# Evaluation of the Huawei UDS cloud storage system for CERN specific data

# Evaluation of the Huawei UDS cloud storage system for CERN specific data

**M Zotes Resines**[1], **S S Heikkila**[2], **D Duellmann**[3], **G Adde**[4], **R Toebbicke**[5], **J Hughes**[6], **L Wang**[7]

[1−5]CERN (European Organization for Nuclear Research), Geneva, Switzerland
[6]Huawei Technologies, Santa Clara, California, USA
[7]IHEP (Institute of High Energy Physics), Shijingshan District, Beijing, China

E-mail: [1]`maitane.zotes@cern.ch`, [2]`seppo.heikkila@cern.ch`, [3]`dirk.duellmann@cern.ch`, [4]`geoffray.adde@cern.ch`, [5]`rainer.toebbicke@cern.ch`, [6]`jphughes@mac.com`, [7]`wanglu@ihep.ac.cn`

**Abstract.**
Cloud storage is an emerging architecture aiming to provide increased scalability and access performance, compared to more traditional solutions. CERN is evaluating this promise using Huawei UDS and OpenStack SWIFT storage deployments, focusing on the needs of high-energy physics. Both deployed setups implement S3, one of the protocols that are emerging as a standard in the cloud storage market. A set of client machines is used to generate I/O load patterns to evaluate the storage system performance. The presented read and write test results indicate scalability both in metadata and data perspectives. Futher the Huawei UDS cloud storage is shown to be able to recover from a major failure of losing 16 disks. Both cloud storages are finally demonstrated to function as back-end storage systems to a filesystem, which is used to deliver high energy physics software.

## 1. Introduction

CERN (the *European Organization for Nuclear Research*) is the largest research centre for particle physics in the world and home of the LHC (*Large Hadron Collider*), a 27 kilometre long circular accelerator located 100 metres below the surface. Four large scale experiments along this ring create every year tens of petabytes of data, which need to be reliably stored for analysis in the CERN computing centre and many partner sites in the Worldwide LHC Computing Grid (WLCG). Physics data is still usually today stored with custom storage solutions, which have been developed for this purpose within the HEP (*High Energy Physics*) community.

The recently emerged cloud storage architecture and its implementations may provide scalable and potentially more cost effective alternatives to current systems [4]. Native cloud storage systems, such as the Amazon S3 [1], are typically based on a distributed key-value store, which divides the storage namespace up into independent units called buckets. This namespace partitioning increases scalability by insuring that access to a bucket is unaffected by data access in other buckets of the same system. In addition the internal replication and distribution of data replicas over different storage components provides fault-tolerance and additional read performance: multiple data copies are available to correct storage media failures and to serve multiple concurrent clients.

The goal of this paper is to present the study of one implementation of the cloud storage concept, the Huawei UDS[1] (*Universal Distributed Storage*) system, with typical CERN data access patterns and volumes. Huawei UDS implements S3 (*Simple Storage Service*) as an access protocol, which has become a de-facto standard across many cloud storage systems. S3 extends HTTP protocol with an authorisation system and provides functions to manage storage system entities such as files and buckets via HTTP REST request, e.g. GET, PUT or DELETE.

Unlike some other cloud storage systems, UDS has full support for multi byte-range reads that are also known as vector reads. Each vector read returns only the specific ranges of bytes requested and hence allows to avoid full file transfer. This is expected to be especially relevant for physics analysis use cases, which are characterised by sparse and non-sequential data access.

The paper is structured as follows. Chapter 2 describes the UDS system design and characteristics. Chapter 3 presents the features and deployment of the benchmark used for evaluating the cloud storage systems. Chapter 4 summarises benchmark and application test results. Finally, Chapter 5 draws conclusions from the study results and outlines the future workplan.

## 2. UDS cloud storage

This chapter gives an overview of the Huawei UDS massive data storage system. The focus is on the hardware and features of the cloud storage setup that is used at CERN.

### *2.1. Hardware components*

The evaluated UDS setup is physically located in the CERN computing centre and placed in three racks with a total storage capacity of 768 terabytes. The system has two main functional components:

- Control Nodes (OSC). The OSCs are user-facing frontend nodes which implement the S3 access protocol and delegate the storage functions to storage nodes. OSC nodes are in charge of scheduling, distributing and retrieving the data of the storage nodes. They are the only components exposed to the outside world. The evaluated setup at CERN consist of seven OSCs.
- Storage Nodes (SOD). The SODs are independent storage nodes, which manage data and metadata on local hard disks. In the tested system SOD nodes consists of a 2 terabyte disk coupled to a dedicated ARM processor and memory. The total setup consists of 384 disk-processor pairs, which are physically grouped in blades of eight SODs. Two of these blades form one chassis within a CERN computer center rack.

The connectivity between these components is provided with three switches. The system is accessed via two 10Gb network connections from a group of CERN-based client nodes.

### *2.2. UDS features*

The UDS is a storage system designed for handling large amounts of data. The stored objects are divided in the OSCs into one megabyte chunks. These chunks are then spread and stored on the storage nodes. The examined UDS setup uses three replicas to ensure the data availability and reliability. Data replicas are distributed to different storage nodes such that a loss of one complete chassis, i.e. 16 storage nodes, will not have impact on data availability.

In case of a disk failure, an automated self-healing mechanism ensures that the data on the faulty disk is handled by the other storage nodes. Corrupted or unavailable data are replaced using the remaining replicas.

[1] http://enterprise.huawei.com/ilink/cnenterprise/download/HW_259595

## 3. Benchmark framework

In the following sections we describe the benchmark setup and application used for our evaluation.

### *3.1. Benchmark design*

We evaluate the cloud storage performance with a S3 benchmark[2] we developed for this purpose. The benchmark framework is implemented in C++ and utilises the ROOT system [3], a widely used scientific data analysis package. A benchmark master process deploys and monitors many parallel client processes via ssh connections on a pool of 21 dedicated client machines. As S3 client library we have utilised the frequently used Amazon AWS Python library[3]. Using the above packages we run a large number of benchmarks in different configurations to measure the aggregated throughput and rate of metadata operations. In addition to basic data upload and download functions we also tested more complex functions such as byte-range read operations and data management functions such as creation and listing of S3 bucket contents.

For all performed benchmarks our benchmark system was collecting key client side metrics to make sure, that our aggregate performance results were not affected by client side bottlenecks. In particular we monitored :

- Operation time: the time in seconds it took each request to complete.
- Transmit rate: the data sending speed to the server in bytes per second.
- Receive rate: the data receiving speed from the server in bytes per second.
- CPU usage: the percentage of the total CPU each client is using.
- Memory usage: the percentage of the total memory each client is using.

### *3.2. Deployment*

Our benchmark software was hosted and distributed among all involved machines via the distributed file system AFS[4], which is used at CERN. Each of the 21 client machines were connected via an 1Gb network link, matching the 20Gb network connectivity (two 10 Gb links) connecting to the UDS server. Each client machine was equipped with 48GB of RAM memory and 24 Intel Xeon 2.27 GHz cores.

## 4. Experiments

In this section we present the experiments performed with the UDS system and the obtained results. Measurement errors estimations are shown as standard deviation among three repeated measurements.

### *4.1. Metadata scalability*

To test specifically the metadata performance of the system we have used very small 4kB files with randomly generated filenames. The small amount of payload data in 4kB allows to avoid any throughput related constraints and stress particularly the metadata handling.

We started by testing data uploads and quickly confirmed that the number of buckets used simultaneously in S3 had the expected performance impact on file creations: with increasing number of buckets, and hence independent namespace providers, we obtained increasing throughput. Figure 1 shows the upload performance with 100 and 2100 buckets. The number of buckets started to limit the performance when the number of parallel uploads increased over 200.

[2] The S3 benchmark software can be requested from the authors.
[3] http://code.google.com/p/awspylib/
[4] http://www.openafs.org/

The number of buckets did not however significantly affect the measured download performance. Maximum performance was already reached using a single bucket with 1,000 files of 4kB. The number of concurrent download processes was increased on each test run. Figure 2 shows that the performance scaled linearly up to a rate of 25,000 files per second. The maximum upload speed for small files is in our setup limited by the capability of frontends to handle the requests. The scalability with increasing number of frontends nodes is examined in more detail in a separate section below.

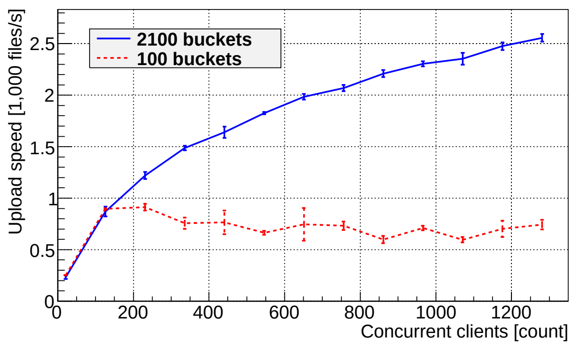

**Figure 1.** Scalability of metadata (uploads of 4kB files)

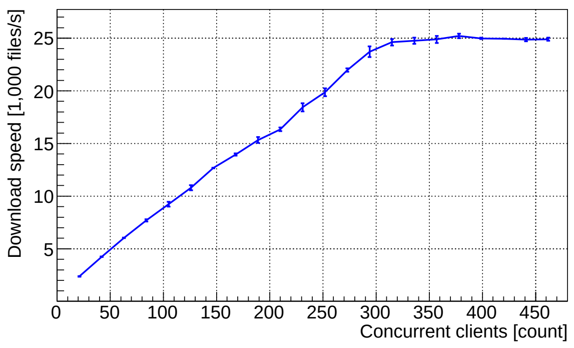

**Figure 2.** Scalability of metadata (downloads of 4kB files)

### *4.2. Data throughput*
In addition to metadata performance we also measured how much data can be transferred in a given amount of time. For these throughput measurements we used larger files of 100MBs. With this payload size the upload performance is not any more strongly affected by the number of concurrently used buckets as rate of metadata operation is several orders lower. As Figure 3 shows, the upload throughputs were almost identical when using 100 and 2100 buckets and we easily filled the available network bandwidth of 20Gb.

Similarly, also the download throughput is largely unaffected by the number of concurrently used buckets. For this reason we could use a single bucket with 100 files of 100MB for downloads. As figure 4 shows, we reached the bandwidth limit already with around 100 processes.

### *4.3. Frontend scalability*
Next we examined frontend scalability up to all seven frontend nodes in our UDS setup. The aim was to verify weather each additional frontend is indeed able to add a similar amount of processing capability to the storage system. For this we repeated the previously presented tests while varying the number of concurrently used frontend nodes.

Figure 5 shows the result with 4kB file downloads. Each frontend adds linearly around 3500 files per second to the total download rate. The achieved maximum 4kB download performance could likely be further increased further by adding more frontend nodes. The frontends were not a limiting factor with 4kB uploads, because already one frontend was able to upload around 800 files per second.

Figure 6 shows the resulting scaling with 100MB uploads. Each additional frontend node was able to upload data around 550MB per second. The network bandwidth limit was however already reached with three frontend nodes, limiting the area in which our setup can confirm linear scalability.

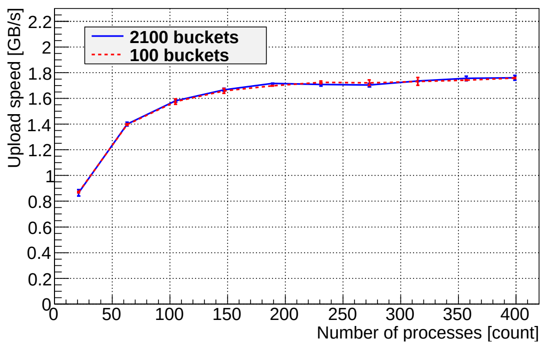

**Figure 3.** Scalability of upload throughput (100MB files)

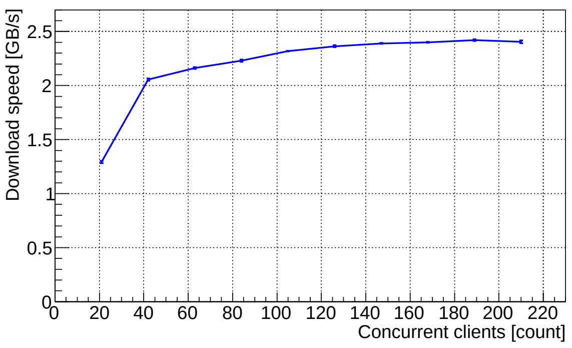

**Figure 4.** Scalability of download throughput (100MB files)

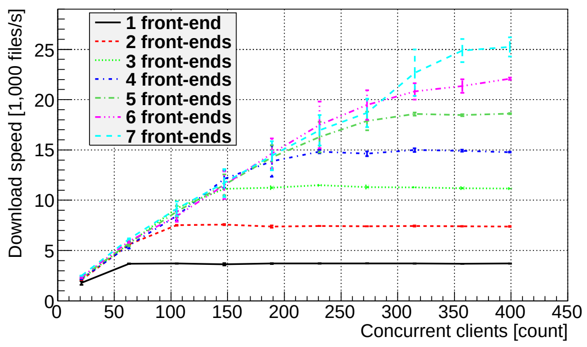

**Figure 5.** Frontend scalability (4kB file downloads)

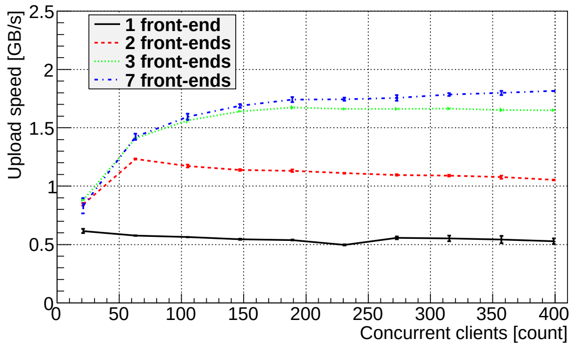

**Figure 6.** Frontend scalability (100MB file uploads)

### *4.4. Impact and recovery from power failures*

This section describes the UDS system response when intentionally forcing a whole chassis of 16 disks to fail. For this test we kept the system actively used while simulating a power failure on one chassis by unplugging its power connection. During the test we performed uploads and downloads with 10MB files for a period of 30 minutes. Figure 7 and figure 8 show the result of the test for write and read operations, respectively. These tests were run first with a fully operational UDS system for 500 seconds until we disconnected the chassis power (first vertical red line). The power was 300 seconds later reconnected (second vertical red line) and the system was left undisturbed.

Figure 7 and Figure 8 show that the affected clients (doing uploads and downloads on this chassis) experienced delays up to around 60 seconds but no errors - all the read and write operations were completed successfully. Further down the UDS automated recovery process, a third vertical red line shows when the first nodes became available again after the chassis rebooted. It took around 200 seconds for the first nodes to be available. At this point we again measured some delays with write operations, while read operations were not affected. The last red line indicates the point when all UDS nodes were fully recovered and available again.

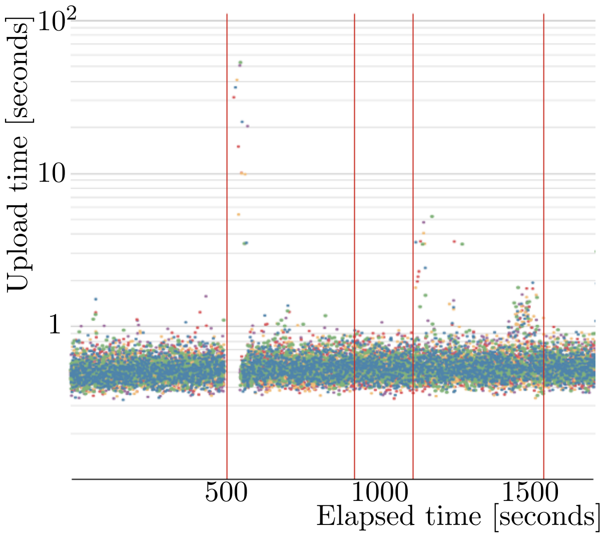

**Figure 7.** Upload times during chassis failure recovery

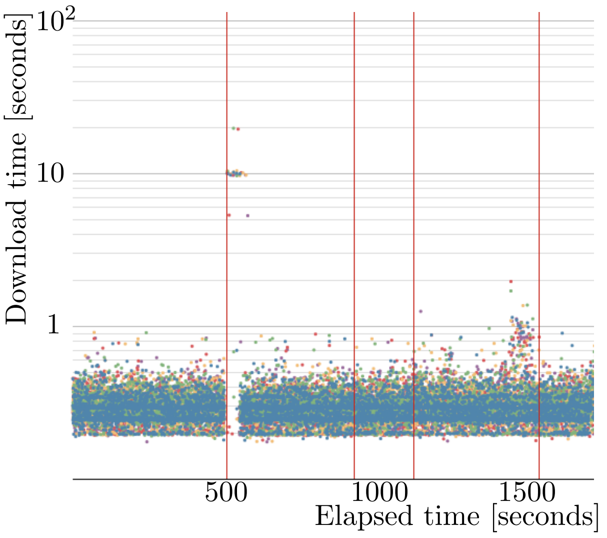

**Figure 8.** Download times during chassis failure recovery

## *4.5. Use as filesystem backend*

In this section we evaluate the ability of cloud storage to serve as storage backend for a file system. We are focussing on CVMFS (*CernVM File System*), as an example, which is a read-only, cached file system that among other applications is widely used to distribute HEP software [2] within the Worldwide LHC Computing Grid (WLCG).

The CVMFS system was recently adapted to store data in multiple buckets in the cloud store in order to achieve the maximum upload performance. For this purpose a Squid[5] proxy server was configured to redirect user file names to corresponding storage names, which include a bucket location, which is deterministically derived from a hash of the user filename. The CVMFS system with cloud storage backends was tested by simulating the publishing step of a software release consisting of 30,000 small files. The files sizes were uniformly distributed from 5kB to 15kB.

The release process using CVMFS consists of two separate steps: in the first step the release coordinator specifies new files for a release. The second publish step then writes the files to the configured backend storage system. In our storage evaluation we hence focused on the performance of the publish step.

Inside the storage backend of CVMFS files are identified by a hash, which is calculated from the actual file content. This allows to rapidly determine if a particular file version has already stored been stored, without the need of a full data transfer. This approach is particularly effective for distributing software since in many cases only small fraction of files changes between consecutive releases.

Figure 9 shows CVMFS upload speeds with different fraction of unchanged files. The UDS back-end was able to publish around 1200 new files per second. The total release speed however more than doubles as the fraction of already previously stored files increases.

A small OpenStack SWIFT setup [5] was also tested as a CVMFS backend. This setup was able to upload around 200 files per second when measuring the metadata upload performance with 4kB files. Running the above publishing tests using CVMFS with OpenStack backend we therefore obtained also a lower publishing rate of 200 files per second as shown in Figure 10.

[5] http://www.squid-cache.org/

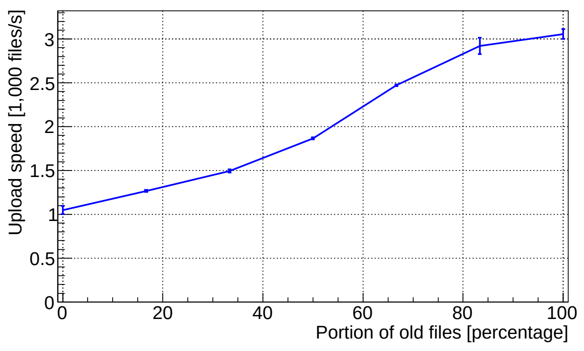

**Figure 9.** CVMFS with UDS back-end

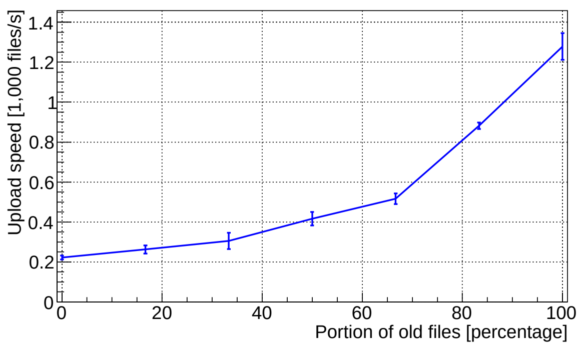

**Figure 10.** CVMFS with SWIFT back-end

## 5. Conclusions

In this study we evaluated a modern cloud storage system in the context of CERN's data storage requirements. The paper focused on the aspects of scalability, reliability and fault tolerance. The evaluation was performed using both synthetic performance benchmarks and a real storage application.

The file uploads and downloads were found to scale with small 4kB files up to 2,500 and 25,000 files per second, respectively. The uploads and downloads were also found to scale with 100MB files until most of the two available 10Gb fibres were utilised. The number of frontends was shown to linearly increase the download speed of small 4kB files and upload throughput of 100MB files.

Concerning the fault-tolerance provided by the UDS system, we demonstrated a transparent recovery after powering off a chassis of 16 disks. We also successfully used Huawei UDS and the OpenStack SWIFT cloud storages as a backend to the CernVM file system. The cloud storage systems behaved as expected and we did not find major problems that would prevent their use as high energy physics data storage.

Planned next steps include the integration with other storage services at CERN and replication tests between cloud storage setups of different vendors and at different sites.

## Acknowledgments

The authors would like to thank the team of engineers from Huawei, who helped to setup and maintain the UDS system at CERN. We also would like to thank the CernVM-FS team, especially René Meusel and Jakob Blomer, for the help with the CVMFS adaptation and tests.

## References

[1] DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, and Vogels W. Dynamo: amazon's highly available key-value store. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, volume 7, 2007.

[2] Blomer J, Aguado-Snchez C, Buncic P, and Harutyunyan A. Distributing LHC application software and conditions databases using the CernVM file system. *Journal of Physics: Conference Series*, 331(4), 2011.

[3] Brun R and Rademakers F. ROOT - an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1), 1997.

[4] Benedict S. Performance issues and performance analysis tools for HPC cloud applications: a survey. *Computing*, 95(2), 2013.

[5] Toor S, Toebbicke R, Zotes Resines M, and Holmgren S. Investigating an open source cloud storage infrastructure for CERN-specific data analysis. In *Proceedings of 7th IEEE International Conference on Networking, Architecture and Storage*, 2012.