U LISBOA | UNIVERSIDADE DE LISBOA

Cloud-of-Clouds Storage: from Theory to Production

Alysson Bessani

Ciências ULisboa

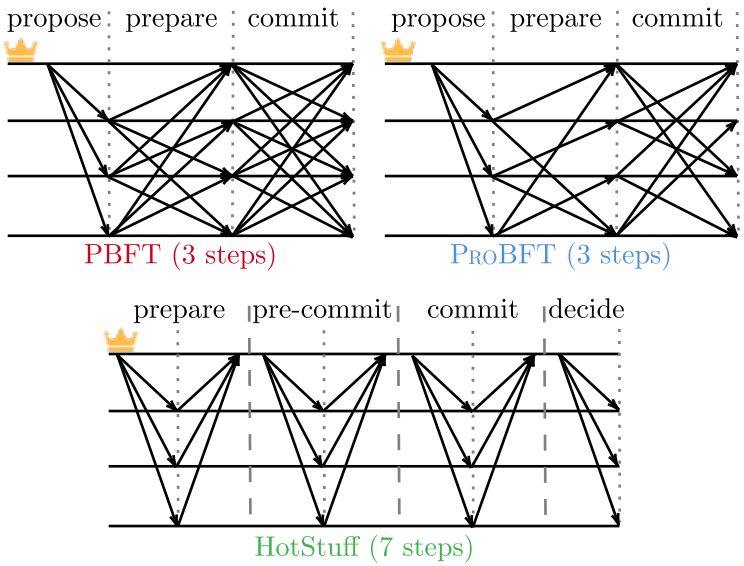
Research in Consensus-based Replication

- Past: BFT-SMaRt, Spinning, MinBFT, BFT-CUP, WHEAT, AWARE, …

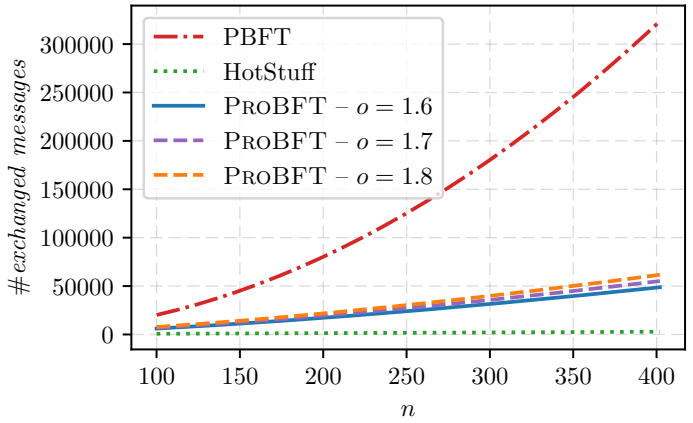
- I'm currently interested in the following topics:
  - **Simple** blockchain consensus protocols
    - Do you know Streamlet?
  - **Confidential** replicated services
    - Secret sharing; Oblivious RAM; Multi-Party Computation
  - **Wide-area** Byzantine-resilient replication
    - Lightspeed consensus; probabilistic protocols

ProBFT [PODC'24]

- New probabilistic protocol
  - PBFT with probabilistic quorums of size $O(\sqrt{n})$ (instead of $O(n)$)
  - Requires a limited network scheduler
  - Liveness (Termination) guaranteed with probability 1
  - Safety (Agreement) guaranteed with probability 1 - exp(-$\Theta(\sqrt{n})$)
- The protocol itself is very simple
- Its analysis is quite complicated



(a) Message pattern and number of communication steps.



(b) Number of exchanged messages.

Consensus-free Replication

Clouds for Data Storage

- According to estimates, 400 million of TBs of data are generated daily
  - Data from diverse sources with different storage requirements
  - Security and regulatory compliance (e.g., GDPR, HIPAA)
- Cloud storage services store a large portion of this data
  - Their scalability, virtually-infinite capacity, and diversified offer are incentives for organizations to move their data to the cloud
  - It is estimated that 80% of companies will move their operations from local datacenters to the cloud by 2025
- **Problem:** although generally more reliable than local infrastructures, clouds are frequently affected by failures and security incidents

THENEWSTA

PODCASTS EBOOKS EVENTS

ARCHITECTURE ENGINEERING

CLOUD SERVICES / DATA

Google Cloud Services Hi
Paris

The incident was later described as "a multicluster failure and has
of multiple zones."

SEARCH

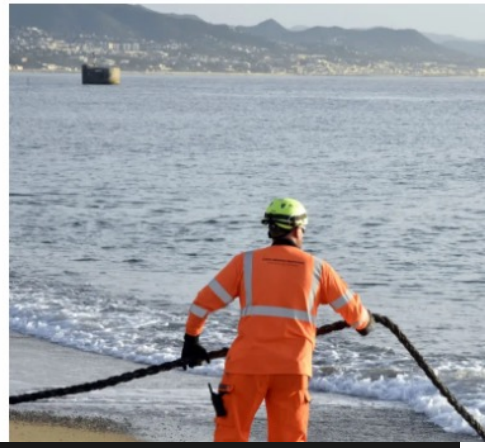
FORTUNE

Europe Energy Economy Tech Retail Lifestyle

TECH

Cloud outages are on the rise. Here's why

BY JEAN-PAUL SMETS
June 7, 2023 at 6:08 PM GMT+1



Topics Newsletters Events

outages serve as warning for
nies relying on cloud
logy

arcserve

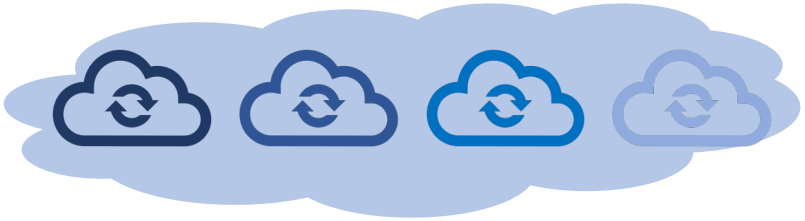
Why Arcserve Products Solutions Partners

CLOUD CYBERSECURITY

7 Most Infamous Cloud Security Breaches

DECEMBER 20TH, 2023

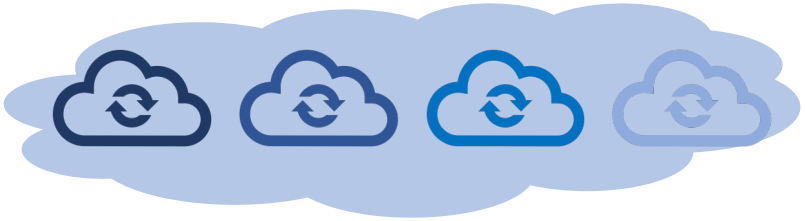
Cloud-of-Clouds Storage



- Use of multiple (independent) cloud providers to store data
- Cloud-of-clouds systems survive faults if no more than a fraction of the underlying clouds are affected at the same time
- They also address the vendor lock-in issue

- Fundamentally changes the **cloud usage trust assumption**:
  - *3rd-party trust*: a cloud application works if its cloud provider is correct
  - *Distributed trust*: a cloud application works if no more than *f* out-of *n* cloud providers are faulty

Cloud-of-Clouds Storage



- Two challenges:
  1. Appropriate middleware for dealing with cloud diversity
  2. Known replication techniques are not easily applied to this model
     - Unless one wants to implement a replicated system with replicas deployed in different cloud providers. **This is not our target!**

- This talk: how to address these challenges
  - (A bit of) Theory: fundamental models and algorithms
  - Practice: some practical systems and a commercial product

Outline

- Introduction
- **Theory**
  - **Model**
  - **Background**
  - **Storage Constructions**
- **Practice**
  - **Two Cloud-of-Clouds File Systems (SCFS, Charon)**
  - **A commercial product (Vawlt)**
- **Conclusions**

Theory

System Model

- Unconstrained set of clients and *n* cloud providers (*clouds*)
- Each cloud implements a service represented by a *base object*
  - Base objects have a well-defined interface and supports access control
- Clients interact with base objects/clouds by invoking *operations*
  - Interactions are asynchronous, i.e., the response time is not bounded

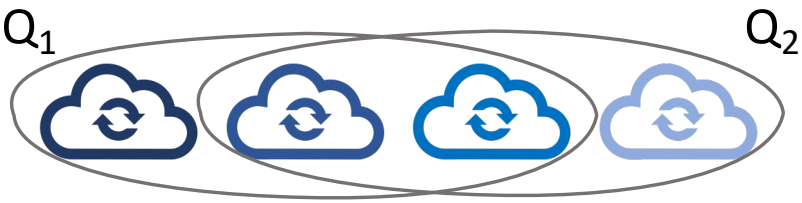
- An unbounded number of clients and up to *f* clouds can experience arbitrary (or *Byzantine*) failures
  - Faulty clients can jeopardize the security of the data they have access

Background: Byzantine Quorum Systems

- A set of subsets (*quorums*) of *n* clouds satisfying:
  - *Consistency*: every two quorums intersect in at least one correct cloud
  - *Availability*: there is at least one quorum of correct clouds

- Dissemination quorum systems:
  - $n > 3f$ (e.g., $n = 3f+1$)
  - $q = \lceil (n+f+1)/2 \rceil$ (e.g., $q = 2f+1$)

Example n=4, f=1:

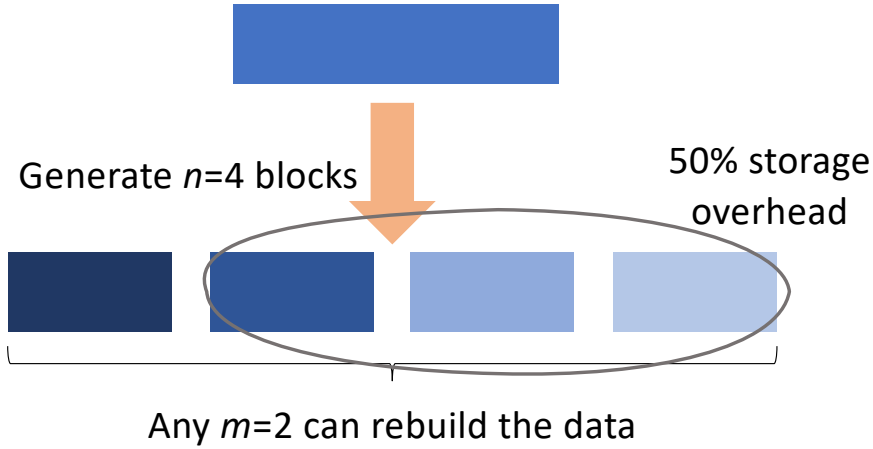


Quorums with q = 3 clouds
Intersections with at least f+1 = 2 clouds

Background: Erasure Code

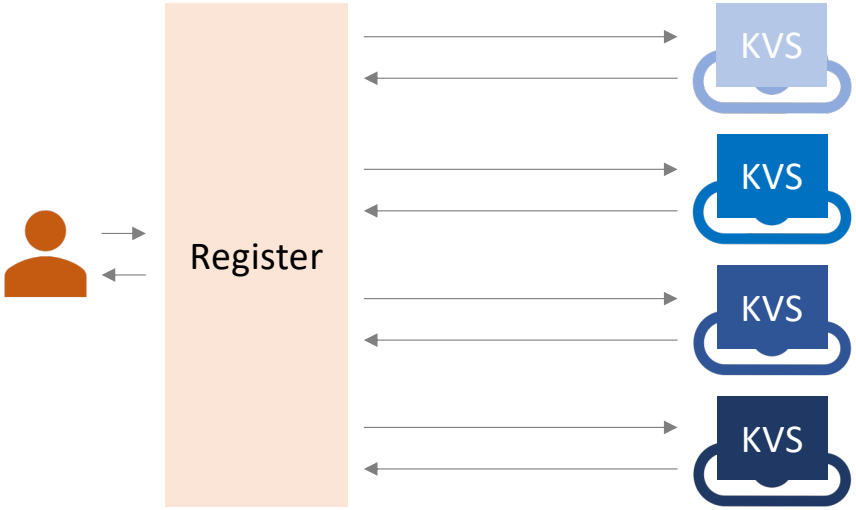
- A type of error correction code
  - Generate *n* coded blocks from a file to be stored
  - Any *m* of those blocks can recover the file (we use *m* = *f*+1)
- The **storage cost** of this technique is *q*/*m* instead of *q*
- Erasure codes can also be used to ensure **confidentiality**, i.e., less than *f*+1 blocks reveal no information about the value

- Example *n*=4, *f*=1:



Storage Abstractions

- Cloud storage services (e.g., AWS S3, Azure Blob Storage, Google Cloud Storage) are fail-prone **key-value stores (KVS)**
- Using these base objects (KVSes), we build Byzantine fault-tolerant **registers** and **lease objects**



Storage Abstractions

Key-value store (the cloud)

- Stores <*key*, *value*> pairs
  - Keys are unique
- Operations: *put(k, v)*, *get(k)*, *list()*, and *remove(k)*
- **Atomic** and **wait-free** operations
- Operations return a real time clock value of the cloud, which also stores the time of the last put for each key
  - Cloud clocks do not need to be synchronized, but drifts are bounded

Register (the multi-cloud)

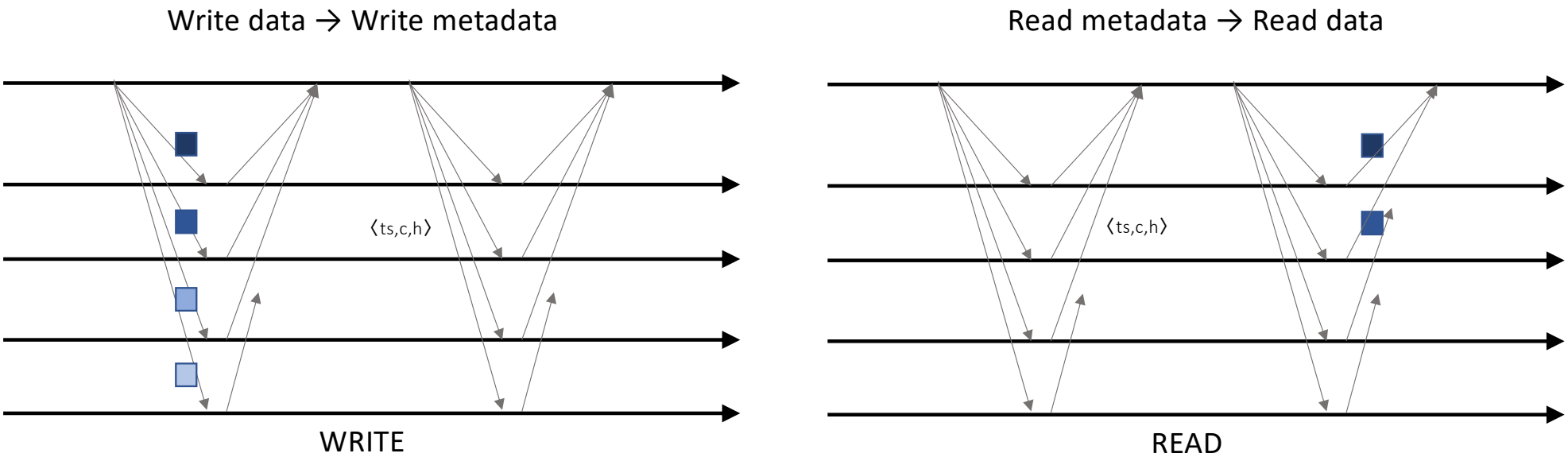
- Store arbitrary values
- Operations: *write(v)*, and *read()*
- Concurrency semantics: **regular**
  - A read returns either the value written in the last completed write or the value being written concurrently
- Types of registers:
  - Write-once
  - Single-writer
  - Multi-writer

Cloud-of-Clouds Constructions
[OPODIS'16, IEEE TCC'21]

- Write-Once **Register**
- Single-Writer **Register**
- Multi-Writer **Register**

- **Lease** Object

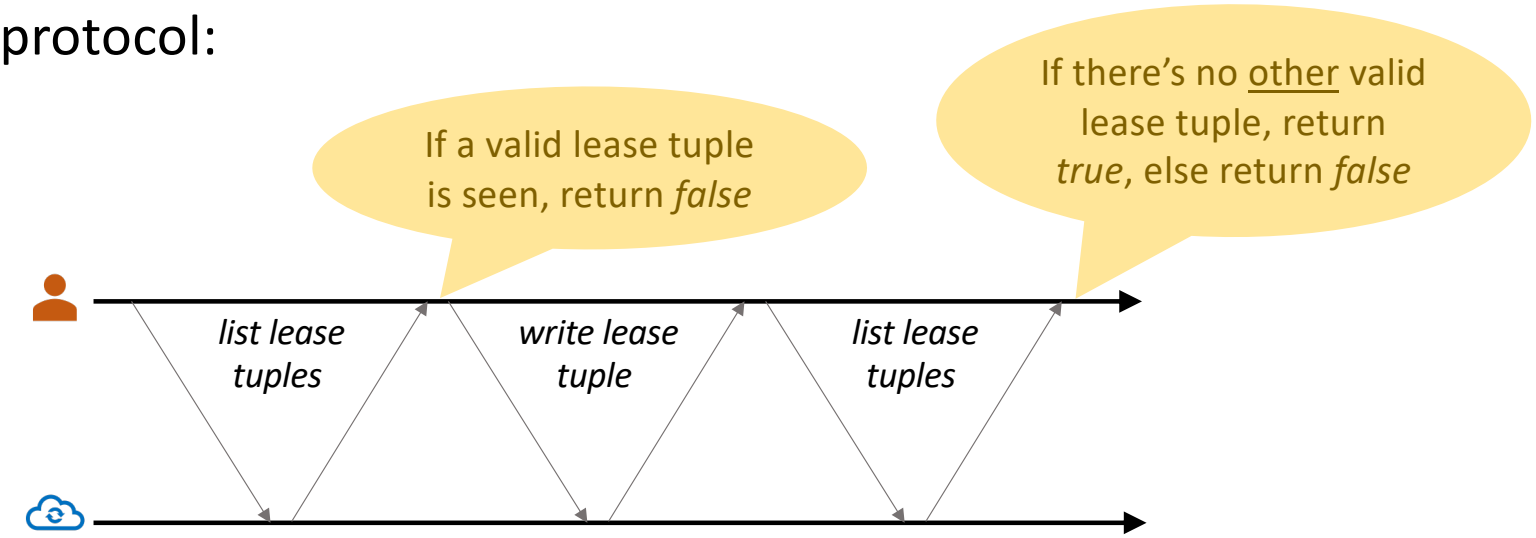
Single-Writer Register: Why does it work?



When a data version is seen, the data content is available

Lease Object

- Lease object is important to support **concurrency control**
  - I.e., avoid concurrent access to shared data
- We show a lease protocol on top of KVS objects
  - It requires real-time timestamps from cloud providers on each interaction
- Base lease protocol:

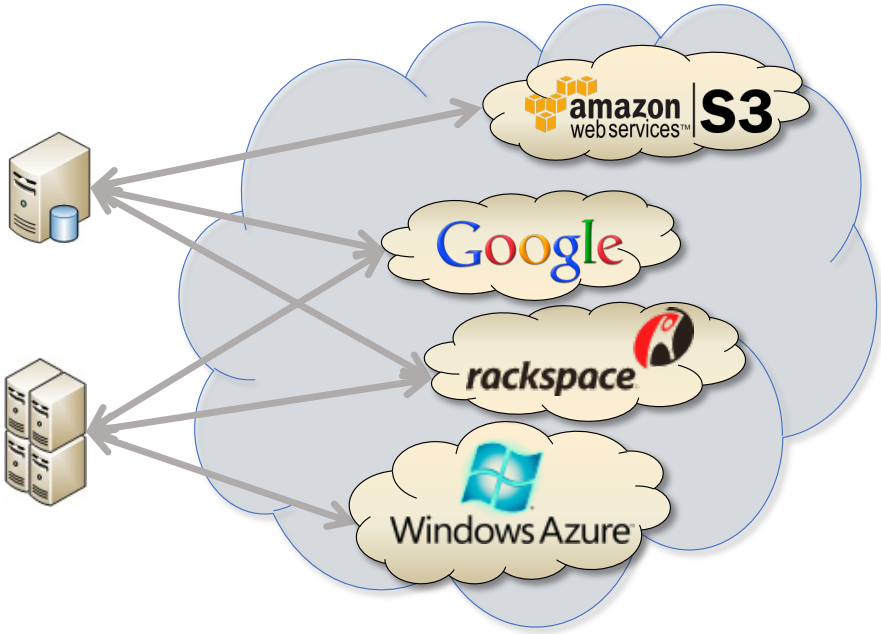


Practice

DepSky

[EuroSys'11, ACM TS'13]

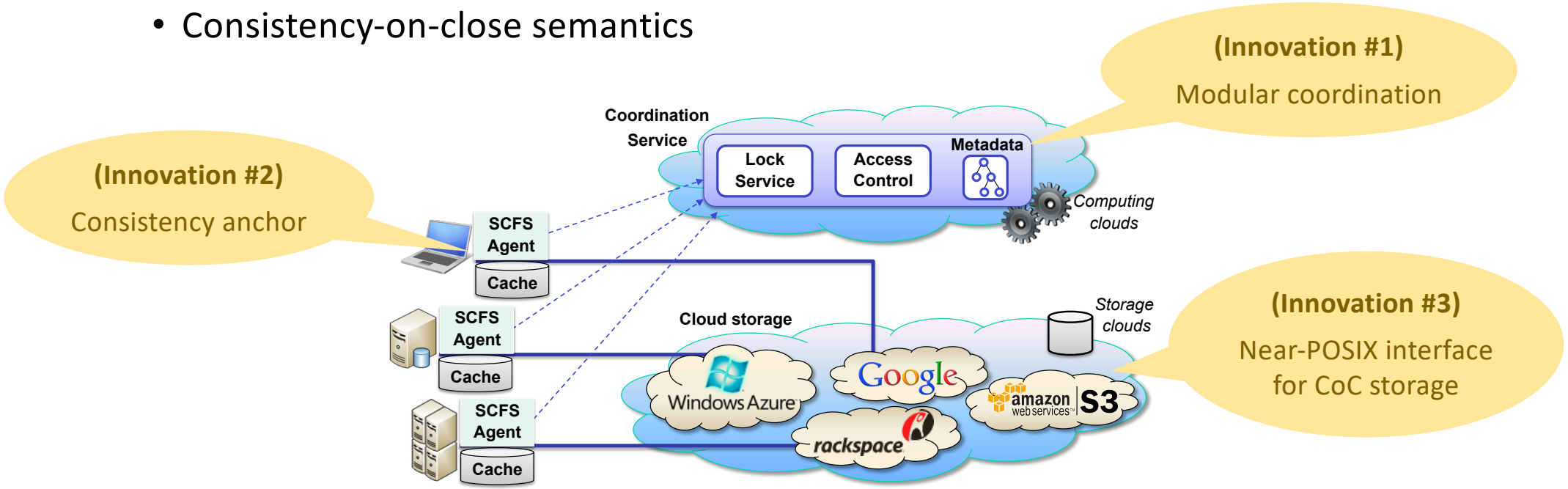
- A Java programming library for using *n* cloud storage services
- Implements the **single-writer register** algorithm with some extensions
  - Creation/destruction of registers
  - Locking/unlocking
- Ensures confidentiality by integrating erasure code with secret sharing



SCFS (Shared Cloud-backed File System)

[USENIX ATC'14]

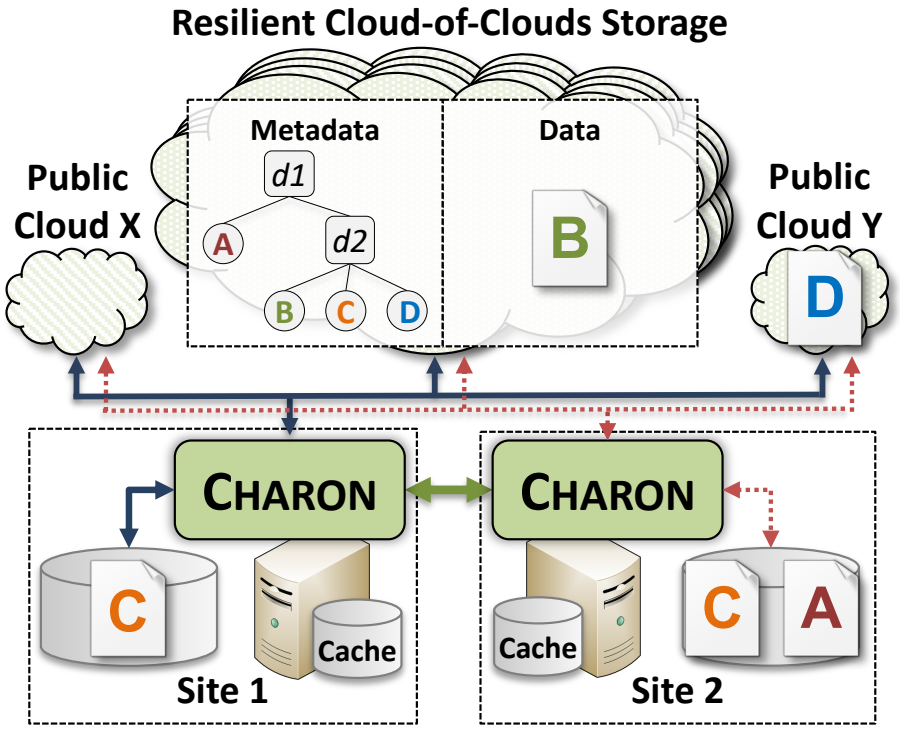
- A file system that makes transparent the use of cloud-of-clouds
  - Separation of FS data and metadata on different (storage) systems
  - Consistency-on-close semantics



Charon

[IEEE TCC'21]

- Address the limitations of SCFS
  - Avoid custom servers in the cloud
  - Deals with big files
- Data can be stored in diverse locations
- Employs three storage constructions
  - **Write-once registers** to store file blocks
  - **Single-writer registers** to store metadata objects (which are updatable)
  - **Composite lease objects** to lock directories for a single writer



Cloud-of-Clouds Practical Usage

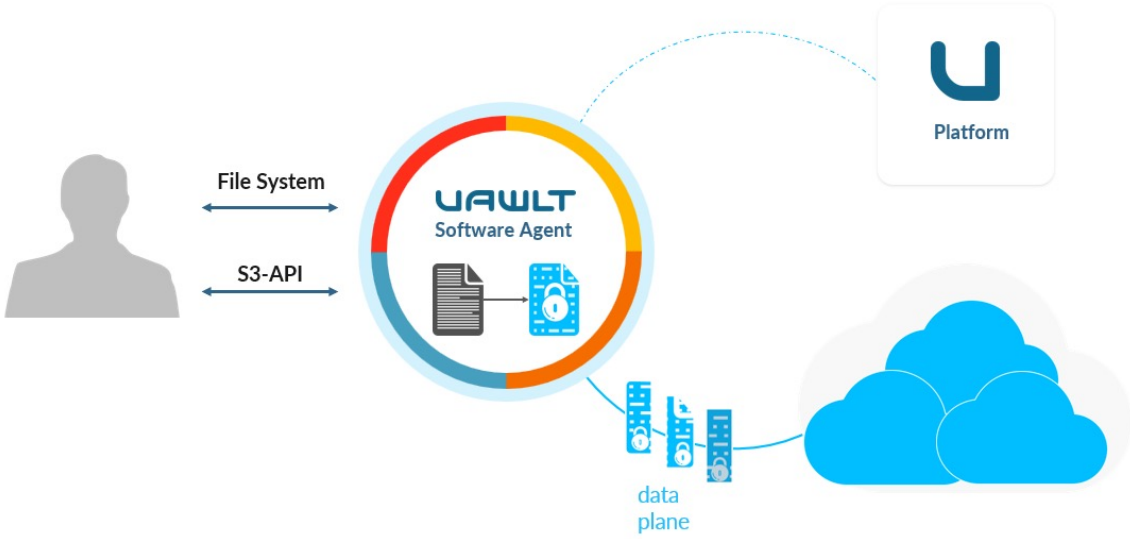
Difficulty #1
The user needs to **choose** the *n* most fitting **cloud services** for instantiating the system

Difficulty #2
The user needs to **set up accounts** on all these clouds and properly **manage** them

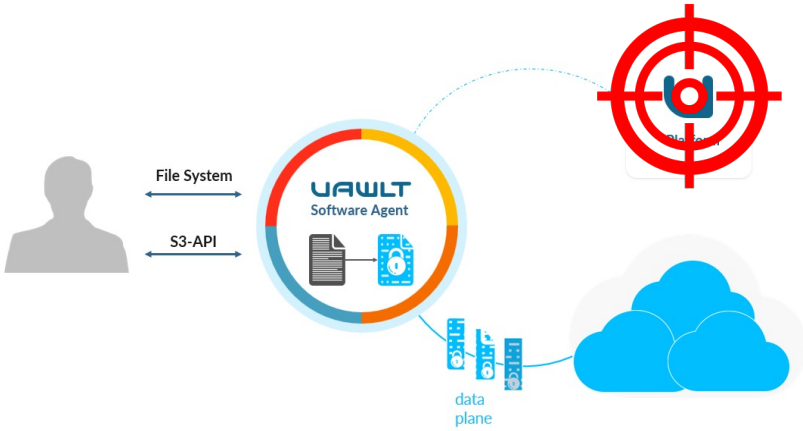
Difficulty #3
The system needs to be **configured** to use the clouds to decrease costs and improve performance

Vawlt

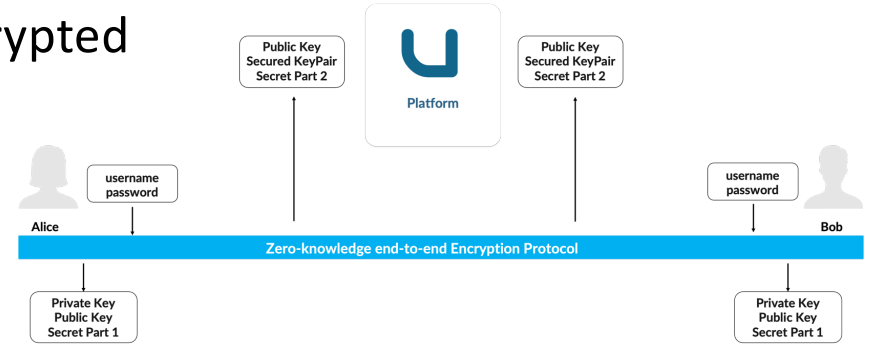
- Startup launched in 2018 offering a cloud-of-clouds storage platform
- At this point, it secured more than 3M euros from VC funds



Vawlt Security Model

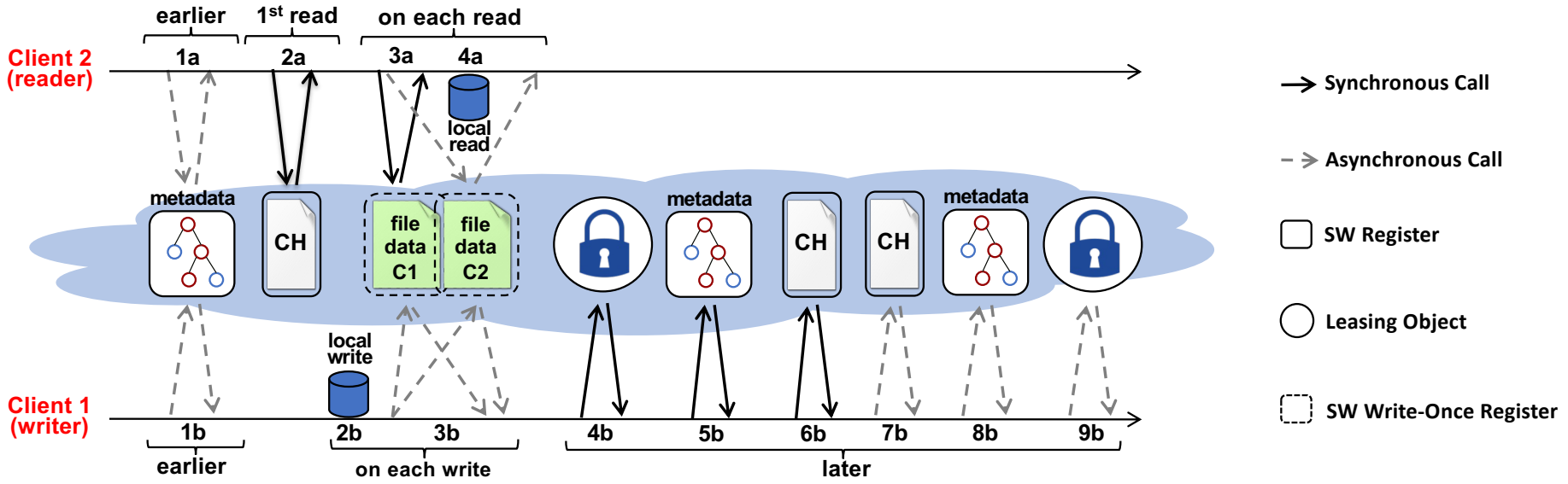


- Vawlt server is a single point of trust
- If it is unavailable
  - It is not possible to create or mount volumes
- If it is compromised
  - The adversary may have access to cloud credentials of users' volumes
  - He/she can access/modify/delete users' data
  - Confidentiality is preserved as the data is encrypted
    - Vawlt does not store users' master keys
    - Thanks to the **end-to-end encryption protocol**
  - Deleted files can be recovered in most clouds



Vawlt Operation

- Lazy locking provides a more responsive experience for users

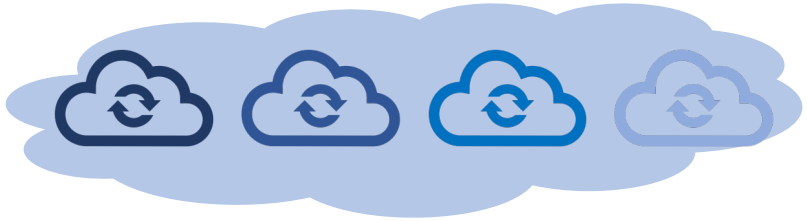


Productization

- Many hours of development were required to redesign the system to multiple platforms, build the UI, test, and maintain deployed versions
- Wrong assumptions/design decisions that needed to be corrected
  - **Metadata scalability**: metadata sometimes doesn't fit memory
    - One client had 200GB of metadata!!!
  - **Mapping of user requirements**: users like it simple and predictable
  - **Data retention**: it is a fundamental requirement for recovering files
    - Soft-delete, versioning, immutability policies (for dealing with Ransomware)
  - **Integration with other systems**: S3 API is often preferable than a POSIX file system; integration with key management systems, LDAP, etc.
  - **Bandwidth throttling**: use of clients' bandwidth needs to be limited
  - **Encrypted scalable cache**: the client is not always trusted; a single file can be bigger than the cache

Conclusions

Final Remarks



- Summary of presentation:
  - Data-centric Byzantine fault-tolerant constructions for storage
  - Academic systems: DepSky, SCFS, and Charon
  - Commercial product: Vawlt
- Remarkably, Charon/Vawlt replication is built using only the described storage constructions (a perfect match between theory and practice)
- However, Vawlt commercial viability required much more development
  - To the best of our knowledge, there is no similar system
  - Vawlt has more than 200 customers with 2000 TBs
  - See more on https://vawlt.io (they are always hiring…)

References

- Alysson Bessani, Miguel Correia, Bruno Quaresma, Fernando André, Paulo Sousa. **DepSky: Dependable and Secure Storage in a Cloud-of-Clouds.** ACM Transactions on Storage. Vol. 9, Num. 4. November 2013.
- Alysson Bessani, Ricardo Mendes, Tiago Oliveira, Nuno Neves, Miguel Correia, Marcelo Pasin, Paulo Verissimo. **SCFS: a Shared Cloud-backed File System.** USENIX'14: USENIX Annual Technical Conference. June 2014.
- Tiago Oliveira, Ricardo Mendes, Alysson Bessani. **Exploring Key-Value Stores in Multi-Writer Byzantine-Resilient Register Emulations.** OPODIS'16: The 20th International Conference On Principles Of DIstributed Systems. December 2016.
- Ricardo Mendes, Tiago Oliveira, Vinicius Cogo, Nuno Neves, Alysson Bessani. **Charon: A Secure Cloud-of-Clouds System for Storing and Sharing Big Data.** IEEE Transactions on Cloud Computing. Vol. 9, Num. 4. October 2021.
- Diogo Avelãs, Hasan Heydari, Eduardo Alchieri, Tobias Distler, Alysson Bessani. **Probabilistic Byzantine Fault Tolerance.** PODC'24: The 43rd ACM Symposium on Principles of Distributed Computing. June 2024.

Obrigado!

- Alysson Bessani
  - anbessani@fc.ul.pt
  - www.di.fc.ul.pt/~bessani