Practical Web-based Delta Sync for Cloud Storage Services

He Xiao Zhenhua Li Ennan Zhai Tianyin Xu

清華大學 Tsinghua University

Yale University

UC San Diego

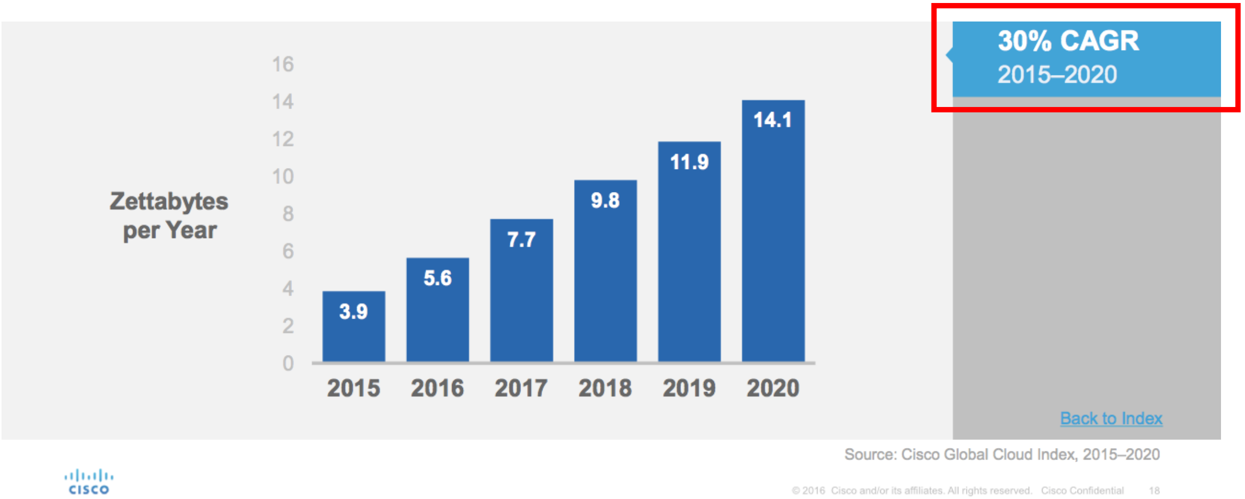
xiaoh16@gmail.com

July 10, 2017

Hotstorage'17

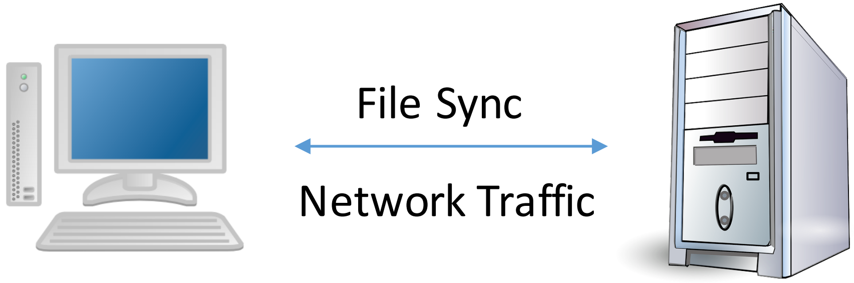
Network Traffic is Overwhelming in Cloud Storage

Cloud Traffic has 30% CAGR (**C**ompound **A**verage **G**rowth **R**ate)



Source: Cisco Global Cloud Index, 2015–2020

© 2016 Cisco and/or its affiliates. All rights reserved. Cisco Confidential 18



Users

Client

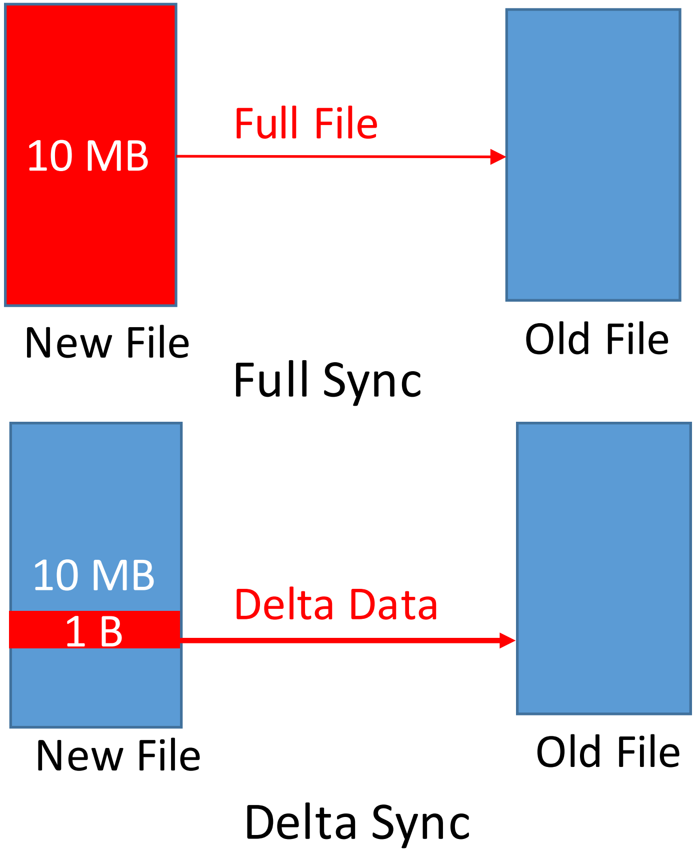
File Sync

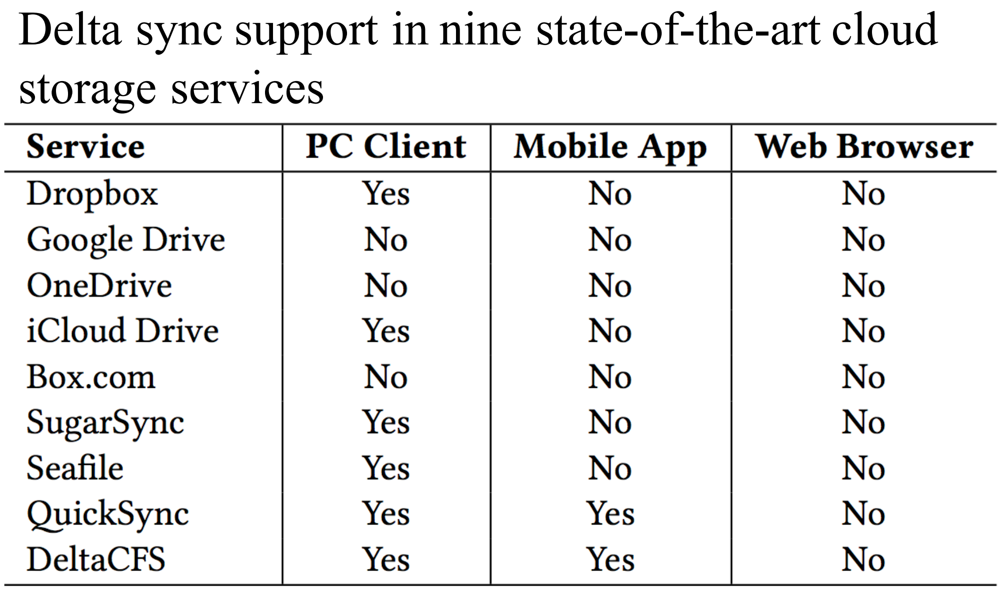
Network Traffic

Sever

Vendors

Delta Sync Improves Network Efficiency





Delta sync support in nine state-of-the-art cloud storage services

Service	PC Client	Mobile App	Web Browser
Dropbox	Yes	No	No
Google Drive	No	No	No
OneDrive	No	No	No
iCloud Drive	Yes	No	No
Box.com	No	No	No
SugarSync	Yes	No	No
Seafile	Yes	No	No
QuickSync	Yes	Yes	No
DeltaCFS	Yes	Yes	No

☺ ☹

Delta Sync is crucial for reducing cloud storage network traffic.

No Web-based Delta Sync

Web-based delta sync is essential for cloud storage web clients and web apps

Web is the most pervasive and OS-independent cloud storage access method

Web Apps with local storage or log files need web-based Delta Sync

Why web-based delta sync is not supported by today's cloud storage services ?

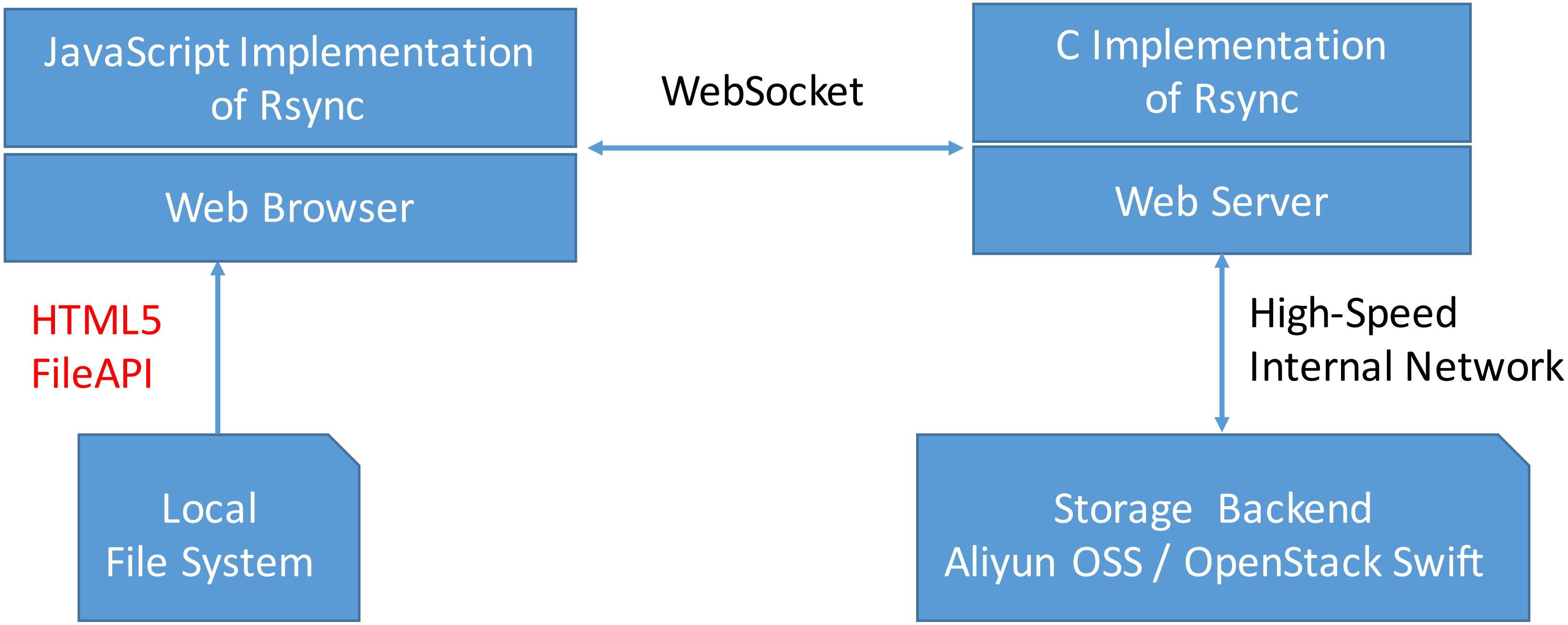
Contribution

- We quantitatively study why web-based delta sync is not offered by today's cloud storage services.

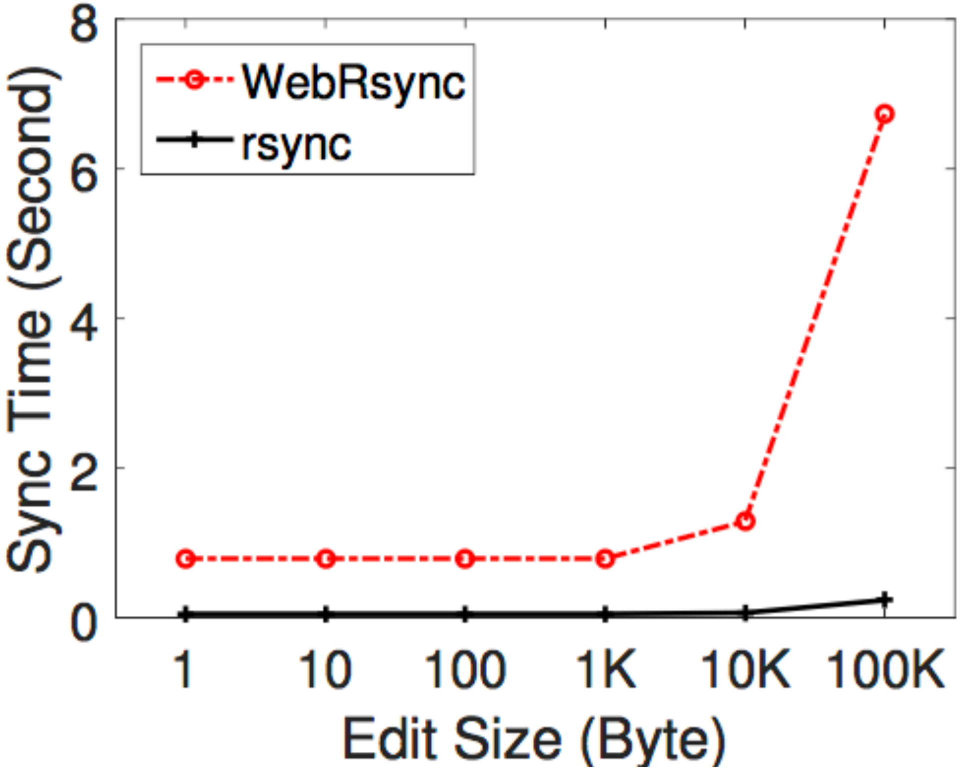
- We build a practical web-based delta sync solution for cloud storage services.
  - By reversing traditional delta sync process, we make the overhead affordable at the web client side.
  - By exploiting the locality of users' edits and trading off hash algorithms, we make the computation overhead affordable at the server side.

WebRsync: Implement Delta Sync on Web

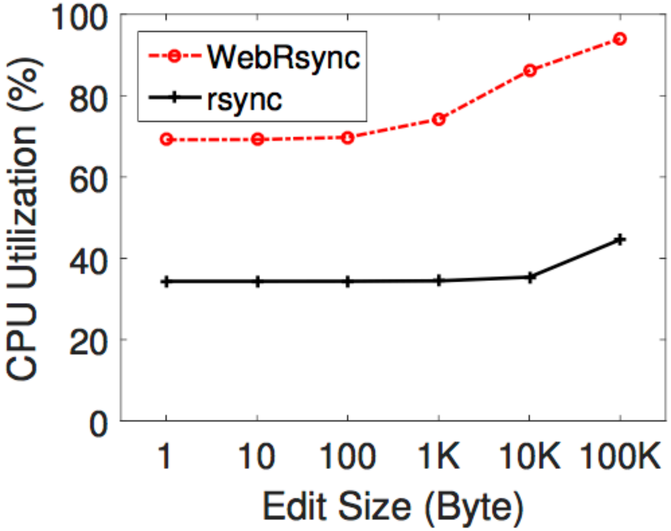
- Implement `rsync` on real cloud storage with native web tech: **JavaScript + HTML5 + WebSocket**
  - `rsync` is the de facto solution of delta sync in cloud storage


JavaScript Implementation of Rsync
Web Browser
WebSocket
C Implementation of Rsync
Web Server
HTML5 FileAPI
High-Speed Internal Network
Local File System
Storage Backend Aliyun OSS / OpenStack Swift

WebRsync vs. `rsync`

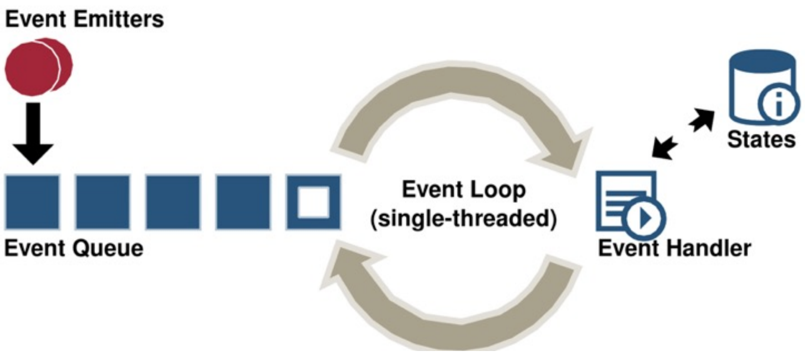


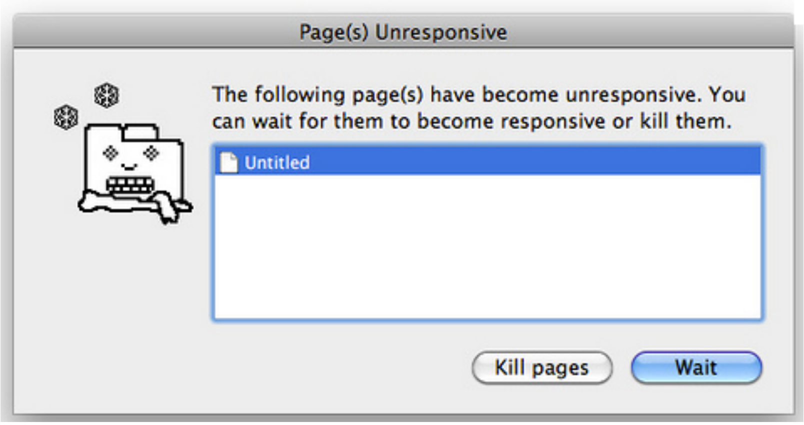
Sync time of WebRsync vs `rsync`



Average Client CPU utilization

Stagnation due to JavaScript's Single-thread Event Loop Model





The following page(s) have become unresponsive. You can wait for them to become responsive or kill them.

StagMeter

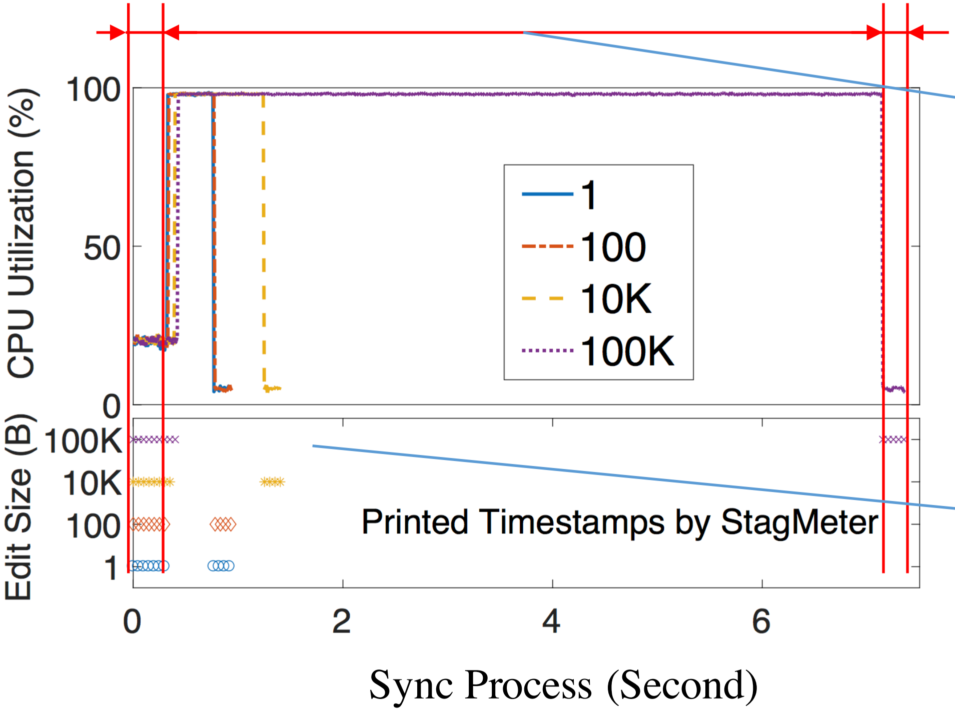
```
//print timestamp every 100ms
setInterval(print(timestamp),100)
//print the timestamp of every keystone( start or end of a task)
on_start(task);  print(task.id,  timestamp)
on_finish(task);  print(task.id,  timestamp)
```

StagMeter on WebRsync

1. Send meta data
Wait server

2. Checksum Search
and Comparison

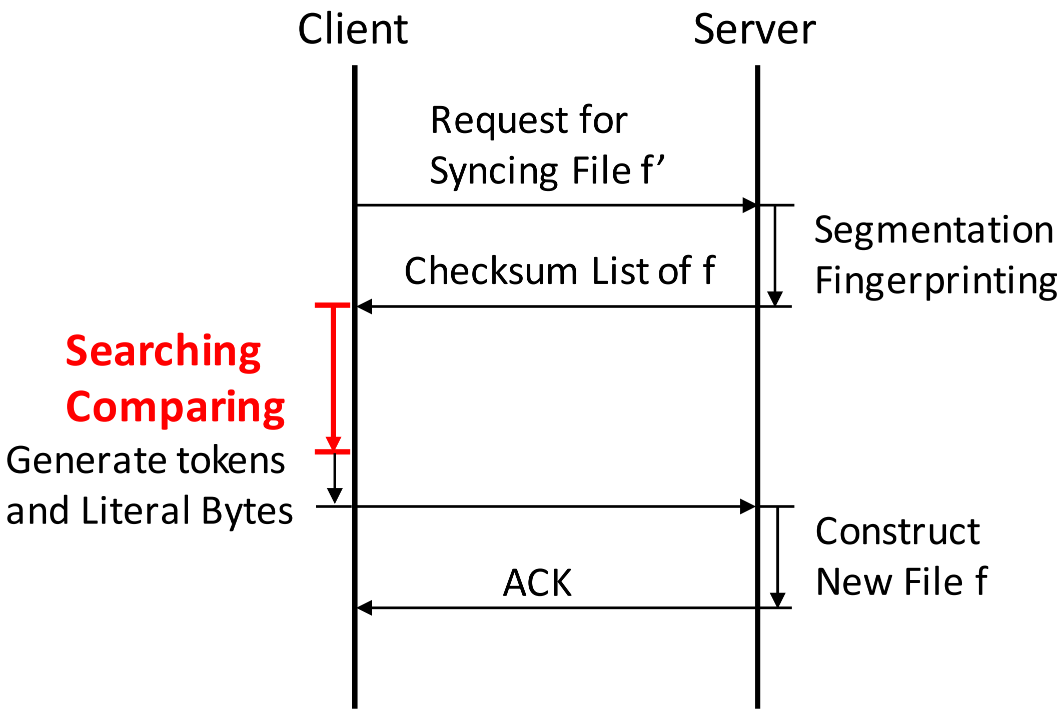
3. Send tokens and literal bytes
Wait server



High CPU Utilization when computing

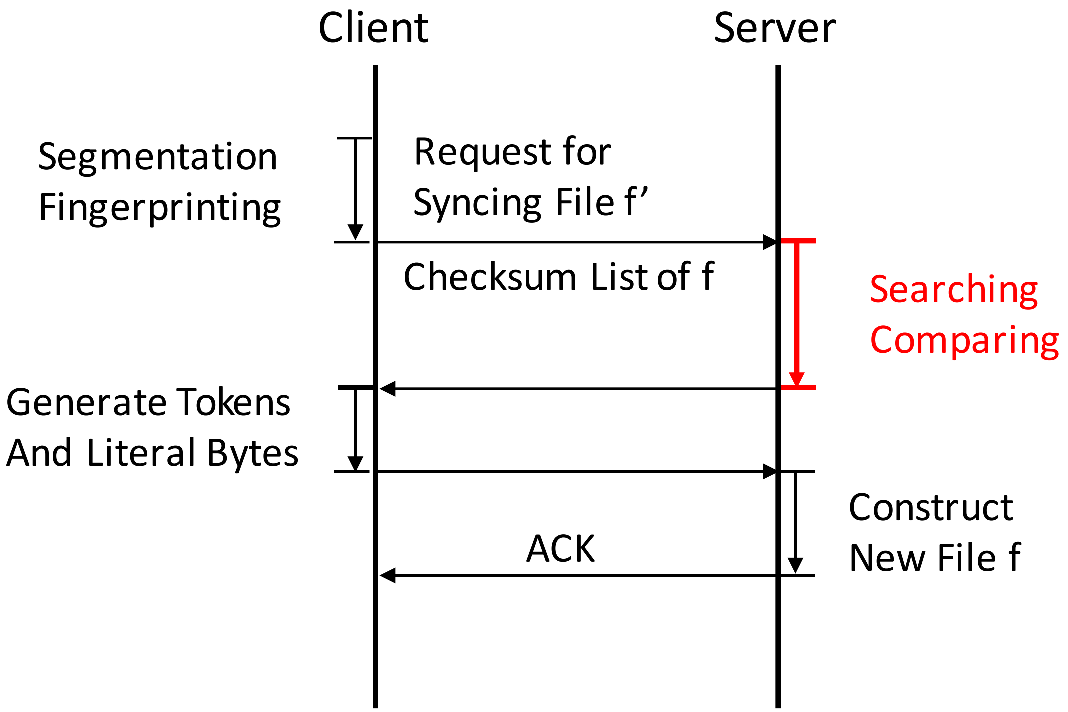
Timestamp Printing is suspended
Web is under stagation state

WebR2sync: Client-side Optimization Reverse Computation Process


Client
Server
Request for Syncing File f'
Segmentation Fingerprinting
Checksum List of f
Searching
Comparing
Generate tokens and Literal Bytes
Construct New File f
ACK

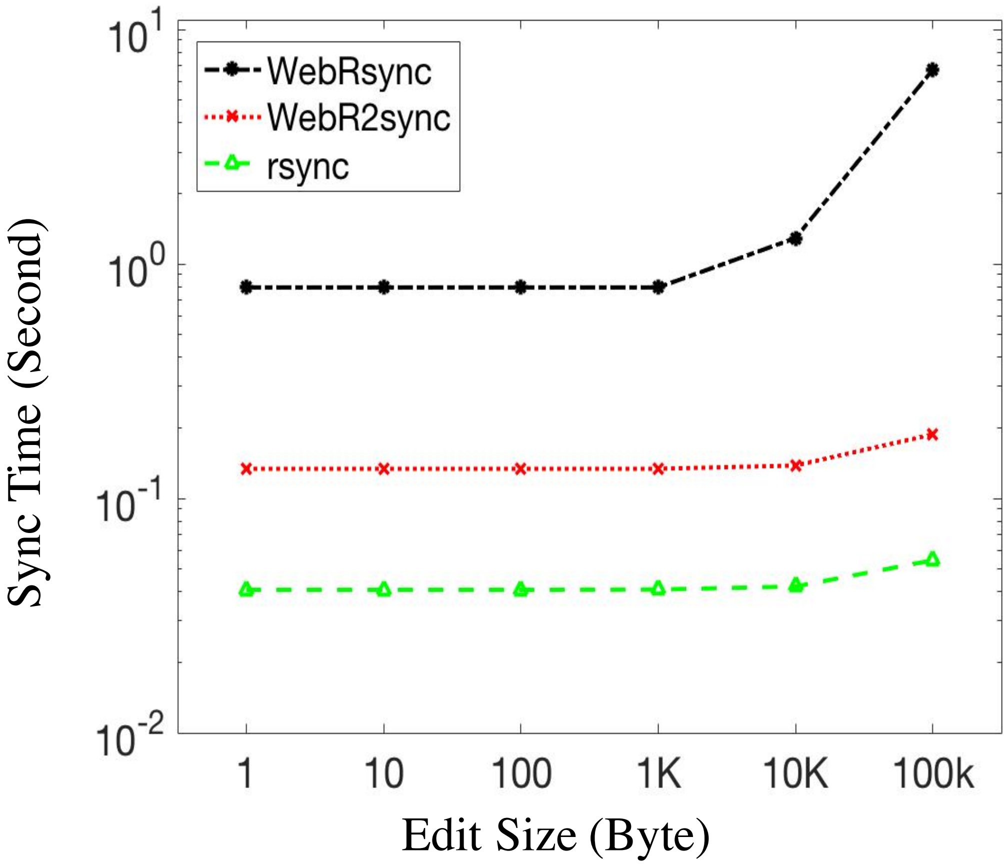
WebRsync

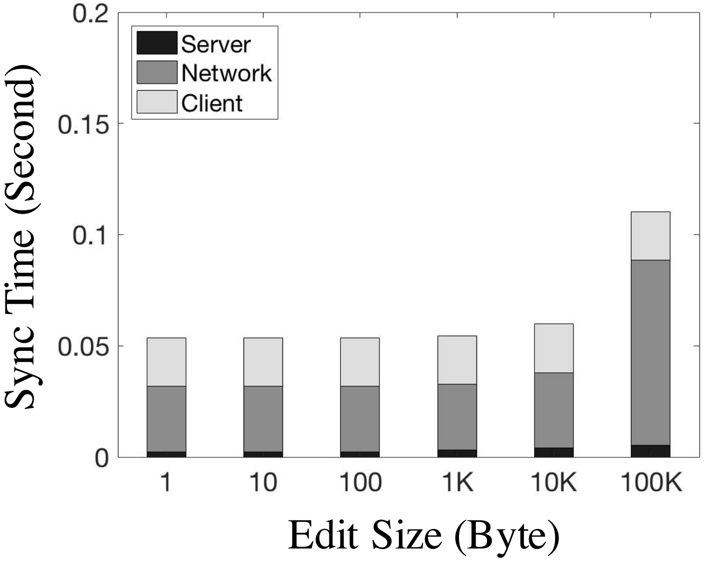
WebR2sync: Client-side optimization Reverse Computation Process



- Web Reverse Rsync: Reverse complicated computation from server to client.

Performance of WebR2sync

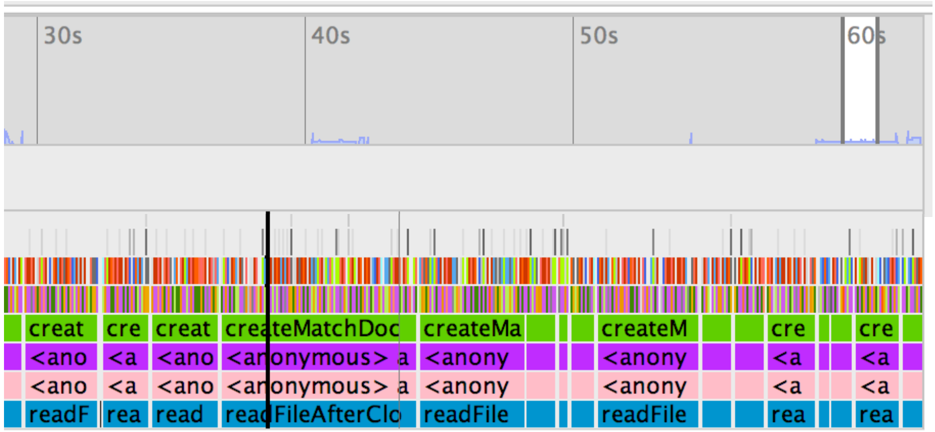




Issue: Server takes severely heavy overhead.

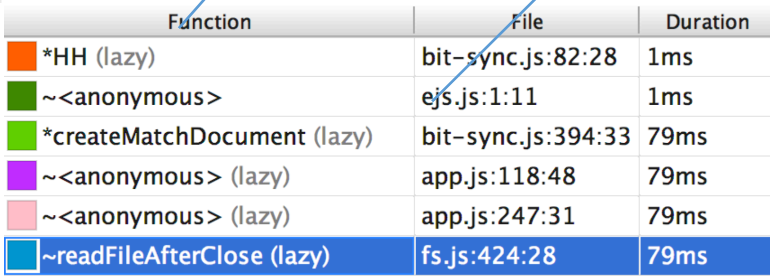
Server-side Overhead Profiling

Checksum searching and **block comparison** occupy **80%** of the computing time



MD5 Computing

Checksum Search



	Function	File	Duration
🟧	*HH (lazy)	bit-sync.js:82:28	1ms
🟩	~<anonymous>	ejs.js:1:11	1ms
🟩	*createMatchDocument (lazy)	bit-sync.js:394:33	79ms
🟪	~<anonymous> (lazy)	app.js:118:48	79ms
🟥	~<anonymous> (lazy)	app.js:247:31	79ms
🟦	~readFileAfterClose (lazy)	fs.js:424:28	79ms

- ➢ Use faster hash functions to replace MD5
- ➢ Reduce checksum searching overhead

Replacing MD5 with SipHash in Chunk Comparison

A comparison of pseudorandom hash functions

Hash Function	Collision Probability	Cycles per Byte
MD5	Low	5.58
Murmur3	High	0.33
Spooky	High	0.14
SipHash	Low	1.13

➡ SipHash remain low Collision Probability at much faster speed

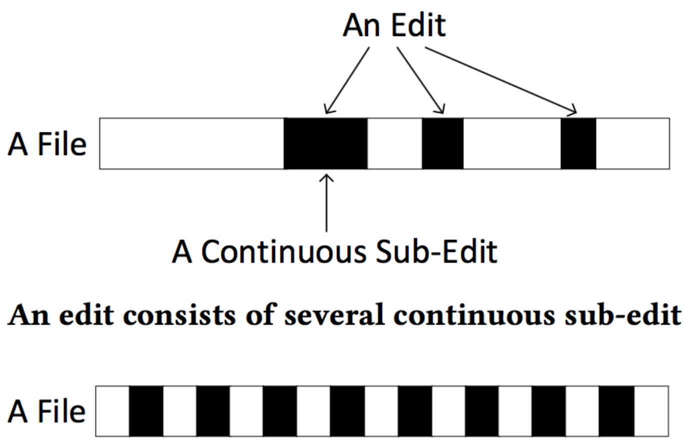
Solve Possible Hash Collision

- Replace MD5 with SipHash, may cause potential collisions (Probability p), so does MD5.

- **Our Solution:** Use Spooky (fastest method, collision probability p’).
  - The probability of collisions is p*p’

- **Alternative**: Use MD5 or other strong hash functions as a global verification.
  - Compute MD5 over whole file is expensive.

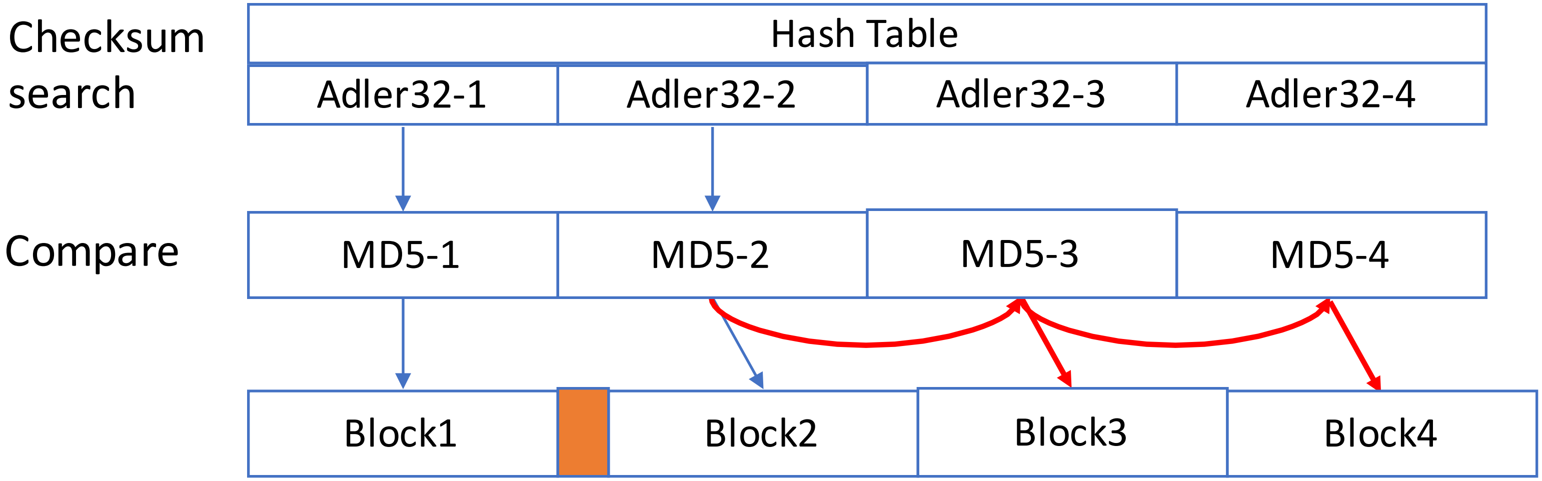
Reduce Chunk Searching by Exploiting Locality of File Edits.



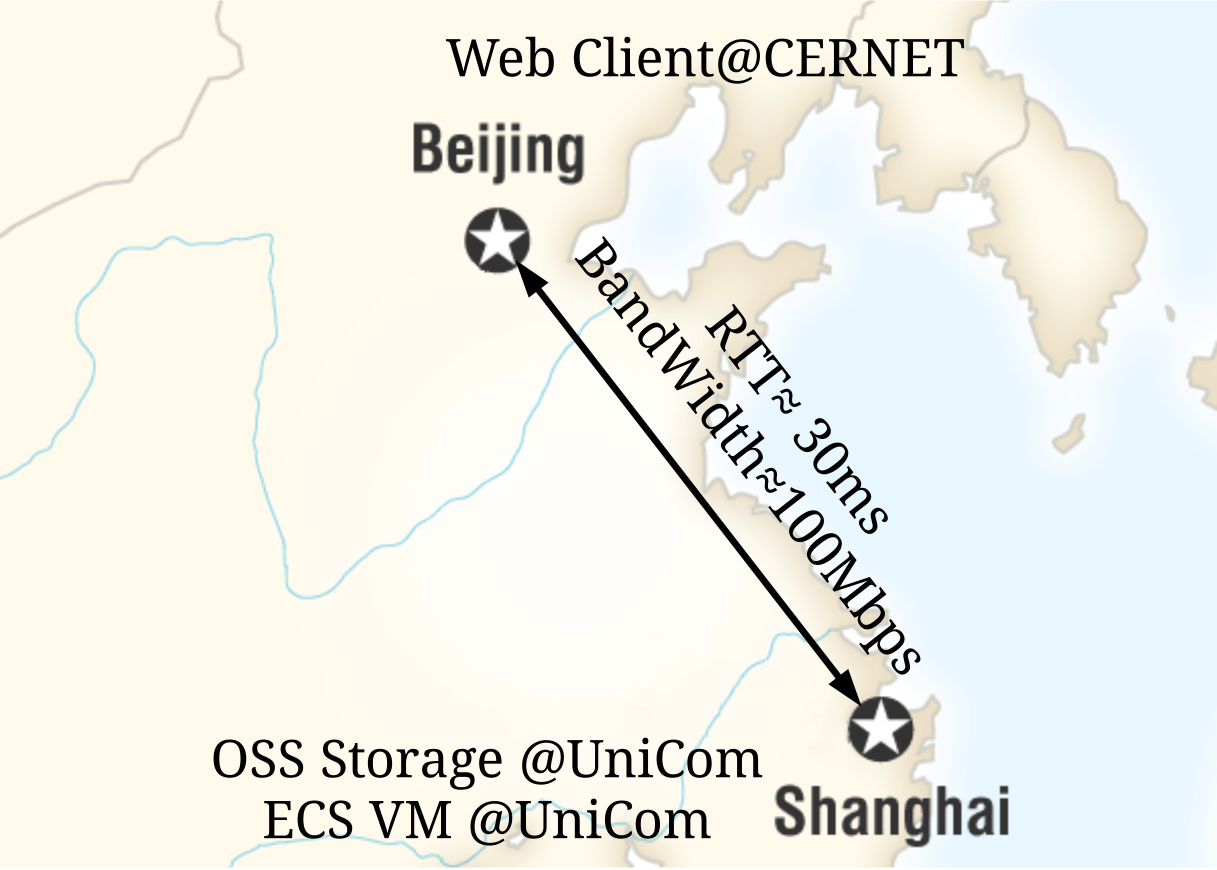
(a) An edit consists of several continuous sub-edits.

(b) The worst case of a file edit in terms of locality.

95% synchronized files have less than 10 edits.

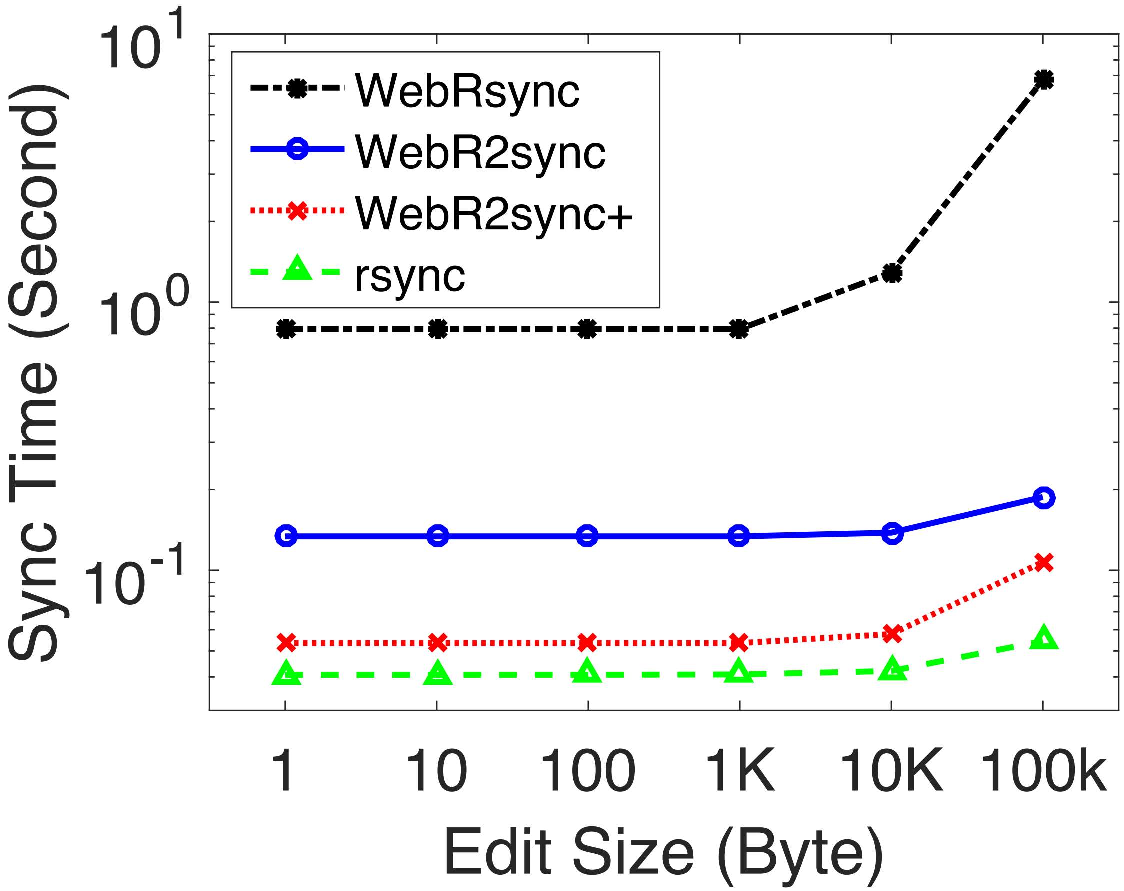


Evaluation Setup



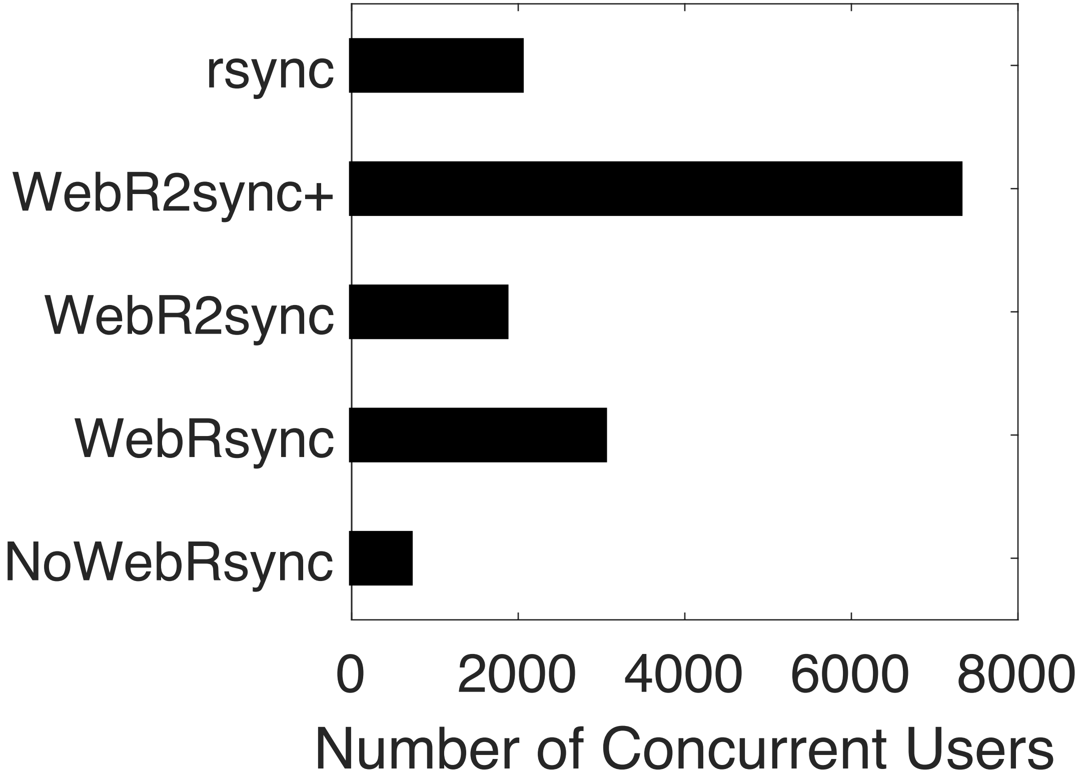
Basic experiment setup visualized in a map of China

Sync Time



WebR2sync+ is 2-3 times faster than WebR2sync and 15-20 times faster than WebRsync

Throughput



This throughput is as **4** times as that of WebR2sync/rsync and as **9** times as that of NoWebRsync.

Future Work

- Evaluate our approach under different edit modes
    - delete, insert, append
- Evaluate traffic efficiency
    - all the methods should have similar traffic efficiency
- Understand the effects of three optimizations
    - evaluate them separately

Discussion

- Probability of collisions of file checksums
- Characteristics of file operations in real-world scenarios from the perspective of sync
- Locality measure for deciding whether to apply locality-based optimization.

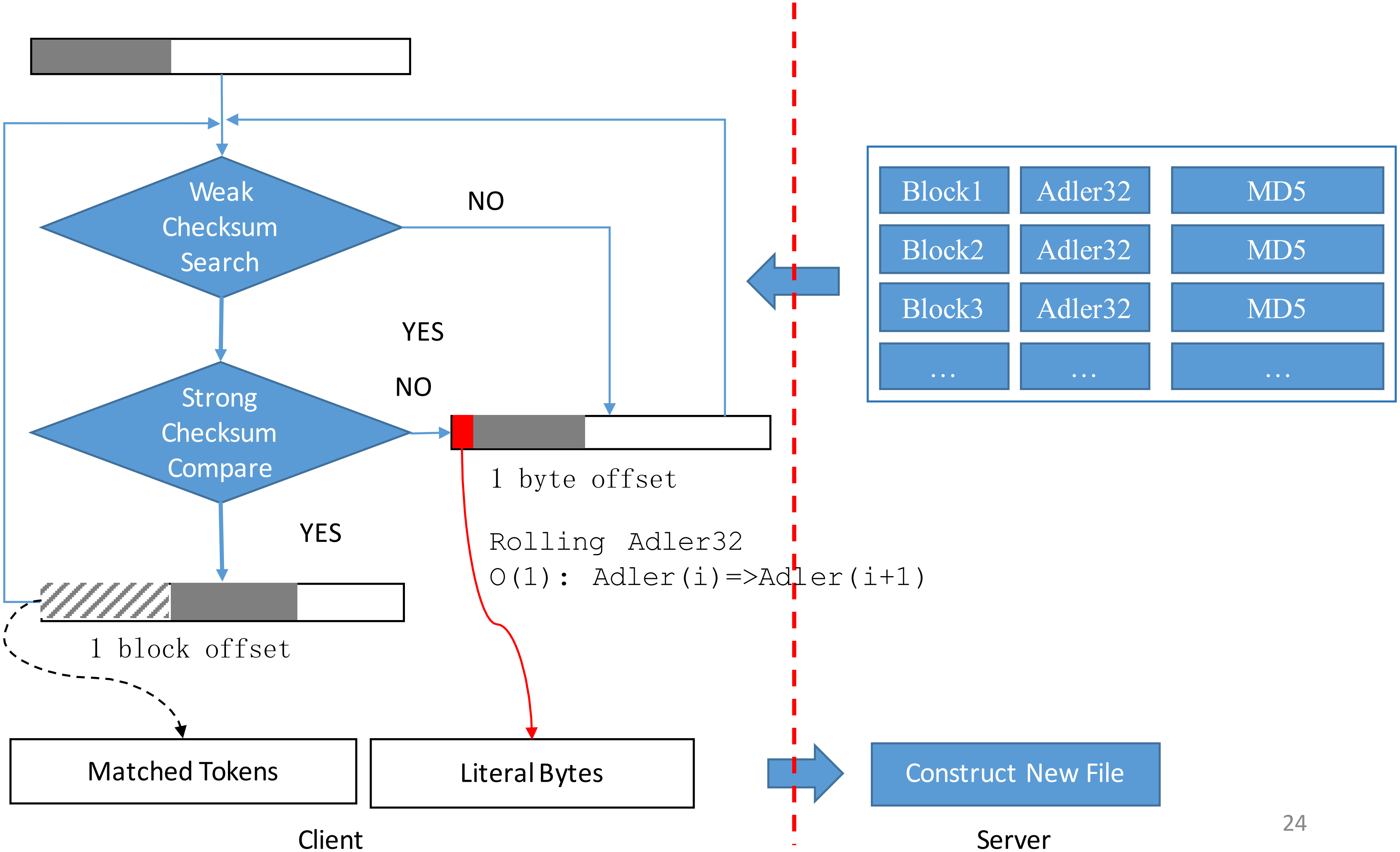
Conclusion

- WebR2sync+ is a practical solution for web-based delta sync
  - lightweight computation at the client side
  - optimized overhead at the server side
  - the server-side optimizations can be adopted in the traditional cloud storage architecture

Thanks!

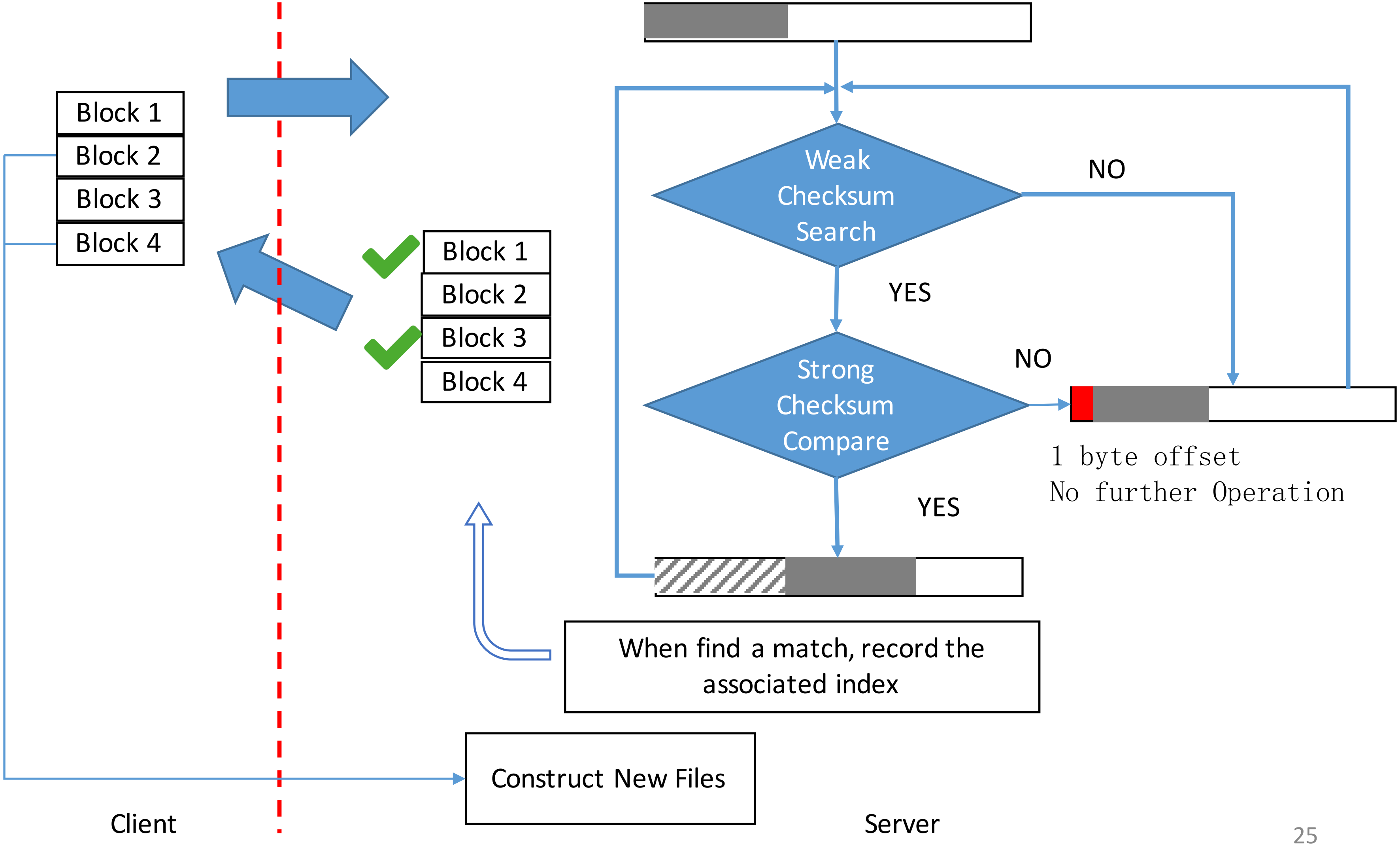
discussion

WebRsync Detailed Description

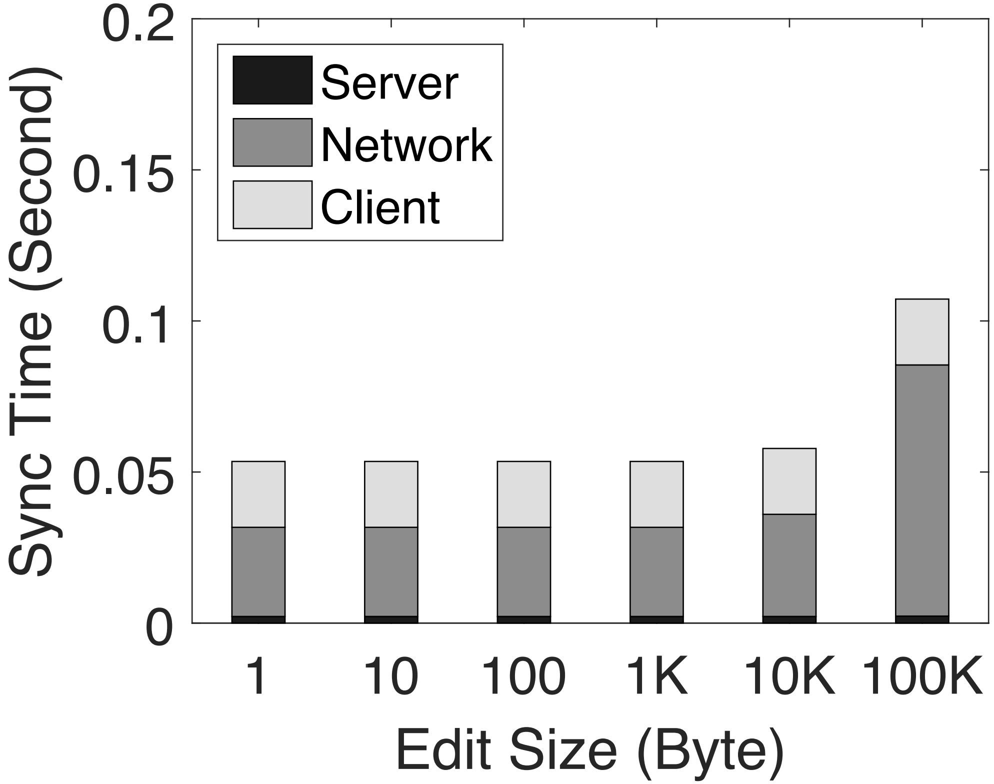


Block1	Adler32	MD5
Block2	Adler32	MD5
Block3	Adler32	MD5
…	…	…

WebR2sync: Flowchart and Data structure



Sync Time decomposed



WebR2sync+ client takes stable and shorter time.
Because of the Server-side optimization, computing time is much shorter both in client and server.