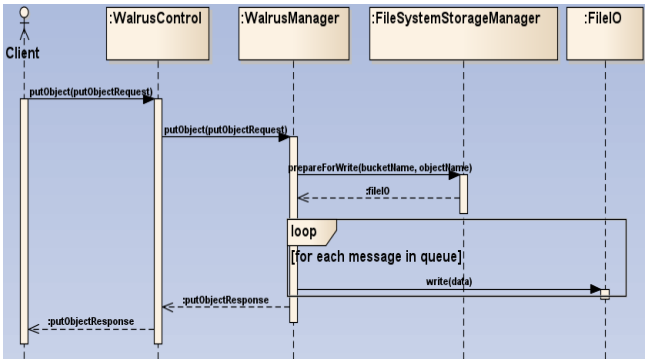

Figure 1: A sequence diagram of the "data storage" operation.

A. Data storage functionality

In terms of data storage, Eucalyptus provides two means for persisting the data generated by applications running in the Cloud: Object Storage and Elastic Block Storage (EBS).

The former one allows for storing virtual machine images along with any other files which are divided into a flat hierarchy of buckets and can be treated as the Amazon Simple Storage Service (S3) counterpart in the Eucalyptus system. Amazon S3 is a Cloud storage service which allows storing any type of data in form of files in a number of buckets (each with a unique name within a bucket) using a simple programming interface, i.e., *put*, *get*, *list* and *del*. The Eucalyptus Object Storage provides exactly the same set of functions which can be executed using a Representational State Transfer (REST) based interface. There are also several tools available which wrap the interface, e.g., a simple command line tool or programming language bindings.

The latter mechanism, i.e., Elastic Block Storage allows for providing virtual machines with block devices which are attached to virtual machines at runtime. However, unlike a virtual machine local disk, such an attached block device is not erased after the VM shutdown.

B. Data storage implementation

A part of the current implementation of storing an object within the Eucalyptus cloud is depicted in Figure 2. Due to high complexity, only one part of the "storing data" use case is presented, namely the one related to actual writing data to physical devices. The first part of the use case is related to handle HTTP requests which contain raw data that is going to be stored. Eucalyptus uses queues to handle incoming requests. Then, the *WalrusManager* object retrieves all the message objects from these queues, opens a file which is accessible with standard IO functions, and finally writes the data to the file.

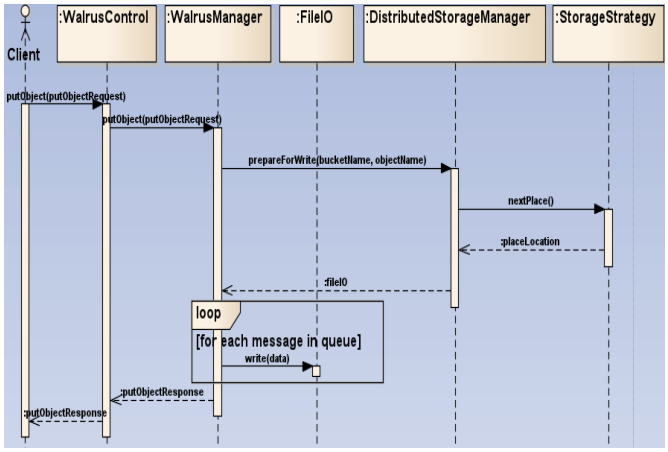

Figure 2: A sequence diagram of the modified storing data use case.

The most important part of the sequence diagram which concerns data distribution is the *preparingForWrite()* call. In the current version of Eucalyptus this method returns an object which uses the Java *FileChannel* class to write data. Moreover, a mapping between Eucalyptus objects and filesystem files implies that all the data has to be stored in a single directory. Moreover, this directory can be located only on a Walrus local disk or a volume that is attached to the Walrus machine, e.g., a disk array via Internet Small Computer System Interface (iSCSI) or a Network Attached Storage via NFS.

However, this means there is only one option to distribute the cloud data, i.e., using a distributed file system on an disk array attached to the Walrus machine which encompasses a number of storage resources. This limitation prevents from exploiting heterogeneous storage systems to build consistent storage cloud from the end user point of view. Moreover, even if heterogeneity is not an issue, distributed file systems do not provide a capability of defining storage strategies for data distribution. In most

cases, distributed file systems aim at either balancing the workload between storage devices or balancing the free space of storage devices. However, if clouds are in our scope, such a basic functionality is not sufficient.

IV. DATA DISTRIBUTION WITH STATIC SCHEDULING ALGORITHMS

In this section, we describe our solution to the problem of data distribution among several, possibly heterogeneous, storage resources. Starting with our motivation, an extension to the Eucalyptus cloud is next presented along with a sample storage strategies which are based on well-known scheduling algorithms.

A. Motivation

As described in the previous section, data distribution is poorly supported in the current version of the Eucalyptus cloud. Low-level mechanisms, i.e., distributed file systems, lack of flexibility in defining the storage strategies that exploit information about the Cloud in particular.

Just to mention a few possible applications of such strategies, let us imagine a situation where we have several disk arrays in our data center which can be used to provide storage capacity for our Cloud. However, we cannot use them all because they are shared between a number of other different projects and users thus their configuration, e.g., filesystem, cannot be modified. In such a situation, we could use only one of the available disk arrays which would probably not meet our needs because Eucalyptus does not provide means for connecting several disk arrays together into a single cloud storage.

Another possible situation is when we would like to separate users' data, based on groups a particular user belongs to. Such a users' group can be bound to a Service Level Agreement between the user and the Cloud provider. From the Cloud provider point of view, each users' group could be handled by a different physical device, i.e., the users who pay more are treated with more reliable and efficient resources.

Also many other situations can be described where support for distributed storage is crucial to succeed but the importance of this functionality should be clearly visible in advance.

B. Design and implementation

When designing an extension to Eucalyptus that provides support for distributed storage, we focused on making it as non-intrusive as possible. Thus, we decided to replace an existing implementation of the *StorageManager* Java interface, namely an instance of the *FileSystemStorageManager* class (depicted in Figure 1) with its another implementation which is aware of the distributed storage. By doing so, we can activate this functionality with only two modifications to the Eucalyptus source code, i.e., in the places where the *StorageManager* variables are instantiated. Even these modifications can be eliminated by using the Dependency Injection pattern [19] and one of its Java implementation, e.g., the Spring framework [20].

A modified version of the "data storage" use case is shown in Figure 2. Due to being part of the Eucalyptus cloud, this extension has access to the whole information about cloud users, user data, etc. Therefore, it can implement a storage strategy on a higher level of abstraction than a distributed file system.

The implemented prototype of this extension enables a Cloud administrator to decide which storage strategy should be used by only modifying one configuration file that besides information about the storage strategy, contains information about available storage resources.

C. Implemented data distribution strategies

Although, the described extension is versatile, i.e., various storage strategies can be implemented and used at runtime, we implemented three strategies as a proof of concept. We exploited algorithms known from the scheduling theory:

- *MonteCarlo* strategy which randomly (with a uniform distribution) chooses a place to store the given data.
- *RoundRobin* strategy which stores the given data alternately on each of the available resources.
- *WeightedQueue* strategy which divides the available bandwidth to a number of channels whose "width" is proportional to weights assigned to storage resources. In the basic version of this algorithm, the weights are assigned to each device arbitrarily by the administrator.

The proposed strategies represent a group of so called *static* scheduling algorithms. As opposed to *dynamic* scheduling algorithms, they do not change the scheduling scheme, i.e., the order of storage resources, as a response to changes in the environment, e.g., infrastructure workload.

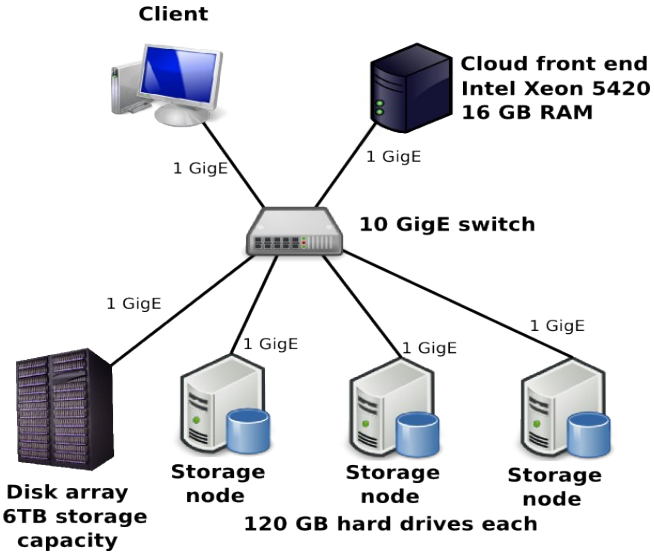

Figure 3: A map of a testing environment.

Although, the static scheduling algorithms can be less efficient than the dynamic ones, they are more predictable and straightforward. Thus, they are more suitable for testing the described functionality comparing the currently available Eucalyptus version. Also, they are more suitable than business-level algorithms because they allow to focus to performance analysis rather than on functional requirements,

e.g., distributing data of different users groups to different storage resources.

V. EXPERIMENTAL EVALUATION

In order to evaluate the implemented extension, a proper testing infrastructure has been composed and a number of tests were performed. The evaluation aimed at finding which storage strategy provides the highest throughput of the Cloud infrastructure. In addition, we would like to find out whether a cloud storage can be built based on commodity hardware, e.g., standard hard drives connected with a commodity ethernet network, instead of expensive disk arrays connected with a special network such as Storage Area Network (SAN) based on FibreChannel, with maintaining the Cloud performance at the same level.

A. Testing environment

Testing environment is a very important aspect of the experimental evaluation. Thus, we prepared a sample configuration for building a small Cloud installation based on a blade-class cluster nodes and a disk array. As a base server for an extended version of the Eucalyptus cloud we use a worker node with the following parameters:

- 2x Intel Xeon CPU L5420 @ 2.50GHz (4 cores each)
- 16 GB RAM
- 120 GB hard drive (5400 RPM)
- Ubuntu Linux 10.04.1 LTS.

Apart from the Cloud front end where the Cloud controller and Walrus components were installed, we also have three similar nodes for running virtual machines connected with the front end by Gigabit Ethernet.

However, a more interesting part of the environment concerns the storage. As a main storage for our cloud installation we used a part of a disk array accessible via iSCSI protocol, with 6 TB of storage capacity. Such a disk array, however, with a greater storage capacity available, could be used in a production cloud. As an additional storage, we decided to use hard drives from the additional worker nodes which are exposed via the NFS protocol.

To summarize, we depicted a map of the testing environment in Figure 4. In our opinion, the presented environment can be effectively used to evaluate different storage strategies because it contains heterogeneous storage resources such as hard drives and disk array distributed among a few machines all connected with open protocols and commodity network fabric.

B. Testing scenario

In the presented case, we proposed a scenario in which a number of users stores files in the Cloud simultaneously. Such a scenario is parametrized with the following elements:

- number of users running in parallel – 10
- file size – 128, 256, 512, 1024, 2048 MB
- storage strategy – MonteCarlo, RoundRobin, WeightedQueue (with a number of different weight vectors)

Each test scenario was perfomed 5 times and the mean value was computed. The performance evaluation metric used in the presented tests is the Cloud write throughput. The metric represents the total rate of writing data by the Cloud to its storage resources. This metric allows to compute the overhead generated by Eucalyptus to the storing data operation. Moreover, we can analyze the utilization rate of the storage resources with respect to different storage strategies.

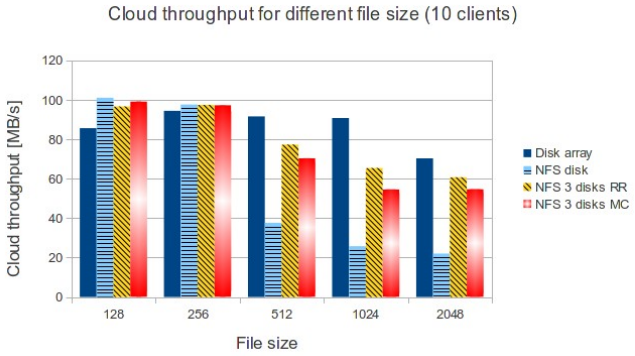

Figure 4: The Cloud throughput depending on a file size with 10 clients run in parallel.

C. Results and discussion

Firstly, the results coming from the tests performed with a single storage resource and with storage resources accessible via NFS are depicted in Figure 4. The results show a huge difference between the performance of the Cloud which uses a disk array and the Cloud which uses a common hard drive connected via NFS. The difference increases with the file size. This is expected due to the cache mechanism. When the file size is greater then the system cache then the performance of the Cloud gets stable. A second thing to notice is the performance of the Cloud which uses three connected hard drives via NFS. The mean performance of this configuration is smaller than in the configuration with a disk array but they are comparable. Also we can notice a slight performance gain when the *RoundRobin* strategy has been used. Also, we should notice a large diversity of measurement values in the storage configuration with a single NFS disk. The smallest diveristy of measurement values was obtained with a configuration of the disk array.

The second part of the results which contains the measured throughput with regard to the selected storage strategy is depicted in Figure 5. This test was performed with a Cloud installation which includes a disk array and three hard drives, exposed via NFS.

The results show that the *MonteCarlo* strategy is the worst one. For 1024 MB files the Cloud throughput for the *MonteCarlo* is less by 1/5 than the Cloud throughput for the *RoundRobin* strategy. The performance achieved in other strategies are similar and are close to 95 MB/s. Since the theoretical network performance is about 125 MB/s the achieved throughput is about 76% of the theoretical value and slightly more then 80 % in the best case.

Comparing the distributed storage to non-distributed storage, the results show about 10% of performance gain. Such a small gain is probably due to the limited network bandwidth rather than storage resources throughput

 ISBN: 978-1-61208-153-3

limitations. Thus, it is highly probable that, if there would be more than one physical network interface (as in our testing environment) coming from the Cloud front end, the Cloud throughput would scale better with the additional resources.

Table 1: Statistical parameters (in MB/s) for the throughput measurement for different storage strategies.

Storage Strategy	Mean	Variance	Confidence interval (α=0.05)
Round Robin	96.98	10.07	[93.96; 100.01]
Monte Carlo	90.50	110.43	[80.48; 100.52]
WQ-32111	96.71	6.91	[94.21; 99.22]
WQ-21111	95.33	0.21	[94.88; 95.77]
WQ-31111	96.37	1.26	[95.29; 97.44]

In Table 1, we gathered important statistical parameters which describe data from the second test case. Although the RR storage strategy leads to the largest mean throughput, the narrowest confidential interval can be obtained with the weighted queue strategy. The MC strategy is the most unpredictable.

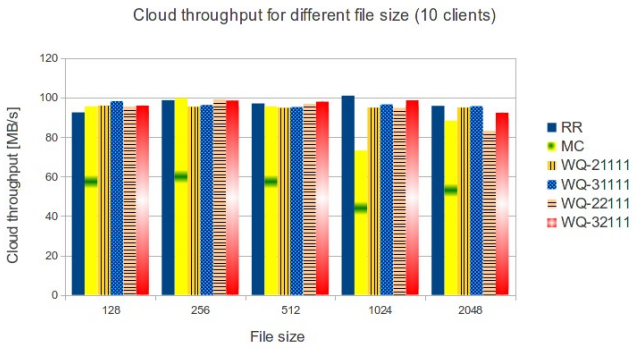

Figure 5: The Cloud throughput depending on a storage strategy.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we aimed at emphasizing the necessity of supporting distributed storage in building storage clouds. Upon having compared several open-source toolkits for building private clouds we decided to use Eucalyptus due to its compatibility at the interface level with the *de facto* standard in Cloud ecosystem, i.e., the Amazon clouds. As described in Section III, the current version of Eucalyptus does not provide sufficient functionality regarding data management. A non-intrusive extension to Eucalyptus has been proposed and implemented. The results from a number of performed tests show that a distributed storage can improve the Cloud throughput comparing the original implementation even if commodity hardware is used. Moreover, when using a distributed storage, the Cloud performance gets stable near the theoretical value of the network bandwidth.

The future work concerns improving the stability of the proposed extension. Also some new storage strategies, similar to those described in Section 4, are going to be provided.

ACKNOWLEDGMENT

This research is supported partly by the European Regional Development Fund program no. POIG.02.03.00-00-007/08-00 as part of the PL-Grid Project . The authors are grateful to Dr. Dr. Łukasz Dutka, Renata Słota and Włodzimierz Funika for valuable discussions.

REFERENCES

[1] Gartners report about the Top 10 Strategic Technologies for 2011, [online: http://www.gartner.com/it/page.jsp?id=1454221, as of April 16, 2011].

[2] The iRODS project website: [on-line: https://www.irods.org, as of April 16, 2011].

[3] S. Jha, A. Merzky, and G. Fox, “Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes”, Journal Concurrency and Computation: Practice & Experience, vol. 21 (8), pp. 1087-1108, June 2009.

[4] Amazon Elastic Compute Cloud website [on-line: http://aws.amazon.com/ec2, as of April 16, 2011].

[5] Microsoft Windows Azure Platform website [on-line: http://www.microsoft.com/windowsazure/, as of April 16, 2011].

[6] Google AppEngine website [on-line: http://code.google.com/appengine/, as of April 16, 2011]

[7] D. Milojičić, I. Llorente, and R. Montero, "OpenNebula: A Cloud Management Tool," IEEE Internet Computing, vol. 15(2), pp. 11-14, Mar./Apr. 2011, doi:10.1109/MIC.2011.44.

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, and A. Warfield, "Xen and the art of virtualization," in SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles. NY, USA: ACM, 2003, pp. 164-177.

[9] Kernel-based Virtual Machine project wiki. [on-line: http://www.linux-kvm.org, as of April 16, 2011].

[10] VMware website. [on-line: http://www.vmware.com, as of April 16, 2011].

[11] NASA Nebula website. [on-line: http://nebula.nasa.gov/, as of April 16, 2011].

[12] RackSpace CloudFiles solution website. [on-line: http://www.rackspace.com/cloud/cloud_hosting_products/files/, as of April 16, 2011].

[13] OpenStack project website. [on-line: http://www.openstack.org, as of April 16, 2011].

[14] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The Eucalyptus Open-Source Cloud-Computing System”, CCGRID '09 Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society Washington, DC, USA 2009.

[15] Amazon Simple Storage Service project website, [on-line: http://aws.amazon.com/s3/, as of April 16, 2011].

[16] Introduction to Storage Area Networks, IBM redbook, [on-line: http://www.redbooks.ibm.com/abstracts/sg245470.html?Open, as of April 16, 2011].

[17] EMC2 Atmos product web site, [on-line: http://www.emc.com/storage/atmos/atmos.htm, as of April 16, 2011].

[18] G. Behrmann, P. Fuhrmann, M. Gronager, and J. Kleist, “A distributed storage system with dCache”, in G .Behrmann et al Journal of Physics: Conference Series, 2008.

[19] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software”, 1995, ISBN: 0-201-63361-2.

[20] Spring Framework website. [on-line: http://www.springsource.org/, as of April 16, 2011].