Storage CloudSim
A Simulation Environment for Cloud Object Storage Infrastructures

Tobias Sturm, Foued Jrad and Achim Streit
Steinbuch Centre for Computing (SCC)
Department of Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany

Keywords: Cloud Computing, Storage as a Service, STaaS, Cloud Broker, CloudSim.

Abstract: Since Cloud services are billed by the pay-as-you-go principle, organizations can save huge investment costs. Hence, they want to know, what costs will arise by the usage of those services. On the other hand, Cloud providers want to provide the best-matching hardware configurations for different use-cases. Therefore, CloudSim a popular event-based framework by Calheiros et al., was developed to model and simulate the usage of IaaS (Infrastructure-as-a-Service) Clouds. Metrics like usage costs, resource utilization and energy consumption can be also investigated using CloudSim. But this favored simulation framework does not provide any mechanisms to simulate todays object storage-based Cloud services (STaaS, Storage-as-a-Service). In this paper, we propose a storage extension for CloudSim to enable the simulations of STaaS-components. Interactions between users and the modeled STaaS Clouds are inspired by the CDMI (*Cloud Data Management Interface*) standard. In order to validate our extension, we evaluated the resource utilization and costs that arise by the usage of STaaS Clouds based on different simulation scenarios.

1 INTRODUCTION

Cloud computing is one of the most emerging technologies of the past few years. Start-ups, big companies and the scientific community explore the benefits of this kind of resource utilization. These Cloud users exploit the advantages of less configuration overhead, less investments, less operational costs, highly flexible systems that scale out to large systems, or scale in to a minimal resource provisioning, depending on the current needs.

The main benefit from using Cloud services is to reduce the cost for computation and storage by sourcing these services out to a third-party provider. The crucial questions are about the costs and the performance of the usage. Users want to know, how much money they are going to spend, if they switch to a Cloud solution. On the other side, providers have to know, what is the best hardware constellation and configuration for them. These questions can be answered by simulating Cloud environments.

CloudSim (Calheiros et al., 2009) is a popular simulation environment, which offers capabilities to simulate computing Clouds that provision computing infrastructures as a service. The focus of *CloudSim* is on the modeling of so-called *Cloudlets* which are jobs, that can be scheduled on VMs (Virtual Machines). Models for hard disks, SAN and files are already included in *CloudSim*. However, capable interfaces to simulate object STaaS are missing, as well as a fine-grained model for the size of files.

Object storage systems pool multiple physical devices together and provide one logical medium to store and retrieve many different pieces of information called objects. Besides user data, an object contains so called *metadata*, like timestamps or number of replicas. Objects can be accessed by their server-wide unique ID and can be created, updated, read (complete or partially) by all authorized clients (Azagury et al., 2003). Containers are virtual collections of objects and may be hierarchical like folders on conventional file systems (CDMI, 2012).

Considering that there is no known simulation environment for object STaaS, which includes a proper modeling of the interfaces, this work extends the existing framework *CloudSim* by adding components for an accurate modeling of object storage.

The rest of this paper is organized as follows: In the next section we present the existing simulation framework CloudSim and review the fundamentals of object storage, before we discuss prior works related to STaaS Cloud simulation. In Section 3 we propose

Sturm T., Jrad F. and Streit A..
Storage CloudSim - A Simulation Environment for Cloud Object Storage Infrastructures.
DOI: 10.5220/0004956401860192
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 186-192
ISBN: 978-989-758-019-2

the extension to enable STaaS Cloud simulation, followed by Section 4, in which we describe a set of experiments we investigated on the extended simulation framework. Section 5 concludes the paper with a brief summary and describes our future research directions.

2 RELATED WORK

In this section we describe the existing *CloudSim* framework, explain the needed changes to simulate STaaS Clouds and then present related works to STaaS simulation.

2.1 CloudSim

CloudSim is a time discrete simulation framework for Cloud computing. The framework consists of three layers as described by the authors (Calheiros et al., 2011) (from bottom to top):

1. *Core Simulation Engine:* Queuing and processing of events, management of Cloud system entities such as host, VMs, brokers, etc.
2. *CloudSim:* Representation of network topology, message delays, VM provisioning, CPU, storage and memory allocation, etc.
3. *User code:* General configuration such as Cloud scenarios and user requirements, User Broker, etc.

Users of the framework can either modify the top layer to model simulation scenarios, or extend the second layer, to test different allocation policies in a Cloud system. The *user code* layer defines so-called *cloudlets* that define a specific amount of computation requirements similar to a Cloud job. Datacenter Brokers forward the cloudlets to Cloud providers and manage and monitor their execution. These jobs are dispatched on available VMs by the *CloudSim* layer. Communication between the entities is done via message events which are sent to the *core simulation engine*, which handles all events in the correct order and manages the simulated time. Events between two remote entities are automatically delayed, if the network topology is predefined (Garg and Buyya, 2011).

CloudSim can simulate SAN storage, hard drives and files, that are stored on hard drives directly or via SAN storage. But their modeling lacks support for object storage as follows:

- *File size magnitude:* CloudSim models the file size in megabyte, but 90% of all web objects fit within 16KB (Dukkipati et al., 2010), which means that a content delivery scenario scenario can not be modeled accurately.
- *Hard drive models:* The hard drive models do not provide all metrics that are required to model the read and write durations accurately.
- *No storage controller:* CloudSim does not offer a controller that determines the storage location of objects.
- *No appropriate object storage interface:* No model for any STaaS interface, as for example CDMI.

2.2 STaaS Simulation

The only work known to us that deals with storage modeling in CloudSim was performed by Zhao and Long (Long and Zhao, 2012). Even after intensive search we have not found any other simulation environment that suits STaaS. Zhao and Long propose an extension of the existing Datacenter model by introducing a *Replica Catalog* and *Block Catalog* (One piece of information is striped and distributed on different physical locations to gain a parallelized and therefore faster access, catalogs can provides a query interface to retrieve those locations) to benchmark different configurations. All operations are started from within a Cloudlet, which means that the VM is the requesting entity. Overall their work presents a mixture of datacenters that offer computing power and storage capacities. This work in contrast focuses on the strict separation of these two capabilities, as currently done in real-world IaaS providers.

In a previous paper (Jrad et al., 2012), we investigated the use of multiple IaaS Compute providers interfaced by the Open Cloud Computing Interface[1] (OCCI). One user assigns a *Meta Broker* to choose between multiple available Clouds based on *Service Level Agreements* (SLA). We provide similar SLA mechanisms for STaaS providers like: Available capacity, bandwidth of Cloud interfaces, restrictions on file sizes, etc.

3 STORAGE CLOUDSIM

We extended the existing simulation framework *CloudSim* to enable the modeling of STaaS Clouds. Therefore we provide additional models in the user code layer and the layer beneath as we can see in Figure 1. *CloudSim* consists of three layers. Our additions to the framework build on top of the *CloudSim Core Engine*, which is responsible for the time discrete message passing between simulation entities. We provide new models for hard disks, servers and

[1]http://occi-wg.org

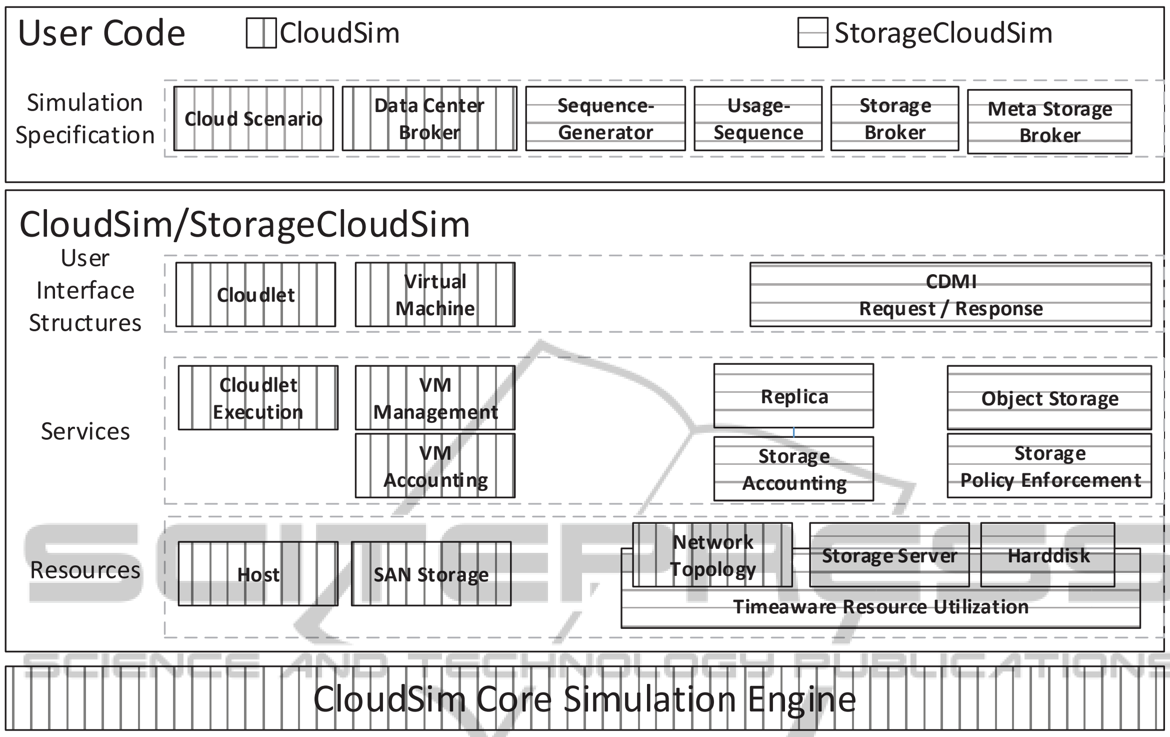

Figure 1: Storage CloudSim architecture Overview.

a more accurate model resource utilization over time. Beside data replication and accounting, we implemented different storage policies enforcements. Users of the simulation framework can generate sequences of CRUD (Create, Read, Update, Delete) operations on objects and containers and implement different multi-Cloud broker policies.

3.1 User Code Extension

The usage of STaaS components is modeled as *usage sequence* - a sequence of CRUD operations on both containers and objects. The order of these operations is fixed (an object can only be put into a container if the container was created before), but the execution of two sequences never interfere or depend on each other. We provide tools for automatically generating such sequences in a way that they are random, defined by statistical distribution constraints[2] and are repeatable by storing them in XML files.

Each sequence is attached to a SLA requirement, where hard constraints and soft requirements can be defined. The hard constraints serve to exclude STaaS Clouds that do not fulfill required features (e.g. capability to store user metadata, available capacity, etc.), whereas the soft requirements lead to a scoring of providers whenever there is more than one provider that fulfills all hard requirements in order to choose the one with the highest score (e.g. prefer providers with the lowest storage cost).

The sequences and their requirements are forwarded to the *Meta Storage Broker*, which query characteristics and capabilities of each available STaaS provider and chooses the best suitable provider for every sequence and dispatches each one to a *Storage Broker*. The former is acting as a proxy for a single user Cloud interaction by dispatching the requests of the sequence and storing all responses from the Cloud for later analysis.

The entire communication between broker and Cloud is wrapped in CDMI calls, which is an industry-wide standard for accessing STaaS Clouds.

3.2 STaaS Provider Models

Every Cloud is described by some characteristics as for example networking capabilities, supported operations, billing information and hardware specifications or limits as for example maximum size of an entity or maximum number of objects in one container. Each instance features a CDMI endpoint, a set of servers and a sets of hard disks. On top of the

[2]size of objects, delay between requests, balance between read and write operations etc.

Table 1: Defined Usage Sequence Patterns.

Name	**Pattern A**	**Pattern B**
Traffic / Sequence	1GB .. 100GB (normal distributed)	1MB .. 16GB (gamma distribution 3,2)
Size / Object	1MB..100GB (normal distributed)	1MB .. 1GB (normal distributed)
Read Write Ratio	0:1	5:11
Hard Constraints	no container size limit sufficient boundary for object size limit enough storage capacity	enough storage capacity
Soft Constraints	low upload and storage costs	all costs low

hardware models is the representation of the logical organization such as each object might have multiple replica, called *blob*s. Communication between brokers and STaaS Clouds is accomplished through CMDI messages, which organize data as *CDMI Objects* whithin Containers (both described as entity in the following). Every object must be stored in one container - containers can not be nested. Each user of the Cloud (represented by a broker) can only see his entities. They can be accessed via their Cloud-wide unique *CDMI ID* or via a human readable name that was chosen by the user. Entities feature so-called *Metadata*, which is a key-value storage, which holds for example the size of an entity, date of creation, number of desired replica or date of last-access. Depending on the capabilities of a Cloud, a user can modify or extend these metadata (e.g. the number of desired replicas).

The storage controller matches certain policies, such as the physical distribution of blobs that belong to one object (replica). A blob can only be stored on a disk, when the disk holds no other blob of the same object, so that the failure of a single disk does not cause the loss of data if replication is used.

Since read, write and update operations involve the transport of data through the internal network, we modeled the payload size of these operations on the most fine-grained level (bytes). As we model the storage of data to the level of disks, we have to model also the data transport between hard disks. Therefore, we assigned a bandwidth to network and disk connections (see Figure 2) inside the STaaS CLoud. The complete data path begins with the broker, which sends a CDMI request to the STaaS provider. The data is then sent to the storage server and then to the hard disk. Every segment of the data path has a certain bandwidth[3] to introduce a certain delay. In addition we model the current utilization of these path sections.

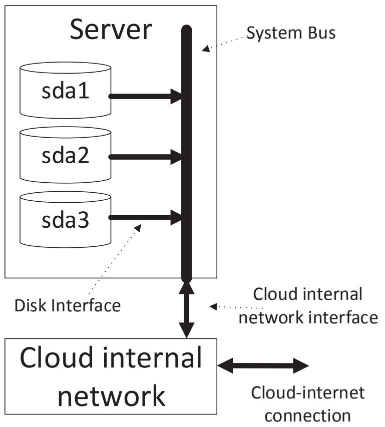

Figure 2: IO limitations on the data path.

The duration of each operation depends on the bandwidth of the path section that has the lowest available bandwidth. Since concurrent operations might interleave or overlap, the available bandwidth might change during one operation. Thus, the total duration of each operation depends on the least available bandwidth on the path over time and thus the following terms must be fulfilled:

$$payloadSize = \int_{t_{start}}^{t_{end}} \operatorname*{argmin}_{i} availableBandwidth_i(t)\,\mathrm{d}t. \tag{1}$$

and

$$\bigwedge_{i \in pathSections, t} availableBandwidth_i(t) < bandwidthLimit_i \tag{2}$$

Read and write operations are dispatched by first-scheduled-first-served strategy.

4 SIMULATION EXPERIMENTS

4.1 Meta Broker Evaluation

We conducted two sets of experiments, one with only one StaaS Cloud provider (*Amazon S3*[4]) and another set with three providers, each with a mixture of patterns of *usage sequence* (see Table 1). Pattern A features only uploads of large data chunks. Therefore the costs for downloads from the Cloud have no impact. The other pattern features up- and downloads.

[3]uni-directional bandwidth specification is possible.
[4]http://aws.amazon.com/de/s3

Table 2: Modeled Cloud Specifications.

	S3	Private	Swift
max. Object size	unlimited	16GB	unlimited
write rate	156MB/s	64MB/s	156MB/s
read rate	156MB/s	156MB/s	156MB/s
capacity	2TB	1TB	2TB
read latency	8.5ms	9ms	8.5ms
write latency	9.5ms	11ms	9.5ms
capacity	72TB	3TB	32TB
# replica	3	3	1
$/GB stored	0.05$	0.04$	0.01$
$/GB up	0.0002$	0.0002$	0.0002$
$/GB down	0.01$	0.0002$	0.1$

We modeled three different Cloud providers *Amazon S3*, a private Cloud and a *Swift Cloud*[5], see Table 2), each with different pricing, storage capacity and throughput. The private Cloud has a hard 16GB limit for objects, but offers the lowest prices.

The meta broker checks all hard constraints against every available Cloud and then calculates a score for each potential Cloud. The score in our example is calculated by the inverse of the relevant costs, e.g.

$$score = \frac{1}{costsperstoredGB} + \frac{1}{costsperuploadedGB}$$

for sequences of pattern A. One hard constraint that is shared among both patterns is the requirement for sufficient storage capacity. Since users in most use cases do not know the exact volume of data they want to store or the size of every single object, we added a 25% uncertainty to these values. A sequence is declined, if no Cloud provider can fulfill all hard requirements.

We can verify the correct behavior of the meta broker by plotting the used storage of each Cloud provider over time, as in Figure 3. Sequences that do not contain objects that are bigger than 16GB can be dispatched on the private Cloud, because it is the cheapest STaaS provider. Sequences that contain objects that are bigger than 16GB will be dispatched on the *Swift Cloud* because it is cheaper than *Amazon S3*, which is never chosen for any sequence. We can see that the *Swift Cloud* usage rises faster since all big objects (greater than 16GB) are stored there. We conducted the above described experiments multiple times and made sure that the randomly generated input sequences did not contain any artifacts. A mixture of both sequence types was used (2486 sequences of type A and 2514 of type B). First, we observed that both experiments (single and multi-Cloud) with 50 and 500 sequences succeeded, but the experiment with 5000 mixed sequences can not be executed on the single-Cloud environment completely (4150 sequences are declined due to capacity exhaustion). If we observe the number of succeeded and failed requests as well as the declined sequences, we observe that at one point of this experiment, the number of succeeded requests stops to rise, the failed requests spike and the number of declined sequences starts to rise. These are declined because the hard requirement for sufficient storage can not be fulfilled. In total 15 requests fail in the single Cloud experiment with 5000 mixed sequences. Three reasons (or any combination of them) explain this behavior:

- *Inaccurate SLA*: *UsageSequence* requests less capacity than it will consume. Cloud storage may be used by 99.99%, but it accepts the request, because the requested size fits into the remaining storage.
- *Unsatisfiable policies*: One reason could be the policy, that forces the storage controller to locate *StorageBlobs* of one object not to be on the same disk. Maybe there is enough physical storage available, but it is not distributed as required.
- *Concurrent access*: Clouds use the currently available storage, when responding to a Cloud discovery request. Requests do not allocate any storage space. Two storage capacity requests can be met, but one of them fills all of the remaining storage, so the other request can not be fulfilled.

One of the most interesting points are the costs that are billed when the sequences are run on the different constellations. We see that multi-Cloud experiments generate less costs than the single Cloud experiment for the runs with 50 and 500 sequences (60% less costs in the 500 sequence experiment, see Table 3). This can be explained by the capacity of the private Cloud, which provides resources for a fraction

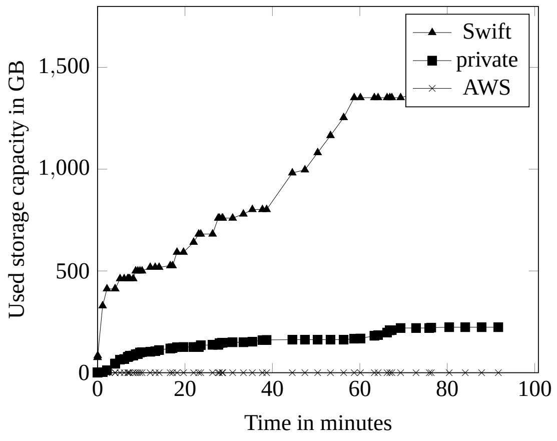

Figure 3: Storage usage for the modeled STaaS Clouds.

[5]http://swift.openstack.org

Table 3: Average Price per accepted Sequence.

# of input Seq.	Single Cloud	Multi-Cloud
50	0.0150$	0.0070$
500	0.0149$	0.0058$
5000	0.0149$	0.0128$

Table 4: Total Costs for 250 Input Sequences.

	Single Cloud	Multi-Cloud
Pattern A	2.62$	2.62$
Pattern B	4.82$	0.30$

of the costs of the *Amazon S3* Cloud. But the total costs for the 5000 sequence experiment shows that the multi-Cloud constellation can generate more costs, which is in the interest of the user, because in this case more sequences are processed, compared to the single Cloud experiment. However, the multi-Cloud setup provides lower prices per accepted sequence in each experiment, as seen in Table 3.

4.2 Effect of the Used Sequence Type

In order to study the impact of the chosen sequence type, we conducted four experiments each of them with a different sequence type and on a single Cloud or a multi-Cloud environment. As we submitted 250 sequences, no sequence was rejected in any experiment. Table 4 shows that the multi-Cloud variant generates less or the same costs as the single Cloud variant. The costs for single and multi-Cloud for the sequences of type A is the same, whereas the difference for sequences of type B is 4.52$ (saving of factor 16). This can be explained by the fact, that the sequences of type A have tighter restriction on the SLA: The minimal accepted value for the maximum object size depends on the largest object of the sequence, which can be between 1 and 100GB. Thus, the majority of A sequences can not be scheduled on the private Cloud, which is deciding for the costs for B sequences

5 CONCLUSIONS

We extended the popular Cloud simulation framework *CloudSim* to enable the simulation of STaaS Clouds. The concurrent use of resources is modeled accurately in order to gain realistic simulation results. Detailed models for storage disks, servers and storage controller are provided and can be easily extended. Monitoring allows users to investigate a broad spectrum of metrics from each simulated STaaS Cloud as well as from the user's perspective.

Different access patterns can be modeled and used to generate sequences of requests. These sequences can be stored and read from file to provide random, but repeatable experiments. In addition, different SLAs can be modeled, such as certain capabilities of a Cloud or some restrictions. A *Meta Broker* enables simulations of multiple Clouds, where different *usage sequences* can be dispatched on the best-matching Clouds. Compared to single Cloud usage users can save huge costs if the SLAs of the request sequences are not very strict.

In a future work, we will investigate more realistic pricing models (no linear price functions) and more complex SLA mechanisms. The framework will be extended to enable the simulation of the storage allocation policies and mechanisms used in current STaaS Clouds, for example mechanisms to prevent bit rotting of failover reactions to outages of hardware. We will also introduce more heuristic parameters to make the simulation more realistic. Also brokers could not only compare requirements against the static SLA but could also calculate metrics based on previous bandwidth usage and then decide which Cloud to use for further requests.

REFERENCES

Azagury, A., Dreizin, V., Factor, M., Henis, E., Naor, D., Rinetzky, N., Rodeh, O., Satran, J., Tavory, A., and Yerushalmi, L. (2003). Towards an object store. In *In Proceedings of the 20th IEEE Conference on Mass Storage Systems and Technologies*, pages 165–176.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50.

Calheiros, R. N., Ranjan, R., De Rose, C. A., and Buyya, R. (2009). Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv:0903.2525*.

CDMI (2012). *Cloud Data Management Interface (CDMI) Version 1.0.2*.

Dukkipati, N., Refice, T., Cheng, Y., Chu, J., Herbert, T., Agarwal, A., Jain, A., and Sutin, N. (2010). An argument for increasing tcp's initial congestion window.

Garg, S. K. and Buyya, R. (2011). Networkcloudsim: Modelling parallel applications in cloud simulations. In *Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing*, UCC '11, pages 105–113, Washington, DC, USA. IEEE Computer Society.

Jrad, F., Tao, J., and Streit, A. (2012). Sla based service brokering in intercloud environments. In *Proceedings of the International Conference on Cloud Computing and Services Science CLOSER2012*, pages 76–81, Porto, Portugal.

Long, S. and Zhao, Y. (2012). A toolkit for modeling and simulating cloud data storage: An extension to cloudsim. In *Control Engineering and Communication Technology (ICCECT), 2012 International Conference on*, pages 597–600.