Prototype, Cloud&Heat SafeCloud-based cloud storage platform

D5.3

Project reference no. 653884

August 2017

Horizon 2020
European Union funding
for Research & Innovation

Document information

Scheduled delivery	31.08.2017
Actual delivery	31.08.2017
Version	1.0
Responsible Partner	Cloud&Heat Technologies GmbH

Dissemination level

Public

Revision history

Date	Editor	Status	Version	Changes
22.08.2017	S. Schmerler	Draft	0.1	Initial version
22.08.2017	H. Mercier	Draft	0.2	UniNE revision
31.08.2017	S. H. Totakura	Draft	0.2	TUM revision
31.08.2017	S. Schmerler	Final	1.0	Incorporate all review suggestions

Contributor

S. Schmerler (C&H)
G. Miegel (C&H)
S. H. Totakura (TUM)
D. R. Matos (IN-ID)

Internal reviewers

H. Mercier (UniNE)
S. H. Totakura (TUM)

Acknowledgements

This project is partially funded by the European Commission Horizon 2020 work programme under grant agreement no. 653884.

More information

Additional information and public deliverables of SafeCloud can be found at http://www.safecloud-project.eu

Glossary of acronyms

Acronym	Definition
DC	Datacenter
IaaS	Infrastructure as a Service
MON	Ceph Monitor
OSD	Object Storage Daemon
RBD	RADOS Block Device
SaaS	Software as a Service
SCFS	SafeCloud File System
VM	Virtual machine
WAN	Wide Area Network

Table of contents

Executive Summary

The prototypes for the SafeCloudBox and CloudBlockStorage solutions based on SafeCloud technology and developed by C&H are described in detail. Both prototypes are storage solutions backed by the Ceph distributed storage software. Being prototypes, both solutions are realized using VMs and in one case Docker containers. The products will be developed further during the remaining project time with the goal of being fully integrated into C&H's infrastructure. In this deliverable, we describe working setups and show that SafeCloud technology can be fully integrated into C&H products. Specifically, the SafeCloudBox product is an entirely new prototype, which relies on the SafeCloud File System to provide customers with a secure and fault-tolerant data storage solution. The CloudBlockStorage product extends C&H's block storage cloud offer with inter-datacenter security by using the SafeCloud Private Communication Middleware.

1 Introduction

The current deliverable outlines the technical development and deployment of prototype setups, showing the functionality of the SafeCloudBox and CloudBlockStorage solutions, which are in development at C&H and based on technology created within the SafeCloud project. The deliverable describes two prototypes developed at C&H in collaboration with IN-ID and TUM, which show that all technologies from within SafeCloud and used by C&H are able to be fully integrated into C&H's production infrastructure, which is the main goal of this deliverable.

2 SafeCloudBox

This chapter describes the implementation of the SafeCloudBox prototype at C&H, utilizing the SafeCloud File System.

1.1 Product design goals

A summary of the design goals described in D5.1 is given here for convenience. The SafeCloudBox product aims to provide users with enhanced data storage security and disaster recovery capabilities compared to services such as DropBox. To accomplish that goal, C&H uses the SafeCloud File System (SCFS) developed within the SafeCloud project. Details about the architecture of SCFS are shown in Fig.1 and are outlined in [SCFS14].

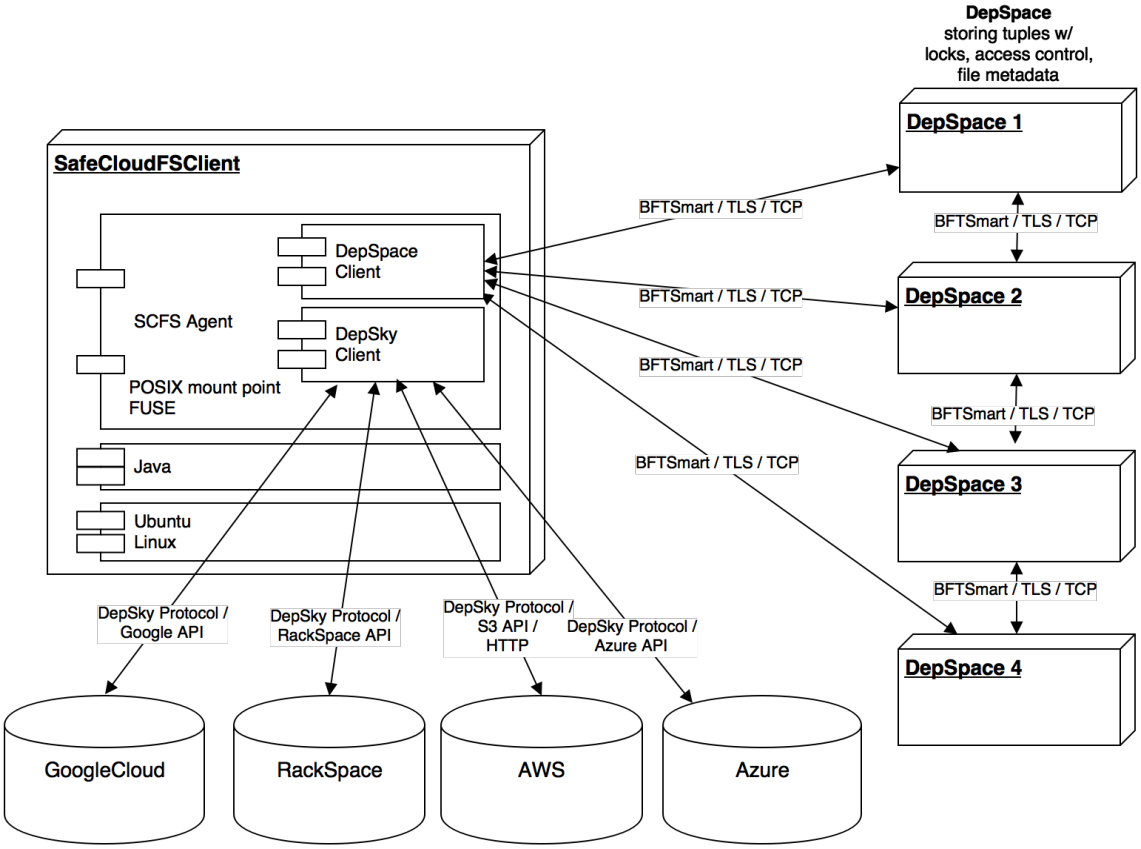

Figure 1: SCFS architecture (image provided by IN-ID)

The SCFS client application provides a FUSE-based mount point to which data can be written. The data is then transferred to cloud storage. The client will be running on the customers infrastructure, such as a datacenter (DC) or a local office machine. One advantage of SCFS is that it writes data encrypted to the cloud storage backends automatically. The second and even more important feature is that SCFS writes data to multiple cloud backends, thus even allowing data recovery in case a cloud provider's datacenter goes down. These points are very important to customers and as such, are equally important to C&H.

C&H operates a number of DCs across Germany and can use SCFS in a quite natural way by using several physically distinct DCs as cloud storage backends.

1.2 Cloud backend storage technology

As can be seen in Fig. 1, SCFS has backend drivers for a number of major cloud providers such as Amazon AWS S3, Google Cloud etc. C&H uses OpenStack[1] as cloud operating system on all DCs. Until recently, C&H used OpenStack Swift [2] as object storage implementation (as well as GlusterFS[3] for block storage). Swift has very similar semantics in terms of data storage compared to AWS S3 (buckets, objects). However, it has its own API, which is not supported by SCFS. There is a project implementing a S3 API layer for Swift (swift3[4]) and C&H spent some time to deploy that layer in production systems. It turned out that the Swift S3 API is not as mature and stable as is needed for production systems. At the same time, C&H was planning several new DCs and coincidentally decided to move to another storage technology entirely, namely Ceph[5].

The Ceph system is a modern and scalable solution which offers object storage and block storage using one underlying storage technology called RADOS, as shown in Fig. 2.

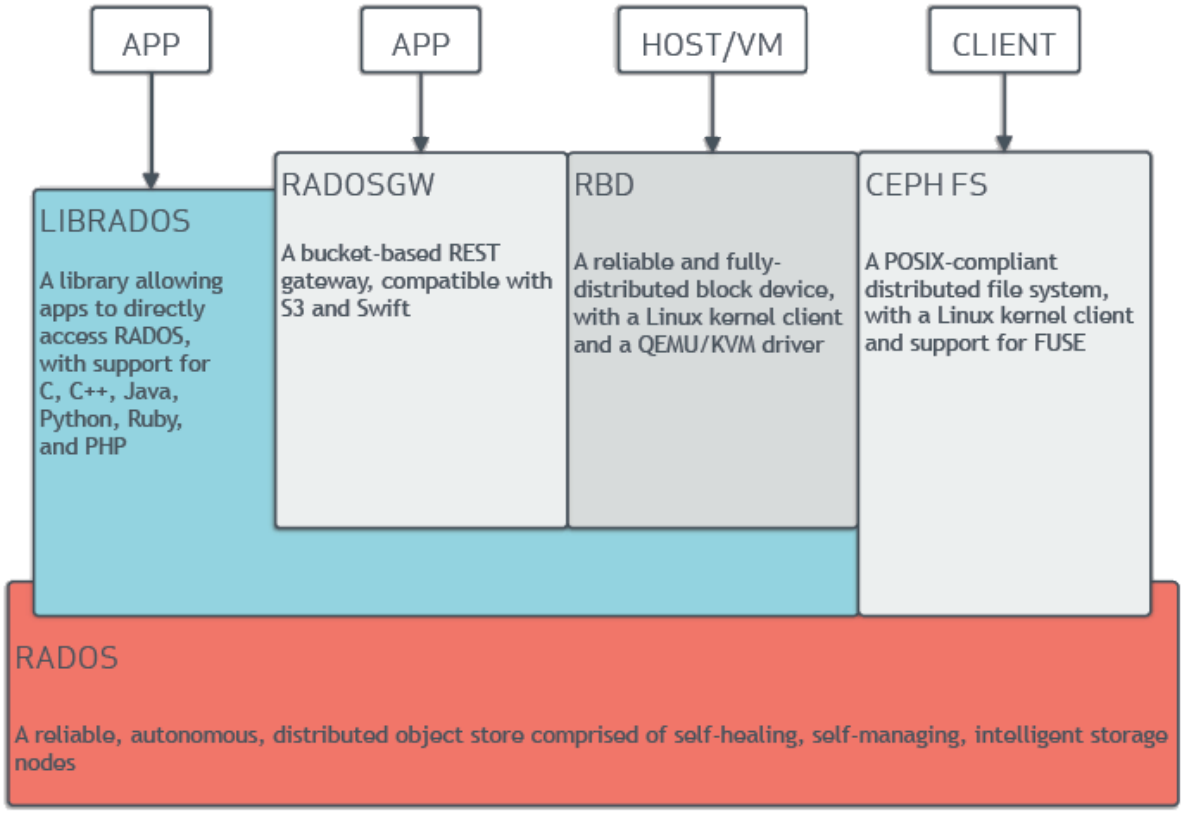

Figure 2: Ceph architecture (from http://docs.ceph.com)

1 https://www.openstack.org
2 https://wiki.openstack.org/wiki/Swift
3 https://www.gluster.org
4 https://github.com/openstack/swift3
5 http://ceph.com

In particular, Ceph's S3 API is provided by a software layer called the RADOS Gateway (RADOSGW). We found it to be much more robust and stable than the Swift S3 API layer. However, as the new DCs are not fully operational and the Swift storage systems are deprecated in old DCs at the time of writing, there was no possibility to deploy SCFS using Ceph backends running on geographically distinct locations. Instead, we focused on building the prototype SCFS deployment setup using VMs and Docker containers. This approach allows us to use all functionality in a controlled testbed and is architecturally equivalent to a fully deployed system. Once Ceph is fully operational in all of our DCs, it is trivial to configure actual DCs running Ceph as storage backend.

1.3 Prototype implementation

Before we could deploy our prototype, a small change in the SCFS code was required. C&H needed to be able to configure an alternative endpoint URL to the default Amazon S3 API endpoint. IN-ID performed the change quickly, such that C&H could proceed to use the code.

The SCFS + Ceph prototype setup is shown in Fig. 3.

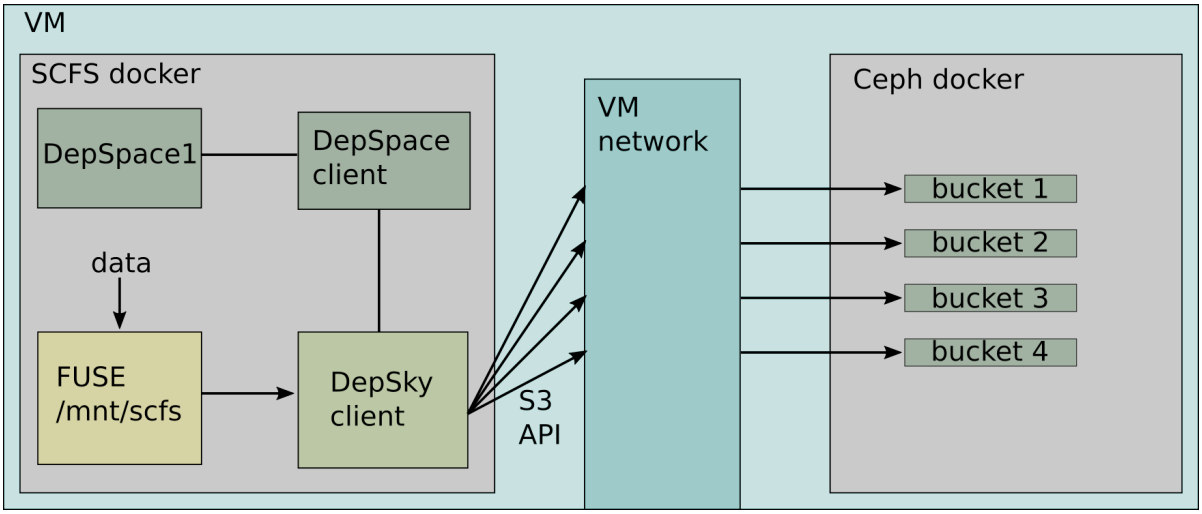

Figure 3: SafeCloudBox SCFS + Ceph prototype setup

We set up a VM running two Docker[6] containers, one running a Ceph cluster and one running all of SCFS (usually, the DepSky client as well as the DepSpace replicas run on different machines, see Fig. 1). SCFS needs at least 4 cloud storage backends to run in a fail-safe mode. In our prototype, the Ceph container is configured 4x in SCFS as cloud storage backend and therefore, the same data is stored in each of the 4 buckets in Ceph. Instead of running 4 DepSpace coordination service replicas, C&H and IN-ID opted for a simpler setup using only one replica for the prototype. Once the SCFS system is started, a FUSE device is provided, which can be mounted and used to read and write data. The data flow is also shown in Fig. 3. Below we show an example session using SCFS. The 4 Ceph S3 buckets are named "**`depskyXXXXXXXXXX-cocY`**".

6 https://www.docker.com

First, we check the content of the SCFS mount from within the SCFS docker container:

```
root@scfs:~# ls -la /mnt/scfs
-rw-r--r-- 1 root root 797 Aug 11 11:42 .statistics.txt
```

There is only one default file here (`.statistics.txt`). Thus there should be already data present in SCFS which represents that file. We check with an S3 API client, which talks to the S3 API endpoint of the Ceph docker container running inside the prototype VM (a script which uses the boto2[7] library):

```
user@vm:~$ ./s3test.py la
depskysbolvkzymdd-coc1
    415024517702370metadata
    415024517702370value1004
depskysbolvkzymdd-coc2
    415024517702370metadata
    415024517702370value1004
depskysbolvkzymdd-coc3
    415024517702370metadata
    415024517702370value1004
depskysbolvkzymdd-coc4
    415024517702370metadata
    415024517702370value1004
```

As can be seen, there is a data and metadata entry for the file in each bucket. Note that each bucket contains the same data, since in a multi-DC deployment case, data is replicated over 4 DCs, each holding one of the buckets.

Now, again in the SCFS container, we write a file to SCFS:

```
root@scfs:~# echo test > /mnt/scfs/file
```

The bucket contents in the Ceph container now look like this::

```
user@vm:~$ ./s3test.py la
depskysbolvkzymdd-coc1
    415024517702370metadata
    415024517702370value1004
    415033170075570metadata
    415033170075570value1004
depskysbolvkzymdd-coc2
    415024517702370metadata
    415024517702370value1004
    415033170075570metadata
    415033170075570value1004
depskysbolvkzymdd-coc3
    415024517702370metadata
    415024517702370value1004
    415033170075570metadata
    415033170075570value1004
depskysbolvkzymdd-coc4
    415024517702370metadata
    415024517702370value1004
```

7 http://boto.cloudhackers.com/en/latest/

```
415033170075570metadata
415033170075570value1004
```

There are new entries in each bucket, holding the data of the written file.

With this prototype, C&H and IN-ID have verified that SCFS is a solution that can be deployed and tested together with real-world storage technology running on C&H's DCs. This is an important step towards a full storage product offering for end users.

1.4 Next steps

As outlined in D5.1, C&H plans to release the SafeCloudBox product as SaaS-type solution. For example, one possible product is a Docker container with SCFS installed and configured to use C&H's cloud storage. In this case, the business model is simply charging for used storage, while the SafeCloudBox client container itself can be distributed as open source, which is possible since the SCFS code hosted at github.com specifies no license and is thus public domain.

Such a storage client can then be deployed by private end users or by system administrators in companies and organizations, thereby extending their storage infrastructure with a secure and fault-tolerant cloud backup.

An extension of the storage client is a SaaS product shipping an open source private cloud solution such as Nextcloud[8], also already pre-configured to use SCFS, at least partly, as secure and fault-tolerant cloud backup.

8 https://nextcloud.com

2 CloudBlockStorage

This chapter describes the implementation of the CloudBlockStorage prototype at C&H, utilizing the Advanced Port-Knocking technique developed by TUM in D1.2 as part of the SafeCloud Private Communication Middleware.

2.1 Product design goals

As described in D5.1, C&H operates a number of DCs across Germany and offers IaaS level products, in particular VMs. Block storage was until now provided by GlusterFS using a 3-fold replication scheme within a DC. With the mentioned switch to Ceph at C&H, the situation is basically unchanged in that C&H offers block storage using 3-fold replication within a DC. The block storage interface to RADOS in Ceph is provided by the RBD layer (RADOS Block Device), as shown in Fig. 2.

There are two related use cases of block storage. First, users are able to attach block storage volumes to VMs (like adding a hard disk to a computer), which allows persistent data storage, even if VMs are shut down or the compute node running the VM dies. This does not protect the complete VM data, but only the data a user chooses to store on the block storage volume, whereas VM data (e.g. the operating system, user home directory, ...) are stored on the compute node's hard disk by default. To address this issue, users have the option of placing the whole VM on a block storage volume. This has performance implications but adds data redundancy and recovery properties independently from compute nodes.

In the SafeCloud context, C&H evaluated the possibilities to go beyond intra-DC data redundancy using inter-DC data replication, where DCs are connected via a WAN. This has, of course, security implications which are met by employing SafeCloud technology. Inter-DC replication gives users additional fault-tolerant data storage options and enhances C&H offers of safe and secure data storage. Especially as C&H operates many DCs, there is a great demand for distributing data across DCs to leverage the present infrastructure for increased fault-tolerant data storage, even in case of DC unavailability (e.g. local power outage, local network failure). It is therefore vital for C&H's products to have access to secure data replication channels between DCs.

2.2 Prototype implementation

In a production setting, we would have two physically distinct DCs. Each would run its own Ceph cluster and the goal is secure data replication over the WAN connecting the two DCs. For the same reasons explained above (lack of production Ceph installations), we built a prototype using two VMs. We deploy two Ceph clusters, each in a separate VM which represent a DC in the real-world production setting. One of the clusters (primary) is the block storage backend used by VMs running in that DC. The second cluster (secondary) is the fallback cluster, to which data is synchronized. This is shown in Fig. 4.

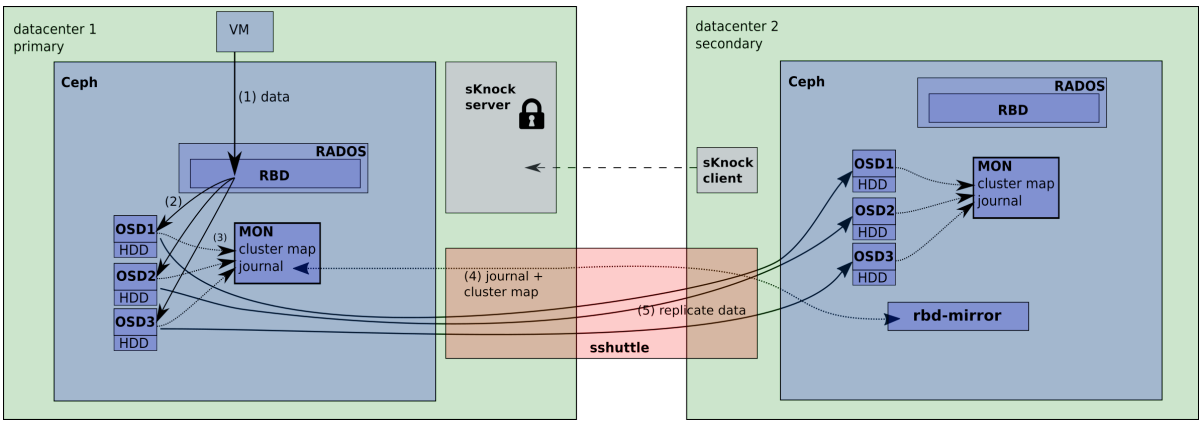

Figure 4: CloudBlockStorage prototype setup

In terms of data replication technology, we employ a Ceph feature called "RBD mirroring". RBD mirroring allows to synchronize data between Ceph clusters. Quoting the Ceph documentation on RBD mirroring[9]:

"RBD images can be asynchronously mirrored between two Ceph clusters. This capability uses the RBD journaling image feature to ensure crash-consistent replication between clusters. Mirroring is configured on a per-pool basis within peer clusters and can be configured to automatically mirror all images within a pool or only a specific subset of images. Mirroring is configured using the rbd command. The rbd-mirror daemon is responsible for pulling image updates from the remote, peer cluster and applying them to the image within the local cluster".

Before we explain the SafeCloud-based security enhancements developed by C&H, we need to establish some Ceph terminology. An OSD (object storage daemon) is a daemon that stores data on a local filesystem (usually a hard disk) and provides access to the data over the network in a Ceph cluster. Each Ceph cluster requires at least one monitor daemon (MON) to monitor the cluster status and to coordinate replication of data among OSDs.

Note that since the primary and secondary cluster are, in general, connected by a WAN and the RBD mirror protocol has no security layer, one needs to secure that channel by other means. This is where C&H employs SafeCloud technology. In the C&H prototype, the WAN communication between primary and secondary cluster is secured be a combination of sshuttle[10] and sKnock. sshuttle is a simple VPN-like technology based on SSH, while sKnock is the port-knocking technology developed by TUM as part of SafeCloud (see D1.2).

The reason why we use sshuttle in addition to port-knocking is because it adds an extra layer of security by encrypting all traffic. A second reason is that RBD mirroring requires an additional daemon (rbd-mirror daemon). The rbd-mirror daemon of the secondary cluster needs to have full layer-3 connectivity to the OSD and MON daemons of the primary cluster in order to replicate data from the primary cluster. The port numbers of

9 http://docs.ceph.com/docs/master/rbd/rbd-mirroring
10 https://github.com/apenwarr/sshuttle

the OSD and MON daemons on the primary cluster are not predictable in a scalable fashion because in real world DC scenarios OSD and MON daemon are spread over a huge number of hosts and are not reachable directly at one public host address.

The security-enhanced version of RBD mirroring in the prototype works as follows. The sKnock server is running on a host on the primary cluster and gets port-knocked from the sKnock client on the secondary cluster to open a specific port. After the sKnock server has opened the requested port, sshuttle on a host on the secondary cluster is used to establish a transparent and encrypted routing of TCP flows into the layer-3 network of the primary cluster. The rbd-mirror daemon running in the secondary cluster then contacts the MON daemon on the primary cluster over this tunnel to receive the cluster map and the journal of the primary cluster. The changes in the dataset of the primary cluster are then replicated ("replayed") by the OSDs of the secondary cluster form their counterpart OSDs in the primary cluster.

2.3 Next steps

C&H plans to deploy the described setup between two DCs, once they run Ceph in production. C&H may replace sshuttle with a full VPN solution if needed. C&H will also evaluate the performance implications of inter-DC data replication and may restrict data synchronization in ways compatible with the user's data redundancy requirements and available bandwidth.

3 Conlcusion

This deliverable presented the developed prototypes for the C&H SafeCloudBox and CloudBlockStorage solutions, both of which are storage products based on the Ceph distributed storage software. It was shown that SafeCloud technology can be used to add levels of data security and fault-tolerance to the solutions and that this technology can be integrated in prototypical C&H systems. It was concluded that it is possible to scale the prototypes to production level systems.

4 References

[SCFS14] Alysson Bessani, Ricardo Mendes, Tiago Oliveira, and Nuno Neves, "SCFS: A Shared Cloud-backed File System", 2014 USENIX Annual Technical Conference (USENIX ATC 14), p169-180