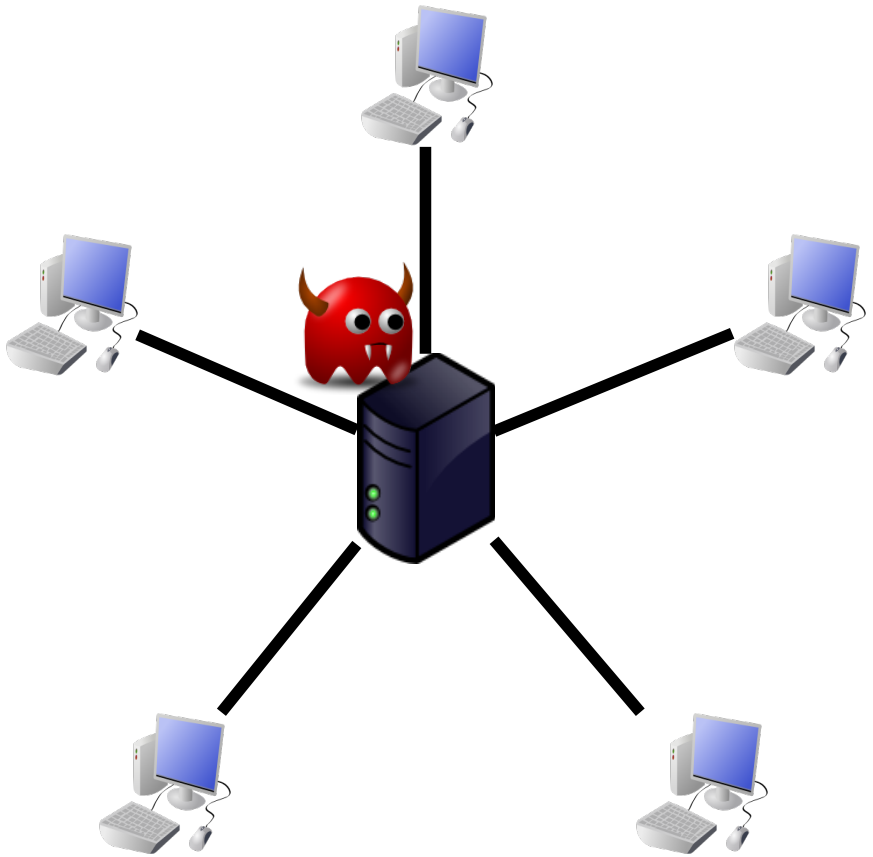
Secure Untrusted Data Repository (SUNDR)

Jinyuan Li, Maxwell Krohn, David Mazières, Dennis Shasha
NYU Department of Computer Science

(as presented by Lonnie Princehouse)

What can a malicious NFS server do?

- Integrity and consistency attacks
  - Arbitrarily change data
  - Falsify clients' actions
  - Lose changes made by clients
  - Present inconsistent views



- Confidentiality attacks
  - Read confidential data

- Availability attacks
  - Ignore user requests

Read-only filesystems with cryptography

... such as SFSRO and CFS.

Block Store

Blocks keyed by hash

put(H(block), block)

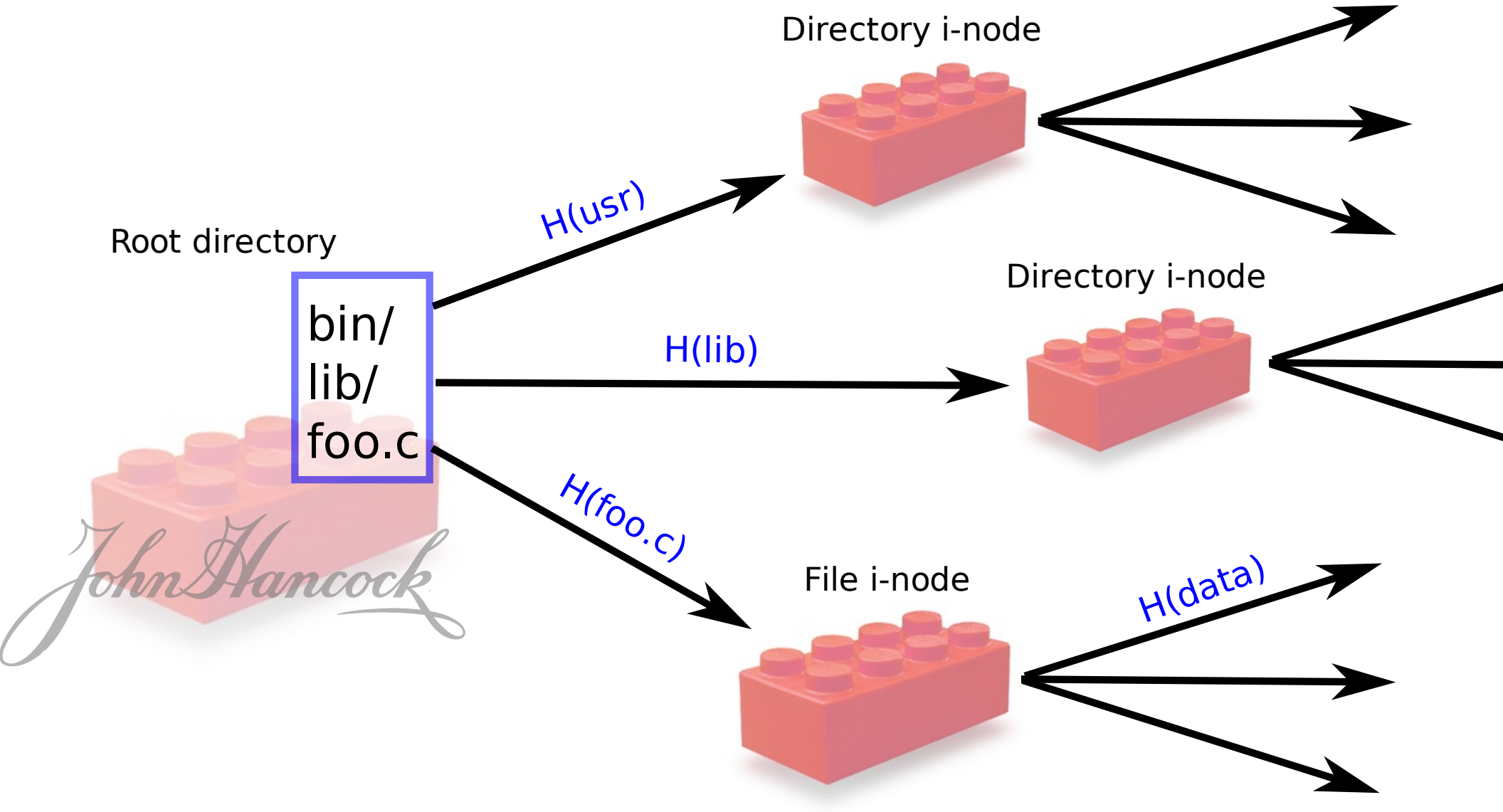
get(H(block))

Signed blocks keyed by public key

put(K, block)

get(K)

Read-only filesystems with cryptography



Read-only filesystems with cryptography

- <key,block> pairs are self-verifying
  - Cannot be forged
- Entire filesystem is self-verifying
  - Can use for any data structure
  - For example, B+Trees
- Signed blocks can be overwritten
  - Limited write capability for SFSRO and CFS

What can a malicious SFSRO server do?

- Integrity and consistency attacks
  - Arbitrarily change data — Detectable
  - Falsify clients' actions — Not applicable
  - Lose changes made by clients — Not applicable
  - Present inconsistent views — No*
- Confidentiality attacks
  - Read confidential data — Not applicable

- Availability attacks
  - Ignore user requests

Comparison

	Writeable	Distributed	Decentralized	Integrity without trusted server
NFS	✓			
SFSRO		–	–	✓
CFS		✓	✓	✓
SUNDR	✓			–

✓ = Yes

– = Sort of

Secure Untrusted Data Repository (SUNDR)

- How could we make SFSRO read-write?

  (While protecting ourselves from a malicious server)

Secure Untrusted Data Repository (SUNDR)

- How could we make SFSRO read-write?

  (While protecting ourselves from a malicious server)

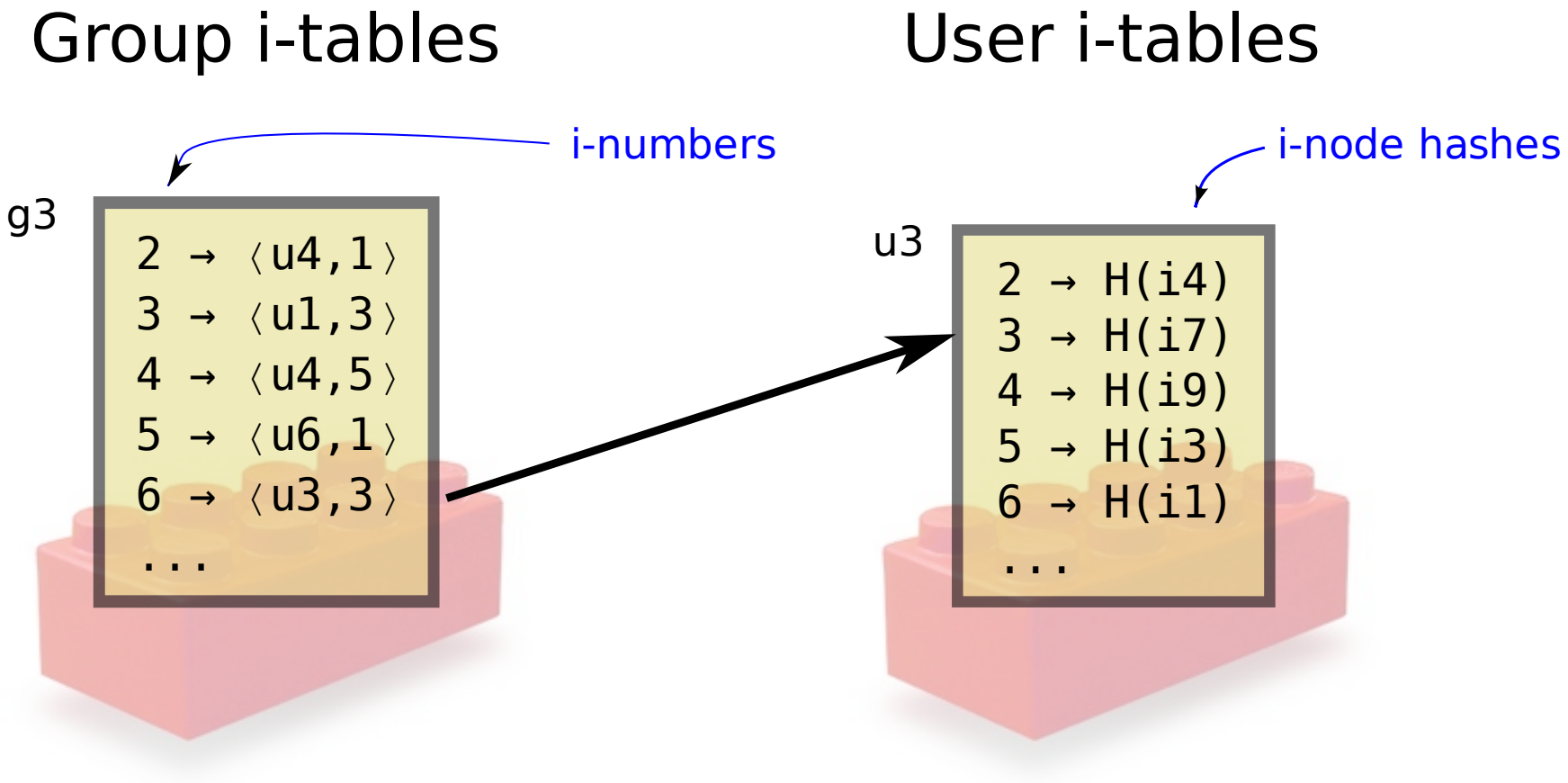
  - Need some concept of users, permissions
  - Prevent server from forging writes
  - Order of read/write operations must be preserved

Secure Untrusted Data Repository (SUNDR)

- Principals (p)
  - Users (u) and groups (g)
  - Each user has public/private key pair
  - One superuser
  - All users know superuser's public key
- Servers
  - Block server stores (key,block) pairs
  - Consistency server does everything else
  - Can be on different machines

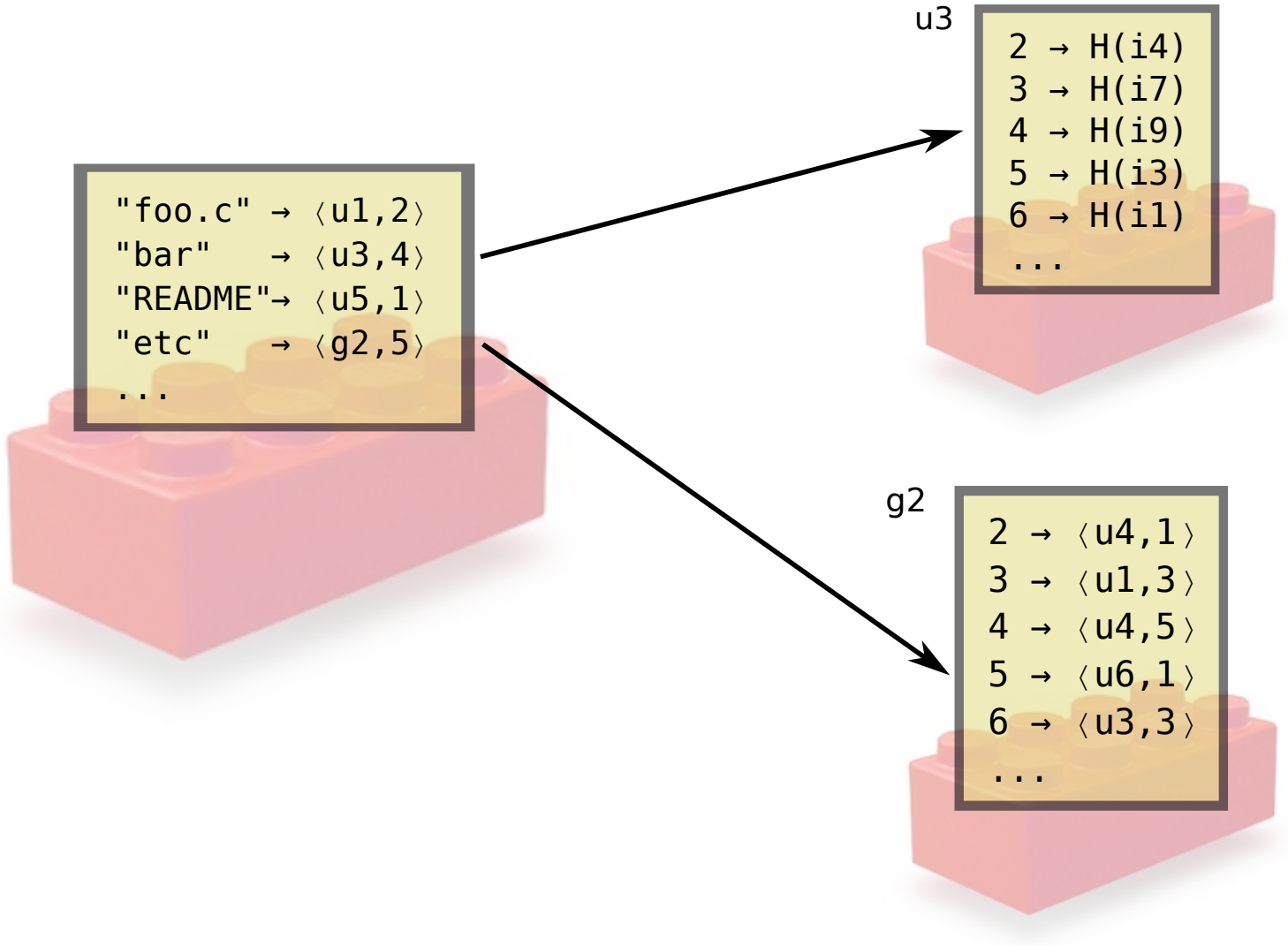
Secure Untrusted Data Repository (SUNDR)

Each principal has an i-table



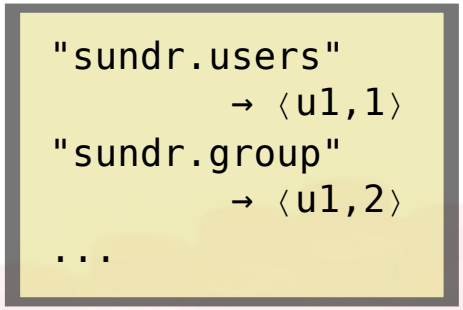
Secure Untrusted Data Repository (SUNDR)

Directory i-nodes associate filenames with i-table entries



Secure Untrusted Data Repository (SUNDR)

- Root directory /sundr
  - Signed by superuser
  - Two special files describe user and group membership
    /sundr/sundr.users: User id → public key
    /sundr/sundr.group: Group membership



Simplified SUNDR model

- Operations totally ordered
  - Global lock prevents concurrent operations
- Users sign every operation
  - Signature reflects both the operation and every operation preceding it

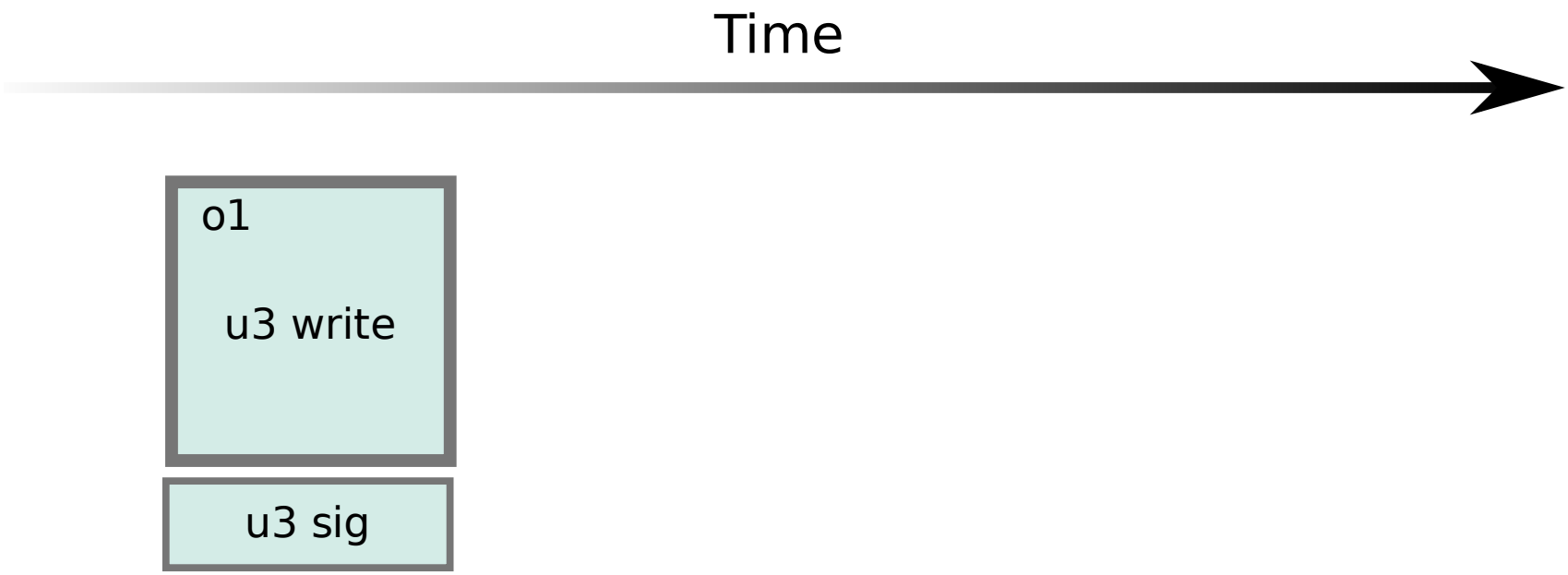
Simplified SUNDR model: fetch and modify

- To fetch or modify a file, a client...
  - Acquires the global lock
  - Downloads entire history of operations
  - Validates each user's most recent signature
  - Checks for own last operation
  - Reconstructs a local copy of the filesystem by replaying history
    - Check validity of each modification
  - Uploads new operation and signature
  - Releases the global lock

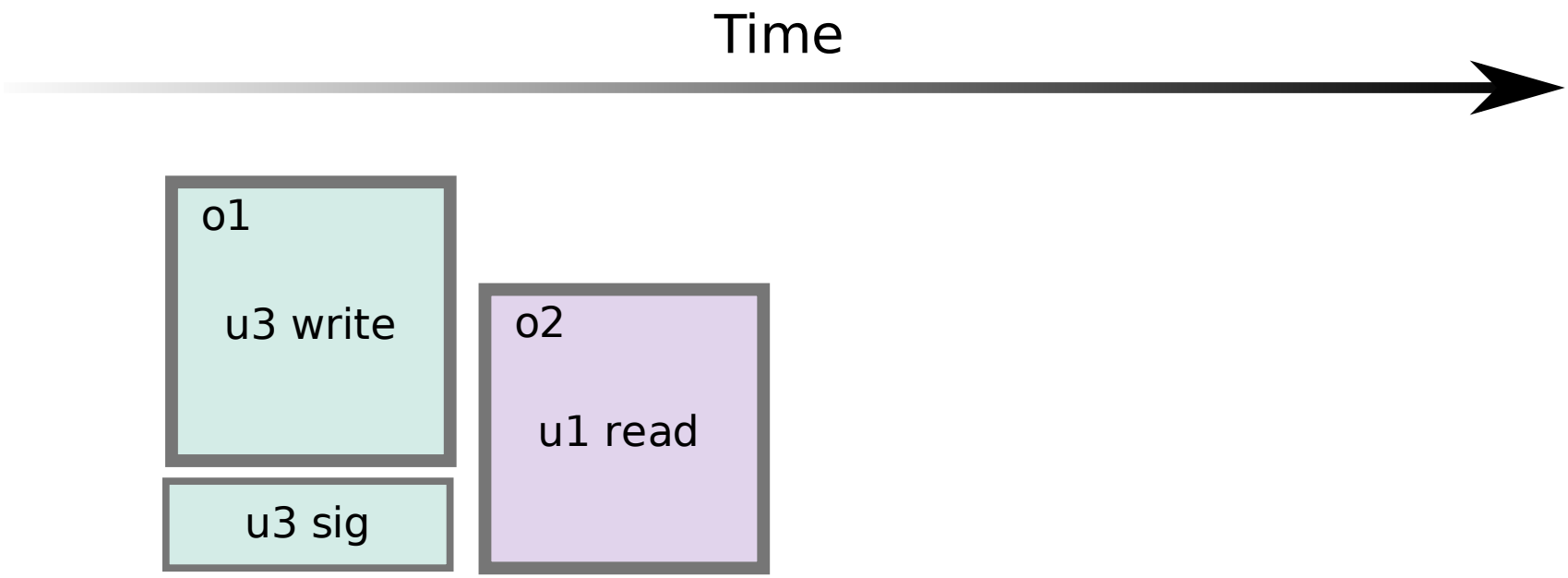
Simplified SUNDR model: fetch and modify

Simplified SUNDR model: fetch and modify

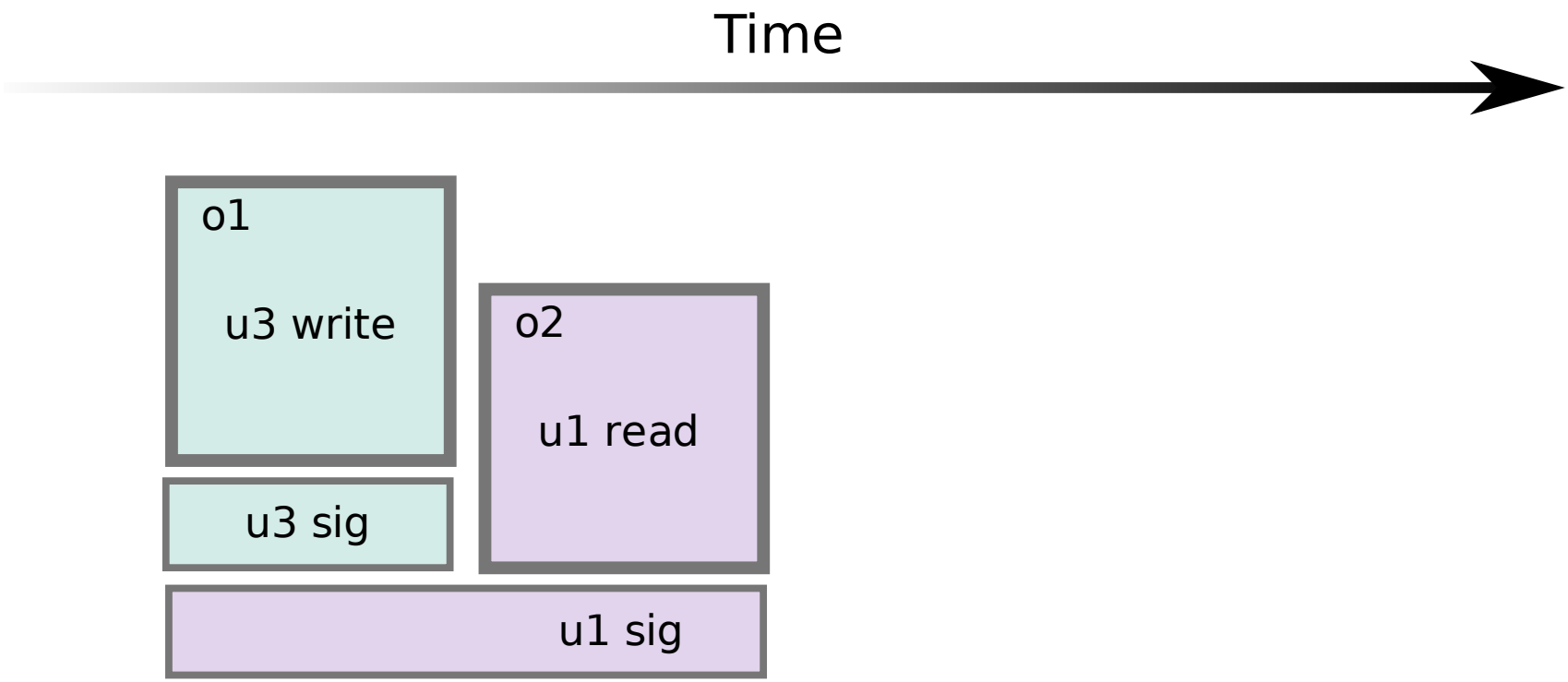
Simplified SUNDR model: fetch and modify



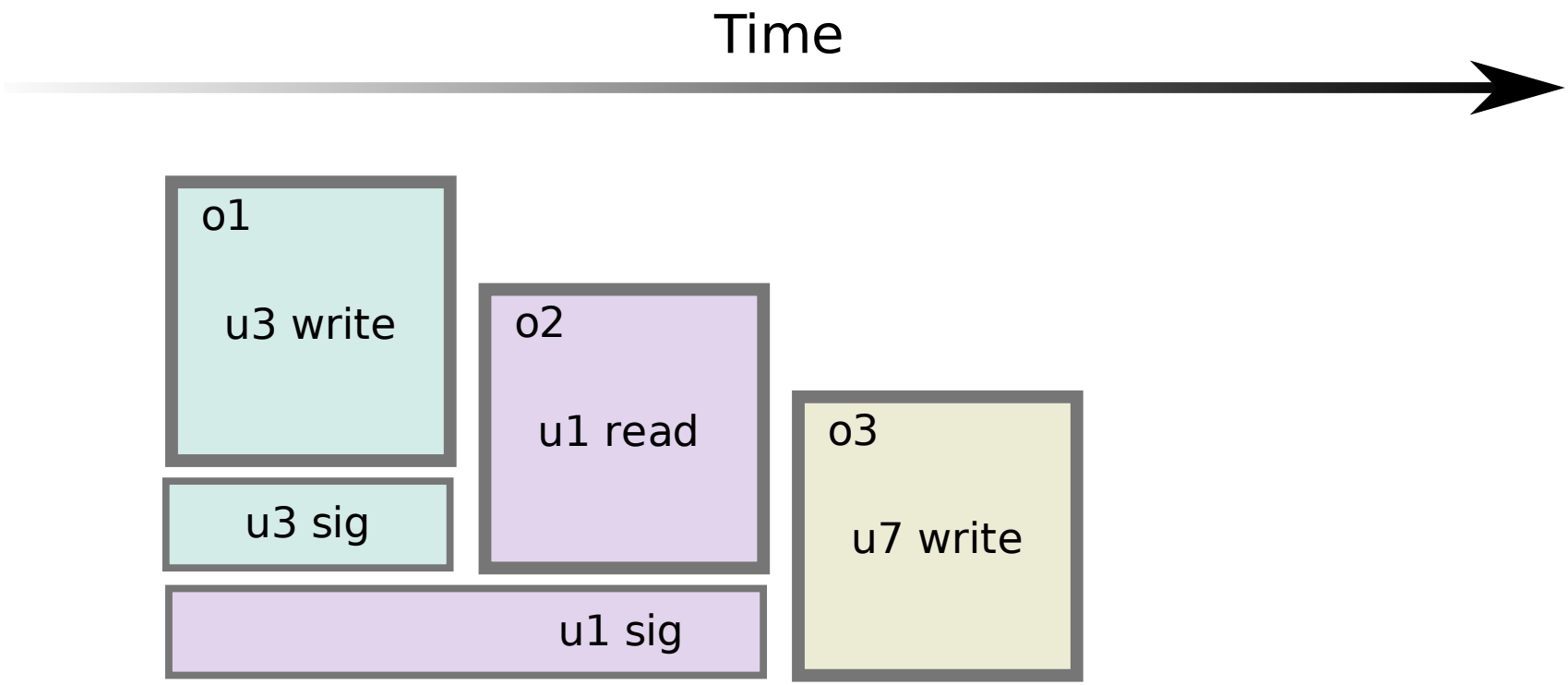
Simplified SUNDR model: fetch and modify



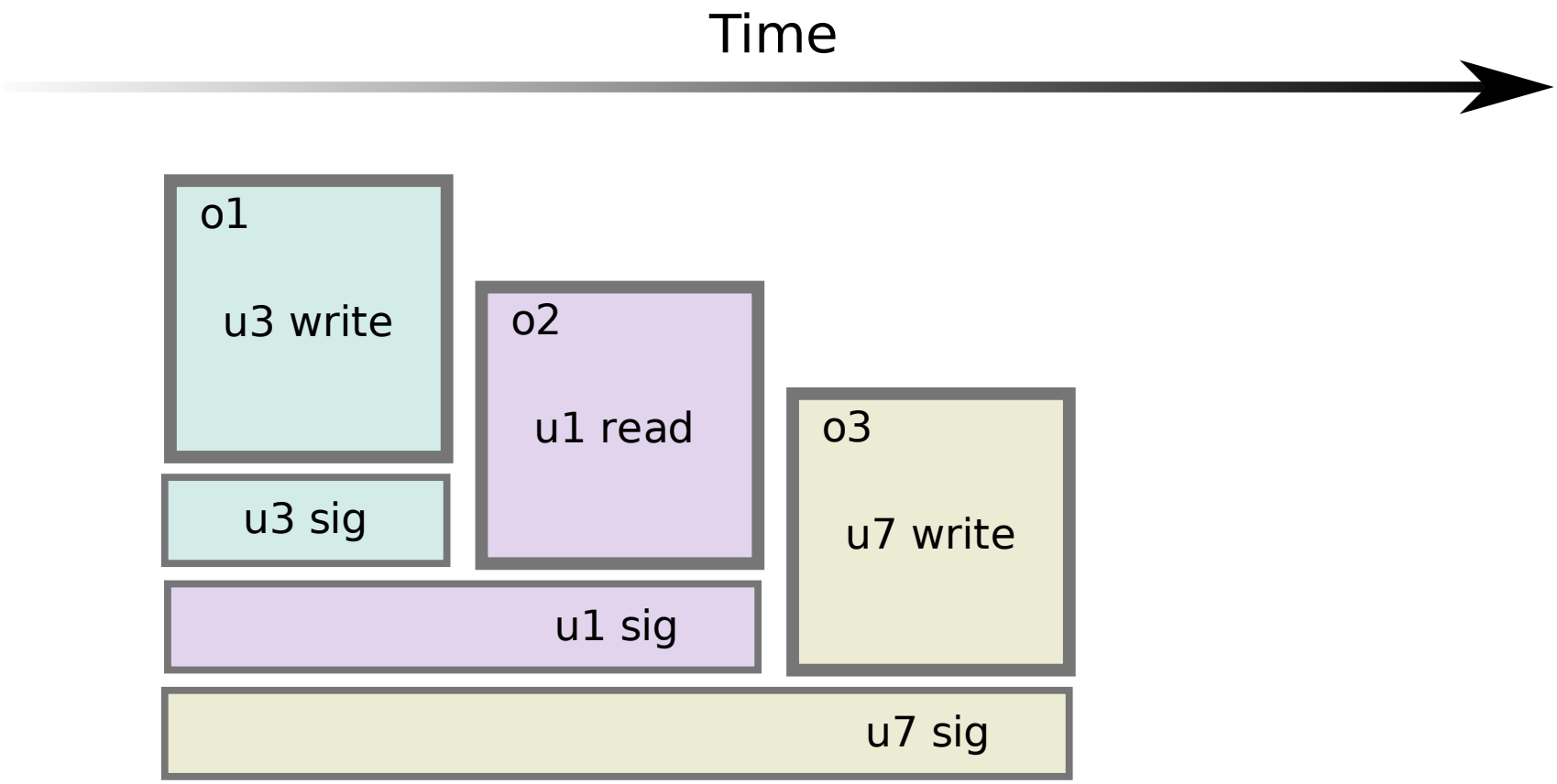
Simplified SUNDR model: fetch and modify



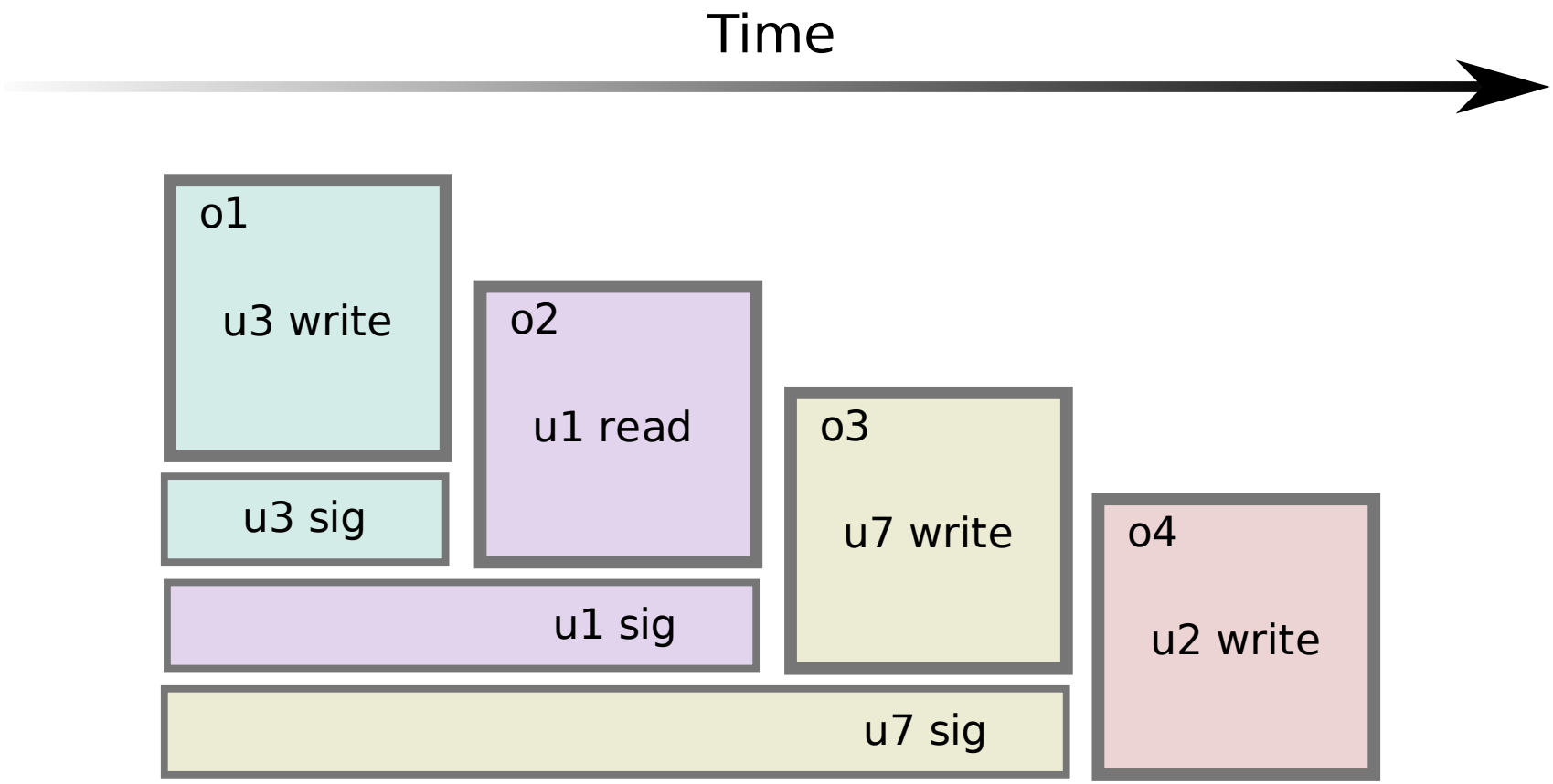
Simplified SUNDR model: fetch and modify



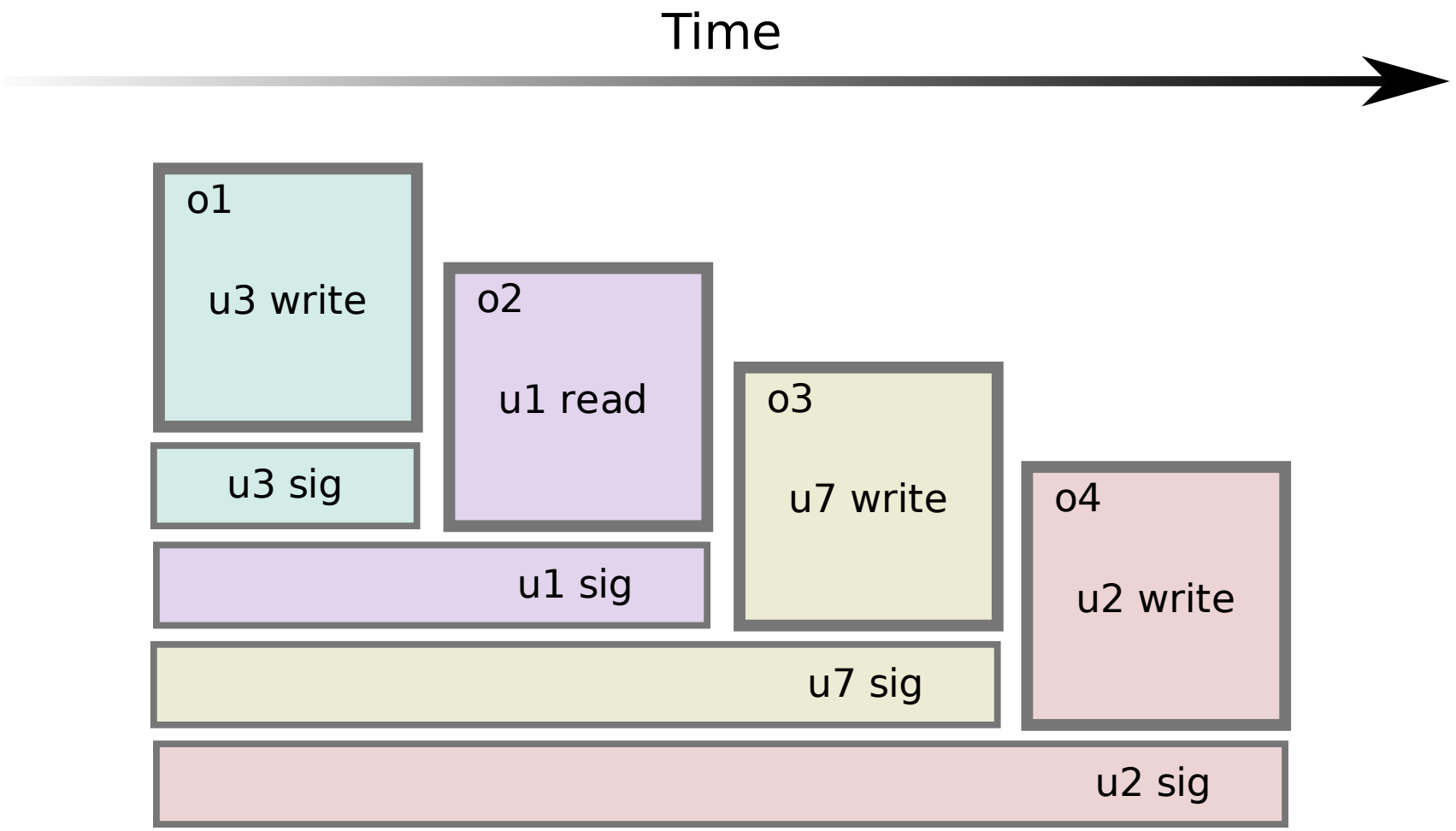
Simplified SUNDR model: fetch and modify



Simplified SUNDR model: fetch and modify

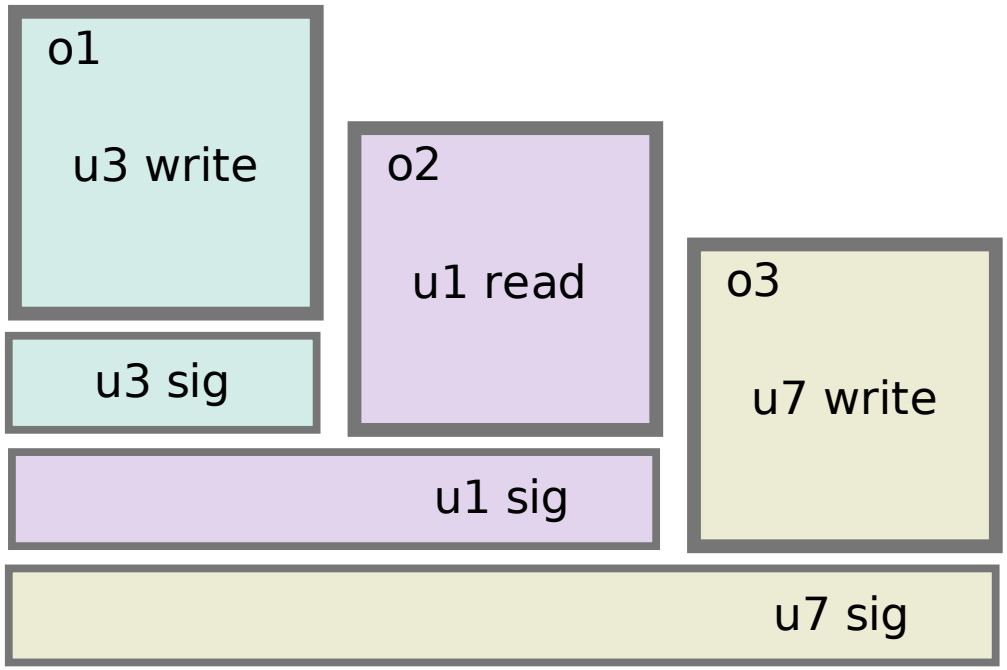


Simplified SUNDR model: fetch and modify



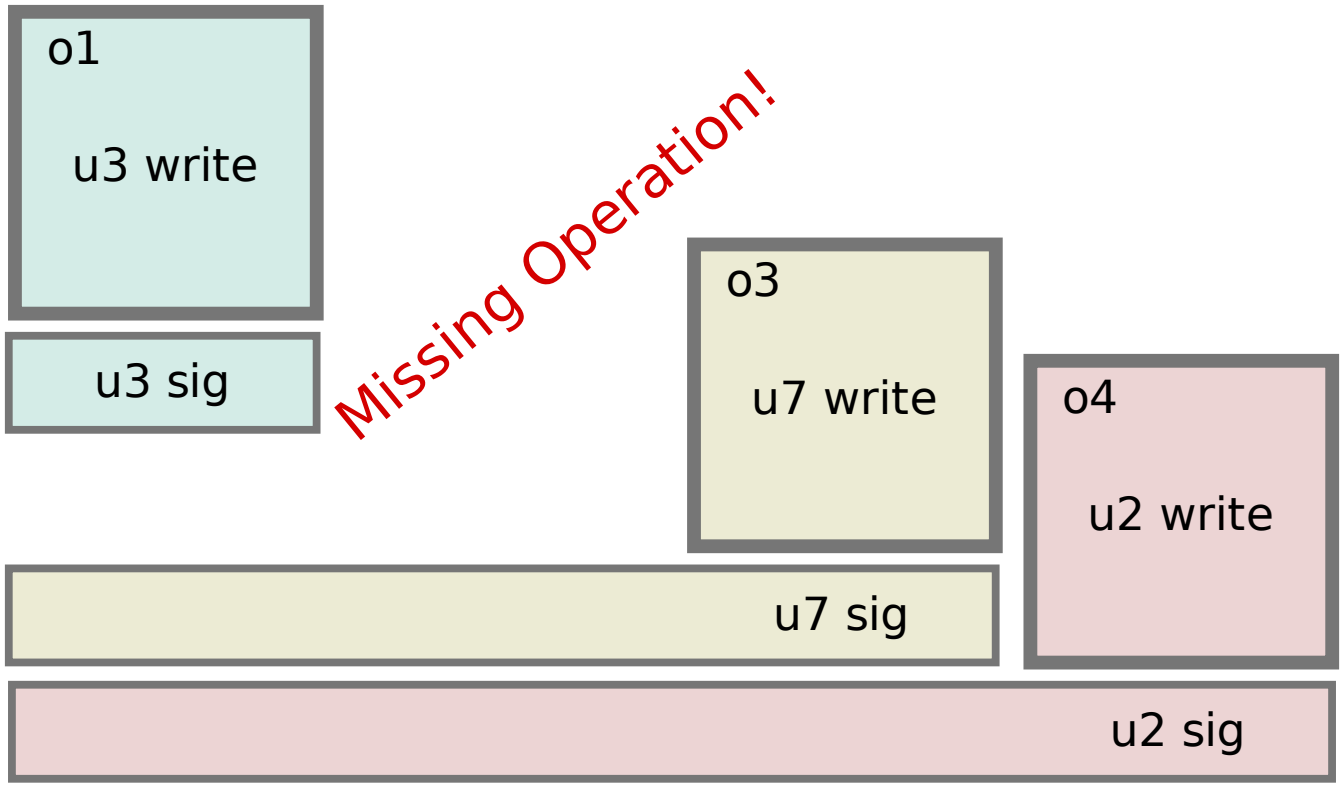
Simplified SUNDR model

- Server's only attack is to omit operations
  - Clients remember their last signature
  - Once a client knows about an operation, cannot be rescinded



Simplified SUNDR model

- Server's only attack is to omit operations
  - Clients remember their last signature
  - Once a client knows about an operation, cannot be rescinded

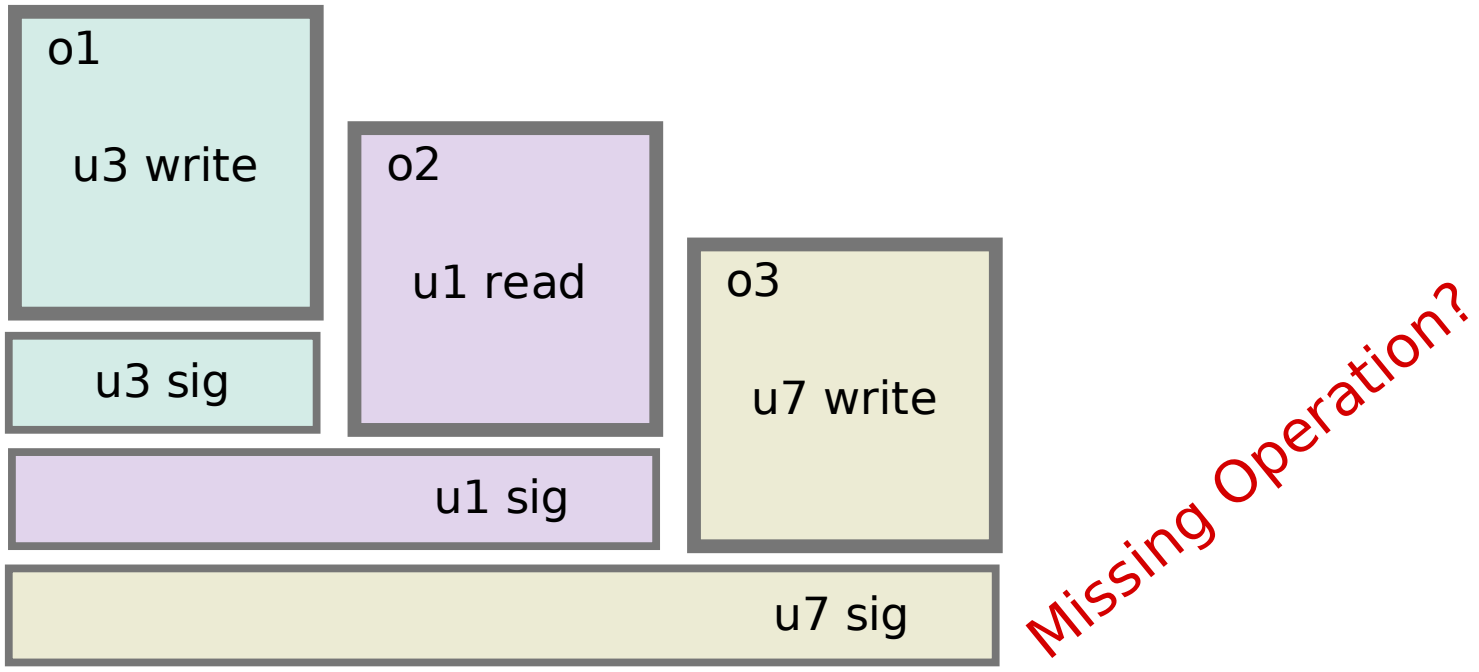


Simplified SUNDR model

- Server's only attack is to omit operations

  … more recent than client's last action.

  This is called a fork.

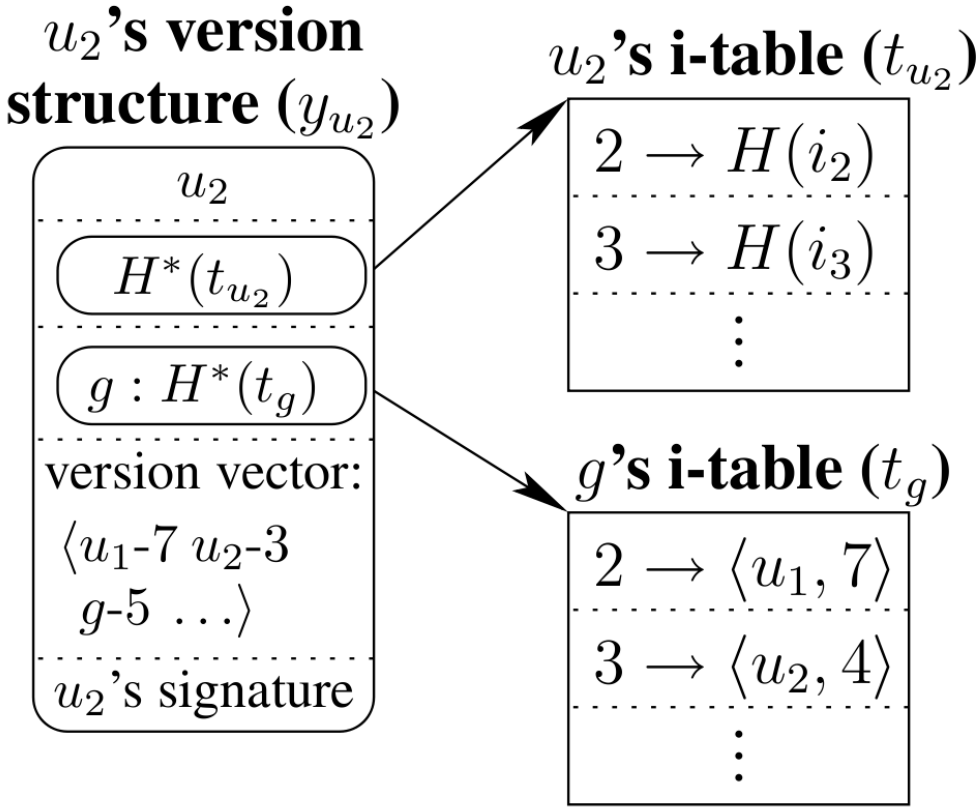


Fork consistency

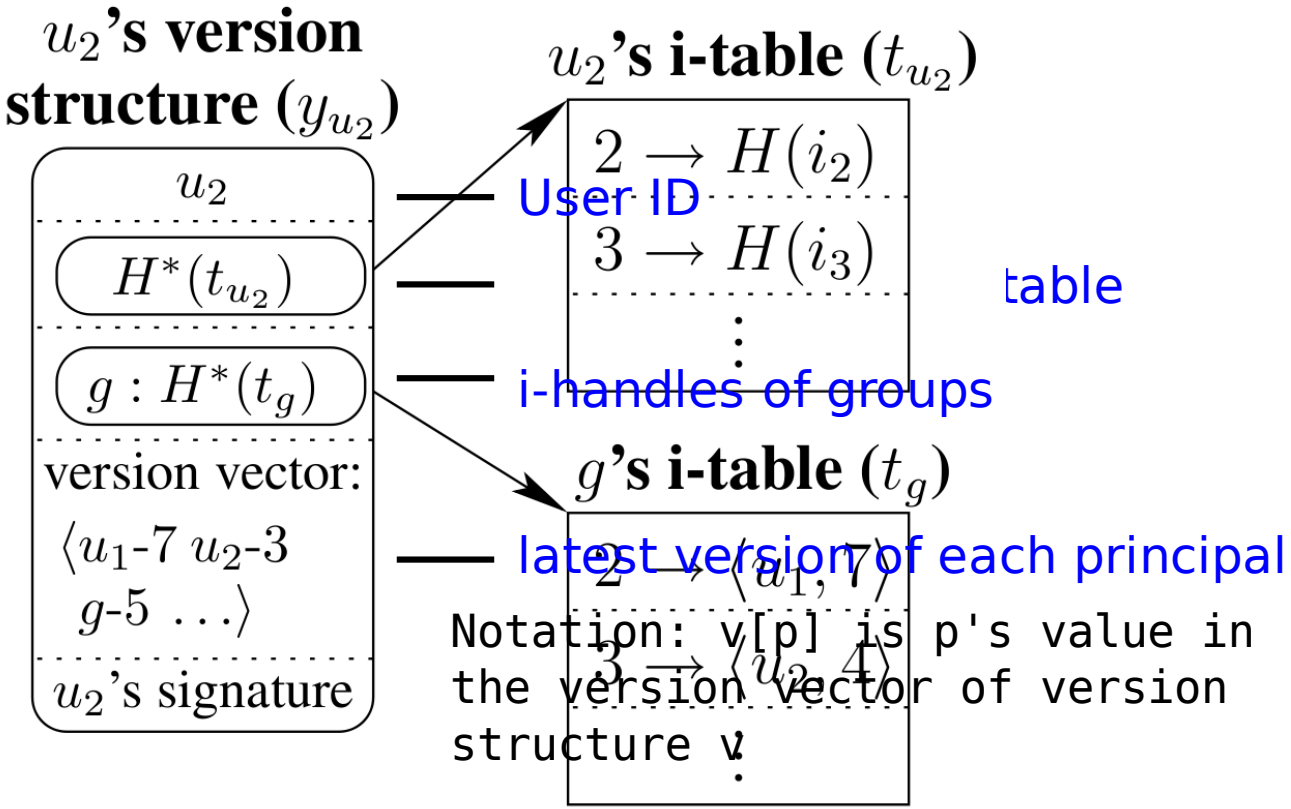
- Clients have differing views due to omission
  - Malicious server must ensure two clients with forked views never again see each others operations
  - A client's internal view remains consistent
- Fetch-Modify Consistency
  - What we expect from a filesystem: All clients see the same total ordering of operations
  - Stronger than fork consistency
  - SUNDR delivers fork consistency, but communication between clients can strengthen it to fetch-modify consistency

Serialized SUNDR

- The simplified model sends the entire history to a client for each operation
  - Ridiculous
- Version Structures
  - We only really need the latest signature and operation from each user
  - Keep a Version Structure for each user



Serialized SUNDR



- Version Structure List (VSL)
  - Stored on server
  - Contains latest version structure for each user

Serialized SUNDR: Fetch/modify

- To fetch or modify, a client must...
  - Acquire global lock
  - Download VSL
  - Create updated version structure V for itself
  - Construct a version vector from the VSL
    - For users, get number from each user's VS
    - For groups, take version from VS with latest group i-handle
  - Increment own version number in version vector
  - Increment version number of any modified groups (and include i-handle)
  - Check consistency of VSL + {V}
  - Sign and upload V. Release global lock

VSL Consistency

- Does VSL contain user's previous version structure?

- Define ≤ for version structures x and y:

$$x \leq y \iff \forall p \; x[p] \leq y[p]$$

- Is VSL + {V} totally ordered by ≤ ?

Version Structures During a Fork Attack

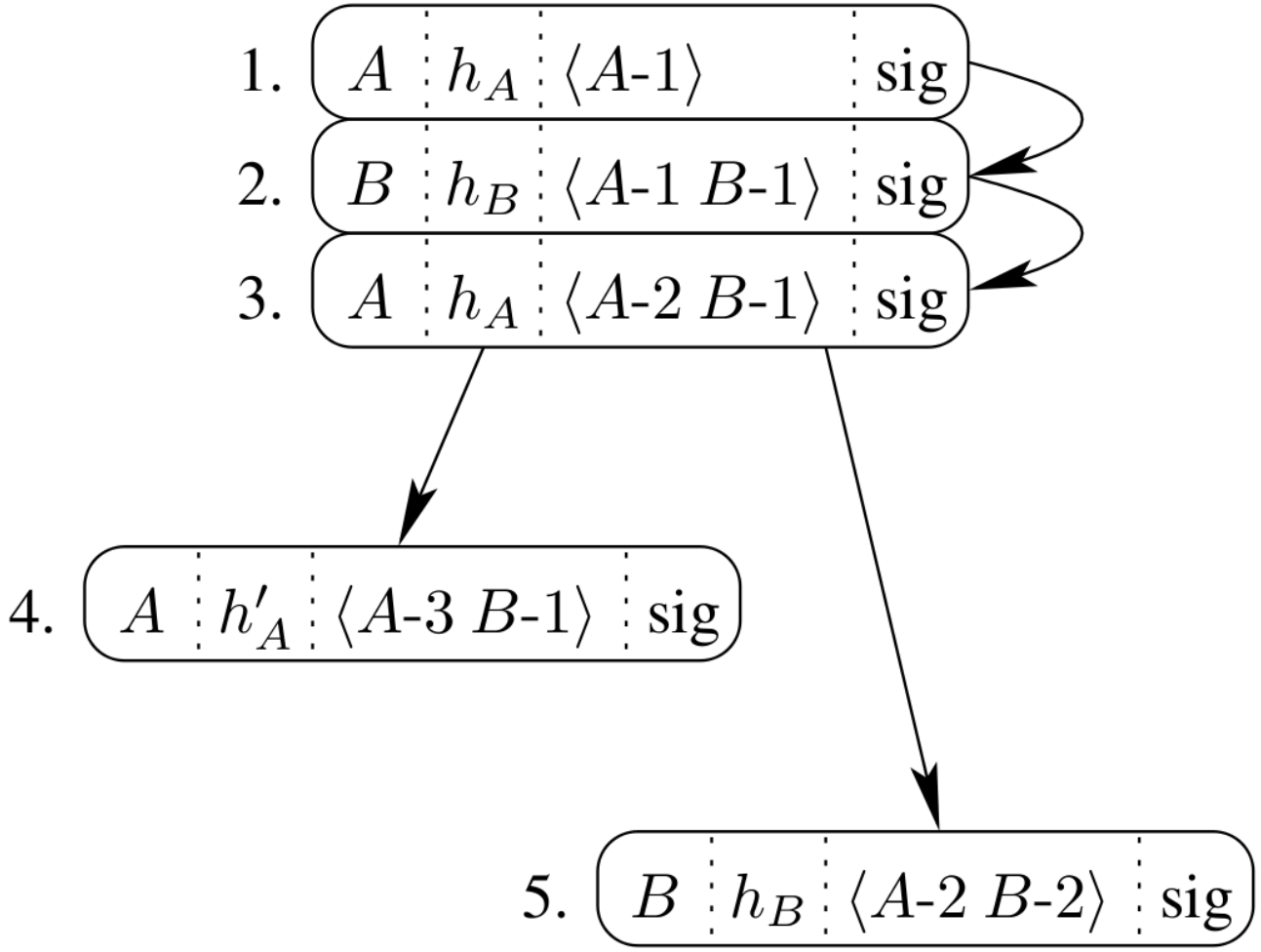


Figure 4: Signed version structures with a forking attack.

Concurrent SUNDR

- Clients pre-declare operations
- Update Certificates notify server of clients' intent

Update Certificate

```
User's next version number

Hash of user's current VSL entry

List of modifications to perform
```

Signature

Concurrent SUNDR

- Possible modifications
  - Set file <user,i#> to i-hash h
  - Set group file <group, i#> to <user, i#>
  - Set/delete named entry in directory <p,i#>
  - Pre-allocate range of group i-numbers

Concurrent SUNDR: fetch/modify

- Server maintains a Pending Version List (PVL)
  - Tuples <certificate, vs>
  - vs is an unsigned version struct generated by the server
  - Client cannot predict the new version vector before it receives the VSL
- Client sends update certificate c before receiving VSL
- Server generates version structure vs, adds <c,vs> to PVL
- Server replies with VSL and PVL
  - An honest server orders entries in the PVL by order of certificate arrival

Concurrent SUNDR: fetch/modify

- Client uses VSL to compute a new version structure V
- Update V with PVL version numbers
- Check consistency of VSL + PVL + {V}
- Send version structure to server

Concurrent SUNDR: Update conflicts

- Read-after-write

  If client is fetching a file and PVL contains modification to that file, then there is a read-after-write conflict.

  Client commits version structure as before, but then waits for fetched files to be committed to VSL before returning.

- Write-after-write for groups

  Extend version structure to contain last seen PVL

What can a malicious SUNDR server do?

- Integrity and consistency attacks
  - ~~Arbitrarily change data~~ — Detectable
  - ~~Falsify clients' actions~~ — Detectable
  - Lose changes made by clients — Limited: Fork attacks
  - Present inconsistent views — Limited: Fork consistency
- Confidentiality attacks
  - ~~Read confidential data~~ — Not applicable
- Availability attacks
  - Ignore user requests

Results

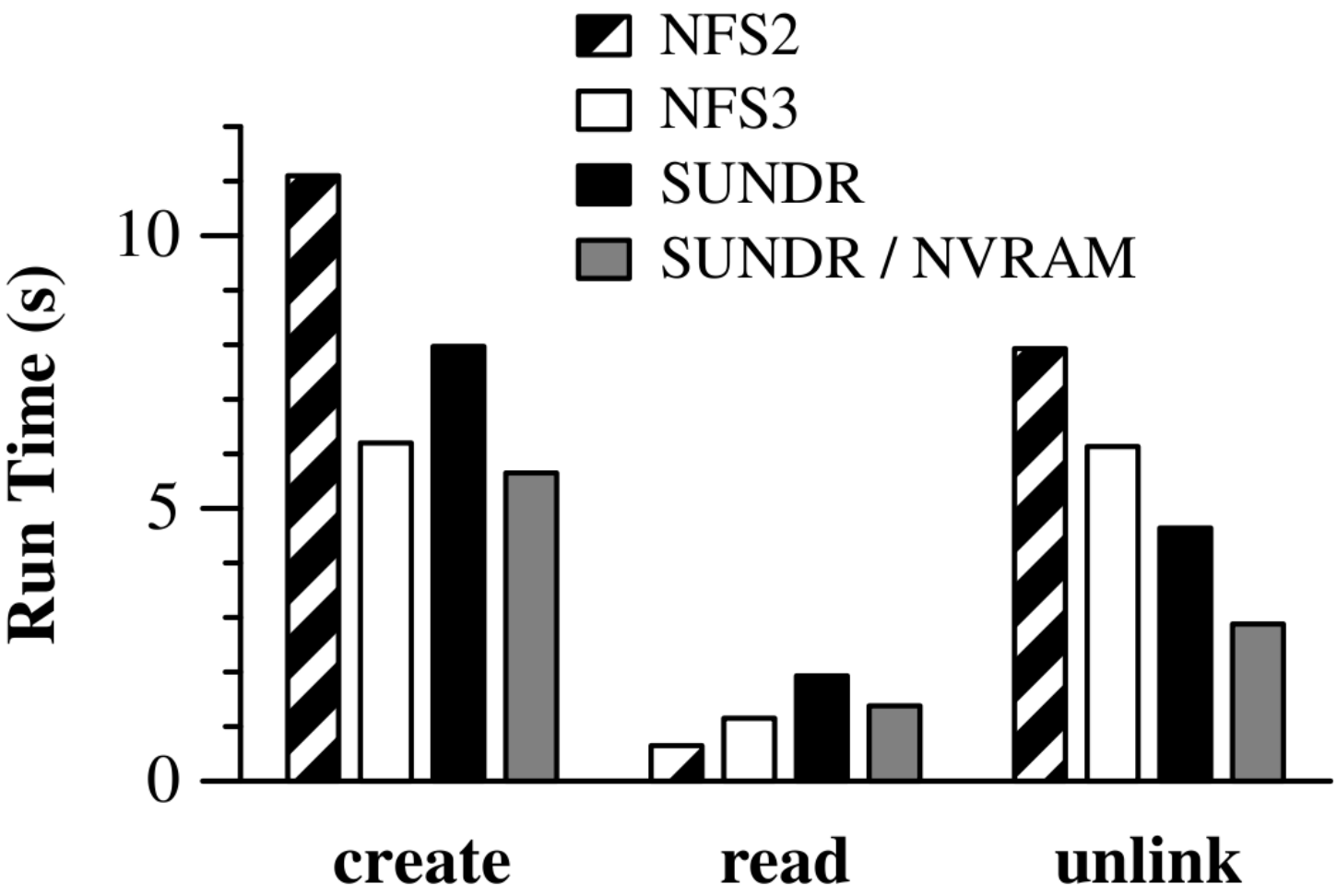
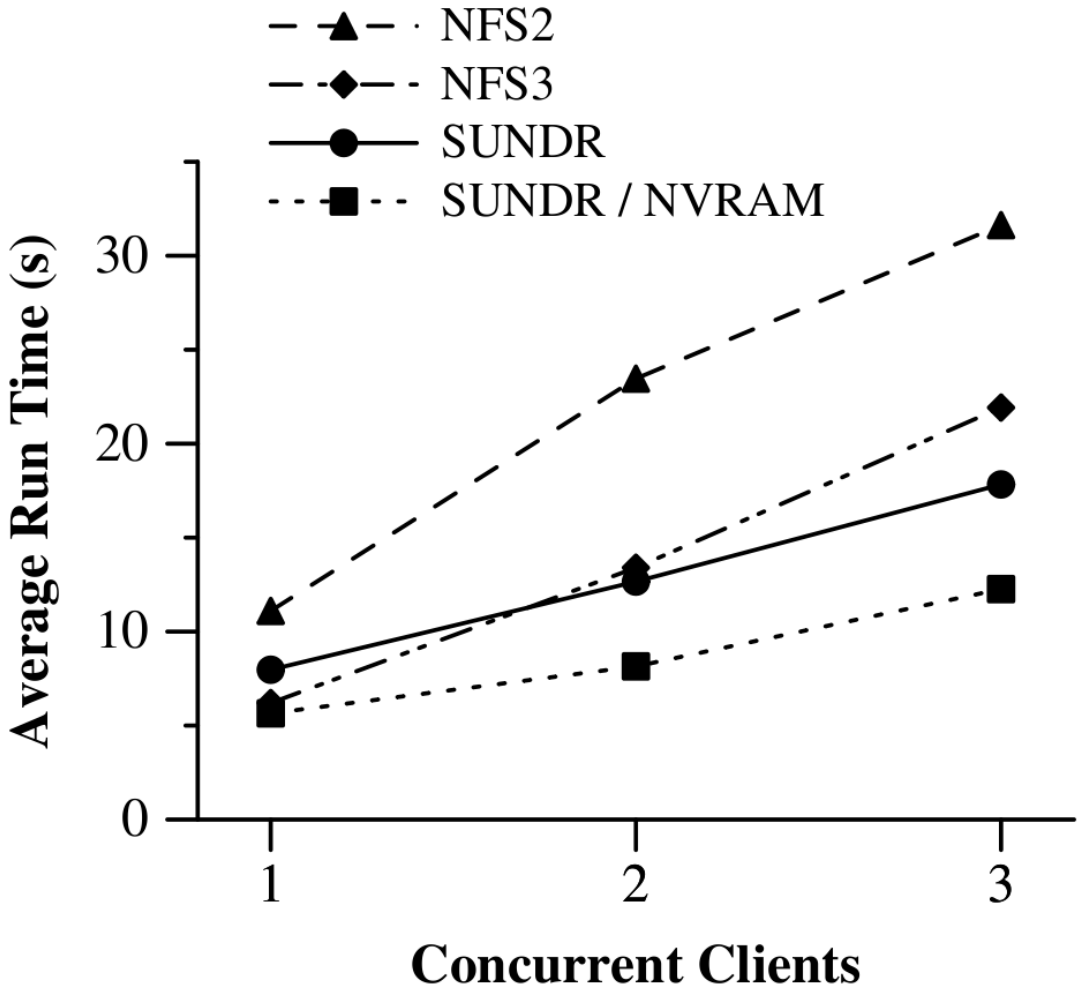


Figure 9: Single client LFS Small File Benchmark. 1000 operations on files with 1 KB of random content.

Results



Concurrent LFS Small File Benchmark, create phase. 1000 creations of 1 KB (SUNDR relative std. dev. in 3 concurrent is 13.7%

Results

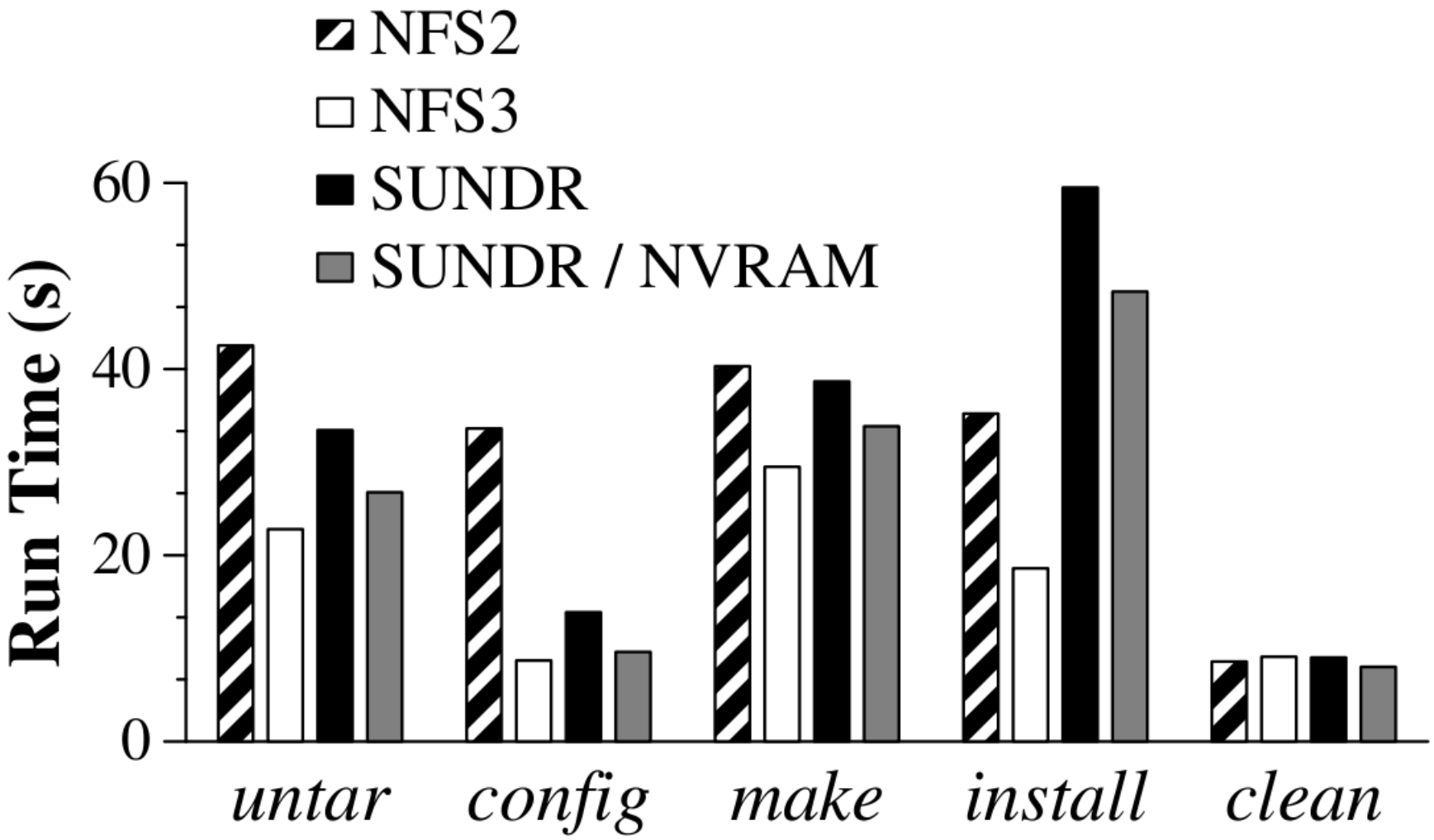


Figure 11: Installation procedure for `emacs_20.7`

Results

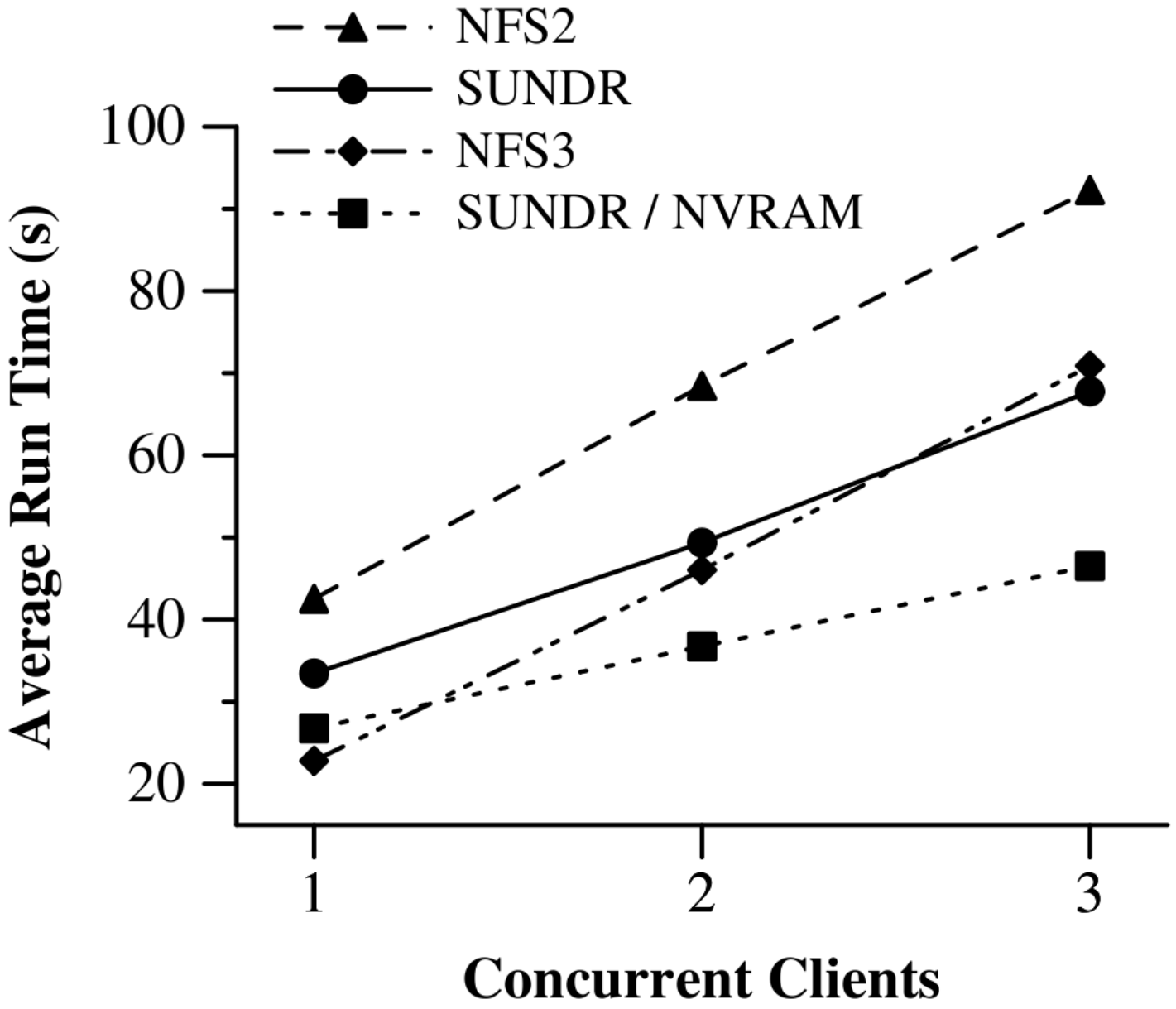


Figure 12: Concurrent *untar* of `emacs_20.7.tar`

Results

Phase	SUNDR	SUNDR NVRAM	NFS3	SSH
Import	13.0	10.0	4.9	7.0
Checkout	13.5	11.5	11.6	18.2
Commit	38.9	32.8	15.7	11.5
Update	19.1	15.9	13.3	11.5

Figure 13: Run times for CVS experiments (in seconds).