Cloud Service Dependability: Masking and Recovering

Miguel P. Correia

15th MDPS Workshop

Lyon, France – Nov. 2015

Joint work with Alysson Bessani, Dário Nascimento, B. Quaresma, F. André, P. Sousa, R. Mendes, T. Oliveira, N. Neves, M. Pasin, P. Verissimo

TÉCNICO LISBOA

inesc id lisboa

ULisboa / IST / INESC-ID

- Universidade de Lisboa
  - largest univ. in Portugal; ~50K students; ~460 programs; 18 schools
- Instituto Superior Técnico
  - largest engineering school in Portugal; ~12K students; 80 programs
- INESC-ID
  - large lab in computer science and electrical engineering; 100+ PhDs (most IST faculty); hundreds of PhD and Master students; many research groups

2

Clouds are complex so they fail

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

3

Outline

- Masking faults
  - DepSky – file storage
  - SCFS – file system
- Recovering from faults
  - Shuttle – recovery system

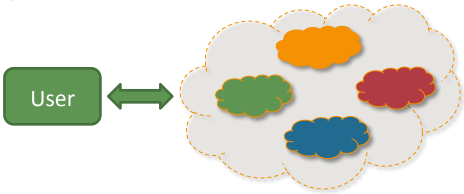
4

DEPSKY: MASKING FAULTS IN STORAGE CLOUDS-OF-CLOUDS

5

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 suggests a virtual cloud created by providers; some level of cooperation between clouds
  - Cloud-of-clouds suggests an ad-hoc virtual cloud created by consumers; no cooperation between clouds



6

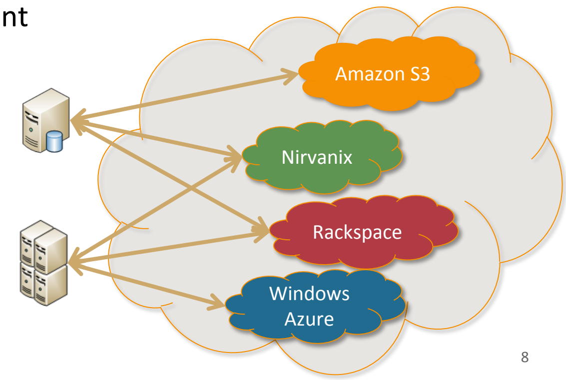
Cloud-of-Clouds dependability+security

- 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

7

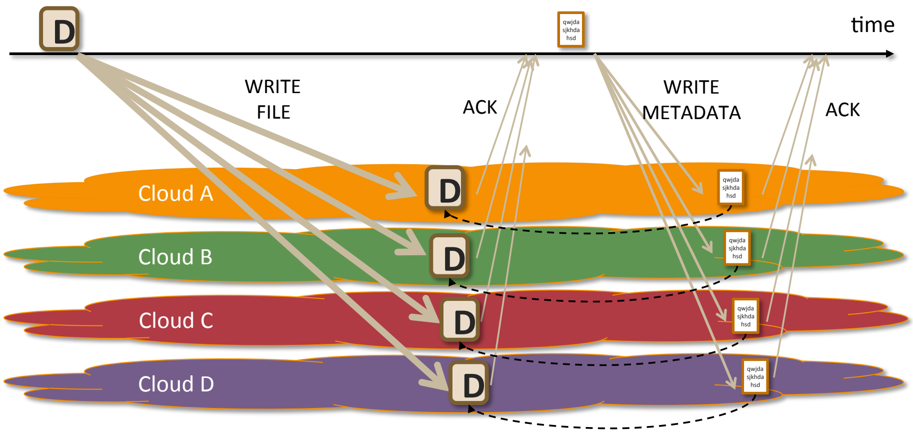
DepSky

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



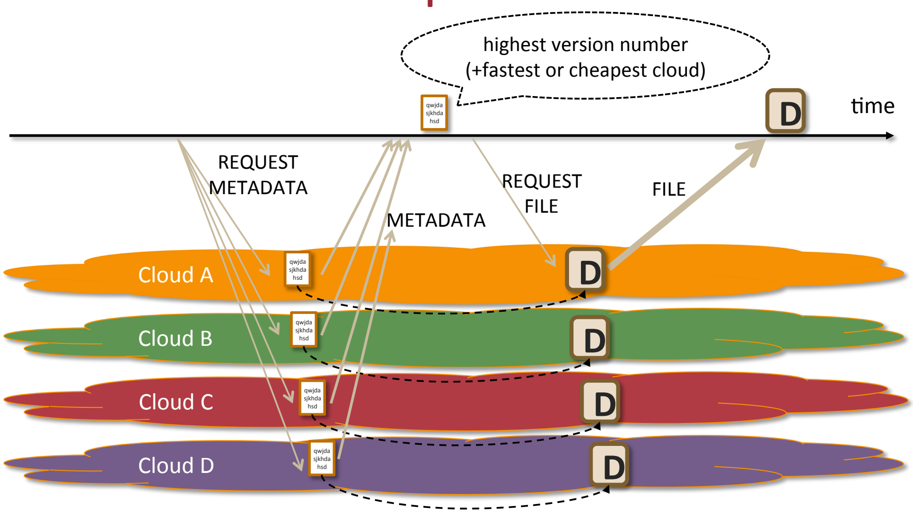
8

Write protocol



9

Read protocol

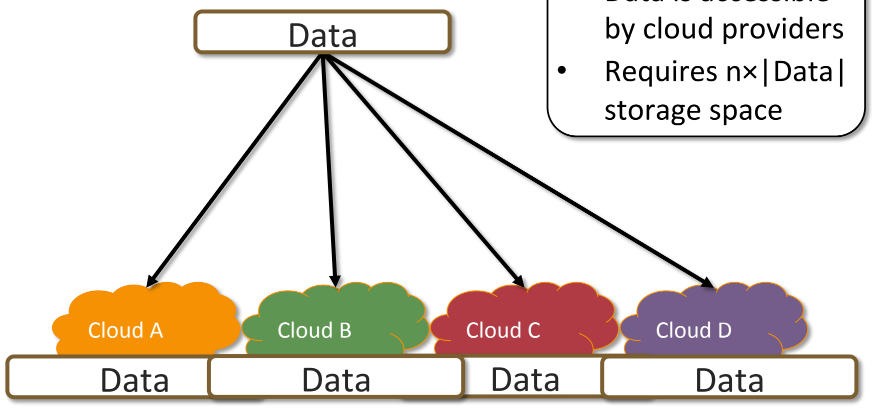


File is fetched from other clouds if signature doesn't match the file

10

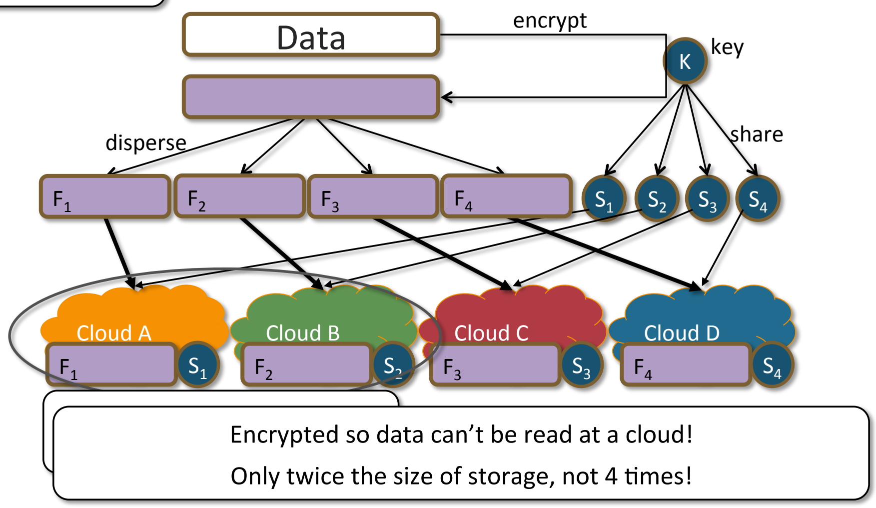
DepSky-A: limitations

- Data is accessible by cloud providers
- Requires n×|Data| storage space



DepSky-CA: combining erasure codes and secret sharing

Only for data, not metadata



Consistency proportionality

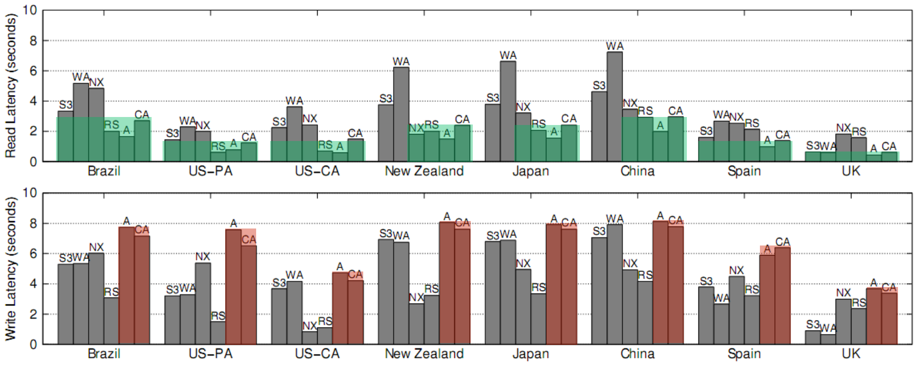
- The consistency provided by DepSky is the same as the base storage clouds
  - If the weakest consistency cloud provides eventual consistency, DepSky provides eventual consistency
  - If the weakest consistency cloud provides read your writes, DepSky provides read your writes
  - If the weakest consistency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

13

DepSky latency

100KB files, clients in PlanetLab nodes

DepSky's **read** latency is close to the cloud with the **best** latency



DepSky's **write** latency is close to the cloud with the **worst** latency

14

DepSky perceived availability

- perceived availability = n. of files read / n. of tries
- 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

SCFS: MASKING FAULTS IN A CLOUD-OF-CLOUDS FILE SYSTEM

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 unmodified apps can use it (uses FUSE)
  - strong consistency

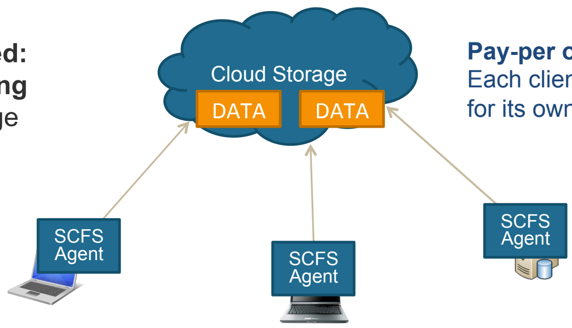
  - **open**(path,flags)
  - **read**(fd,buffer,length,offset)
  - **write**(fd,buffer,length,offset)
  - **chmod**(path,mode)
  - **mkdir**(path,mode)
  - **flush, fsync, link, rmdir, symlink, chown,...**

17

Shared Cloud-backed File System-SCFS

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

Pay-per ownership: Each client **pays** for its own files



Strong Consistency

Controlled sharing: Access control for security and concurrency

Redundant Cloud Services

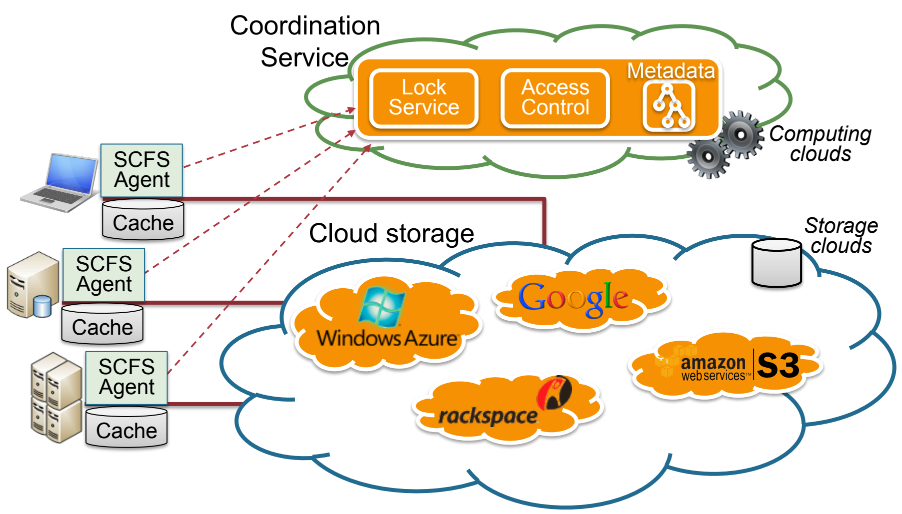
Features

- 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/most client files
    - Opened files are also maintained in main-memory
  - Metadata cache: short-lived, main-memory
    - To deal with bursts of *metadata* requests

Features

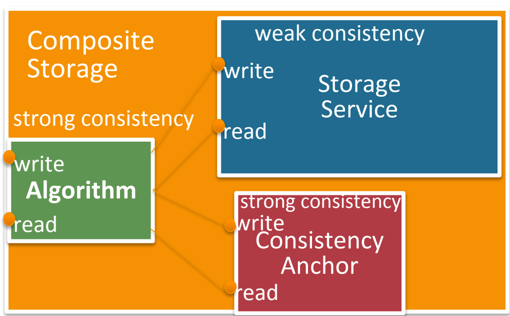
- Consistency
  - Consistency-on-close semantics
    - when user closes a file, all updates he did become observable by the rest of the users
  - Locks to avoid write-write conflicts
- Modular coordination
  - Metadata is stored in a coordination service
    - e.g., Zookeeper (crash fault-tolerant),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

SCFS architecture



Consistency anchor

- Problem: How to provide strong consistency on top of weak consistent storage clouds? (typ. eventual consistency)

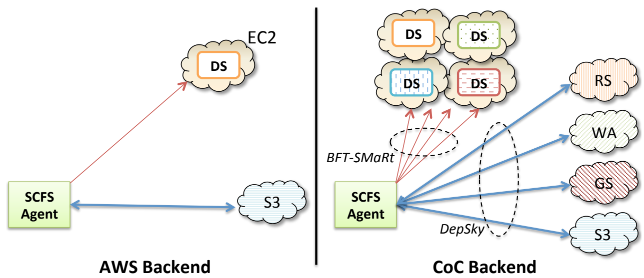


- Key property: the composite storage' consistency is the same of the consistency anchor (typ. atomic consistency)

SCFS configurations

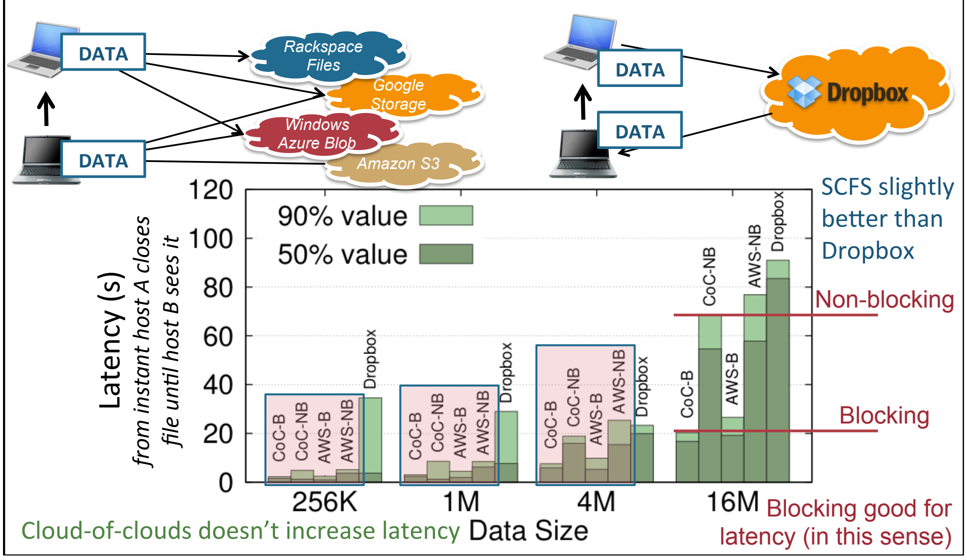
- SCFS can use different configurations/backends

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

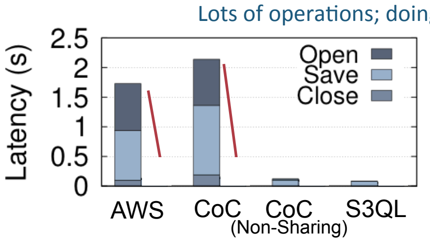
OpenOffice Writer

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)

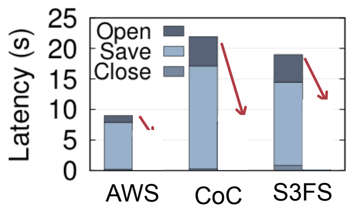
55%
40%
80%

lock file ops; may be done locally

Lots of operations; doing this remotely...



Non-blocking



Blocking

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

Doing locks locally reduces much the latency

SHUTTLE: RECOVERING FROM INTRUSIONS IN PAAS CLOUDS

26

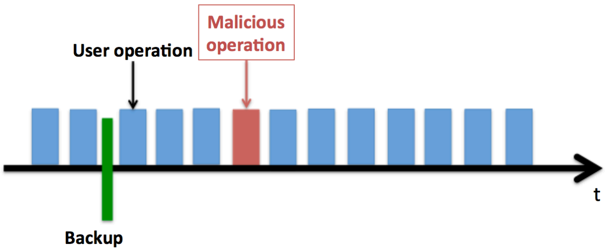
Platform as a Service (PaaS)

- PaaS services allow running applications
- Consumer develops application to run in that environment, using
  – Supported languages, e.g., Java, Python, Go, PHP
  – Supported components, e.g., SQL/noSQL databases, load balancers
  – Examples: Google App Engine, Windows Azure Cloud Services, Salesforce Force.com,...

27

Shuttle

- Recovers PaaS applications' state integrity when there are intrusions
- Isn't it what backups do?
  – Backups: remove both bad and good operations
  – Shuttle: removes bad operations but keeps good ones



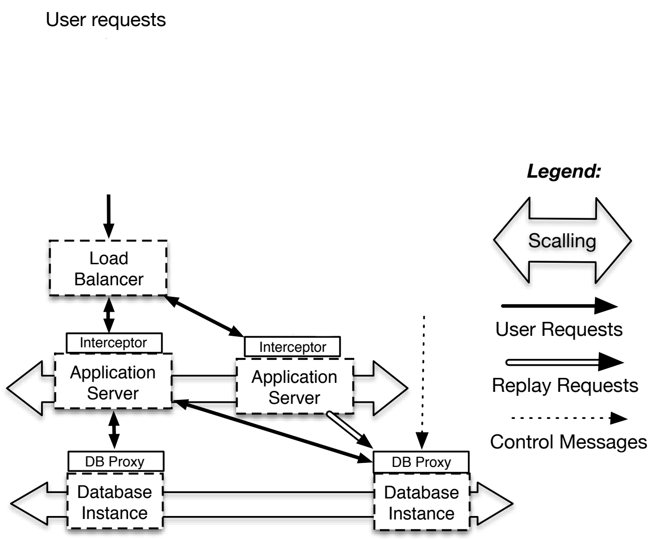
28

Features

- Supported by the cloud: available without consumer setup
- Supports applications deployed in various instances
- Avoids application downtime as no need to stop the application during recovery
- Leverage elasticity to make recovery faster

29

Shuttle architecture



30

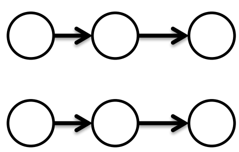
Replay Process

1. Detect/identify the malicious operations (not Shuttle)
2. Start new instances of the application and database
3. Load a snapshot previous to intrusion instant; create a new branch (application stays running in previous branch)
4. Replay requests in new branch
5. Block incoming requests; replay last requests
6. Change to new branch; shutdown unnecessary instances

Replay Modes

- Full-Replay: Replay every operation after snapshot
- Selective-Replay: Replay only affected (tainted) operations

- Serial: Replay all dependency graph sequentially
- Clustered: Independent clusters can be replayed concurrently; allowed by the cloud elasticity



- Modes supported:

	Full-Replay	Selective-Replay
1 Cluster (Serial)	✔	✔
Clustered	✔	✗

Evaluation Environment

- Amazon EC2, c3.xlarge instances, Gb Ethernet
- WildFly (formely JBoss) application servers
- Voldemort database
- Ask Q&A application; data from Stack Exchange

33

Performance overhead

- in normal execution

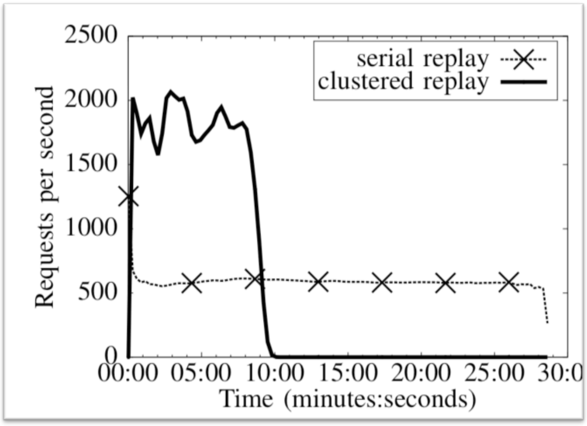
	Workload A	Workload B
Shuttle	6325 ops/sec [5.78 ms]	15346 ops/sec [3.62 ms]
No Shuttle	7148 ops/sec [5.07 ms]	17821 ops/sec [3.01 ms]
overhead	13% [14%]	16% [20%]

Overhead seems acceptable; penalty mostly due to single proxy

34

Recovery time

- for 1 million requests



Clustering greatly reduces recovery time

35

Storage overhead

- for 1 million requests

	# objects	size (MB)
Shuttle Storage:		
Request	1 million	212
Response	1 million	8 967
Start/end timestamps	2 million	16
Keys	137 million	488
Total		9 684
Database node:		
Version List	14 593	1.4
Operation list	9 million	277
Total		282
Manager:		
Graph	1 million	718

Storage is considerable but mostly due to storing full responses

36

WRAP-UP

Conclusions

- Masking / recovering faults / intrusions in the cloud
- 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

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
- Shuttle: a recovery service for PaaS offerings
  - Supports applications running in multiple instances
  - Leverages elasticity/pay-per-use to reduce the recovery time and costs

39

Thank you

- Papers:
  - DepSky: Dependable and Secure Storage in a Cloud-of-Clouds. ACM Transactions on Storage, 2013 (also at EuroSys 2010)
  - SCFS: a Shared Cloud-backed File System. Usenix Annual Technical Conference (ATC), 2014
  - Shuttle: Intrusion Recovery for PaaS. International Conference on Distributed Computing Systems (ICDCS), 2015
- Code:
  - DepSky: http://cloud-of-clouds.github.io/depsky/
  - SCFS: http://cloud-of-clouds.github.io/SCFS/
  - Shuttle: https://github.com/dnascimento/shuttle
- My web: http://www.gsd.inesc-id.pt/~mpc/

TÉCNICO LISBOA

inesc id lisboa