SeqCDC: Hashless Content-Defined Chunking for Data Deduplication

Sreeharsha Udayashankar, Abdelrahman Baba and Samer Al-Kiswany

UNIVERSITY OF WATERLOO

ACM / IFIP International Middleware Conference, 2024

Introduction

- Data explosion
  - Global data production expected to exceed 180 ZB by 2025 [1]
- Mechanisms
  - Distributed file systems [2]
  - Storage Architectures [3]
  - Data Deduplication [4]

[1] Arne Holst. *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025*. Statista, 2021.

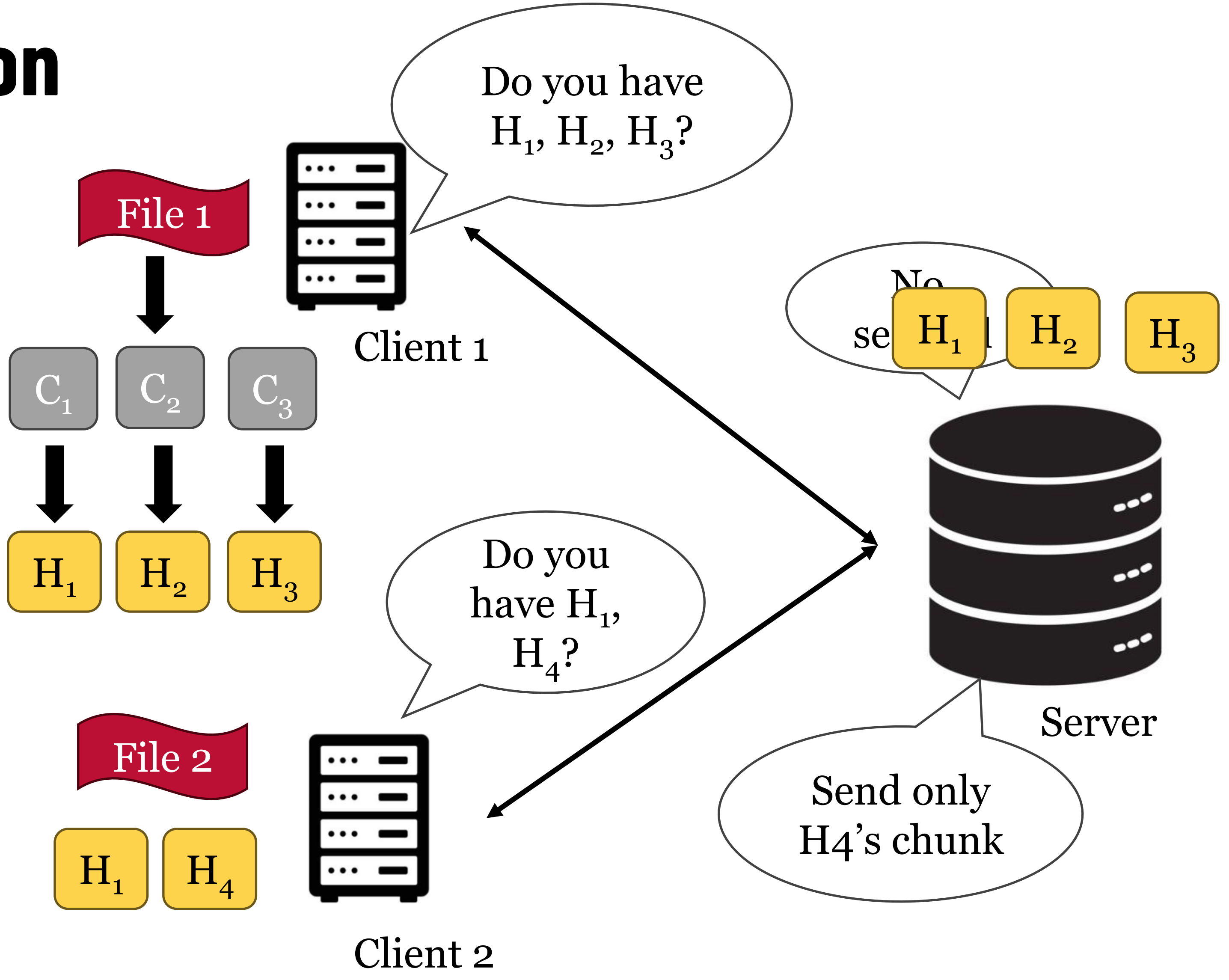
[2] Sanjay Ghemawat et al. *The Google File System*. SIGOPS Oper. Syst, 2003.

[3] Peter M Chen et al. *RAID: High-performance, reliable secondary storage*. ACM Computing Surveys (CSUR), 1994.

[4] Nagapramod Mandagere et al.. *Demystifying data deduplication*. ACM/IFIP/USENIX Middleware'08 Conference, 2008

Introduction: Deduplication

- Data Deduplication [5]
  - Identify and eliminate duplicate data
- Deduplication Overview [6]
  - File Chunking and Hashing
  - Fingerprint Comparison
  - Data Storage



[5] Dutch T Meyer et al. *A study of practical deduplication*. ACM Transactions on Storage (ToS), 2012.
[6] Alan Liu et al. *Dedupbench: A Benchmarking Tool for Data Chunking Techniques*. IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2023.

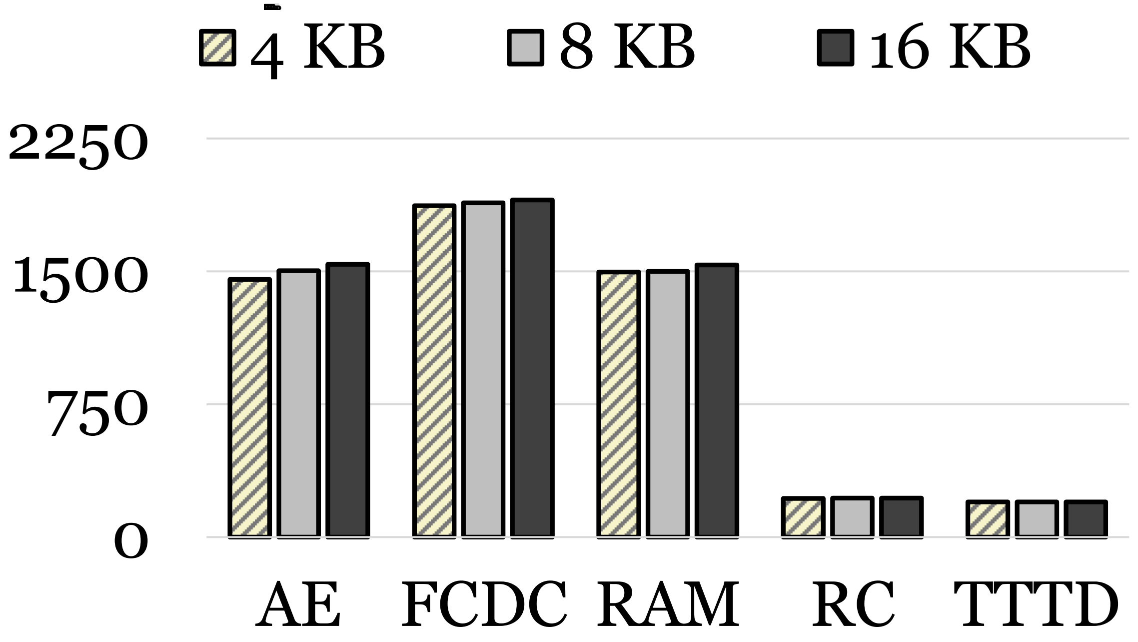
Introduction: File Chunking

- Content-Defined Chunking (CDC) [7, 8, 9]

Existing CDC algorithms are slow!

Existing CDC algorithms designed for small chunks!

- Systems in production favor larger chunks
  - Metadata concerns
  - Storage fragmentation concerns



(a) Chunking Throughput on Random Data

[7] Athicha Muthitacharoen et al. *A low-bandwidth network file system*. SOSP, 2001.

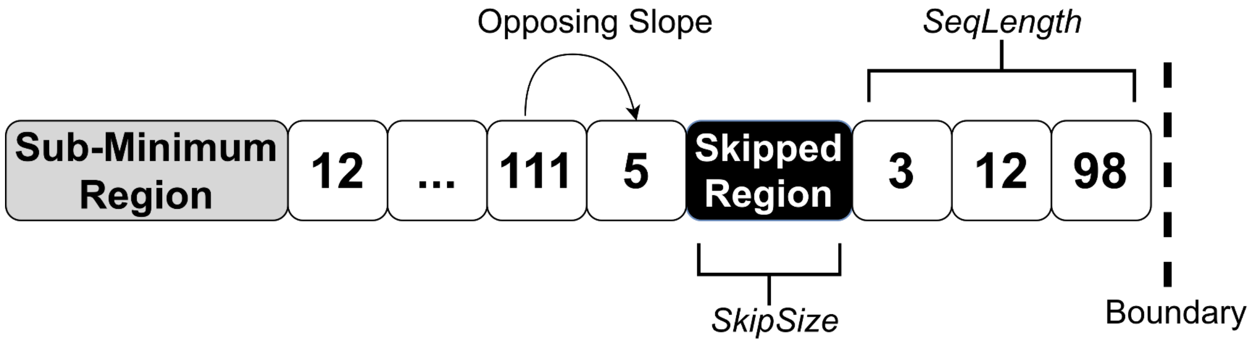
[8] Yucheng Zhang et al. *AE: An asymmetric extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication*. INFOCOM, 2015.

[9] Kave Eshghi et al. *A framework for analyzing and improving content-based chunking algorithms*. Hewlett-Packard Labs Technical Report, 2005

UNIVERSITY OF WATERLOO

SeqCDC

- Novel CDC algorithm
  - Lightweight boundary detection to reduce complexity
  - Content-defined skipping to selectively avoid scanning *unfavorable regions*
  - **1.5x – 3x higher throughput** than state-of-the-art



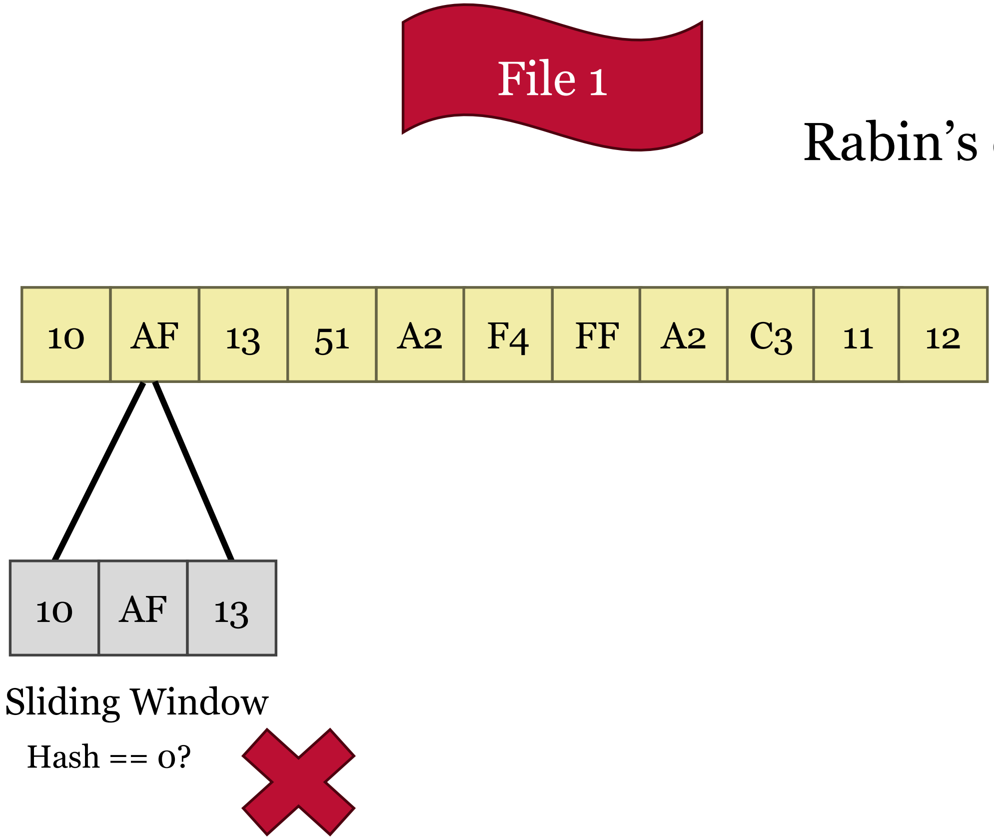
(a) SeqCDC

Outline

- Introduction
- Background
- Design
- Evaluation
- Conclusion

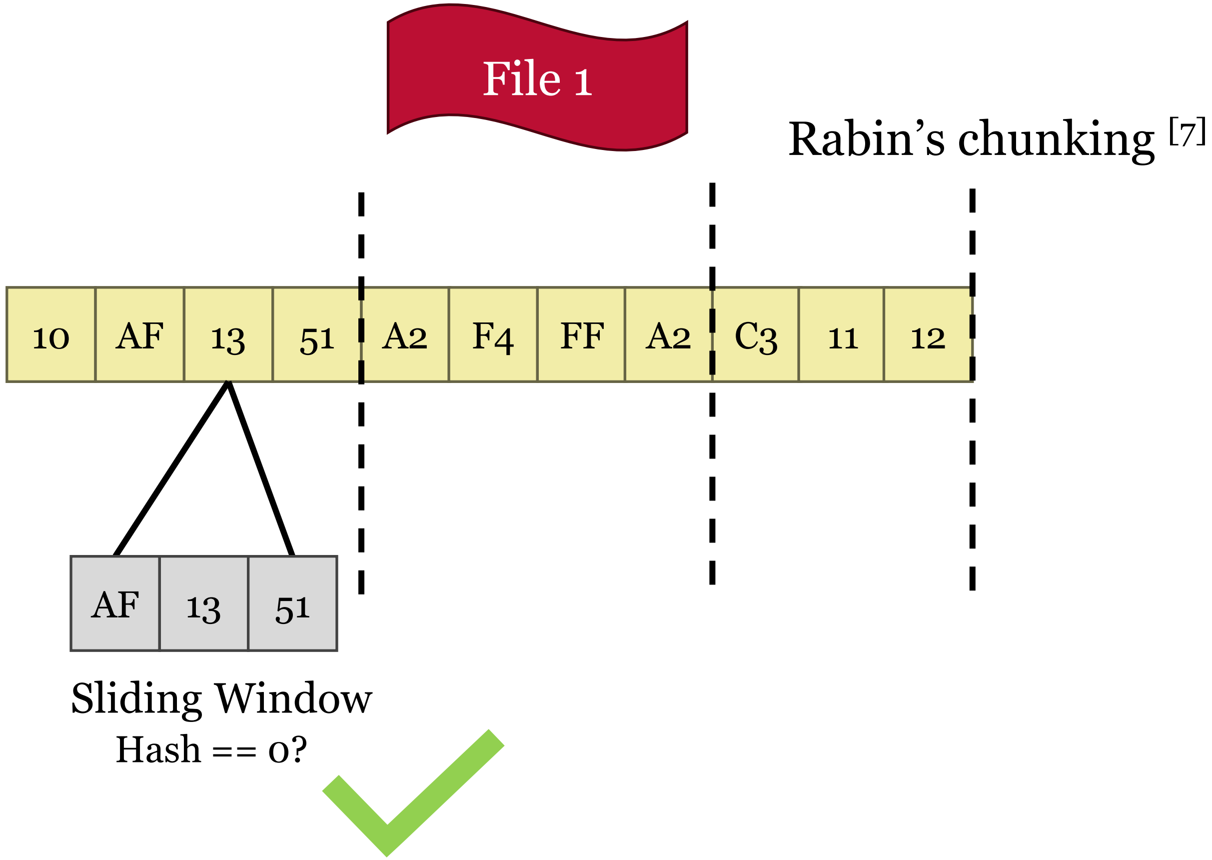
Background: Content-Defined Chunking

Rabin's chunking [7]



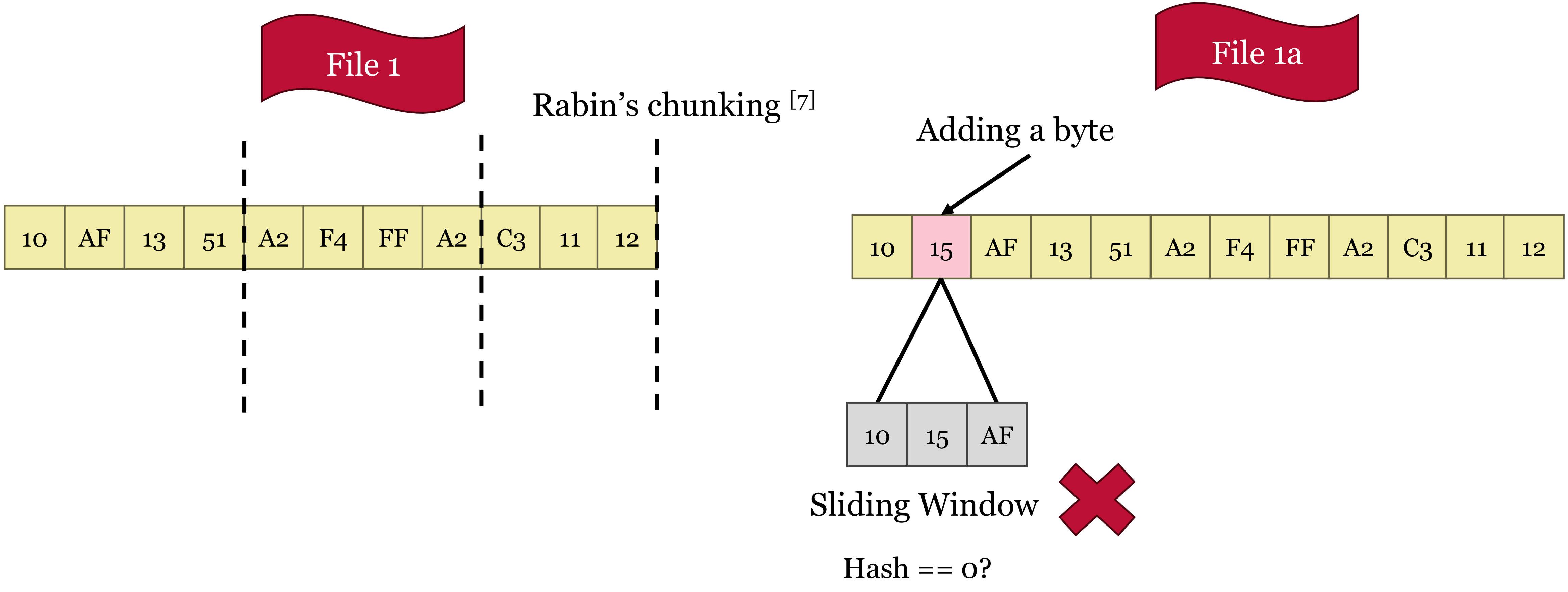
[7] Athicha Muthitacharoen et al. *A low-bandwidth network file system*. SOSP, 2001.

Background: Content-Defined Chunking



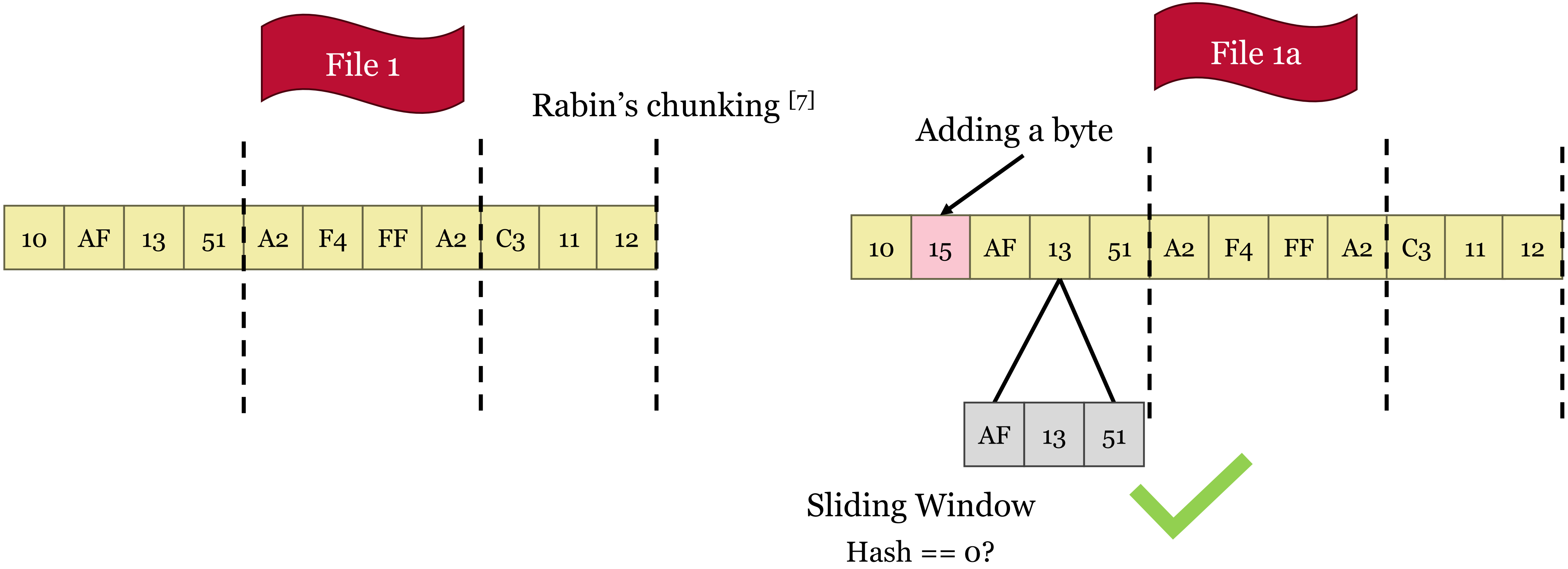
[7] Athicha Muthitacharoen et al. *A low-bandwidth network file system*. SOSP, 2001.

Background: Content-Defined Chunking



[7] Athicha Muthitacharoen et al. *A low-bandwidth network file system*. SOSP, 2001.

Background: Content-Defined Chunking



Only one chunk is different, the rest are the same

[7] Athicha Muthitacharoen et al. *A low-bandwidth network file system*. SOSP, 2001.

Background: Issues with Traditional CDC

Traditional CDC

- Expensive boundary detection
- Large amount of data to scan
- Scanned amount does not change with chunk size

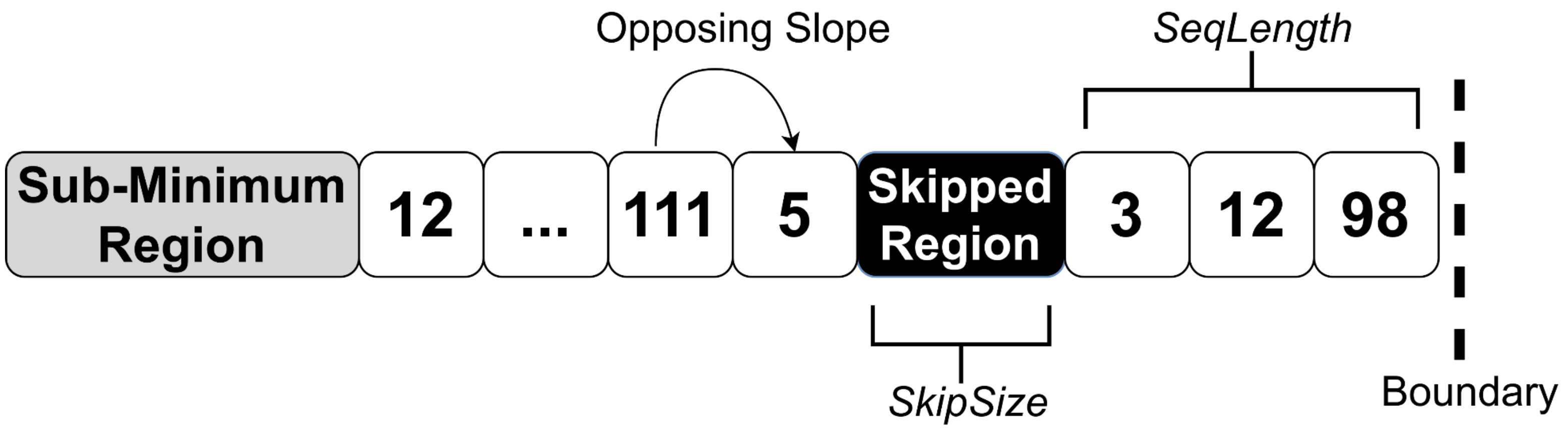
Can we chunk the data without scanning *all* of it?

Outline

- Introduction
- Background
- Design
- Evaluation
- Conclusion

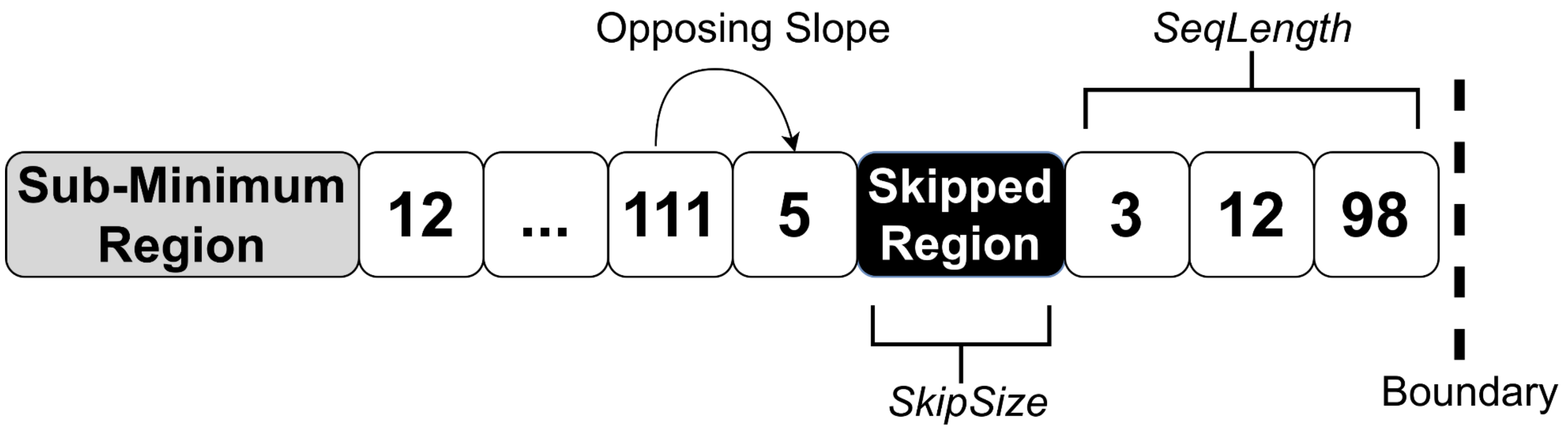
SeqCDC

- Insert chunk boundaries when fixed-length sequences are detected
  - Monotonically increasing / decreasing
  - *SeqLength*



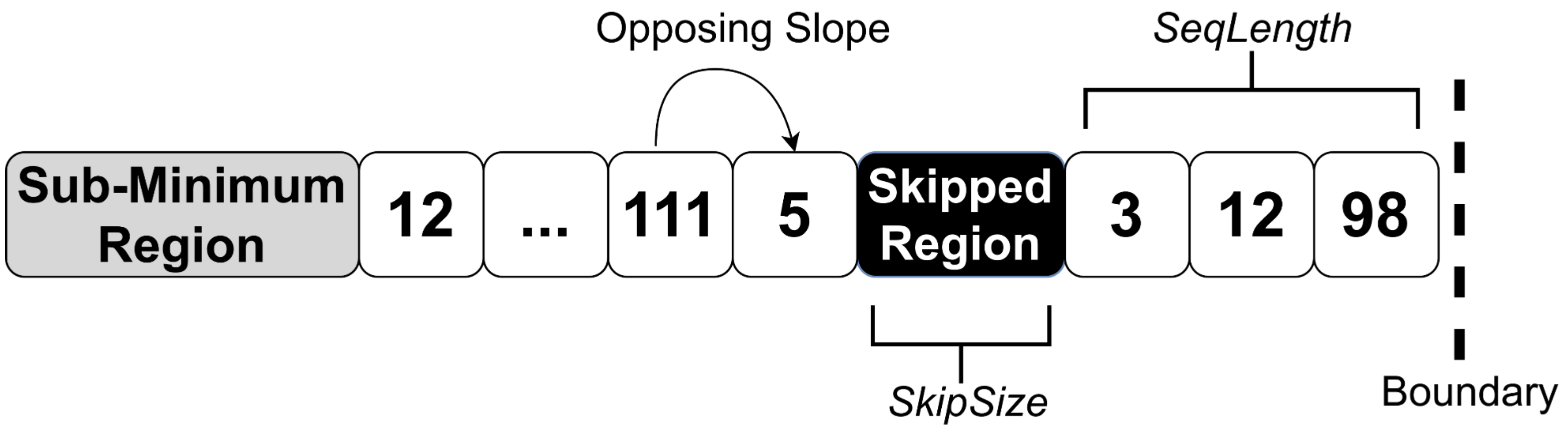
SeqCDC

- Skip scanning the sub-minimum region
  - Minimum chunk size
  - Similar to existing CDC algorithms



SeqCDC

- Skipped regions
  - *Unfavorable:* Opposing slope bytes
  - When triggered, skip scanning the next *SkipSize* bytes



SeqCDC

Traditional CDC

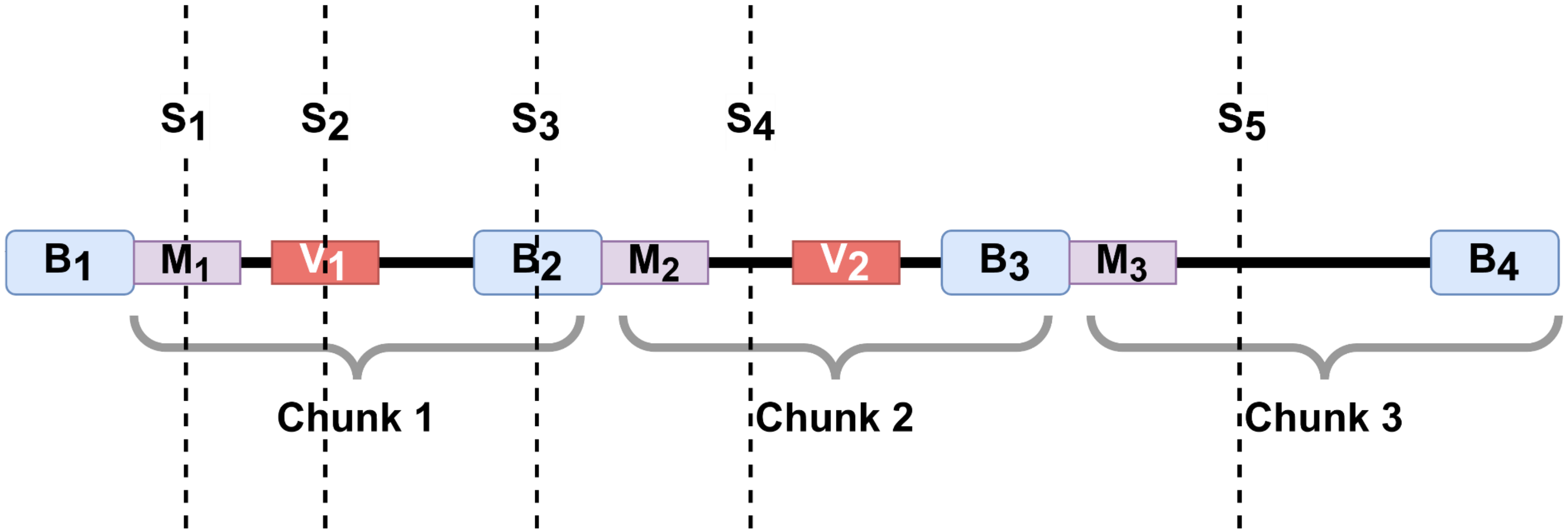
- Expensive boundary detection
- Large amount of data to scan
- Scanned amount does not change with chunk size

SeqCDC

- Lightweight and hashless
- Selectively skip *unfavorable regions*
- Larger chunk size => Larger *SkipSize*

SeqCDC

- How much is byte-shift detection affected?
  - Small amounts on real datasets
  - Detailed analysis in paper



a) Different kinds of byte shifts with SeqCDC

Evaluation

Datasets

Rust
bitnami by vmware
redis
Linux
MySQL

Metrics

Space Savings

Speed / Throughput

Chunk Size Distribution

Alternatives

AE [8]

FastCDC [12]

Rabin's Chunking [7]

RAM [13]

TTTD [9]

[7] Athicha Muthitacharoen et al. *A low-bandwidth network file system*. SOSP, 2001.

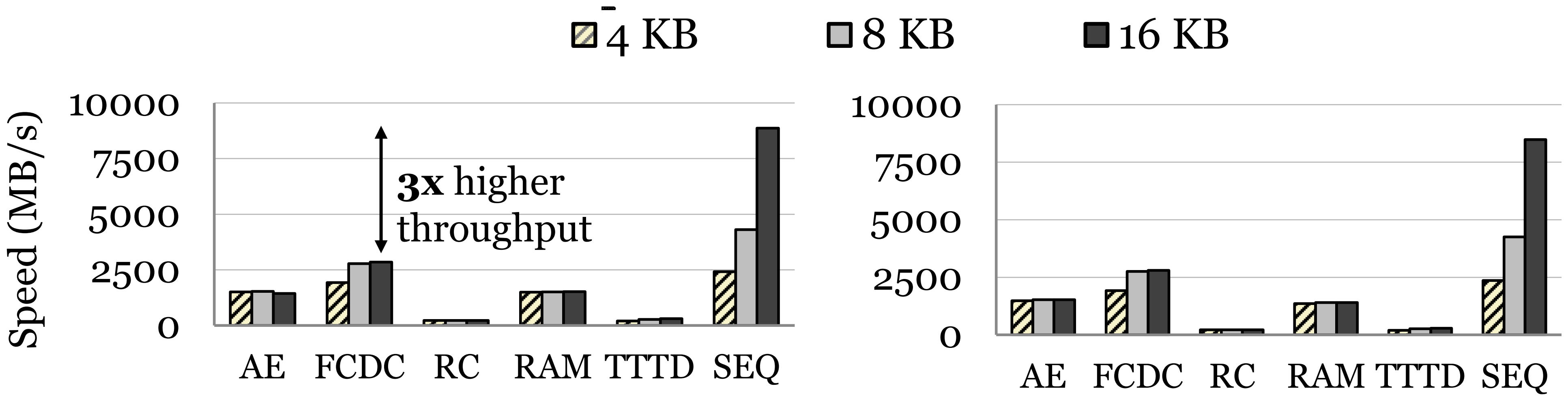
[8] Yucheng Zhang et al. *AE: An asymmetric extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication*. INFOCOM, 2015.

[9] Kave Eshghi et al. *A framework for analyzing and improving content-based chunking algorithms*. Hewlett-Packard Labs Technical Report, 2005

[12] Wen Xia et al. *FastCDC: A fast and efficient content-defined chunking approach for data deduplication*. USENIX ATC, 2016.

[13] Ryan NS Widodo et al. *A new content-defined chunking algorithm for data deduplication in cloud storage*. Future Generation Computer Systems, 2017.

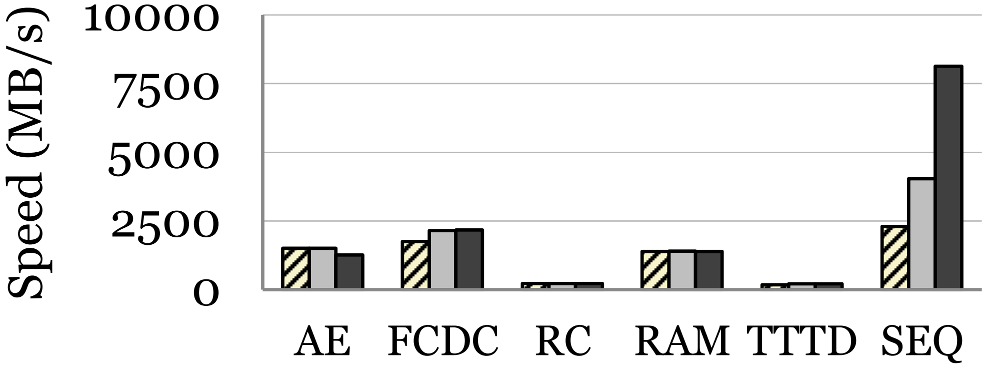
Evaluation: Chunking Throughput



(a) Bitnami VMs

(b) Rust build server backups

SeqCDC achieves **3x higher throughput** at large chunk sizes



(c) Redis backups

Summary

- Data deduplication is used to improve storage efficiency
  - Content-defined chunking algorithms critical to system performance
- SeqCDC
  - Lightweight boundary detection and content-defined data skipping
  - 1.5x – 3x higher chunking throughput with similar space savings
- **Code:** https://github.com/UWASL/dedup-bench

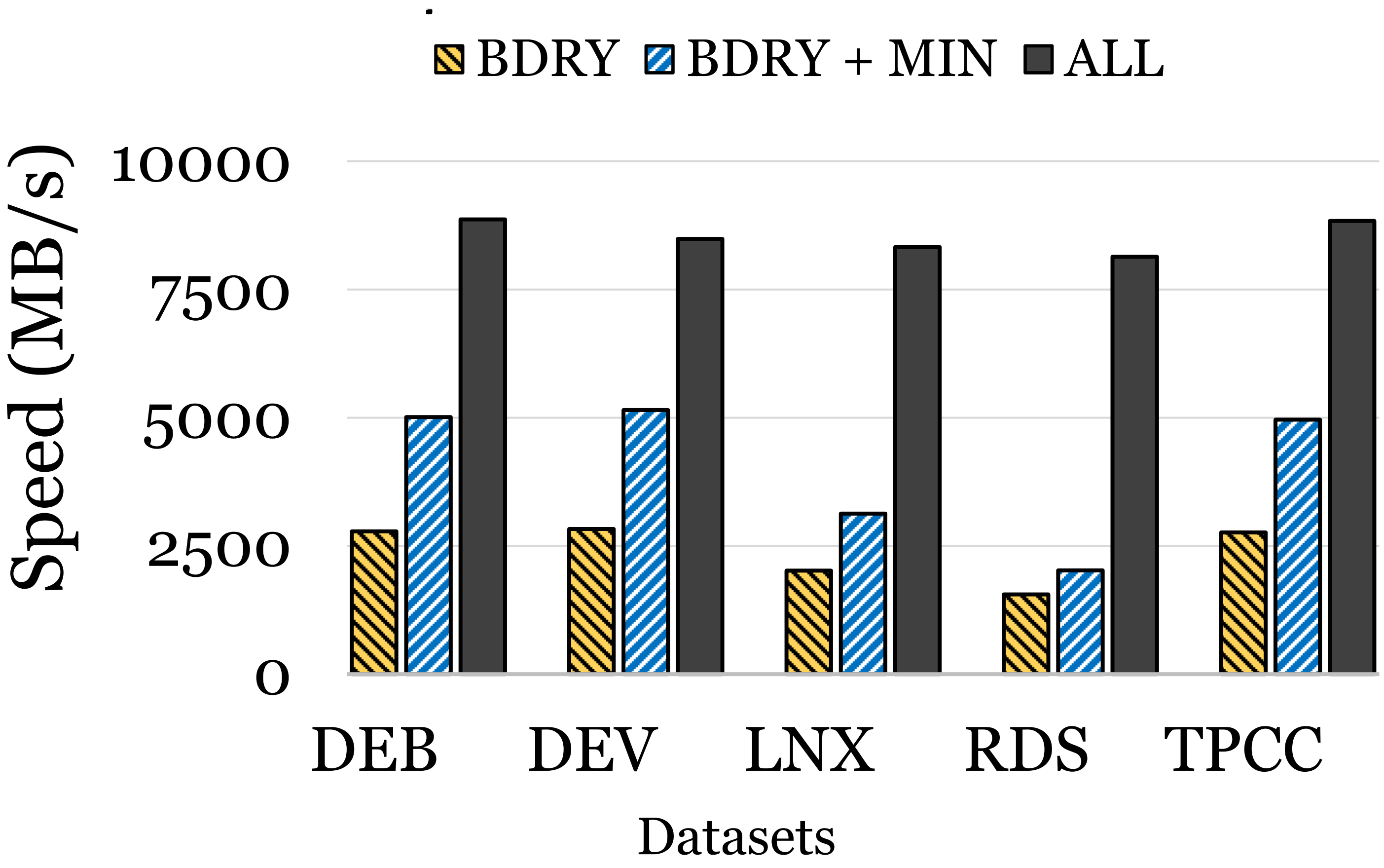
Artifacts Available acm

Artifacts Evaluated Functional acm

UNIVERSITY OF WATERLOO

UNIVERSITY OF
WATERLOO

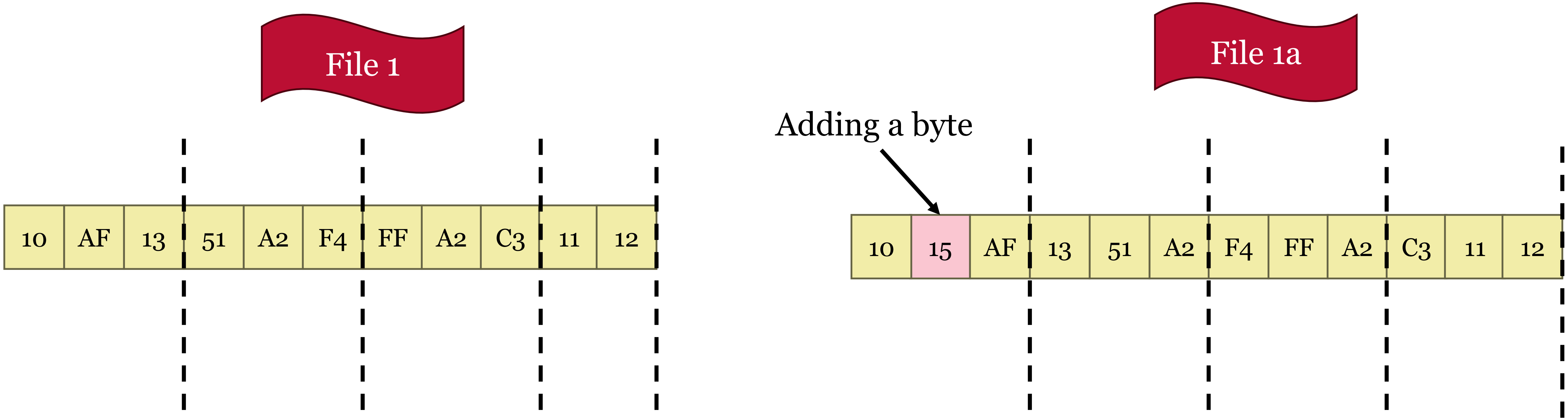
Evaluation: Throughput Breakdown



All optimizations contribute to SeqCDC's throughput

(d) SeqCDC Speed Breakdown – 16KB

Do we really need CDC?



Fixed-size chunking

Similar files but completely different chunks!

Datasets

Dataset	Size	Information	XC_4K
DEB	40GB	65 Debian VM Images obtained from the VMware Marketplace [28]	28.1%
DEV	230GB	100 backups of a Rust [29] nightly build server	90.9%
LNX	65GB	160 Linux kernel distributions in TAR format [30]	34.8%
RDS	122GB	100 Redis [31] snapshots with `redis-benchmark` runs	33.7%
TPCC	106GB	25 snapshots of a MySQL [32] VM running TPC-C [33].	54.6%

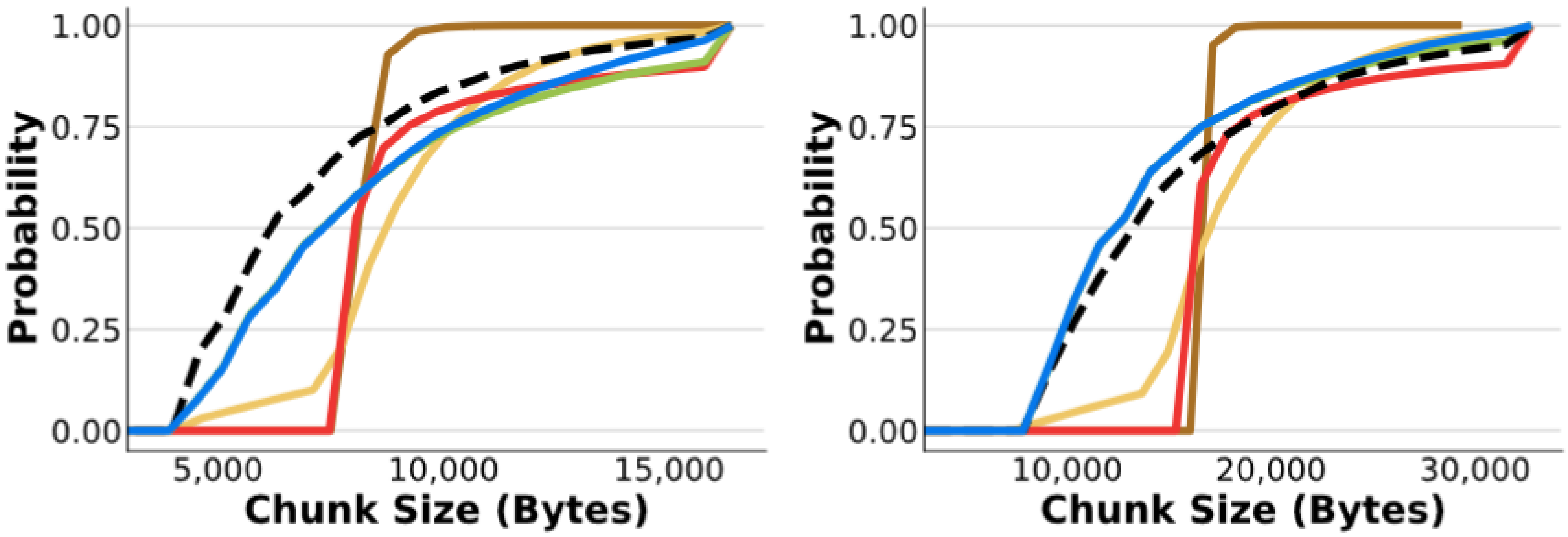
Table 1: Dataset Information

SeqCDC Space Savings

Dataset	CDC	4KB	8KB	16KB
DEB	AE	41.99%	33.23%	21.47%
	FCDC	43.83%	36.97%	27.77%
	RC	44.38%	36.16%	27.22%
	RAM	42.98%	34.63%	22.61%
	TTTD	**45.06%**	37.13%	**27.94%**
	SeqCDC	42.77%	**37.76%**	27.62%
DEV	AE	98.00%	97.72%	97.21%
	FCDC	98.17%	98.06%	97.90%
	RC	98.21%	98.09%	97.97%
	RAM	98.05%	97.79%	97.31%
	TTTD	**98.22%**	**98.10%**	**97.98%**
	SeqCDC	98.13%	98.03%	97.82%
LNX	AE	59.41%	45.33%	31.67%
	FCDC	59.16%	44.35%	33.64%
	RC	67.02%	49.28%	35.40%
	RAM	57.94%	42.90%	29.12%
	TTTD	**68.46%**	**51.06%**	**36.92%**
	SeqCDC	63.13%	49.46%	33.26%
RDS	AE	94.66%	92.86%	91.04%
	FCDC	93.82%	92.04%	90.15%
	RC	94.31%	92.32%	90.57%
	RAM	**95.67%**	**94.09%**	**92.03%**
	TTTD	95.2%	93.30%	91.55%
	SeqCDC	94.86%	92.54%	88.78%
TPCC	AE	86.58%	84.96%	81.58%
	FCDC	87.18%	86.74%	86.17%
	RC	87.24%	86.80%	86.37%
	RAM	86.71%	85.21%	81.67%
	TTTD	**87.29%**	**86.84%**	**86.40%**
	SeqCDC	87.04%	86.68%	85.83%

Table 3: Space savings of CDC techniques

SeqCDC: Chunk Size Distribution



(a) 8 KB

(b) 16 KB

UNIVERSITY OF WATERLOO