



# Cloud File System Security and Dependability with SafeCloud-FS

Miguel P. Correia  
KTH – Stockholm – June 2017

Joint work with [Alysson Bessani](#), B. Quaresma, F. André, P. Sousa, R. Mendes, T. Oliveira, N. Neves, M. Pasin, P. Verissimo, M. Pardal, E. Silva, F. Apolinário, D. Matos



UNIVERSIDADE DE LISBOA  
TÉCNICO LISBOA



European Commission  
Horizon 2020  
European Union funding for Research & Innovation




inescid lisboa



FCT Fundação para a Ciência e a Tecnologia  
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

## Clouds are complex so they fail



**Ma.gnolia Suffers Major Data Loss, Site Taken Offline**

**Cloud computing takes hit in Sidekick data loss**

**Facebook helps you connect and share with the people in your life.**

**More Details on Today's Outage**

**Google App Engine Downtime Notify**

**Stalked Teens, Spied Chats (Updated)**

**These faults can stop services, corrupt state and execution: Byzantine/malicious faults**

**Message from discussion App Engine Datastore Outage - May 25, 2010**

**App Engine Team**

**May 25th Datastore Outage Post-mortem**

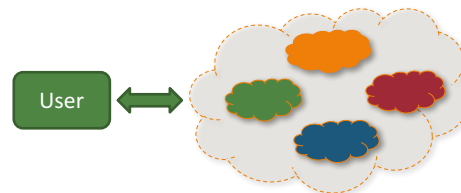
**Summary**

On May 25th, App Engine's Datastore experienced a failure causing an unexpected read-only period while traffic moved to the secondary data center. The outage affected all App Engine applications using the

2

## Cloud-of-Clouds

- Consumer runs service on a **set of clouds** forming a **virtual cloud**, what we call a **cloud-of-clouds**
- Related to the notion of federation of clouds
  - **Federation of clouds** – a virtual cloud created by cloud providers; requires cooperation between providers
  - **Cloud-of-clouds** – an ad-hoc virtual cloud created by consumers; no cooperation between clouds needed



3

## Cloud-of-Clouds dependability+security

- There is **redundancy** and **diversity** between clouds
- so even if some clouds fail a **cloud-of-clouds** that implements **replication** can still guarantee:
  - **Availability** – if some stop, the others are still there
  - **Integrity** – if some corrupt data, data is still at the others
  - **Disaster-tolerance** – clouds can be geographically far
  - **No vendor lock-in** – several clouds anyway
- plus, although, not specific to cloud-of-clouds:
  - **Confidentiality** (from clouds) – encryption
  - **Confidentiality/integrity** (from users) – access control

4

## Outline

- DepSky – file storage in clouds-of-clouds
- SCFS – file system in clouds-of-clouds
- SafeCloud-FS – file system in clouds-of-clouds

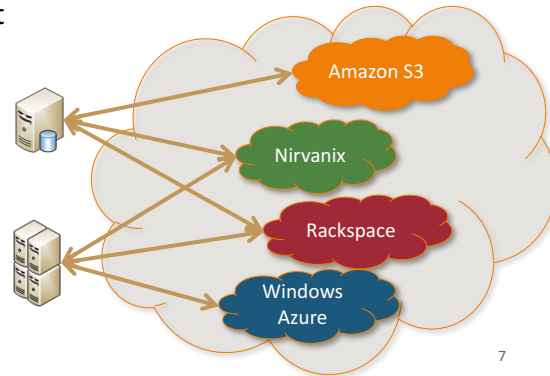
5

## DEPSKY: FILE STORAGE IN CLOUDS-OF-CLOUDS

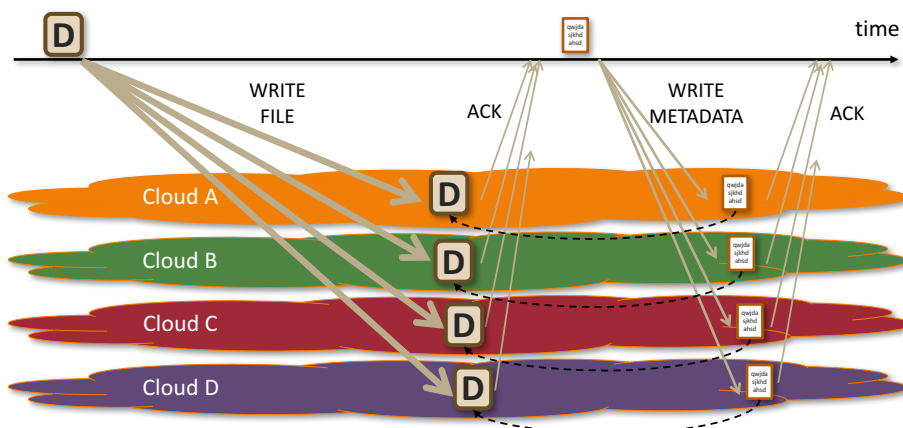
6

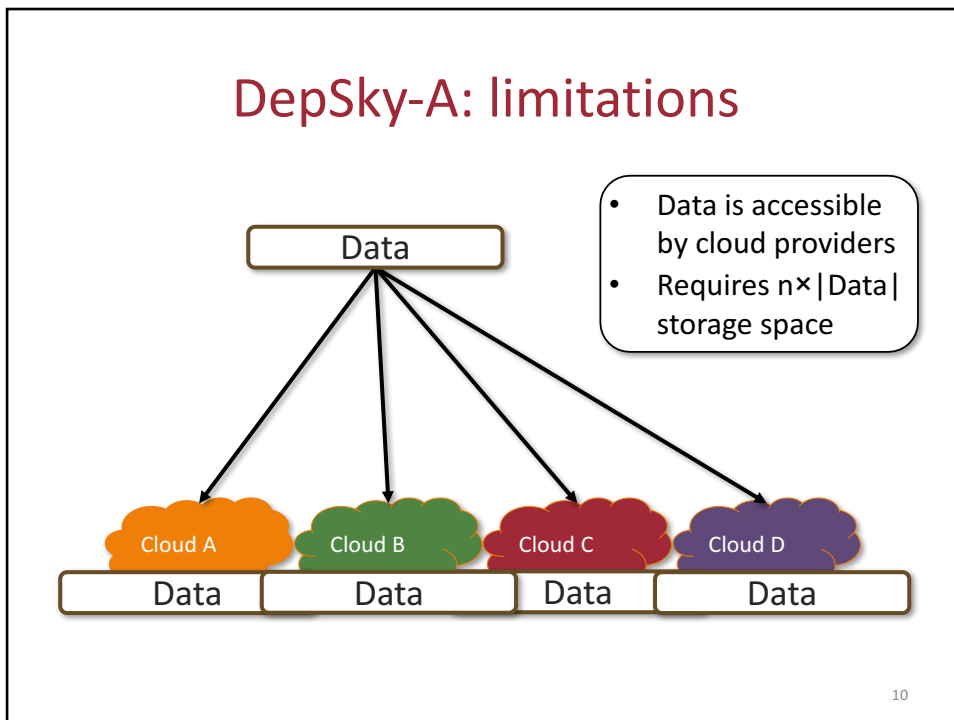
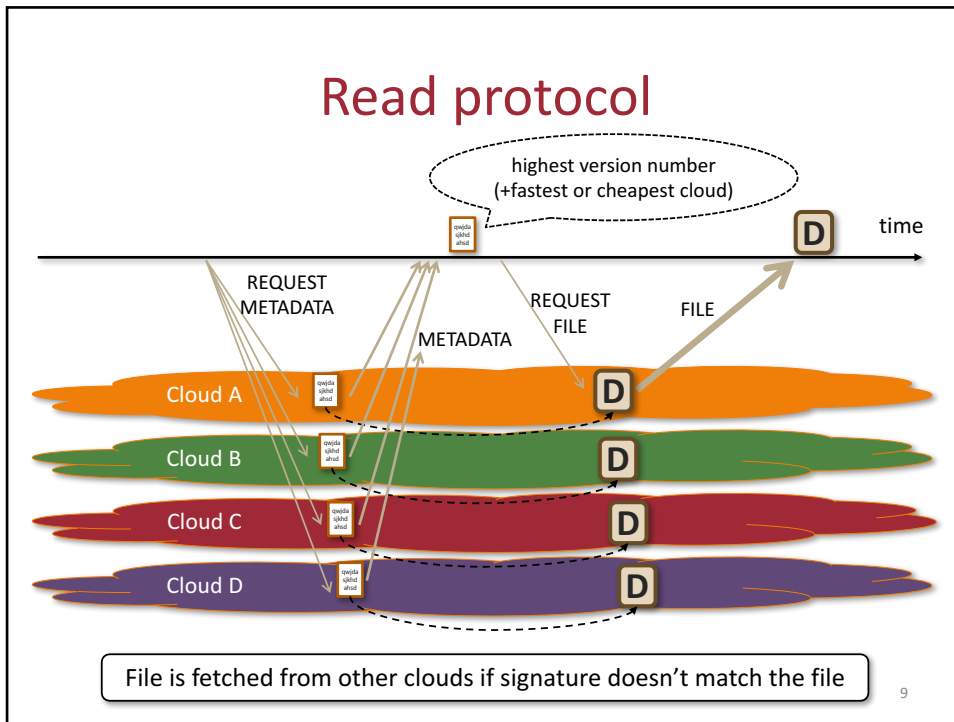
## DepSky

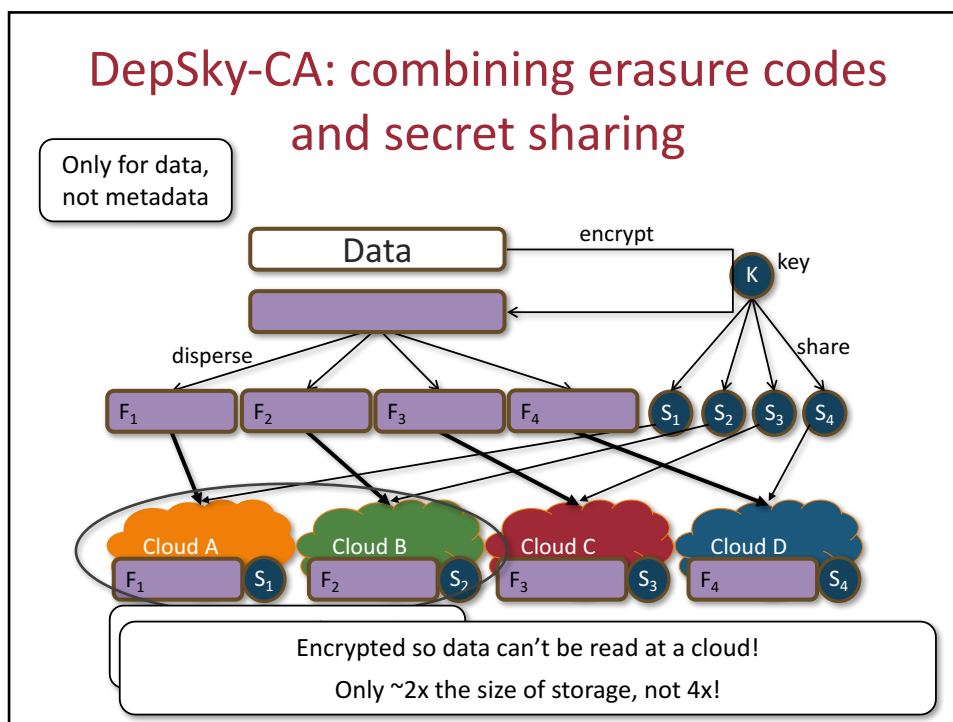
- Client-side library for cloud-of-clouds storage
  - File storage, similar to Amazon S3: read/write files, etc.
- Use storage cloud services (S3, etc.) as they are:
  - All code at the client
- Data is updatable
  - Requires Byzantine quorum replication protocols for consistency



## Write protocol

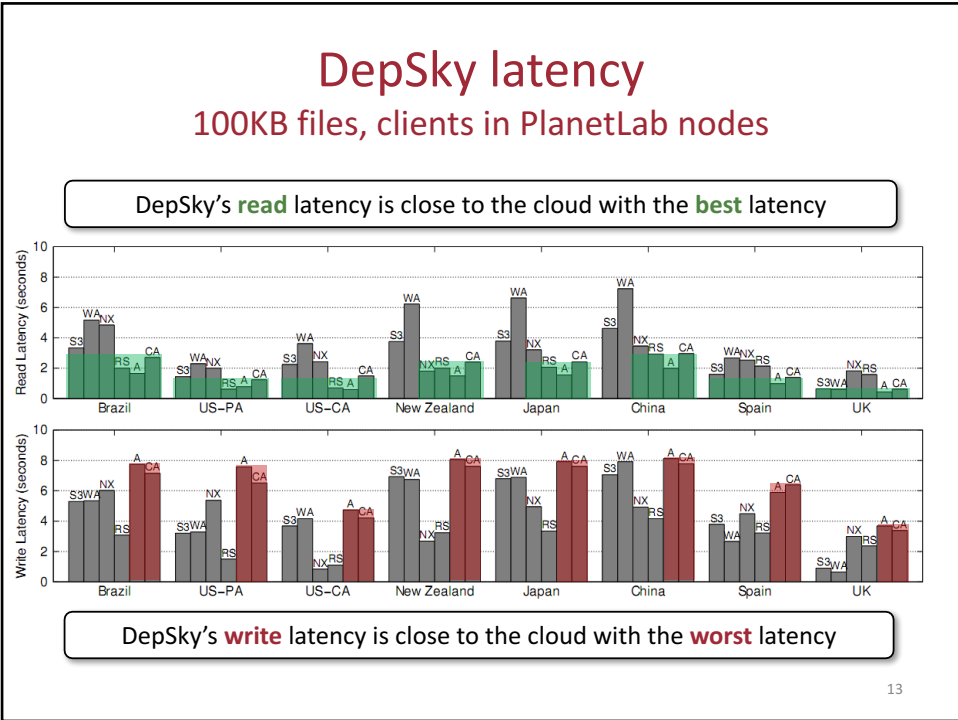






## Consistency proportionality

- The consistency provided by DepSky is the same as the base storage clouds
  - If the weakest consistency cloud provides **eventual consistency**, DepSky provides **eventual consistency**
  - If the weakest consistency cloud provides **regular storage**, DepSky provides **regular storage**
  - ...



### DepSky perceived availability

- **perceived availability** = n. of files read / n. of read attempts
- impacted by the cloud and Internet availability

Location	Reads Tried	DEPSKY-A	DEPSKY-CA	Amazon S3	Rackspace	Azure	Nirvanix
Brazil	8428	1.0000	0.9998	1.0000	0.9997	0.9793	0.9986
US-PA	5113	1.0000	1.0000	0.9998	1.0000	1.0000	0.9880
US-CA	8084	1.0000	1.0000	0.9998	1.0000	1.0000	0.9996
New Zealand	8545	1.0000	1.0000	0.9998	1.0000	0.9542	0.9996
Japan	8392	1.0000	1.0000	0.9997	0.9998	0.9996	0.9997
China	8594	1.0000	1.0000	0.9997	1.0000	0.9994	1.0000
Spain	6550	1.0000	1.0000	1.0000	1.0000	0.9796	0.9995
UK	7069	1.0000	1.0000	0.9998	1.0000	1.0000	1.0000

14

## SCFS: FILE SYSTEM IN CLOUDS-OF-CLOUDS

15

### Storage vs. File System (DepSky vs. SCFS)

- **Storage (DepSky)**
  - API: simple operations over data blocks
  - same consistency as clouds
  - **create**(id)
  - **read**(fd)
  - **write**(fd,block)
  - **delete**(fd)
  - **lock**(fd)
  - **unlock**(fd)
  - **setACL**(fd)
- **File system (SCFS)**
  - API: ~POSIX, so it's mounted and unmodified apps can use it (uses FUSE)
  - strong consistency
  - **open**(path,flags)
  - **read**(fd,buffer,length,offset)
  - **write**(fd,buffer,length,offset)
  - **chmod**(path,mode)
  - **mkdir**(path,mode)
  - **flush, fsync, link, rmdir, symlink, chown,...**

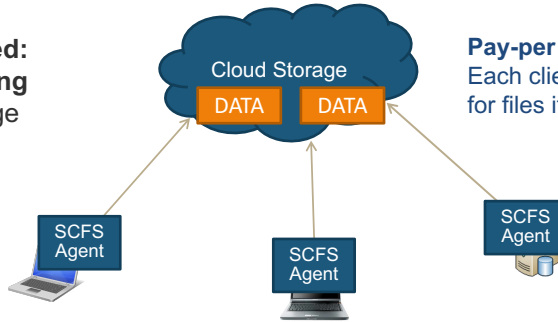
16



## Shared Cloud-backed File System-SCFS

**Client-based:**  
Uses **existing**  
cloud storage  
services

**Pay-per ownership:**  
Each client **pays**  
for files it creates

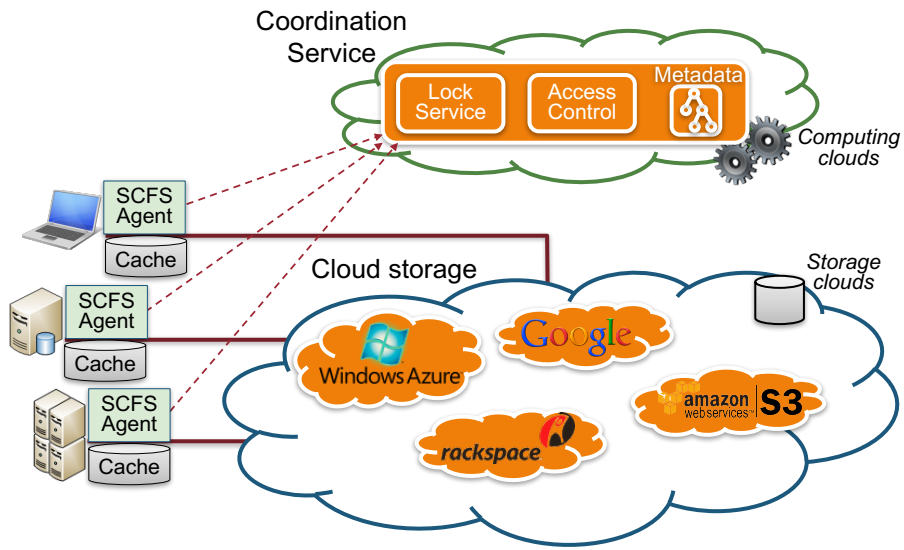


**Strong Consistency**

**Controlled sharing:**  
**Access control** for  
security and concurrency

**Redundant  
Cloud Services**

## SCFS architecture



## Features

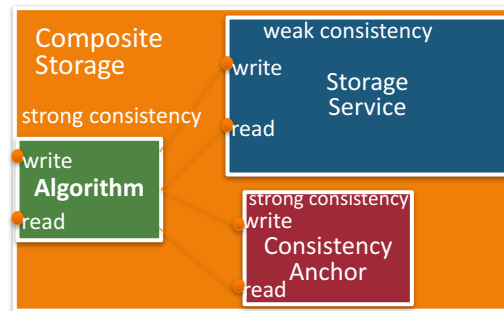
- Data layout/access pattern
  - Each file is an object (single-block file)
  - Multiple versions of the files are maintained
  - Always write, avoid reading (exploiting free writes)
- Caching
  - File cache: persistent (to avoid reading)
    - Local storage is used to hold copies of all client files (that fit)
    - Opened files are also maintained in main-memory
  - Metadata cache: short-lived, main-memory
    - To deal with bursts of *metadata* requests

## Features

- Consistency
  - Consistency-on-close semantics
    - when user closes a file, all updates he did become observable by the rest of the users
  - Locks to avoid write-write conflicts
- Modular coordination
  - Metadata is stored in a coordination service
    - e.g., Apache Zookeeper (crash fault-tolerant), our own DepSpace (Byzantine/intrusion-tolerant)
  - Also used for managing file locks
  - Separate data from metadata

## Consistency anchor

- **Problem:** How to provide strong consistency on top of weak consistency storage clouds? (typically eventual consistency)

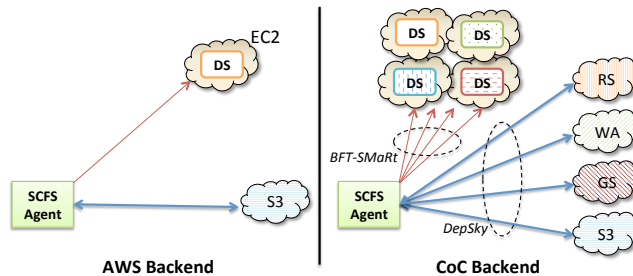


- **Key property:** the composite storage' consistency is the same of the consistency anchor (typ. atomic consistency)
- **Coordination service** serves as consistency anchor

## SCFS configurations

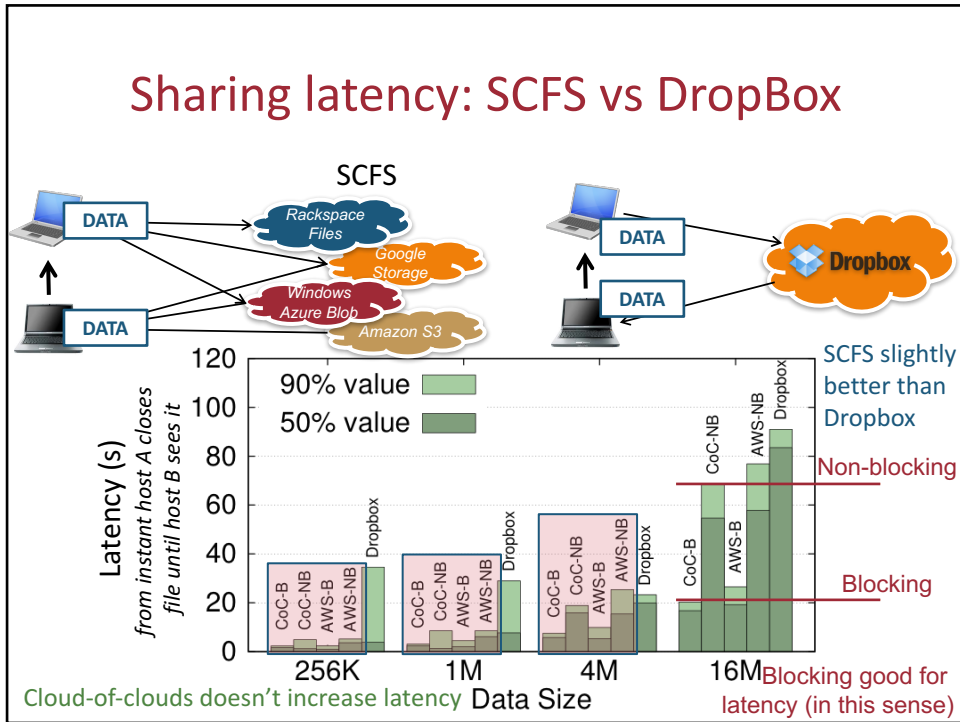
- SCFS can use different configurations/backends

### Intrusion-tolerant configuration (uses DepSky)



- Operation: **blocking**, non-blocking and non-sharing

## Sharing latency: SCFS vs DropBox



## Benchmarking unmodified desktop applications

1.2 MB file



**Open Action:** 1 open(f,rw), 2 read(f), 3-5 open-write-close(lf1), 6-8 open-read-close(f), 9-11 open-read-close(lf1)

**Save Action:** 1-3 open-read-close(f), 4 close(f), 5-7 open-read-close(lf1), 8 delete(lf1), 9-11 open-write-close(lf2), 12-14 open-read-close(lf2), 15 truncate(f,0), 16-18 open-write-close(f), 19-21 open-fsync-close(f), 22-24 open-read-close(f), 25 open(f,rw)

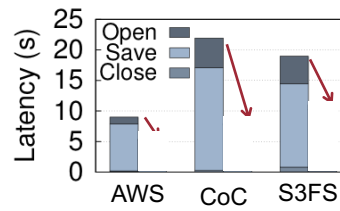
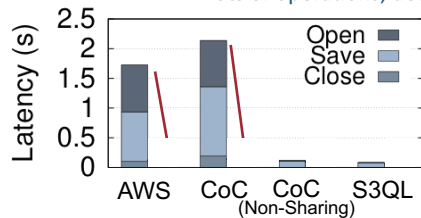
**Close Action:** 1 close(f), 2-4 open-read-close(lf2), 5 delete(lf2)

55% lock file ops; may be done locally

40%

80%

Lots of operations; doing this remotely...



Cloud-of-clouds per se doesn't increase latency much

Doing locks locally reduces much the latency

## SAFECLOUD-FS – AN ENHANCED CLOUD-OF-CLOUDS FILE SYSTEM

25

### SCFS: opportunities

- **Coordination service** stores metadata (e.g., filenames, directories) in clear
- **Integrity verification** of data stored in a cloud requires first downloading the data
- **Intrusion recovery** – when a user account is compromised and data corrupted, recovery has to be done manually

26

## SafeCloud-FS

- Based on SCFS, with the features just explained, but:
- **Coordination service** HomomorphicSpace
  - Based on DepSpace but supports homomorphic operations
  - Based on the MorphicLib library (Java)
    - Operations: searchable, order preserving, summable, multipliable
  - Stores file metadata encrypted
- **Integrity verification:** SafeAudit
  - integrity verification of stored data without downloading it, using homomorphic signatures
- **Intrusion recovery** automatically with SafeRCloud

27

## WRAP-UP

28

## Conclusions

- Masking faults / intrusions using clouds-of-clouds
- **DepSky**: storage clouds-of-clouds
  - Availability, integrity, disaster-tolerance, no vendor lock-in, confidentiality
  - Faults in clouds + versions, so **Byzantine quorum system protocols**
  - Same consistency as the storage clouds
  - **Erasure codes** to reduce the size of data stored
  - **Secret sharing** to store cryptographic keys in clouds

29

## Conclusions

- **SCFS**: a cloud-backed file system
  - Based on DepSky and providing similar guarantees but **near-POSIX API**
  - so it needs strong consistency provided by coordination service
  - caching and careful design allows good performance
- **SafeCloud-FS**: an enhanced cloud-backed file system
  - Encrypted metadata
  - Integrity verification
  - Intrusion removal

30



## Thank you

- Papers:
  - DepSky: Dependable and Secure Storage in a Cloud-of-Clouds. ACM Transactions on Storage, 2013 (also EuroSys 2010)
  - SCFS: a Shared Cloud-backed File System. Usenix Annual Technical Conference (ATC), 2014
- Code:
  - DepSky: <http://cloud-of-clouds.github.io/depsky/>
  - SCFS: <http://cloud-of-clouds.github.io/SCFS/>
- My web: <http://www.gsd.inesc-id.pt/~mpc/>



FCT Fundação para a Ciência e a Tecnologia  
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

