Restart-Rollback: a novel fault model for Efficient TEE replication

Baltasar Dinis

Instituto Superior Técnico (ULisboa) / INESC-ID / MPI-SWS

NDSS – February 28 - March 2, 2023

joint work with
Rodrigo Rodrigues (INESC-ID),
Peter Druschel (MPI-SWS),

Trusted execution environments (TEEs)

Security properties of TEEs

Teechain: A Secure Payment Network with Asynchronous Blockchain Access

Joshua Lind　　Oded Naor　　Ittay Eyal

ROTE: Rollback Protection for Trusted Execution

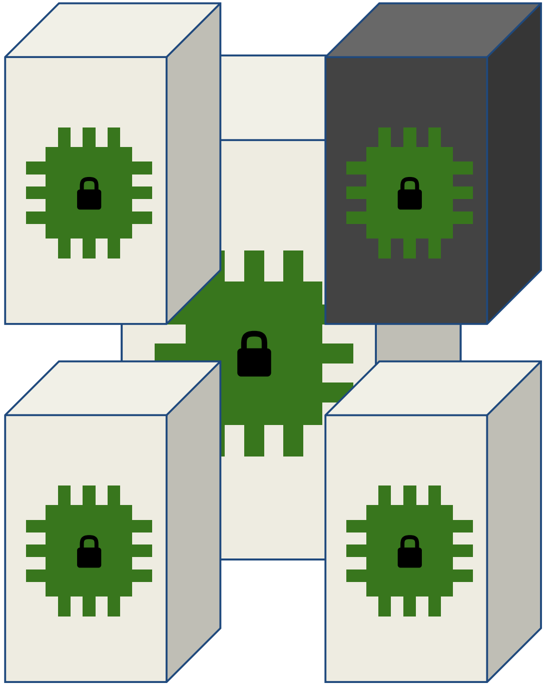
Sinisa Matetic　　Mansoor Ahmed　　Kari Kostiainen　　Aritra Dhar
ETH Zurich　　ETH Zurich　　ETH Zurich　　ETH Zurich

Avocado: A Secure In-Memory Distributed Storage System

Maurice Bailleu[1], Dimitra Giantsidi[1]
Vasilis Gavrielatos[1], Do Le Quoc[2]; Vijay Nagarajan[1], Pramod Bhatotia[1,3]
[1] *University of Edinburgh*　[2] *Huawei Research*　[3] *TU Munich*

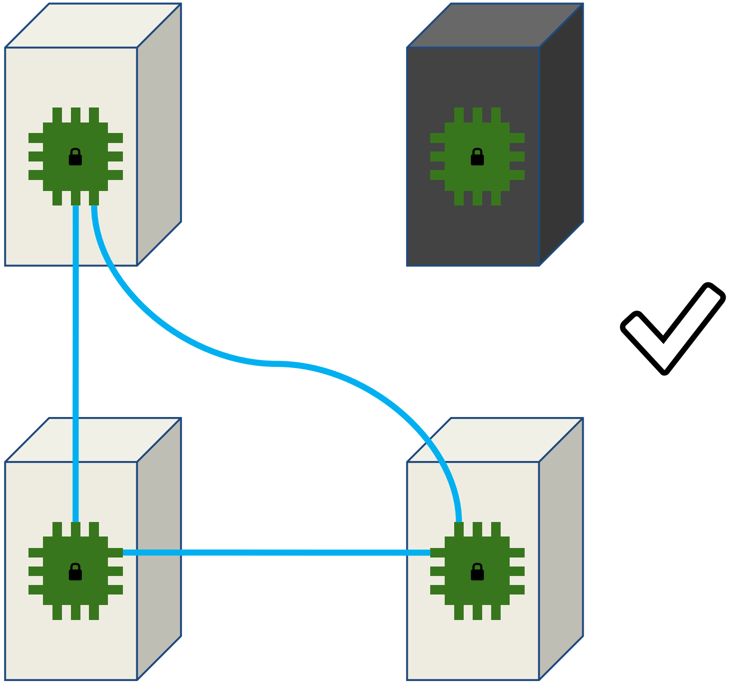
Robust P2P Primitives Using SGX Enclaves

Yaoqi Jia[1]　Shruti Tople[2]　Tarik Moataz[3]　Deli Gong[1]　Prateek Saxena[4]　Zhenkai Liang[4]
[1] ACM Member {jiayaoqijia, gnnnnng}@gmail.com [2] Microsoft Research t-shtopl@microsoft.com
[3] Aroki Systems tarik@aroki.com [4] National University of Singapore {prateeks, liangzk}@comp.nus.edu.sg



What is the right model for TEE replication?

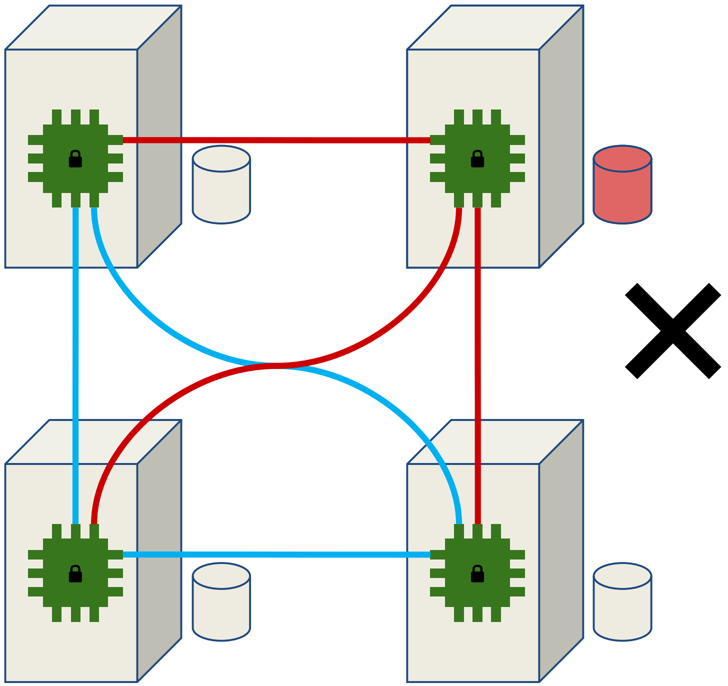
- Crash Fault Tolerance (CFT) assumes
  - Correct computation
  - Replicas may fail silently



What is the right model for TEE replication?

- Persistent storage complicates things, since it is not under direct control of the TEE

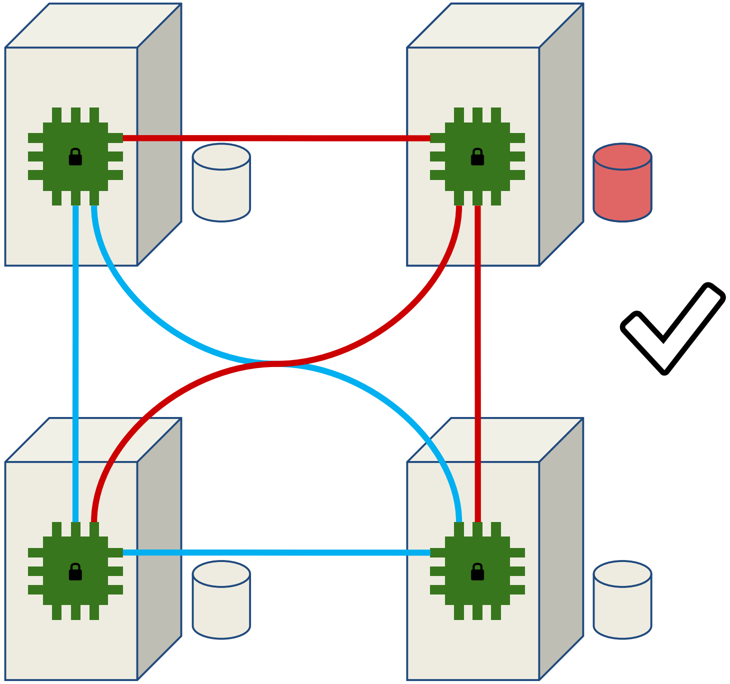
- An adversary may replace the persistent storage with an older version: a **rollback attack**



What is the right model for TEE replication?

- Byzantine Fault tolerance (BFT) tolerates arbitrary faults
  - By extension, rollbacks

- BFT mechanisms are **expensive**



What is the right model for TEE replication?

- CFT is sufficient, ***unless*** the TEEs have external persistent state
  - If so, the system is vulnerable to **rollback attacks**

- For this reason, BFT is necessary
  - More expensive and intuitively sounds **pessimistic**, given TEE guarantees

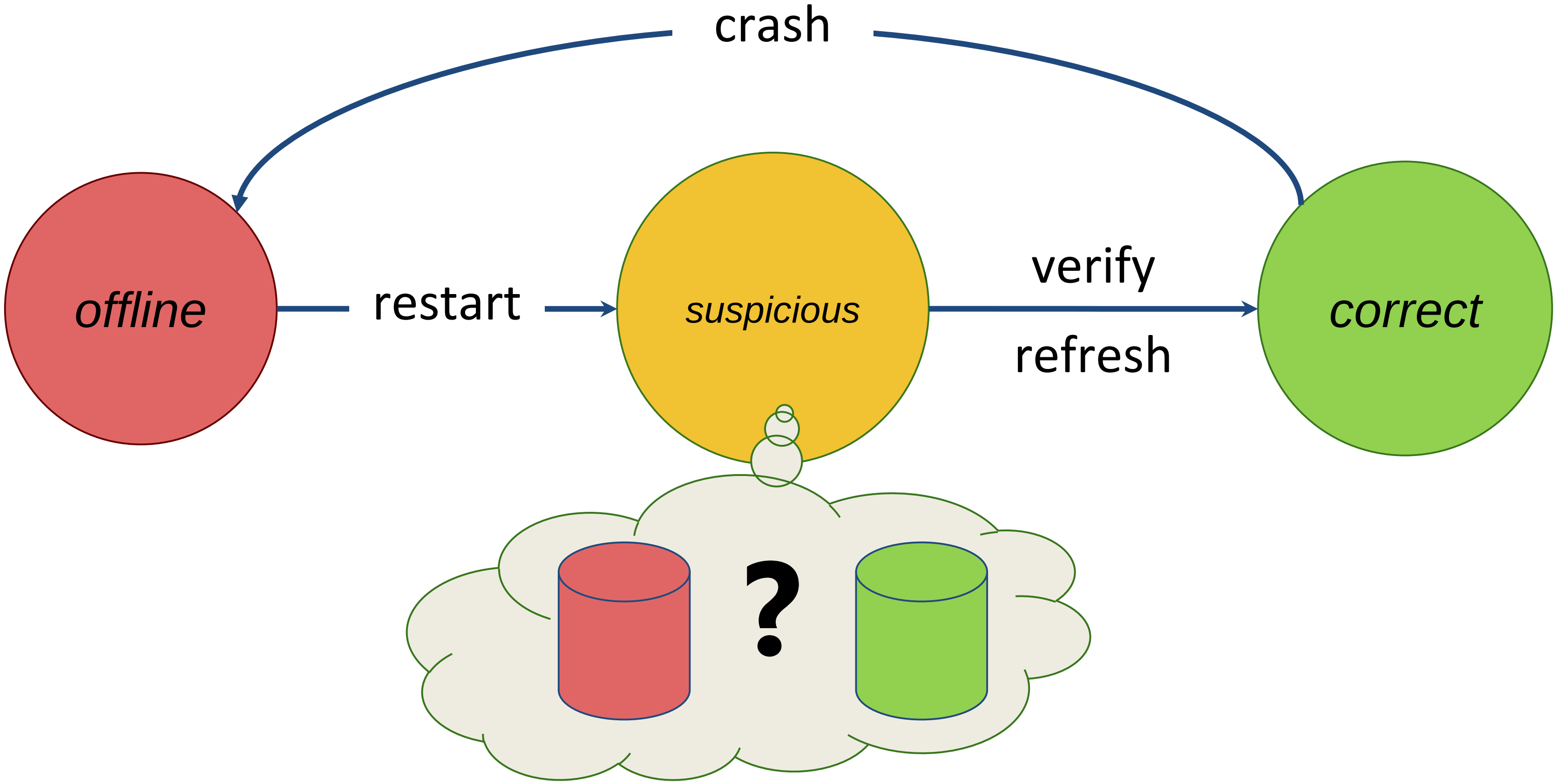
- Can we achieve the best of both worlds?

Yes, if the fault model accurately reflects TEE behavior

Talk outline

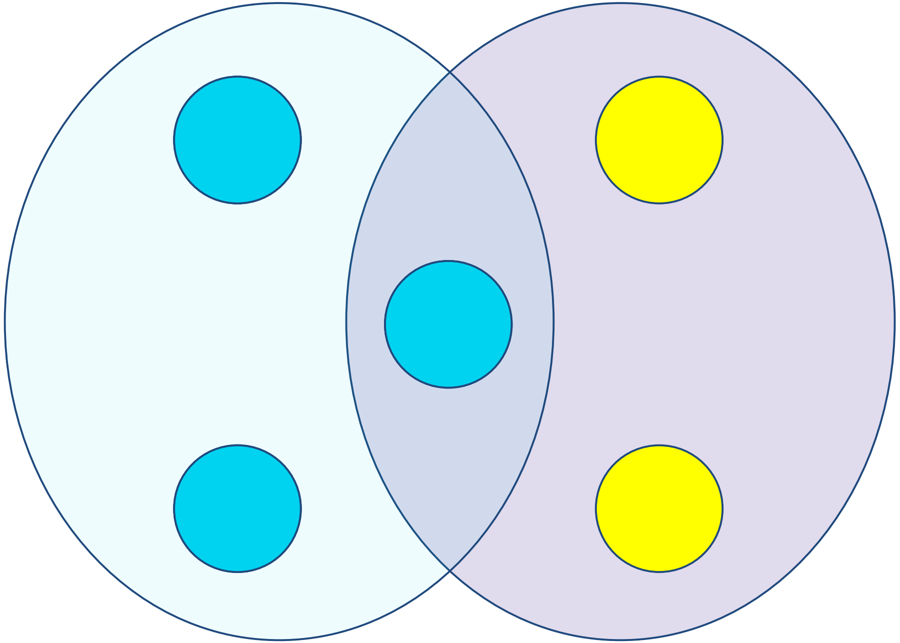
- Motivation
- The Restart-Rollback fault model
- Adapting replication protocols for the new model
- TEEMS: a metadata service for TEE-grade cloud storage

Restart-rollback model

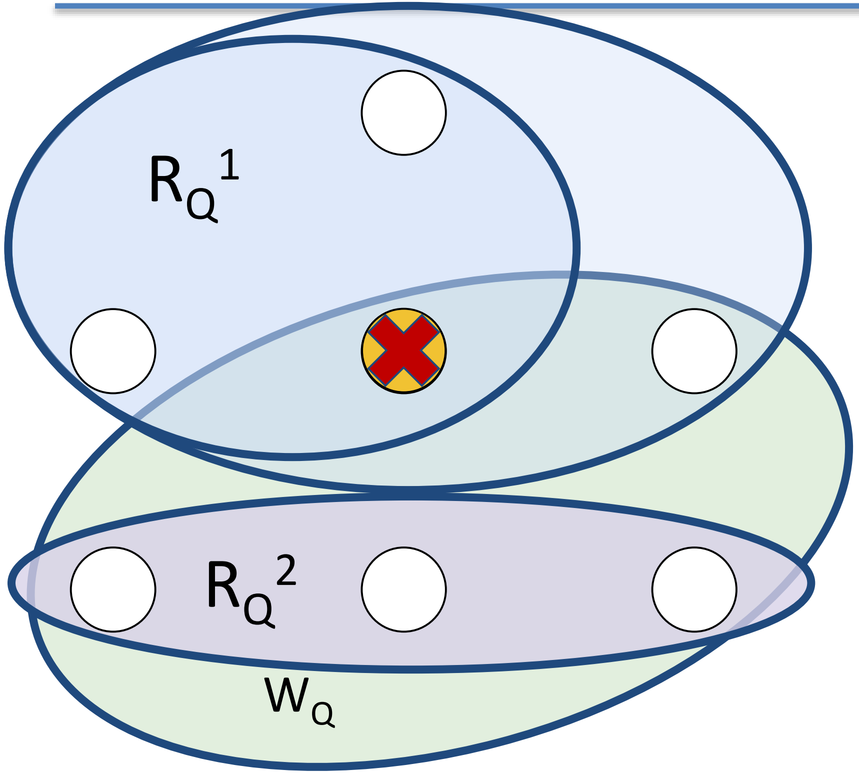


We have a fault model, now what?

- A major primitive for replication protocols are **quorums**:
  - A subset of replicas required to execute an operation
  - Guarantee correctness because pairs of quorums **intersect**
- We need a **quorum system** in the RR model



RR Quorum System Example



- Read and write quorums intersect in at least one non-rolled back replica
- They have different sizes (i.e, they are **asymmetric**)
- Two read quorums may not intersect
- Read quorums scale up when needed (they are **dynamic**)

Talk outline

- Motivation
- The Restart-Rollback fault model
- Adapting replication protocols for the new model
- TEEMS: a metadata service for TEE-grade cloud storage

Adapting CFT replication protocols

1. After a restart, flag replies with **suspicion**

2. Adjust quorum sizes
   a. When reading system state, use R_Q
   b. When writing system state, use W_Q

3. Deal with split brain
   1. Because read quorums may not intersect
   2. This will be protocol specific

Adapting replication protocols

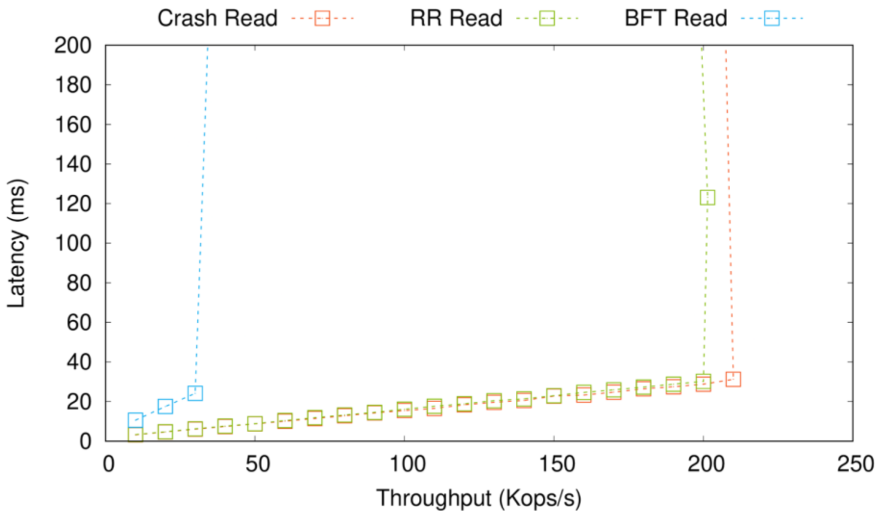
Distributed register

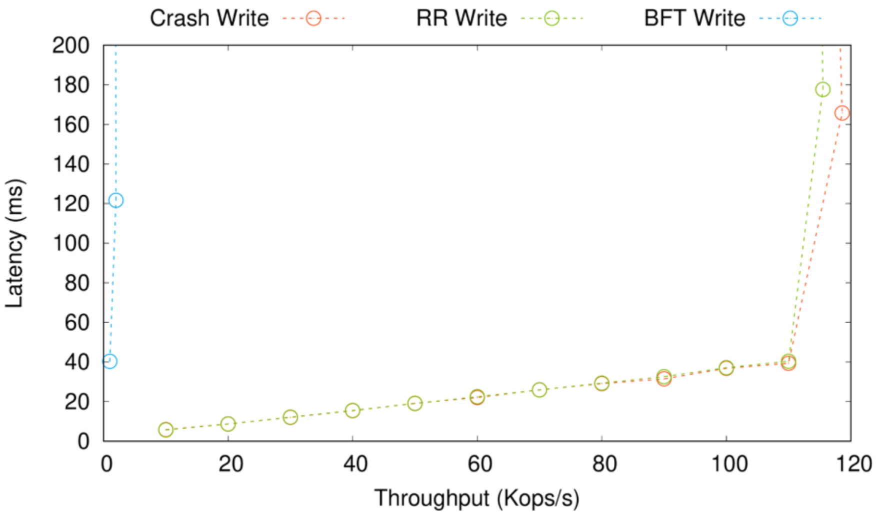
- Read/write operations
- CFT: ABD
- BFT: Byzantine Quorums

State machine replication

- Any deterministic service
- CFT: Paxos, Raft
- BFT: PBFT, BFT smart, etc.

Evaluation of the read/write protocols





Talk outline

- Motivation
- The Restart-Rollback fault model
- Adapting replication protocols for the new model
- TEEMS: a metadata service for TEE-grade cloud storage ☜

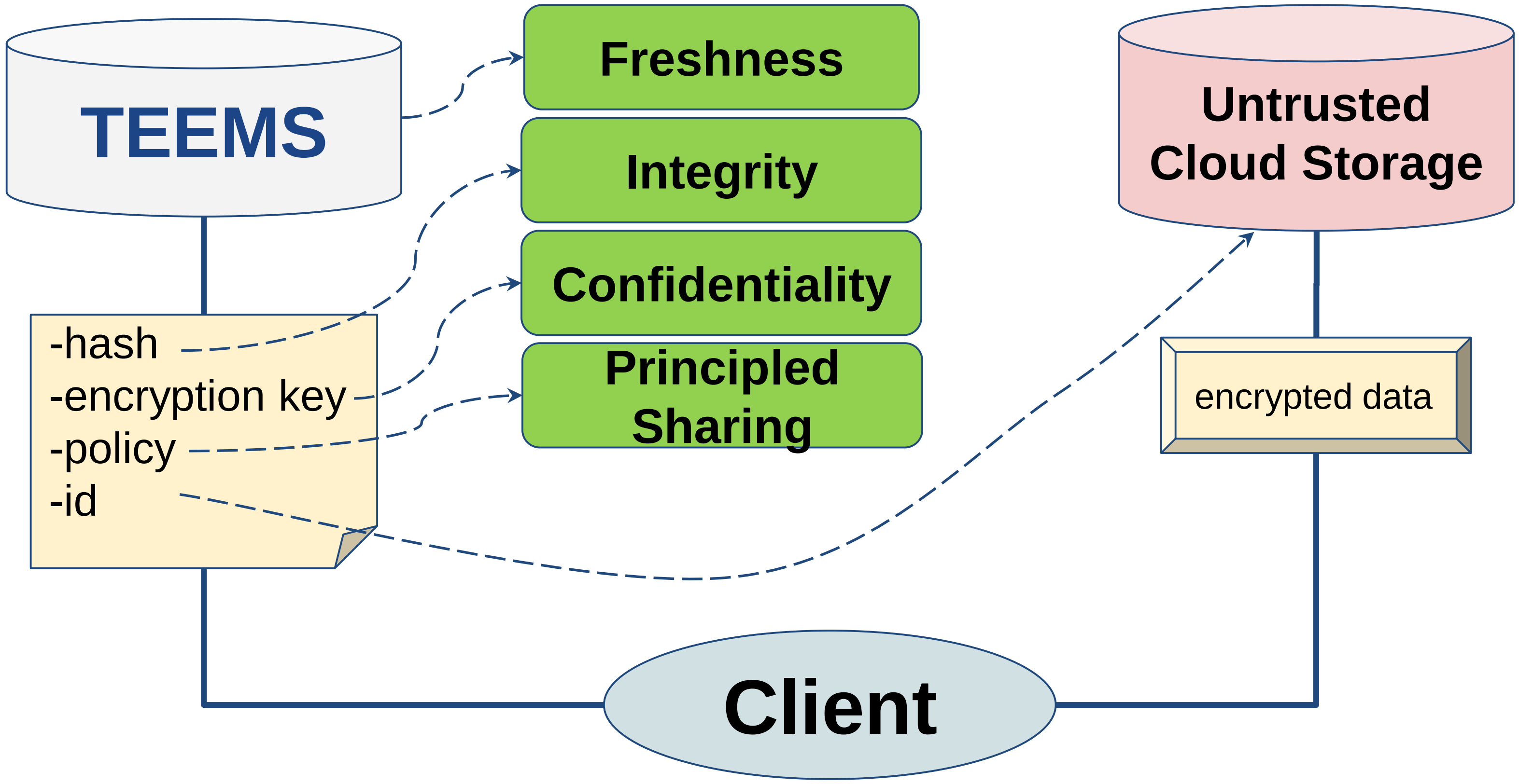
TEEMS (TEE metadata service)

- TEEs are key element to increasing trust in cloud services
  - E.g., Azure confidential computing, Google confidential VMs
- Can we build cloud storage with similar guarantees?
  - Freshness
  - Integrity
  - Confidentiality
  - Principled Sharing

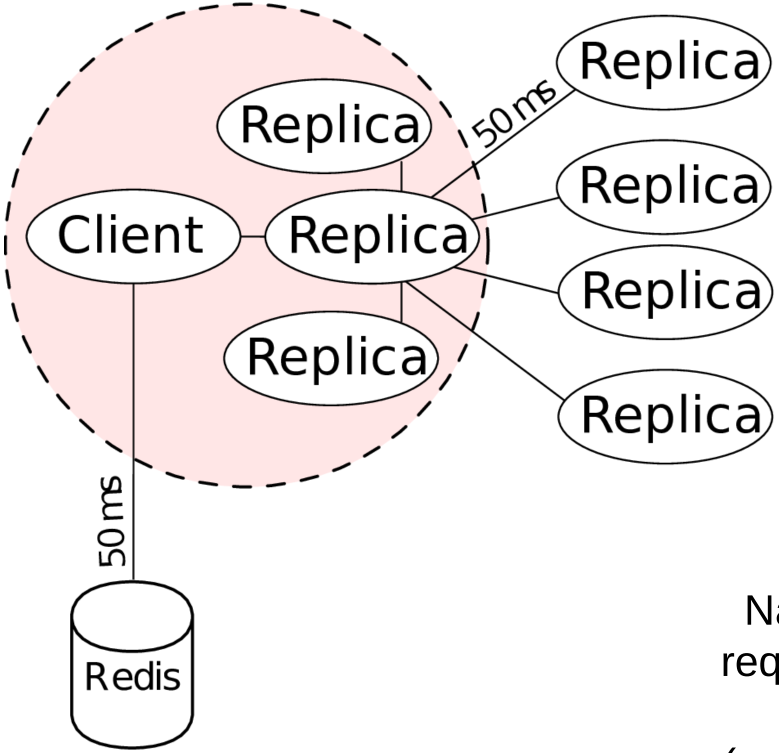
- Strawman solution: Encryption
  - does not ensure **freshness**
  - no mechanism for **principled sharing** among clients

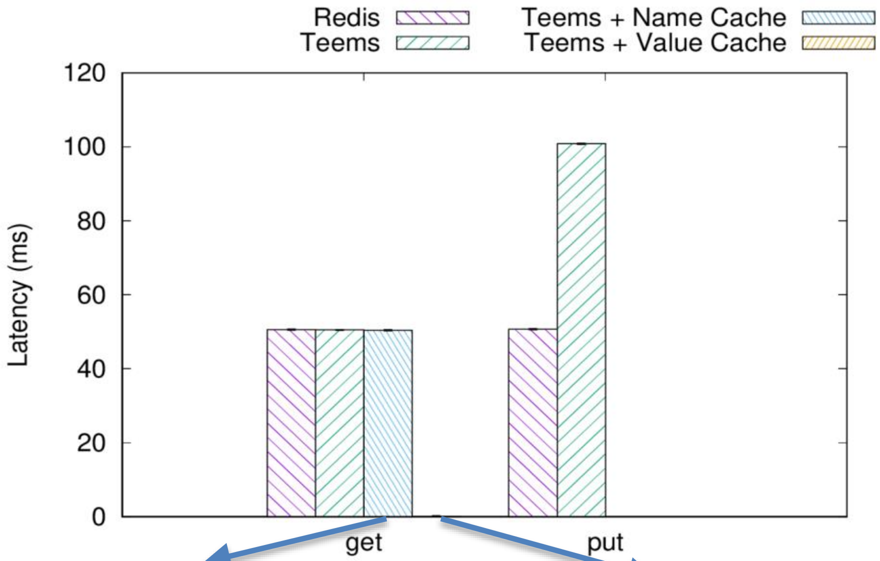
Leverage our new protocols to build **trusted metadata service**

TEEMS


TEEMS
Freshness
Integrity
Confidentiality
Principled Sharing
Untrusted Cloud Storage
-hash
-encryption key
-policy
-id
encrypted data
Client

Evaluation of TEEMS-based storage





Name hint still requires fetching the data (expensive step)

Caching the entire value allow for a fast (local) check of the metadata

Concluding remarks

- Restart-Rollback **precisely** models TEEs with persistent state
  - Ensuring correctness
- With RR, we can intuitively adapt existing CFT protocols, yielding similar performance
- Using RR, we built TEEMS a system that provides the analog to the "confidential VM" paradigm for cloud storage

Thank you

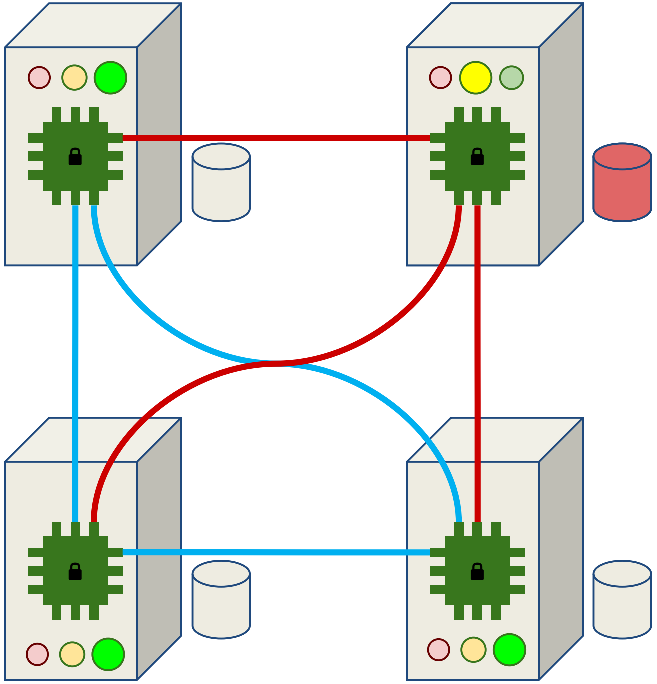
Adapting replication protocols

1. Pick your favourite CFT protocol (read/write register, replicated state machine)
   - Remember, **computation is correct**
2. Identify quorums where the protocol reads state
   - Replace with $\mathbf{R_Q}$
3. Identify quorums where the protocol writes state
   - Replace with $\mathbf{W_Q}$
4. We're done?

Restart-rollback model

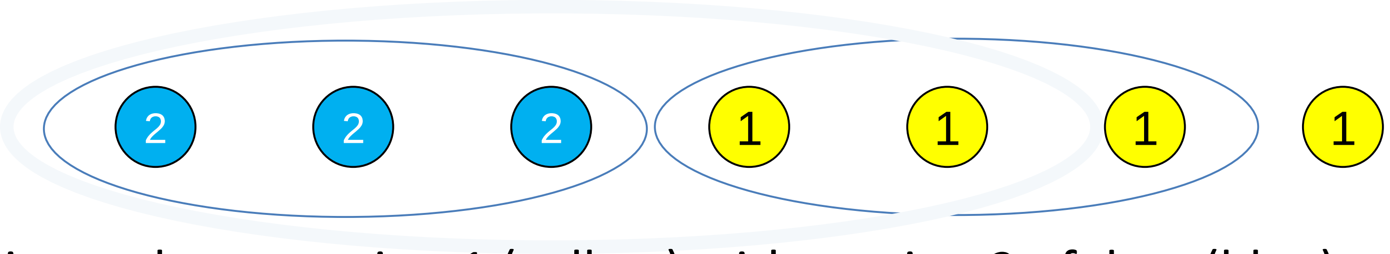
- If more than **F** replicas are simultaneously in the state, the system becomes **unavailable**

- If more than $\mathbf{M_R}$ replicas have simultaneously been rolled back (i.e, the storage is in the state) then the system suffers a **rollback**

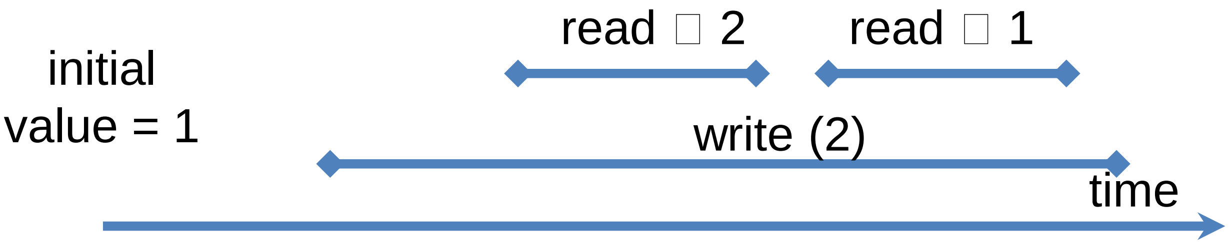


Reading in parallel with ongoing write

- $M_R = 4$, $F = 2$, $N=7$, $W_Q = 5$



- Write replaces version 1 (yellow) with version 2 of data (blue)
- First read sees unanimously version 2
- Second read sees unanimously version 1



Key replication design decision: fault model

Model	Parameters	N	Quorum Size
Crash [AD ICSE'76]	f	2f + 1	f + 1
Byzantine [PSL jACM'80]	f	3f + 1	2f + 1
Hybrid [MP TPDS'91, CLNV SRDS'02, C+ SOSP'09]	u, r	2u + r + 1	u + r + 1

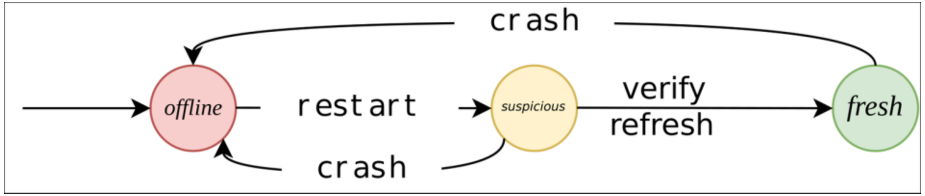
- Replication in the asynchronous model relies on quorums
- Smaller quorum sizes lead to more efficient operations

Deriving quorum systems

- During each quorum-RPC, count number of replies with suspicion flag: **s**
  - **s** is the number of replicas in the quorum that **may** have been rolled back
- Per-RPC maximum number of rolled back nodes: **min(s,M_R)**

M_R: simultaneous rollbacks

F: simultaneous crashes



Deriving quorum systems

M_R: simultaneous rollbacks
F: simultaneous crashes
s: restarts in this RPC
$min(s, M_R)$: maximum number of rolled back nodes in this RPC

- Correctness (safety) intersection requirement becomes:

  $R_Q + W_Q > N + min(s,M_R)$
- In conjunction with liveness constraints, we derive the following:

$$N = max(M_R,F) + F + 1$$
$$W_Q = max(M_R,F) + 1$$
$$R_Q = F + min(s, M_R) + 1$$

CFT Distributed Register

```
Read()

Send <Read> to all replicas
Wait for majority of replies
if unanimous:
        return value
else // writeback
        Send <Write, value, hts>
                to replies with ts < hts
        Wait for majority of replies
return value
```

```
Write(value)

Send <Read> to all replicas
Wait for majority of replies
Send <Write, value, hts + 1>
Wait for majority of replies

return
```

Distributed Register in the new model

```
Read()

Send <Read> to all replicas
Wait for RQ replies
if unanimous:
        return value
else // writeback
        Send <Write, value, hts>
        to replies with ts < hts
        Wait for WQ replies
return value
```

```
Write(value)

Send <Read> to all replicas
Wait for RQ replies
Send <Write, value, hts + 1>
Wait for WQ replies

return
```

But: Read quorums may not intersect! ⚠