Redundant Cloud Storage with Octopus

Christian Baun   Marcel Kunze

Steinbuch Centre for Computing
Karlsruhe Institute of Technology
christian.baun/marcel.kunze@kit.de

Abstract

Several ways exist to store data inside public and private cloud infrastructure and storage services. Regarding the availability of the stored data, all theses offerings and solutions face drawbacks. In this paper the software service Octopus is analyzed, that is designed to provide a high-availability cloud-based storage solution.

Keywords Cloud Computing, Storage Services, Amazon Web Services, Google App Engine

1. Introduction

Storing data is beside running virtual server instances one of the major tasks that is offered by cloud services. The growing number of public and private cloud infrastructure and storage services offers the customers several ways to store data. This paper analyzes the advantages and drawbacks of different ways to store data with cloud services that are compatible to the Amazon Web Services (AWS). The service level agreements of Amazon and Google are compared and the design and implementation of a service called Octopus is analyzed. This service connects several S3-compatible storage services to a high-availability RAID-1 storage solution.

2. State of the art

AWS-compatible cloud services provide their customers several ways to store data (see table 1). The customers can store data inside their Elastic Compute Cloud (EC2) instances. Disadvantages of this approach are limited capacity and that all changes are lost after the termination of an instance. To access data from outside the local network of the service provider, the instance need to be in the status running and a server software that grants access to the data via protocols like NFS, SMB or FTP need to installed and maintained. Depending of the instance type and server load, the instance itself can be a bottleneck for accessing the data.

A alternative and also persistent way to store data are Elastic Block Store (EBS) volumes. The volumes are block-based and the handling is similar to an non-formated block storage device like a hard disk drive (hdd). A volume can include any filesystem and the customers can create/erase volumes and attach/detach them to/from their EC2 instances at any time. A volume remains available until the customer implicit erases it. Disadvantages are that a volume can only be attached to one instance at a time and volumes can only be attached to instances that reside inside the same availability zone. To access the data stored inside a volume from outside the local network of the service provider, at least a single instance with a server software need to be in the status running with the volume attached and this instance can also be a bottleneck for accessing the data.

The object-based S3-compatible (Simple Storage Service) services provide persistence too. Data is stored there in form of web objects. Accessing these objects from anywhere on the web via REST or SOAP is possible at any time and no instance is needed. Using a file system inside an object-based storage service is not foreseen, but it is possible to create a file system inside an object with tools like s3fs[1].

Additional approaches and services to store data inside the AWS and compatible public and private cloud services are the database services Relational Database Service (RDS) and SimpleDB. These services are working table-based and therefore cannot be used in an all-purpose way like the storage services EBS and S3.

3. Availabilty of cloud storage services

Public cloud provider usually guarantee a level of availability of 99% and more for their services. Amazon declares for EBS that volumes with 20 GB or less data can expect an annual failure rate (AFR) of between 0.1% and 0.5%. But Amazon does not guarantee the availability of the data inside the EC2 Service Level Agreement[2]. The Service Level Agreement of Amazon S3 guarantees 99.999999999% durability and 99.99% availability of objects over a given year for the standard storage method. The Reduced Redundancy Storage is designed to provide 99.99% durability and 99.99% availability of objects over a given year[3]. Google guarantees 99.9% availability and operability of the Google Storage service and the developers APIs in any

[1] `http://code.google.com/p/s3fs/`

[2] `http://aws.amazon.com/ec2-sla/`

[3] `http://aws.amazon.com/s3-sla/`

Table 1. Different ways to store data inside the AWS and compatible services

	Instance storage	EBS	S3	RDS	SimpleDB
Type of storage	block-based	block-based	object-based	table-based	table-based
Persistent (instance-independent)	no	yes	yes	no	yes
Access independent of instances	no	no	yes	yes	yes
Connectable via software RAID	no	yes	no	no	no
Any filesystem supported	not at boot partition	yes	none	none	none
Public cloud services	AWS EC2	AWS EBS	AWS S3, Google Storage, Host Europe Cloud Storage	AWS RDS	AWS SimpleDB
Private cloud services	Eucalyptus, Nimbus OpenNebula, OpenStack Compute	Storage Controller (Eucalyptus)	Walrus (Eucalyptus), Cumulus (Nimbus), OpenStack Object Storage		M/DB

Table 2. Amazon S3 Service Level Agreement

Monthly Uptime Percentage	Percentage of monthly bill that will be refunded
99.0% – $<$ 99.9%	10% of bill
$<$ 99.0%	25% of bill

Table 3. Google Storage Service Level Agreement

Monthly Uptime Percentage	Percentage of monthly bill that will be refunded
99.0% – $<$ 99.9%	10% of bill
95.0% – $<$ 99.0%	25% of bill
$<$ 95.0%	50% of bill

billing month[4]. Host Europe declares for its Cloud Storage[5] service that it is based of an infrastructure with 99.99% availability but Host Europe cannot guarantee any availability.

In the case of a data loss, when using S3 or Google Storage, the customer receives a service credit with is credited back to his account. The value of the service credit depends of the monthly uptime percentage (see table 2 and 3). For Amazon and Google, service credits are only applied against future service payments and are not entitled to any refund or other payment from the provider.

When using private cloud services, the level of availability depends on the local installation, its architecture, software and hardware used.

The open source solution Eucalyptus [6] 2.x that allows to run a cloud infrastructure (IaaS) that is API-compatible to EC2, EBS and S3 inside a linux cluster, provides no high-availability (HA) services because it is impossible to run more than just a single instance of the storage services Walrus (S3) and Storage Controller (SC). Eucalyptus 3.x is planed to provide the HA functionality with multiple service instances. The S3-compatible storage service of Nimbus 2.7 [5], called Cumulus [2], supports just a subset of the S3 API and functionality. The OpenStack Object Storage[6] supports a subset of the S3 API and uses a distributed architecture, avoiding a central point of control to provide HA and scalability due to redundancy. Objects are replicated to multiple hardware devices automatically across the cluster. OpenNebula 2.2.1 [7] includes no storage service.

In an abstract view, when focusing availability, a storage service can be considered as something similar to a hdd. If a hdd fails, in most cases, the data is lost. The same happens when a storage service fails. Losing data that is stored by a public cloud provider is not impossible and did already happen. In April 2011, a service crash of Amazon EC2 permanently destroyed some of the customers data inside EBS

[4] `http://code.google.com/apis/storage/docs/pricingandterms.html`

[5] `http://www.hosteurope.de/produkt/Cloud-Storage`

[6] `http://www.openstack.org/projects/storage/`

volumes. In February 2008, Amazon S3 and EC2 suffered a major outage. No data was lost but one geographic location (availability zone) was unreachable for approximately two hours. This experiences make the development of a storage solution reasonable, that is based of several different cloud services, to improve availability.

A well established method to improve the availability of local storage is a Redundant Array of Independent Disks (RAID) that connects several physical or virtual storage devices to a virtual storage device with a bigger size and/or a better availability.

When using instance storage or EBS volumes, a high-availability and redundant storage can be realized with distributed filesystems like Ceph or GlusterFS and it is possible to connect volumes via software RAID. Using a distributed filesystem with a S3-compatible cloud service may be possible when using s3fs but this seems not to be a robust solution with an adequate performance. A promising approach is using several S3-compatible services as a RAID that provides a better level of availability.

In September 2010 we started to design and implement the Octopus Cloud Storage Service[7], a software service that allows to connect multiple different public and private cloud storage services as a RAID-1 (mirroring without parity or striping).

4. Octopus Design

Octopus is designed to run as a service inside a cloud platform (PaaS) like the Google App Engine (see figure 4). This design offers several interesting choices to deploy the software as a service (SaaS) using public or private clouds. The tool may be installed and run either on the public Google cloud platform or on private platform services like AppScale [4] [3] and typhoonAE[8]. Octopus has been written in Python using JavaScript (jQuery) and the web framework Django. For the communication with cloud services that are API-compatible to the AWS, Octopus uses the library boto, a python interface which is open source. There is a limited but powerful set of APIs inside the App Engine. Octopus uses the persistent Datastore BigTable (with a SQL-like query language called GQL) to store the customers imported credentials encrypted. The user management and authentication is based on Google accounts.

The customers can import their credentials to S3-compatible services. Octopus checks if a bucket with the naming scheme `octopus_storage-at-<username>` exists. If not, the bucket will be created and the customers can upload files (objects) with one click to the connected storage services into the Octopus buckets. Figure 4 demonstrates the steps to upload an object. After the customers login, his client requests (1) the Octopus website with the HTML form and the list of objects. The object list is requested (2) from

[7] `http://code.google.com/p/octopuscloud/`

[8] `http://code.google.com/p/typhoonae/`

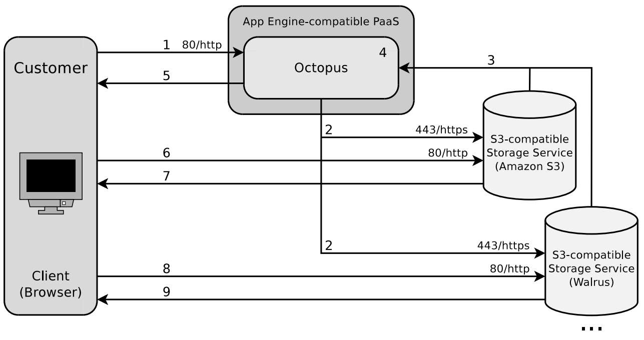
App Engine-compatible PaaS
Customer
Octopus
1
80/http
4
5
3
2
443/https
S3-compatible Storage Service (Amazon S3)
6
80/http
7
2
443/https
S3-compatible Storage Service (Walrus)
8
80/http
9
Client (Browser)
...

Figure 1. Direct upload from the customers browser to the storage services.

the storage services and transferred (3) to Octopus. The synchronicity of the objects is checked (4) by Octopus using checksums. All S3-compatible store a MD5 checksum for each object. These checksums are transferred automatically when a list of objects is requested and they allow to verify simple if the objects located at the different storage services are synchronized. Any time, when a list of objects is requested, Octopus checks if the objects are still synchronized across the storage services. After the synchronicity check, the web site with the HTML form is transferred (5) to the customers browser. After the customer selected the local file and started the upload with the submit button, the object is transferred (6) to the first storage service. If the upload was successful, a confirmation message is send (7) back to the browser. Step 6 and 7 are repeated for each additional storage service used.

Parallel uploads are impossible because inside the App Engine the web applications are not allowed to spawn threads for securities reasons.

Because each object is transferred directly from the customers browser to all connected storage services, the amount of data that need to be transferred, increases linear with each additional storage service used. Therefore, the use of multiple storage services leads to disproportionately long transfer times.

5. Lessons learned

The basic functionality of all S3 storage services are identical, but the exact behaviour is often slightly different. Google Storage and Amazon S3 for example always enclose the MD5 checksums of the objects with double quotes. Walrus sends the checksums without any quotes.

Another characteristic of the Walrus service is, that the service attaches faulty data to each object that is transmitted via POST, if no submit-element exists at the end of the HTML message. The submit-element is recognized by Walrus as the end of an object. Is the submit-element missing, Walrus attaches all data to the object until the end of the transfer.

When using Walrus of Eucalyptus 1.6, inside all buckets a defective object called `None` exists automatically. This object cannot be erased and therefore Octopus need to ignore it.

When using Google Storage, the submit-button, that is used to send the objects, may contain no `name`-attribute. Otherwise an error `InvalidPolicyDocument` occurs. S3 and Walrus in contrast ignore the `name`-attribute of the submit-button.

When using S3 and Walrus, it is possible to define the value `redirect` or alternatively `success_action_redirect` inside the HTML form and the policy document[9]. With this values it is possible to define where the browser will be redirected if the upload was successful. Google Storage only interprets `success_action_redirect` and ignores `redirect`.

A useful extension of Octopus would implement a kind of proxy service, in which the customers' objects are stored initially by Octopus and then redistributed to the S3-compatible storage services. Such a proxy would prevent multiple transfers of the same objects from the customers browser (see figure 5). The Google Datastore is not suitable as a buffer because objects there have a maximum size of 1 MB. Alternatively, Google offers the Blobstore which allows the persistent storage of data in form of so-called blobs that have a maximum size of 2 GB. Objects can be uploaded to the Blobstore via POST and then accessed from App Engine web applications directly. Another possible proxy solution could base of Google Storage because objects within Google Storage can also be accessed directly by web applications inside the App Engine, which is a prerequisite for the transmission via POST to the other storage services. A disadvantage of the Blobstore and Google Storage scenarios is that outgoing data connections from the App Engine are limited to 1 MB per transmission.

6. Next Steps

Supporting Cumulus from the Nimbus project was not successful yet because this storage service does not support uploads via POST.

Octopus can check for synchronicity, but the implementation of an automatic repair routine when the check for synchronicity failed, still need to be done.

If an adequate proxy for Octopus is found, it shall be evaluated. Such a proxy for Octopus could help to avoid multiple uploads from the browser to the storage services.

The implementation of a RAID-5 (block-level striping with distributed parity) mode is a future goal. Advantages would be that no provider has a full (working) copy of the customers data, which improves security, and if a provider is not operational any more, the customers data is still available.

[9] The policy document contains i.a the object name, a content description, the bucket and the destination address to which the browser will be redirected if the upload was successful.

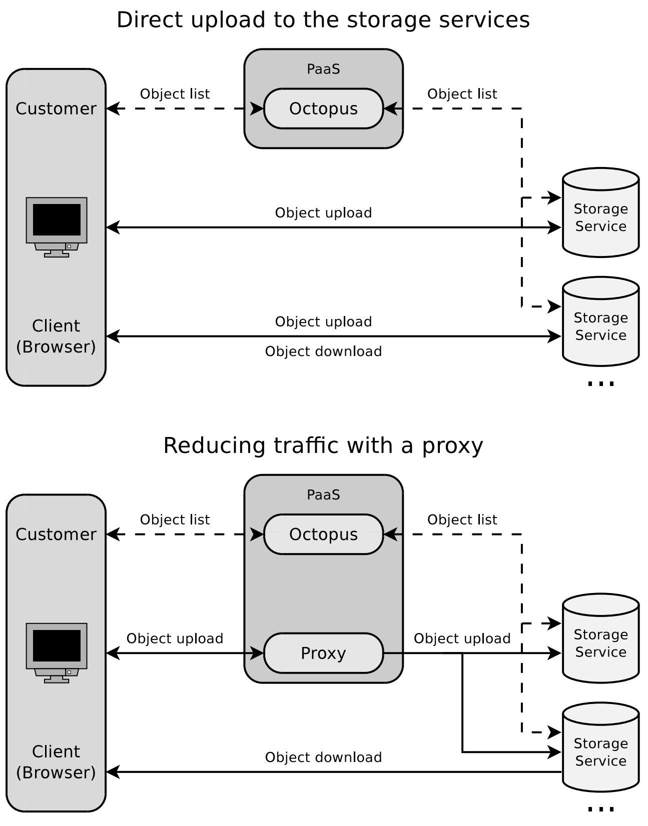

Figure 2. Object upload with and without a proxy.

7. Related Work

Since February 2011, at the University of Kent, a secure front-end to S3-compatible resources, called proxyS3[10], is under development. It's focus is federated identity management access to cloud storage resources and the ability of service customers to grant access to their cloud resources to other customers. The service is based of PHP, MySQL, LDAP and the Apache web server.

An attempt to improve private cloud S3 services, based of Walrus (Eucalyptus) with parallel file systems to leverage their benefits like scalable performance and the POSIX interface with partial reads and writes, the Project pWalrus [1] at Carnegie Mellon University implemented some changes to Walrus. As a result, pWalrus can work with HDFS by accessing it through FUSE. A disadvantage of pWalrus is that adjustments to the source code of Walrus and thus are even at the services' installation itself is necessary, which is not always possible or desirable. Therefore the simultaneous use of public and private storage services with pWalrus is impossible.

[10] `https://authz.tas3.kent.ac.uk:5443/proxyS3/`

Popular services like the web-based file hosting service Dropbox[11] base of S3 and refine the Amazon service. These proprietary services cannot be used with private clouds and self owned credentials for storage services.

8. Conclusion

Object-based storage services can used by the Octopus cloud storage system to create a redundant storage solution that is virtually connected to a RAID with mirroring. As a result, the availability of data and the dependence to individual service providers will be reduced.

References

[1] Yoshihisa Abe and Garth Gibson. pWalrus: Towards Better Integration of Parallel File Systems into Cloud Storage. In *IASDS10: Workshop on Interfaces and Abstractions for Scientific Data Storage*, 2010.

[2] John Bresnahan, David LaBissoniere, Tim Freeman, and Kate Keahey. Cumulus: An Open Source Storage Cloud for Science. San Jose, CA, 2011.

[3] Chris Bunch, Navraj Chohan, Chandra Krintz, Jovan Chohan, Jonathan Kupferman, Puneet Lakhina, Yiming Li, and Yoshihide Nomura. An Evaluation of Distributed Datastores Using the AppScale Cloud Platform. In *IEEE Cloud10: International Conference on Cloud Computing*, July 2010.

[4] Navraj Chohan, Chris Bunch, Sydney Pang, Chandra Krintz, Nagy Mostafa, Sunil Soman, and Rich Wolski. AppScale: Scalable and Open AppEngine Application Development and Deployment . First International Conference on Cloud Computing, 2009.

[5] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes. Sky Computing. *Internet Computing*, 13(5):43–51, September 2009. ISSN 1089-7801.

[6] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The Eucalyptus Open-source Cloud-computing System. In *CCA'08: Proceedings of Cloud Computing and Its Applications workshop*, 2008.

[7] Borja Sotomayor, Rubén S. Montero, Ignacio M. Llorente, and Ian Foster. Virtual Infrastructure Management in Private and Hybrid Clouds. *Internet Computing*, 13(5):14–22, September 2009. ISSN 1089-7801.

[11] `http://www.dropbox.com`