Brief Announcement: Auditable Register Emulations

Vinicius V. Cogo
https://www.di.fc.ul.pt/~vcogo/
vcogo@fc.ul.pt

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

Full version: *https://arxiv.org/abs/1905.08637*

DISC 2021: October 4–8, 2021

The Question

How to ***extend***

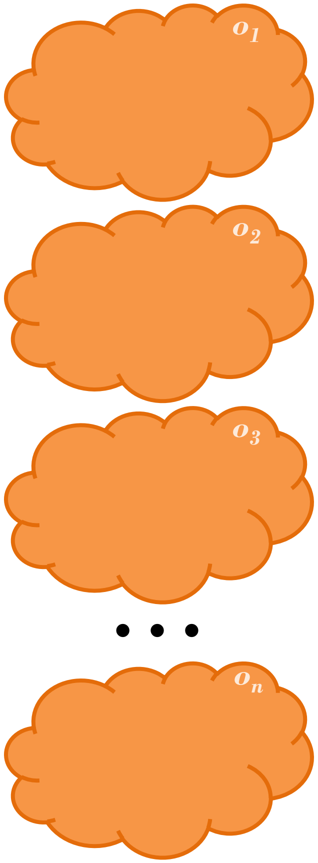
resilient storage *emulations*

with *the capability of* ***auditing***

who *has* ***effectively read*** *data?*

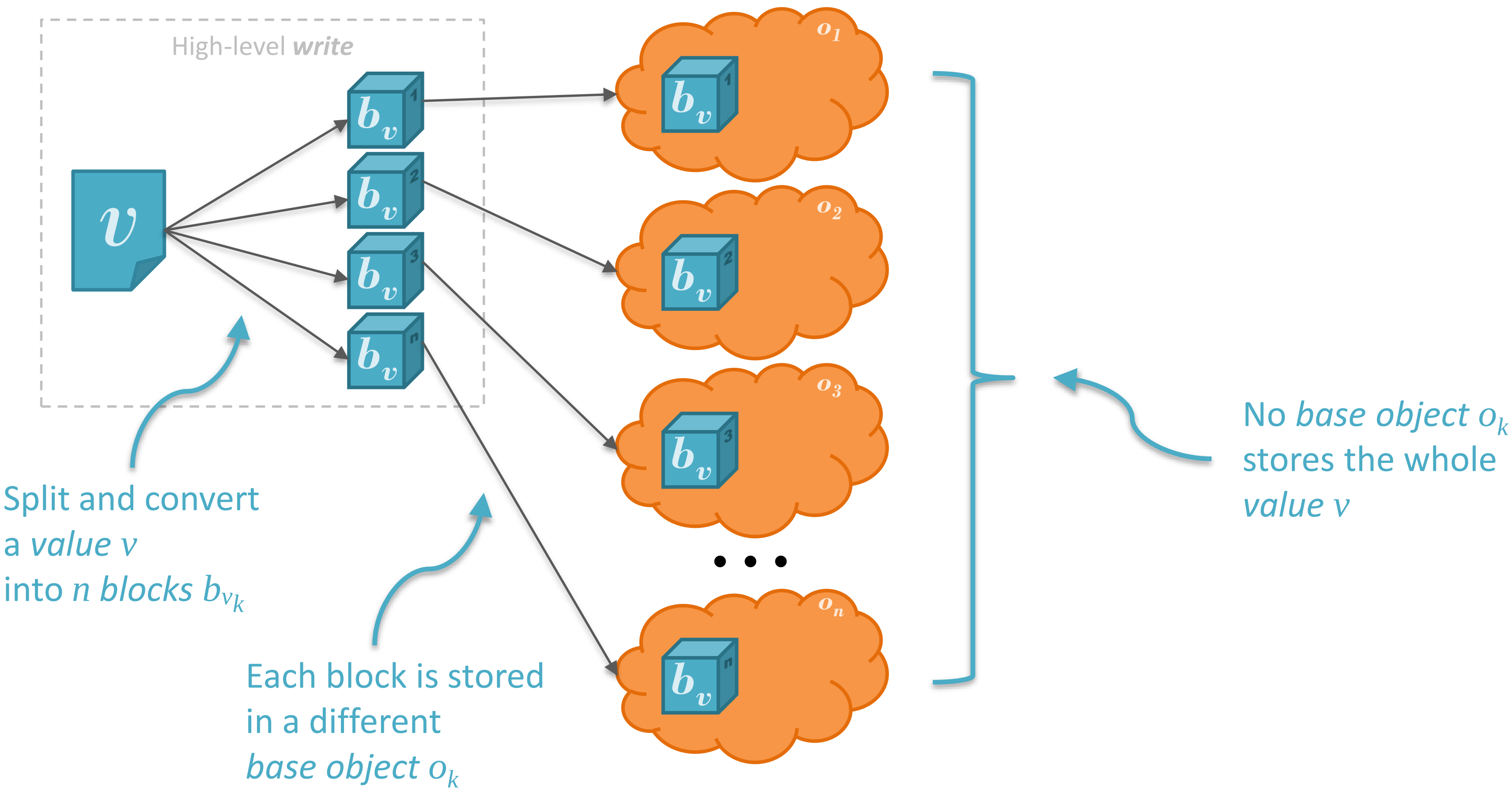
Resilient Storage System

- Composed of ***n*** **storage objects** (e.g., RW registers)



Resilient Storage System

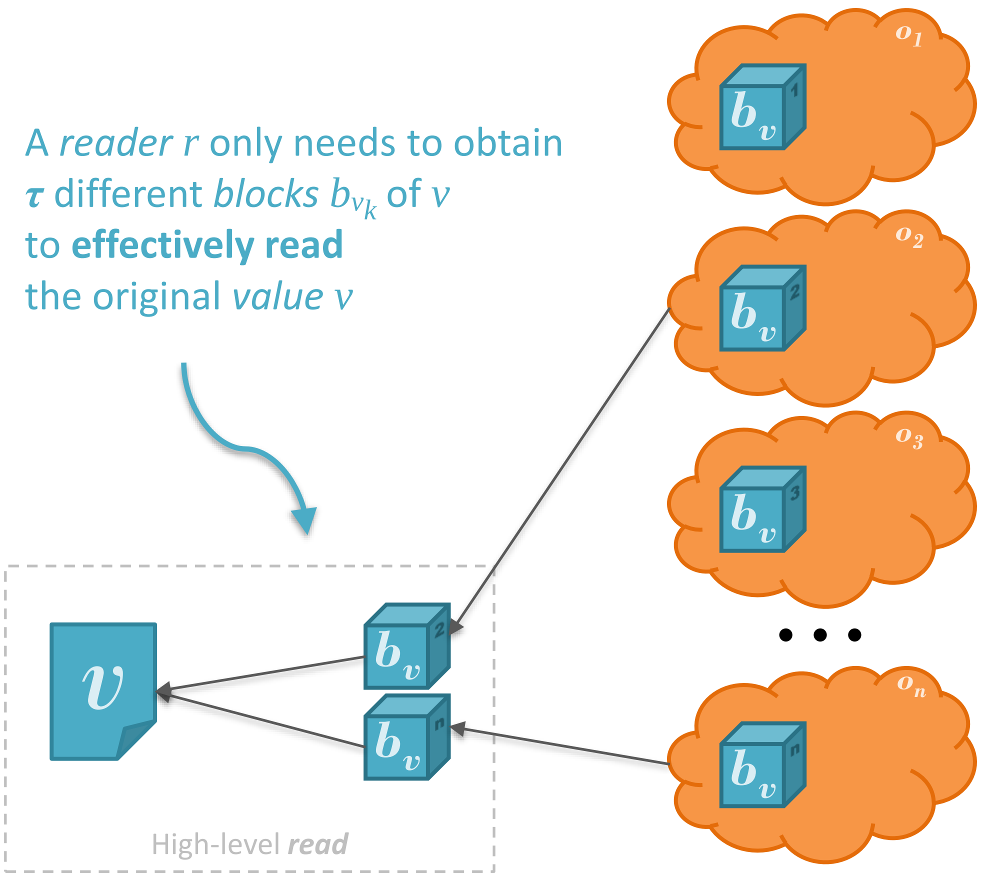
- **Information dispersal** techniques (e.g., secret sharing, erasure codes)
- Write operations



Resilient Storage System

- Information dispersal techniques (e.g., secret sharing, erasure codes)
- **Read** operations

A *reader* r only needs to obtain
τ different *blocks* b_{v_k} of v
to **effectively read**
the original *value* v



Back to The Question

How to ***extend***

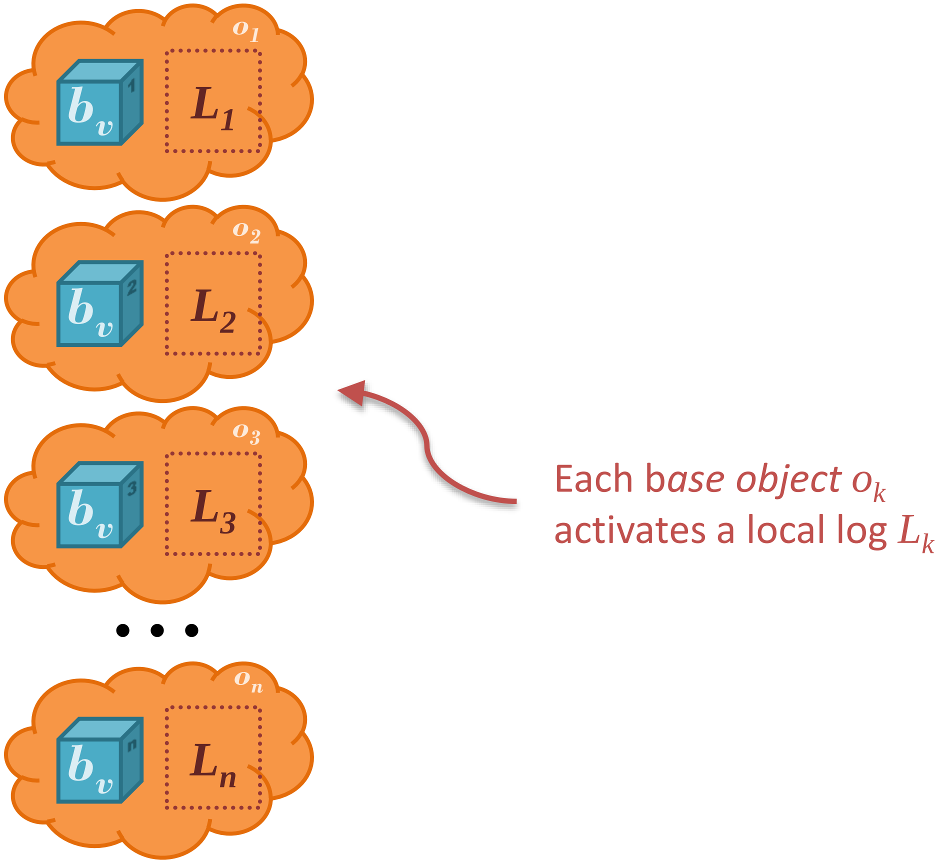
resilient storage *emulations*

with *the capability of* ***auditing***

who *has* ***effectively read*** *data?*

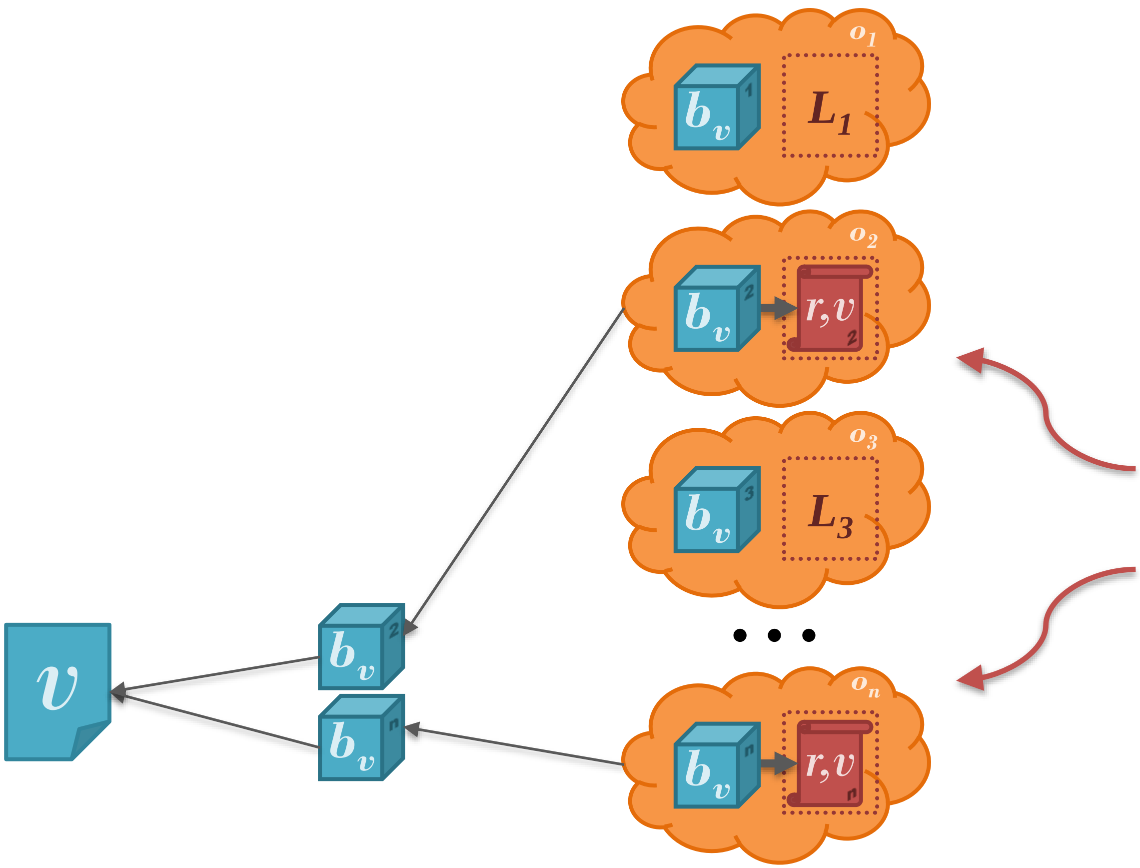
Auditable Resilient Storage System

- Extensions to make resilient storage auditable
- Base objects (i.e., RW registers) become **Loggable RW registers**



Auditable Resilient Storage System

- Extensions to make resilient storage auditable

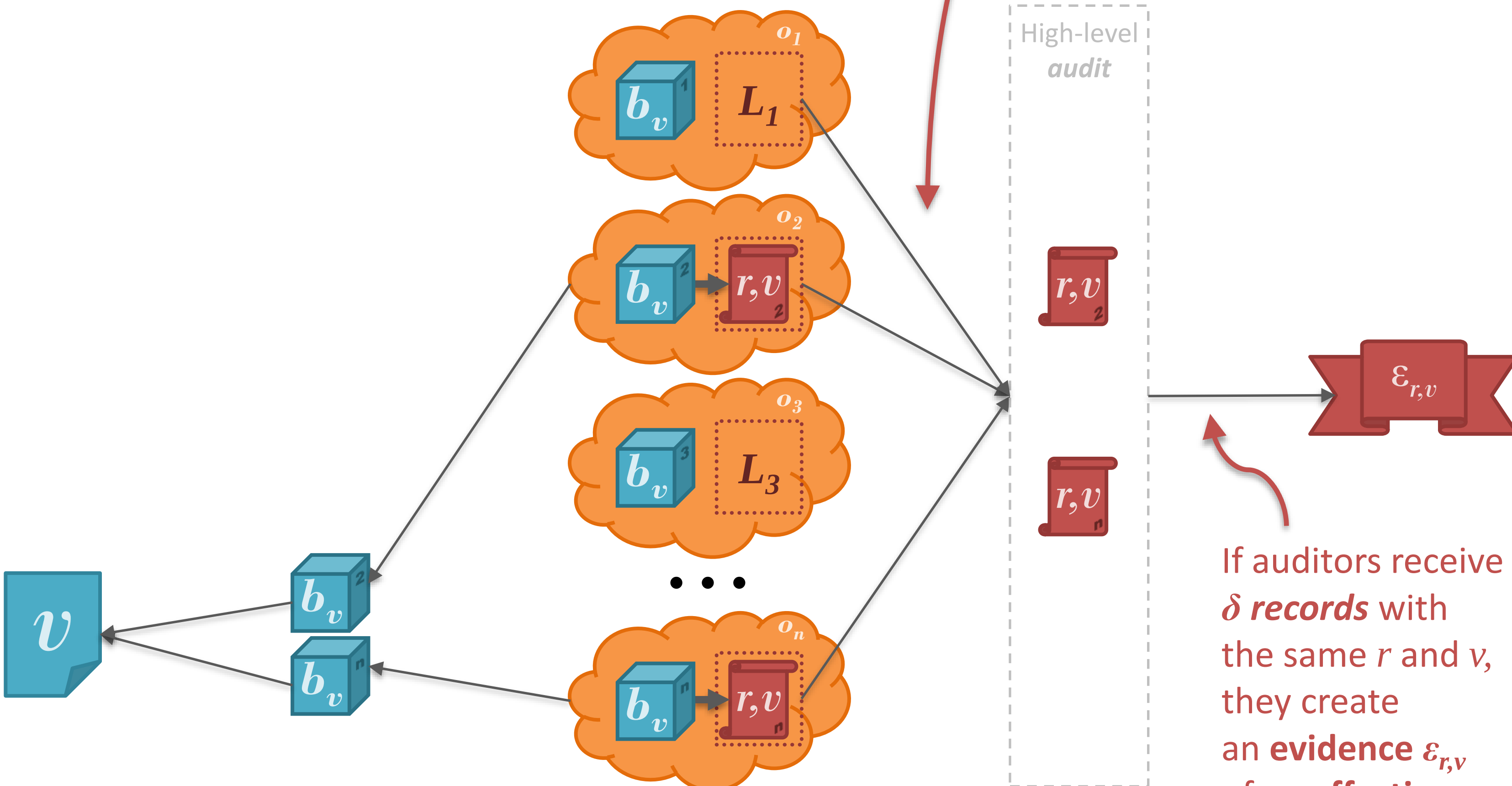


In each low-level *read* operation, the *base object* creates a **record** ***<reader r, label of value v>*** in its logs L_k

Auditable Resilient Storage System

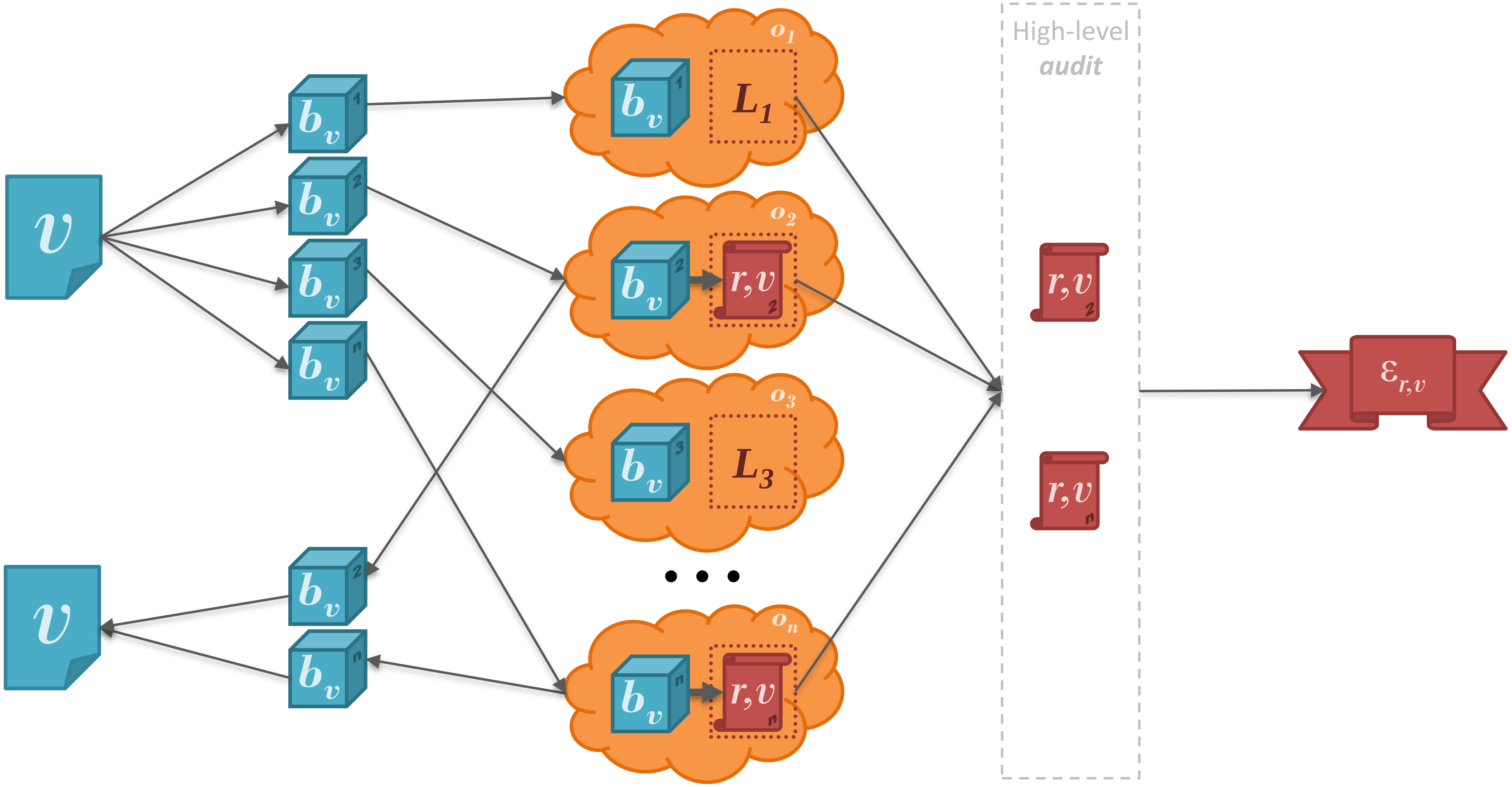
- Extensions to make resilient storage auditable

A **high-level *audit*** operation receives the logs L_k from a **quorum** of base objects



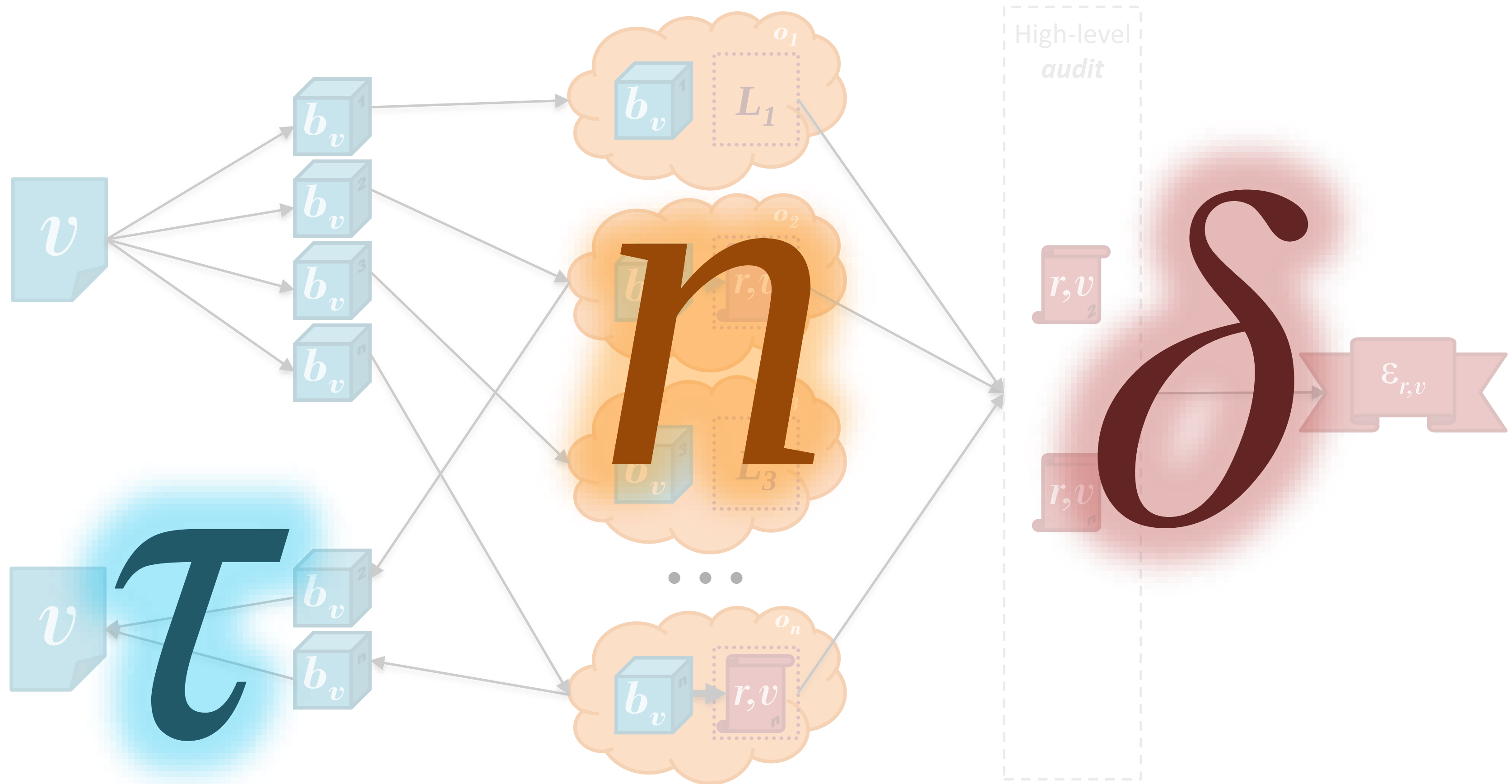
If auditors receive at least ***δ records*** with the same r and v, they create an **evidence** $\varepsilon_{r,v}$ of an **effective read**

Auditable Resilient Storage System



Auditable Resilient Storage System

- Lower-bounds to auditable resilient storage systems based on τ, n, and δ



System model

- Arbitrary number of **client** processes
  *(**writers**, **readers**, or **auditors**)*

- Clients interact with
  a set of ***n*** **base** storage **objects**
  by **invoking** operations
  (e.g., RPC on top of asynchronous reliable, authenticated channels)

- High-level **reads** are **fast**
  (i.e., 1 communication round-trip)

- **Audit** quorums are **available**
  (i.e., auditors wait responses from $n - f$ objects)

Fault model

- Faulty **writers** and **auditors** can only fail by **crashing**

- Faulty **readers** may be **Byzantine**

- Faulty **storage objects** can:
  - crash,
  - omit their blocks to readers,
  - omit read records to auditors, or
  - record nonexistent read operations.

*… fail by **omission** when accessing **their blocks***

*… fail **arbitrarily** when providing their **log records***

- No more than ***f*** **storage objects** are **faulty**

Properties of Auditable Registers

- **Completeness**
  Protect the system from readers obtaining data without being reported
  "Every value v effectively read by a reader is reported by auditors"

- **Accuracy**
  Protect correct readers from faulty objects trying to incriminate them
  - ***Weak accuracy:***
    "A reader that has never invoked a read will never be reported by auditors"
  - ***Strong accuracy:***
    "A reader that has never effectively read a value v will never be reported by auditors as having read it"

- Weak auditability = completeness + weak accuracy
- Strong auditability = completeness + strong accuracy

Results (in the model with fast reads)

1. **Auditability** is **impossible** with $\boldsymbol{\tau \leq 2f}$
   - *$\tau \geq 2f + 1$ guarantees that auditors receive*
     at least 1 record of every effective read from correct objects

2. **Weak auditability** requires $\boldsymbol{\tau \geq 3f + 1}$
   - *Guarantee completeness*
     and that f faulty objects are not enough to incriminate a reader

3. **Strong auditability** is **impossible**
   - *Auditors may receive **less records** of an **effective read***
     than of a read that is not effective
   - ***Impede** auditors from **choosing** any single **threshold** δ*
     without compromising either completeness or strong accuracy

Results (in alternative models)

4. **Signing reads** generically **overcomes** the **weak accuracy** lower bound and enables **weak auditability** with $\tau \geq 2f + 1$
   - *$\delta \geq 1$ becomes enough for weak accuracy because objects cannot forge records if they have never received a request from r*

5. **Totally ordering** (TO) operations or using **non-fast (NF) reads** enables **strong auditability** with $\tau \geq 3f + 1$
   - *TO limits the number of different values in storage objects to only one value*
   - *NF reads ensure correct readers only fetch blocks for the most up-to-date value*
   - *$\delta \geq f+1$ becomes enough for strong accuracy*

6. Combining **non-fast reads** with **specific signed read** requests enables **strong auditability** with $\tau \geq 2f + 1$
   - *NF enables correct readers to discover the most up-to-date value and to sign read requests specifically for that value*
   - *$\delta \geq 1$ becomes enough for strong accuracy*

Results (practical consequences)

- **Logging** accesses is **viable** since many storage **objects already support** it
  - *Data-centric public cloud services (e.g., AWS S3, Google Cloud Storage, Azure Storage)*
  - *Network file servers (e.g., FTP)*
  - *Local file systems (e.g., NTFS)*

- **Practical** resilient storage **emulations need f to $2f$ additional objects**
  - *(**compared to** their **original** lower bounds)*
    to support the auditability of who has effectively read data from them

- Requiring **more** base objects **decreases** the storage **overhead** of resilient storage systems
  - *since it allows **smaller blocks** for the **same** f*

Thank you!

Vinicius V. Cogo
https://www.di.fc.ul.pt/~vcogo/
vcogo@fc.ul.pt

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

Full version: *https://arxiv.org/abs/1905.08637*

DISC 2021: October 4–8, 2021