SPORC

Group Collaboration using Untrusted Cloud Resources

Paper by Ariel Feldman, William Zeller, Michael Freedman, and Edward Felten

OSDI '10

Presentation by Riley Simmons-Edler

What is SPORC?

- A system for running collaborative cloud apps on Untrusted servers
  - Works with any application where operation records can be mirrored locally as well as on a server
- Prevents server admin snooping, malicious interference

Why do we care?

- Google Docs is popular

- People like to write things that can get them in trouble

- Growing use of web/cloud services to organize by protest movements and other groups

- Anonymization of cloud file hosting

Goals

- Flexible
- Fast
- Asynchronous
- Protect data from server
- Detect malicious servers
- Recover from malicious servers

Model

- Operational Transformations
- Fork* Consistency
- Server functions as a centralized access point/message passer
- Hash chain of ops to guarantee consistent operation basis, avoid meddling

Operational Transformation(OT)

Take two operations and transform one of them relative to the other

Then apply the non-transformed and transformed operation in order

$$T(\text{del }\ x, \text{del }\ y) = \begin{cases} (\text{del }\ x-1, \text{del }\ y) & \text{if} \quad x > y \\ (\text{del }\ x, \text{del }\ y-1) & \text{if} \quad x < y \\ (\text{no-op}, \text{no-op}) & \text{if} \quad x = y \end{cases}$$

Doesn't need to be the optimal merge, just needs to be consistent/automatic

Fork*

Guarantees that the server can not add/alter data

Only permissible server interference is forking clients
Into multiple operation histories

Hash chain of operations used to enforce Fork*

Hash stores all ops seen by a client,
plus operation # for last op.

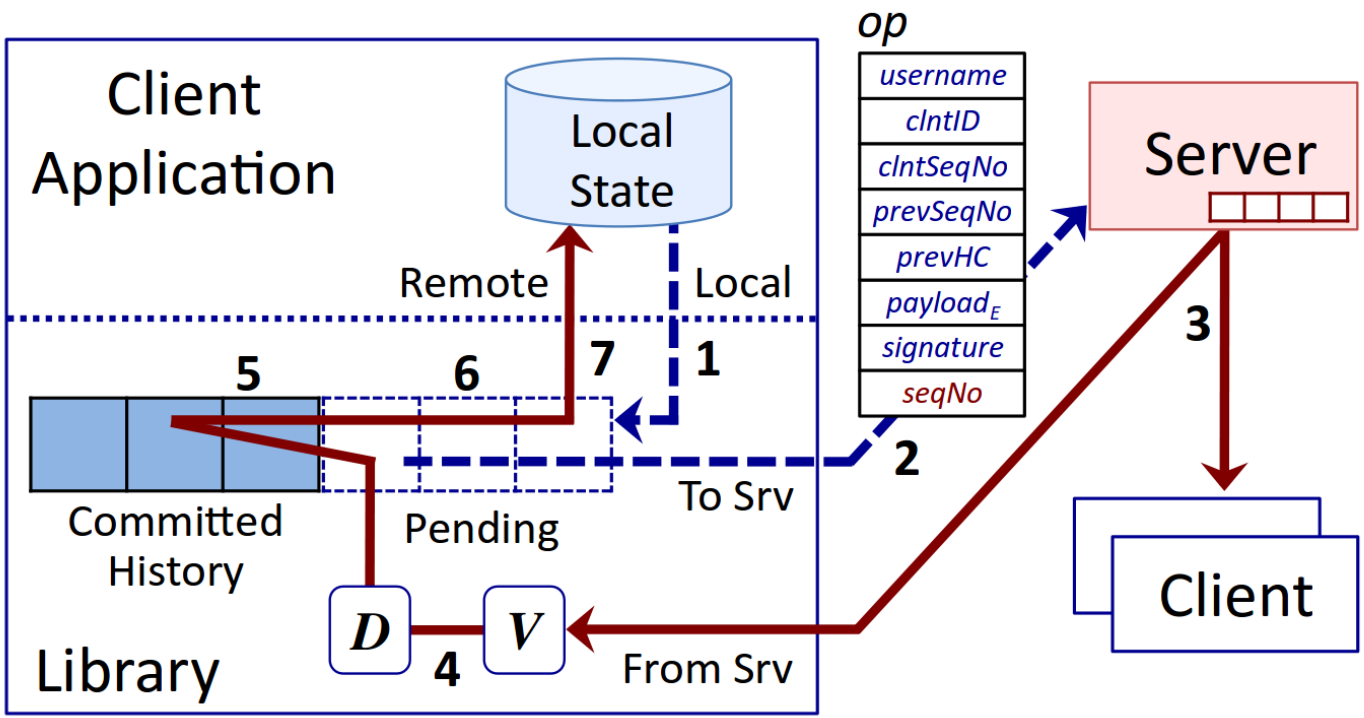
If histories diverge up to that op #,
server interfered or failed.

Fork Checking/Resolution

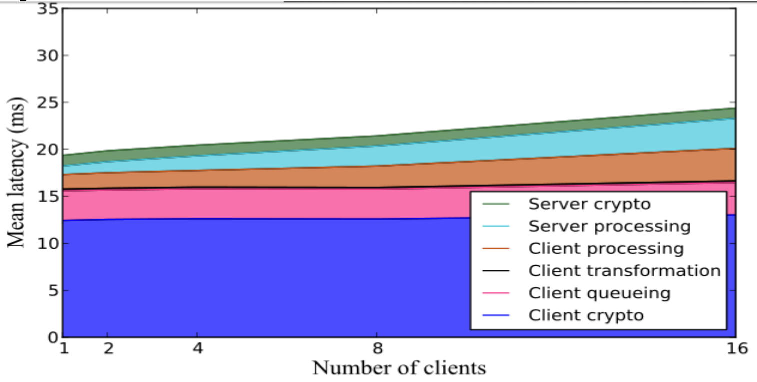
- Forks are detected out-of-band(e.g. via socket connection)
  - Clients compare op#-hash chain pairs, difference indicates fork

- Detected forks are resolved via OT
  - Replay changes since fork on trusted server,
    OT merges them automatically

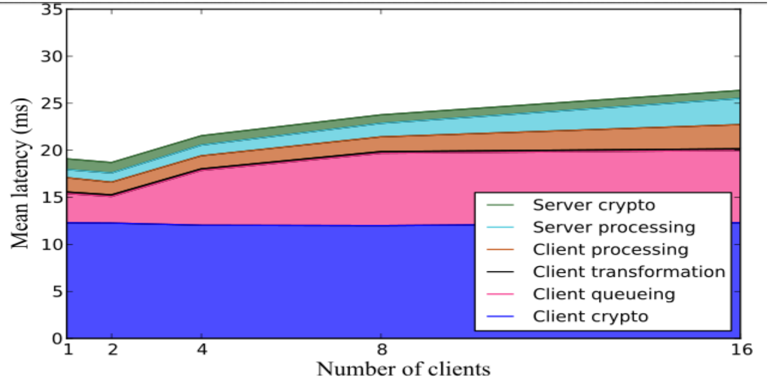
Execution Sequence



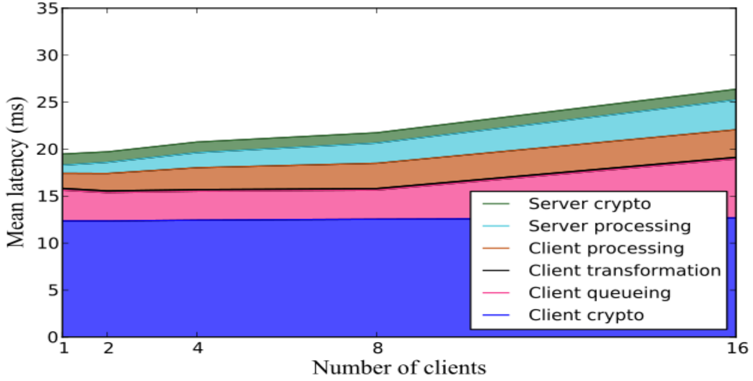
Experimental Validation



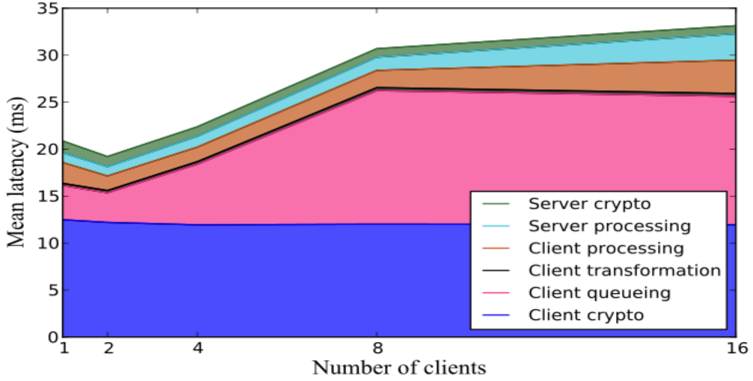
(a) Unloaded key-value store



(a) Loaded key-value store



(b) Unloaded text editor



(b) Loaded text editor

Throughput and Fork Merge Time

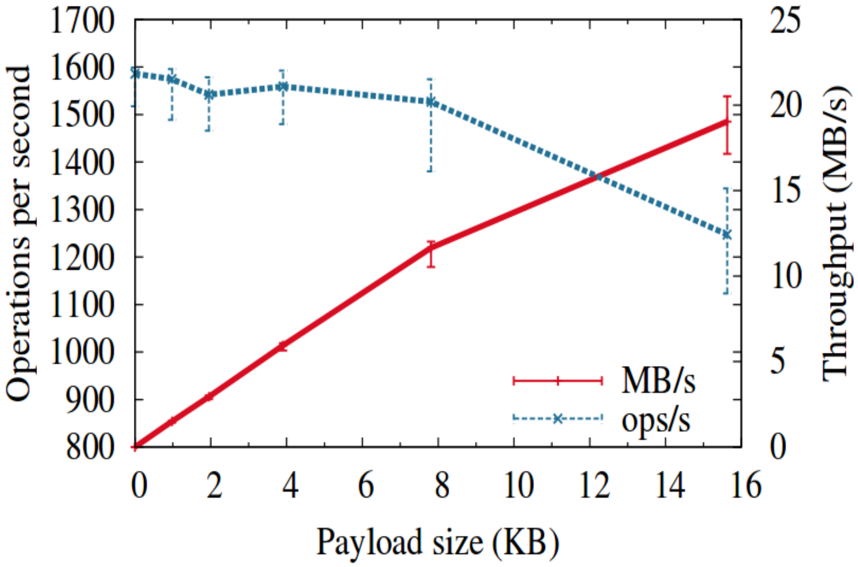


Figure 4: **Server throughput as a function of payload size.**

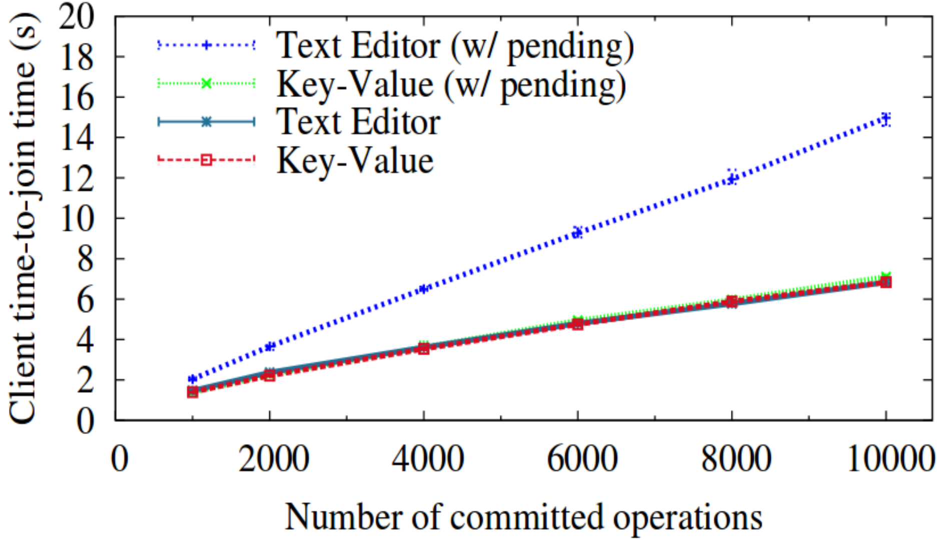


Figure 5: **Client time-to-join given a variable length history**

Strengths

- Highly resistant to malicious server attacks
- Minimal reduction in user experience relative to unprotected cloud services
- Able to recover automatically in the event of malicious server activity

Weaknesses

- Side-channel/OOB attacks are still a thing
  - SPORC makes a lot of assumptions about peripheral security
- Malicious server still knows client IP's
- Traffic analysis and client monitoring could allow contents to be inferred.
- Many details(e.g. fork merging/detection) not implemented

Conclusions

- SPORC hides activity from the server effectively
- Low overhead on top of normal cloud app operations
- General security of SPORC is dependent on external functionality, good user behavior.
- Hard to make better without restricting user behavior or removing server entirely(e.g. peer-to-peer system)

Bottom Line: Probably a good everyday solution
for those who don't want to get snooped by Google/NSA