SiLo: A Similarity-Locality based Near-Exact Deduplication Scheme with Low RAM Overhead and High Throughput

Wen Xia[†] Hong Jiang[‡] Dan Feng[†] Yu Hua[†,‡]

[†] Huazhong University of Science and Technology
[‡] University of Nebraska-Lincoln

Data Deduplication

- **Why deduplication ?**
  - **Reduces the storage space overheads.**
  - **Minimizes the network transmission of redundant data.**
- **Deduplication Technique.**
  - **Data fingerprints: MD5, SHA-1, SHA-256.**
  - **Remove duplicate data by checking its fingerprints.**
- **Deduplication granularity.**
  - **File-level.**
  - **Chunk-level.**
    - Fixed-length Chunking; Content Defined Chunking.

Deduplication Challenges

Files $\xrightarrow{\text{Chunking}}$ Chunks $\xrightarrow{\text{Hashing}}$ Fingerprints $\xrightarrow{\text{Indexing}}$ Store

The Scalability of Deduplication Indexing

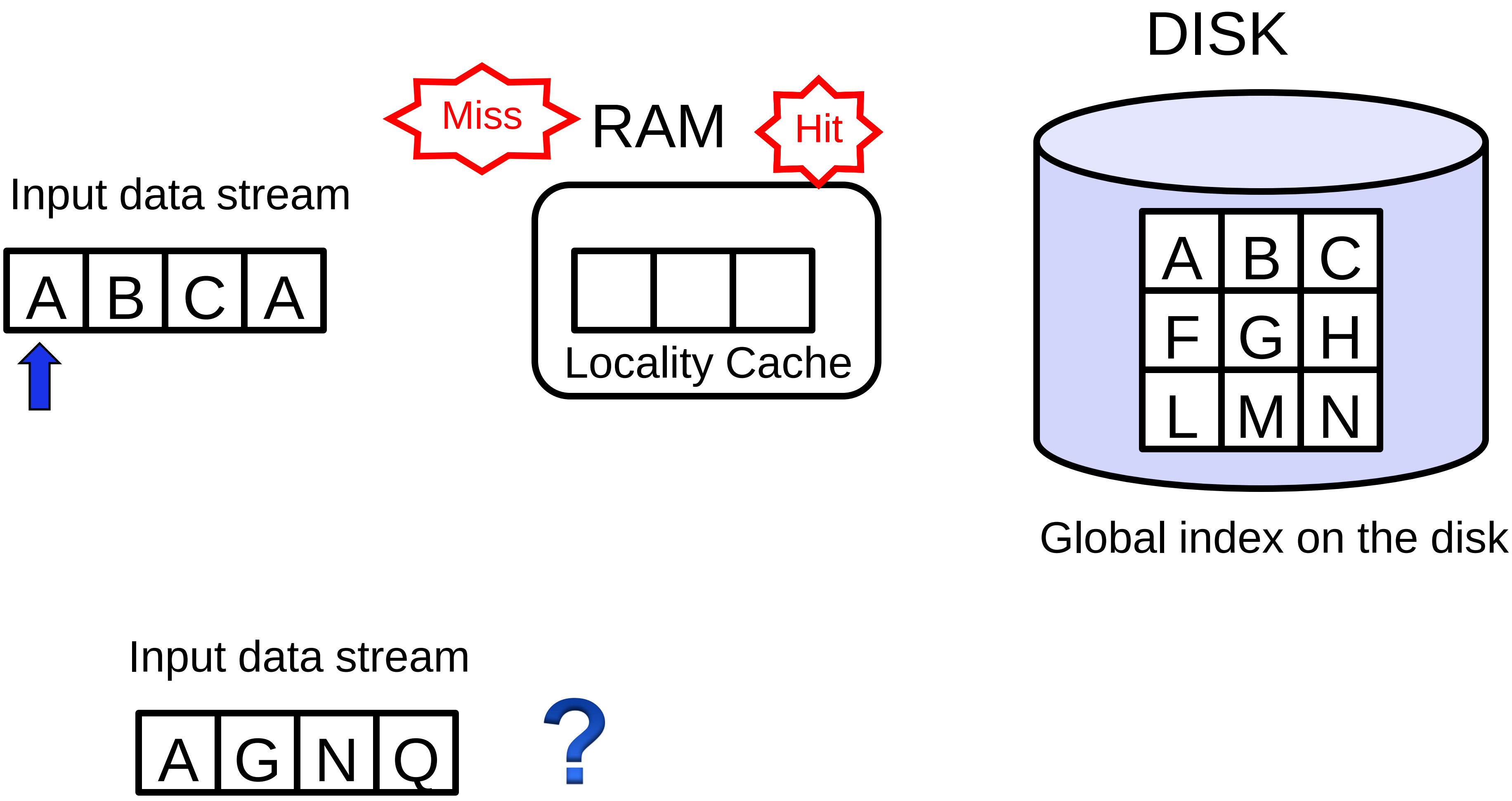
Deduplicate 800 TB unique data.

SHA-1 signature.
Avg. 8KB Chunk.

2TB Fingerprints are generated .

Global indexing.
Disk bottleneck.

Locality-based Approaches (1)



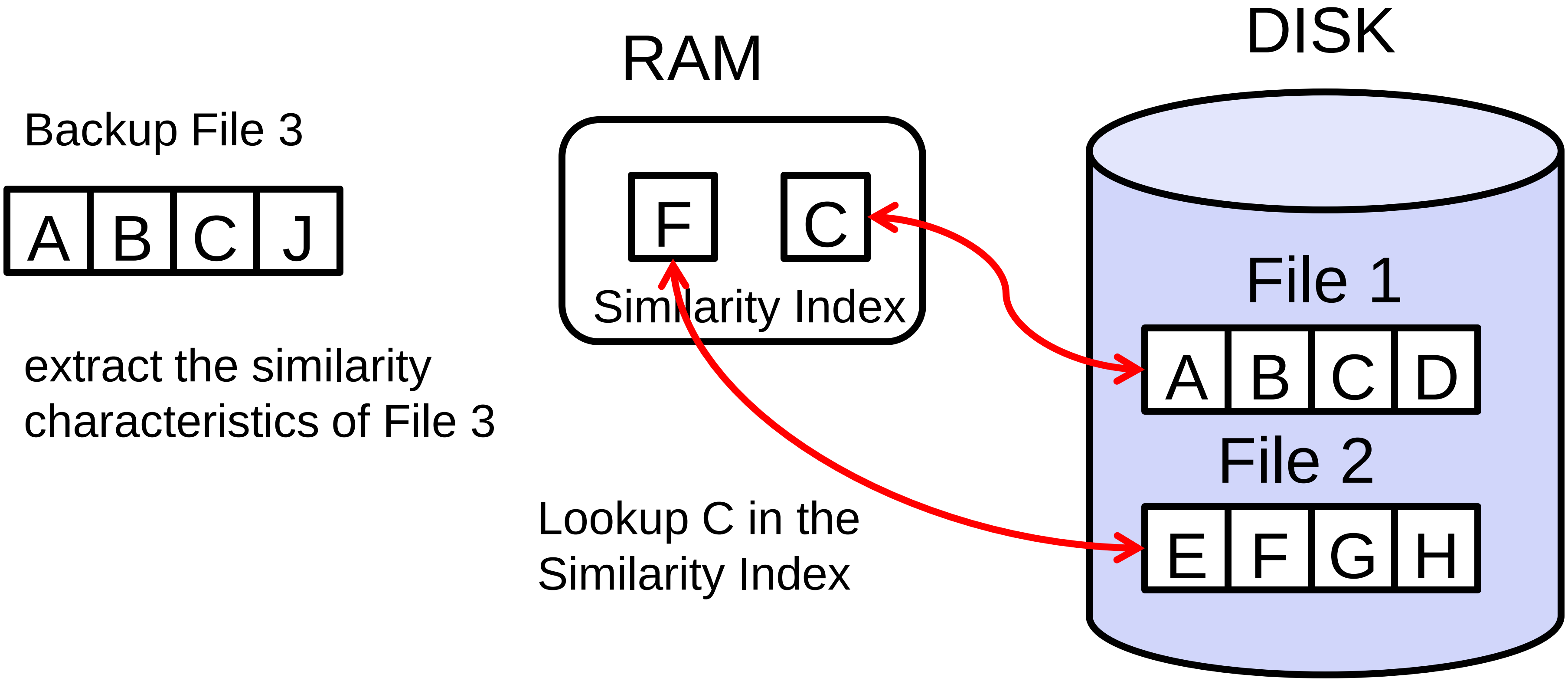
Locality based approaches (2)

- DDFS, Sparse Indexing, ChunkStash.

- Exploit locality of backup streams.
  - It maximizes the RAM utilization and reduces frequent accesses to on-disk index by retaining access locality in the locality cache.

- Limitations.
  - Work poorly when backup stream lacks locality.
  - High RAM consumed.

Similarity-based Approaches (1)



Achieve a single on-disk index access for chunk lookup per file thus avoid global index on the disk.

Deduplicate File 3 with File 1

Limitation of These Approaches (2)

- Exploit similarity of backup streams.
  - Avoid global indexing and achieve a single disk read.
  - Minimize the RAM overhead for indexing fingerprints.

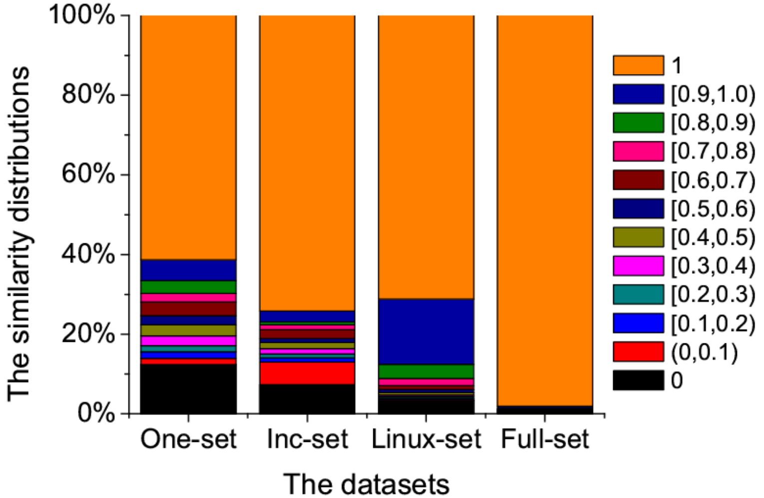
- Limitation.
  - Degradation of Deduplication efficiency.

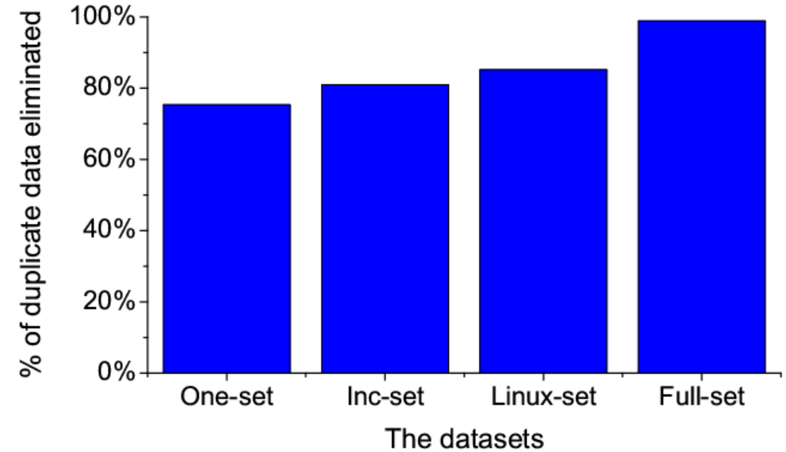
Theorem 1: Consider two files S1 and S2, Let min(H(S)) denote the similarity characteristic of file S. Then ***similarity degree*** *between the two files is quantified by the probability that min(H(S1))= min(H(S2)), which is dependent on the percentage of data common to both files:*

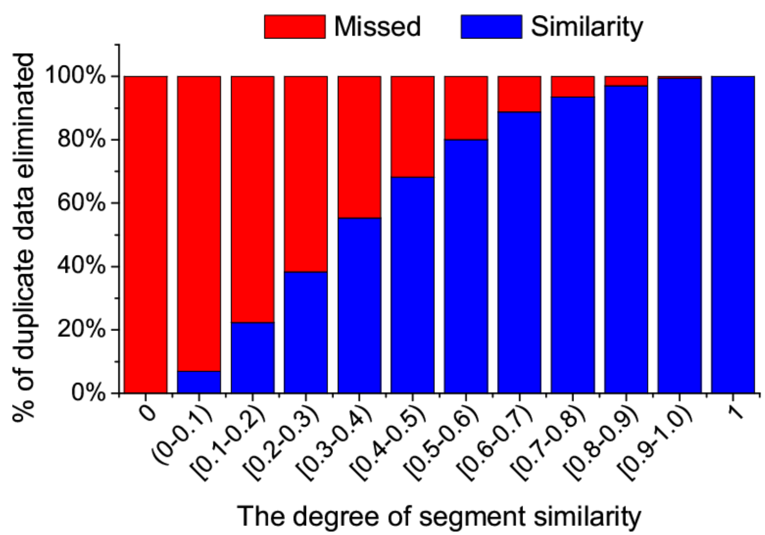
$$\Pr[\min(H(S_1)) = \min(H(S_2))] = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Evaluation of Similarity Approach

Similarity based Deduplication efficiency is dependent on the similarity degree of data stream







Observation

- The deduplication of small files and large files.

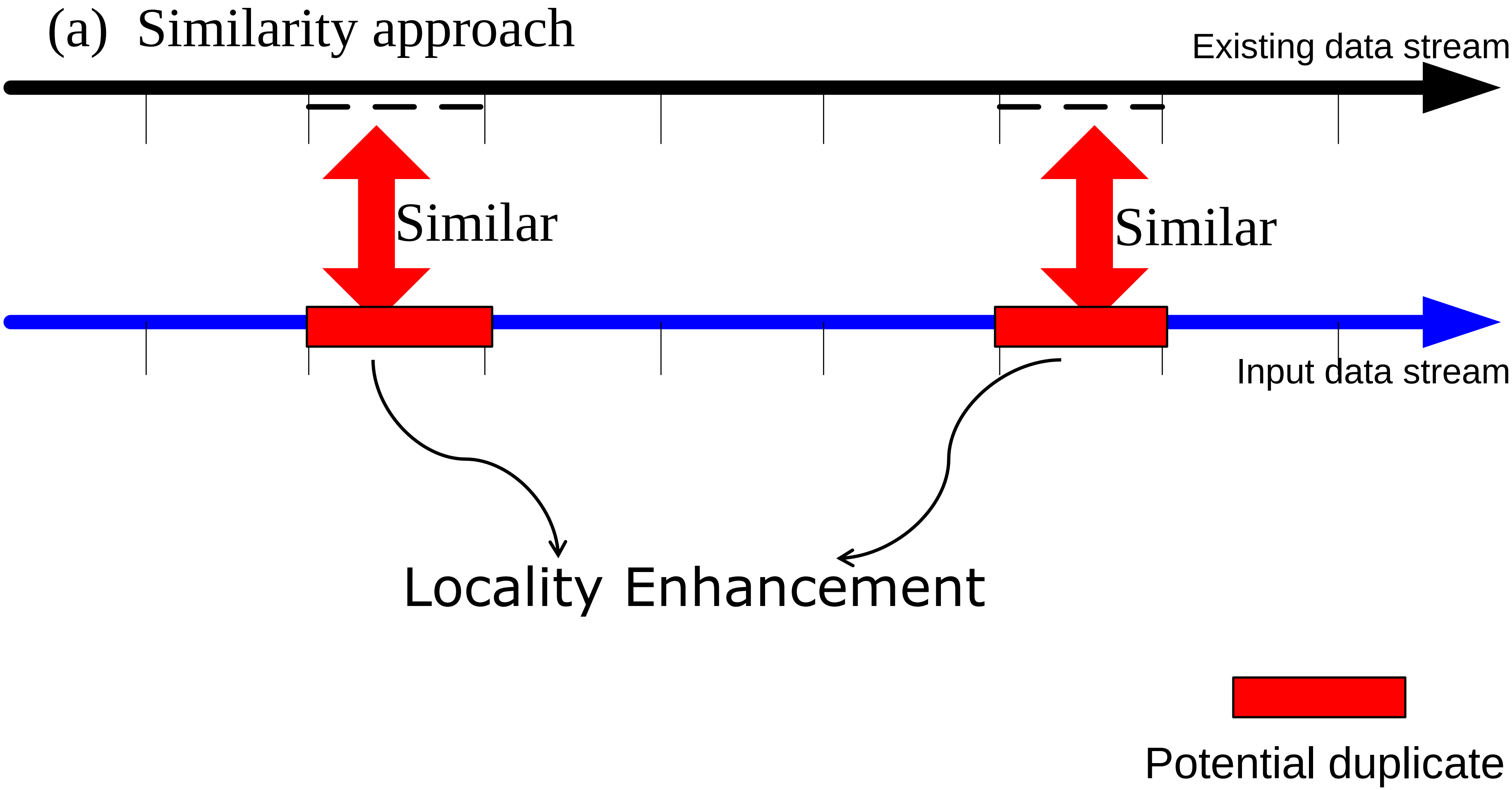
	Small files (≤ 64KB)	Large files (≥ 2 MB)
Percentage of total file number	≥ 80%	≤ 20%
Percentage of total space	≤ 20%	≥ 80%

Grouping many highly correlated small files into a segment to minimize dedupe overheads

Dividing the large files into many small segments to expose more similarity characteristics

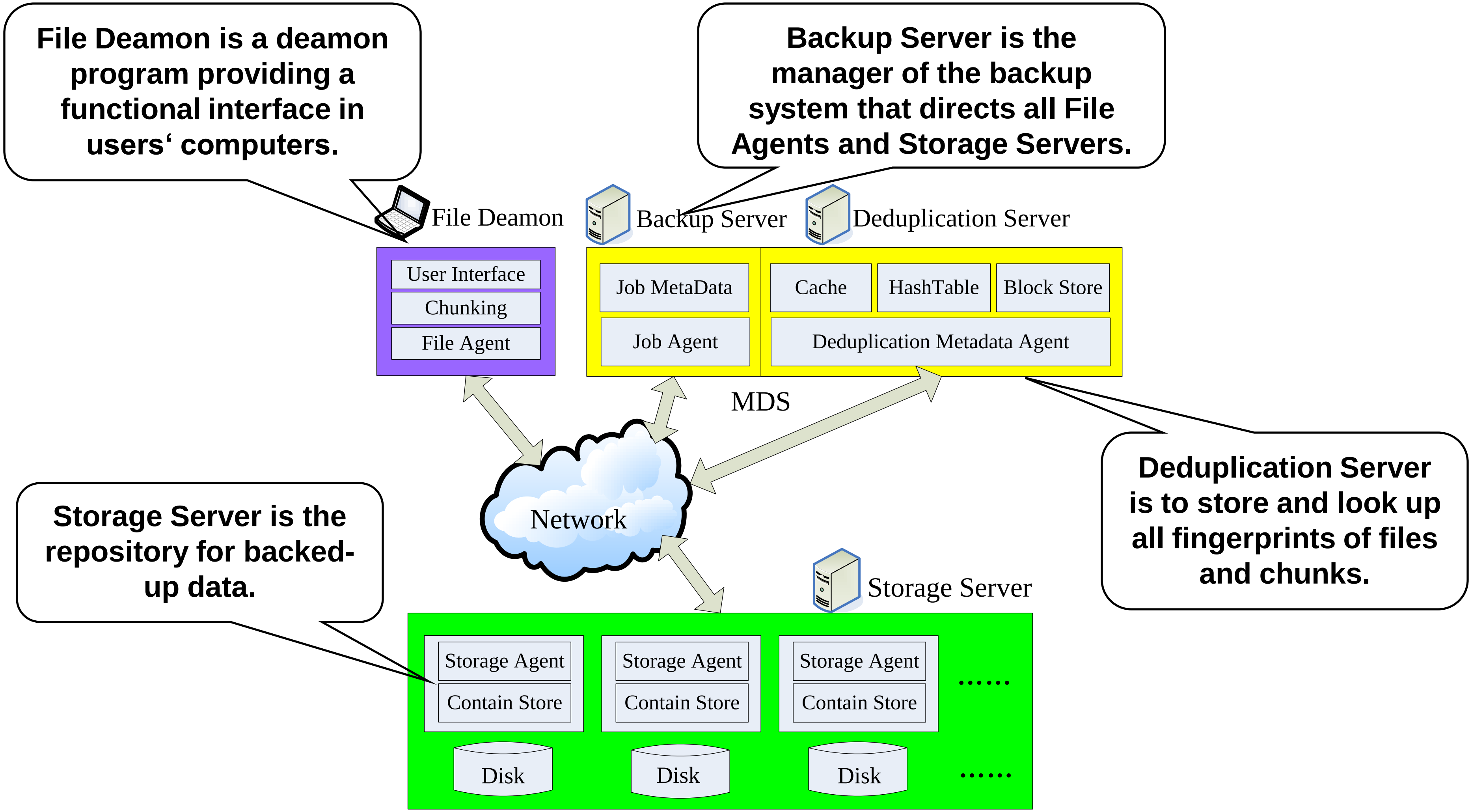
Intuition

- The combination of similarity and locality.

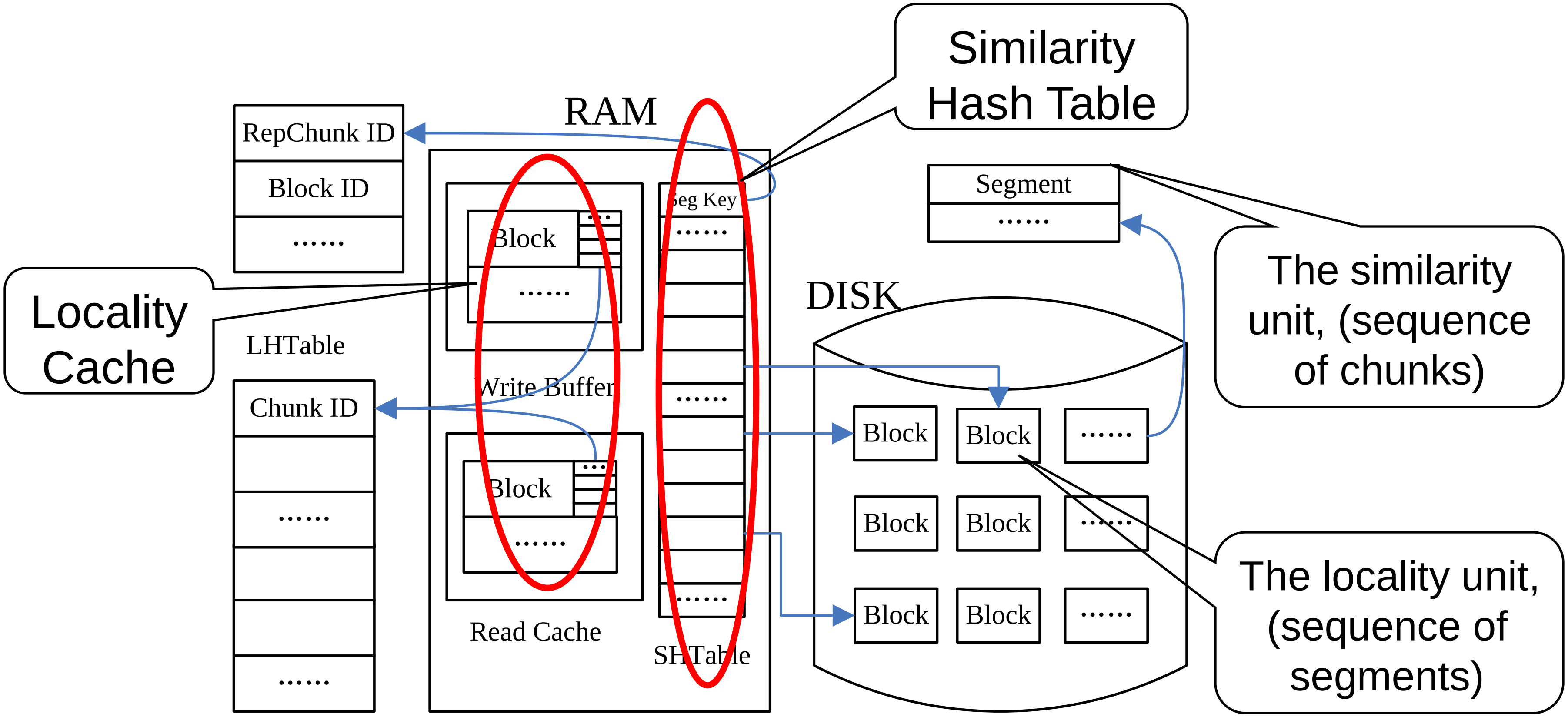


System Architecture Overview

- A disk-inline backup storage system.



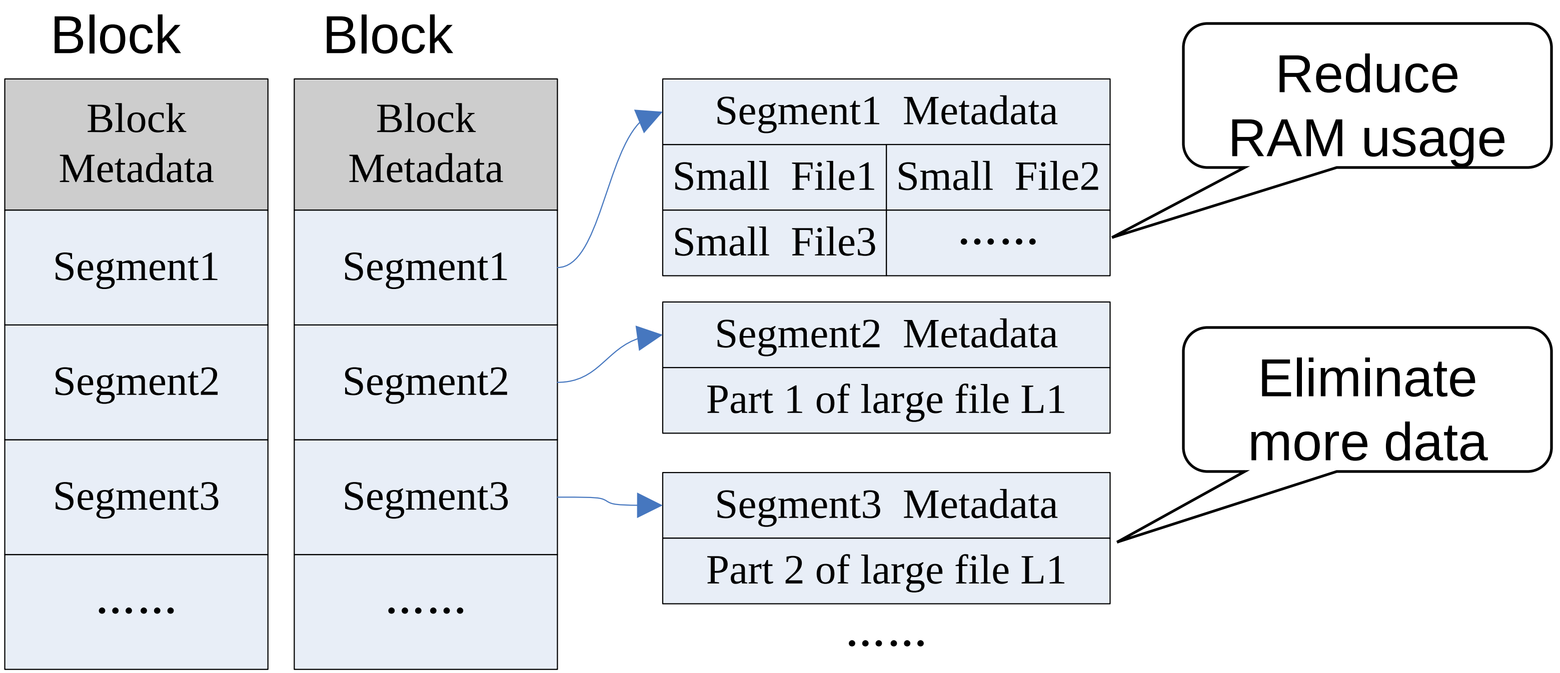
Deduplication Server



- Deduplication Server.
  - It is most likely the performance bottleneck.

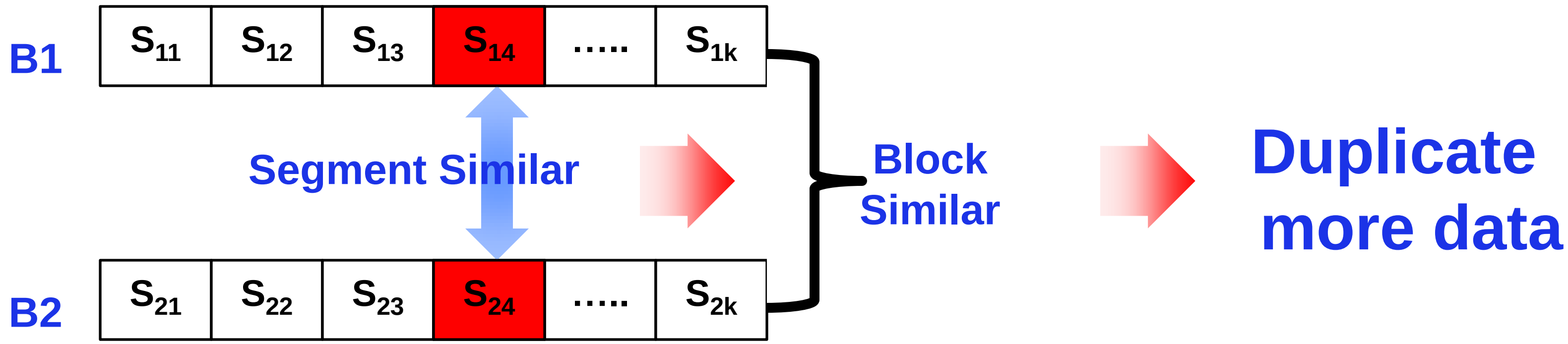
The Similarity Algorithm

- **Structuring data from backup streams into segments according to the following three principles.**
  - P1. Correlated small files in a backup stream are to be grouped into a segment.
  - P2. A large file in a backup stream is divided into several independent segments.
  - P3. All segments are of approximately the same size (e.g., 2MB).



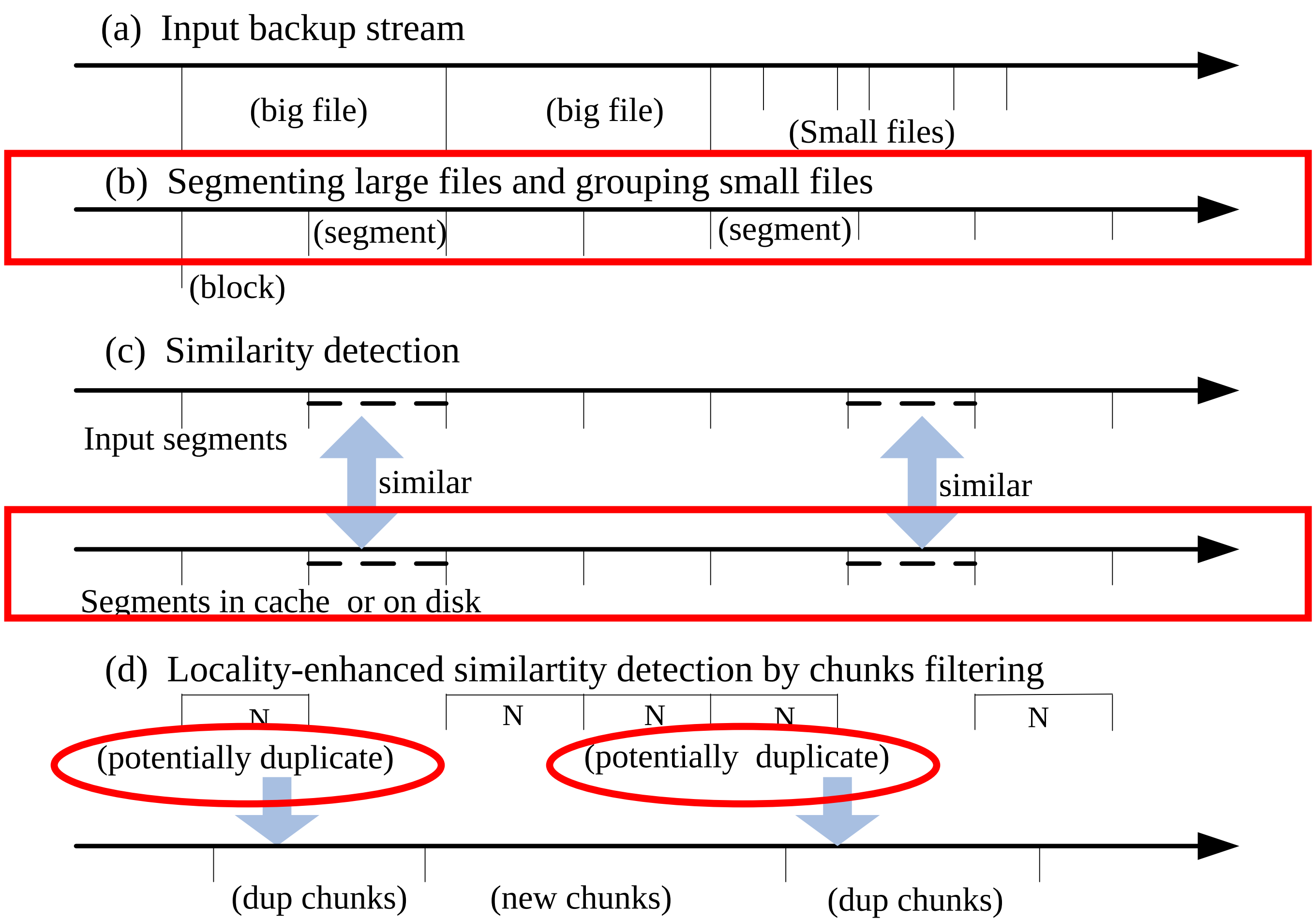
The Locality Algorithm

- The locality algorithm groups several contiguous segments into a block and preserves their locality-layout on the disk.
  - It maximizes the RAM utilization and reduces frequent accesses to on-disk index by retaining access locality in the locality cache.
  - By exploiting the inherent locality in backup streams, the block-based SiLo locality algorithm can eliminate more duplicate data.



SiLo Workflow

- The locality algorithm helps detect more potentially duplicate chunks that are missed by the similarity algorithm.



RAM Consideration

- RAM usage of SiLo:
  - The locality cache?　　　A small portion.
  - The similarity hash table?　　The main portion.

- RAM usage analysis:
  - SiLo requires only 30 MB for deduplicating 1TB unique data.
  - Extreme Binning requires 300 MB for deduplicating 1TB unique data. (Avg. file size of 200KB).
  - Sparse Indexing uses 170 MB of RAM space for a TB-scale deduplication system, whereas the Sparse Indexing paper estimates that DDFS would require 360 MB RAM to maintain a partial index depending on locality in backup streams.

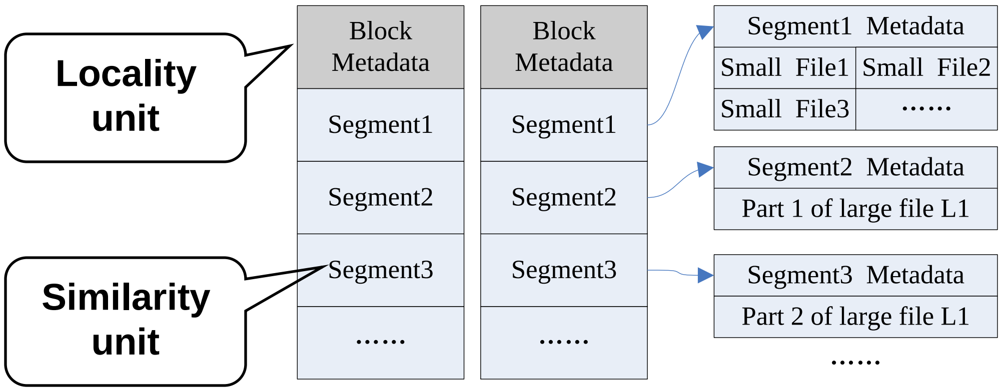
Performance Evaluation

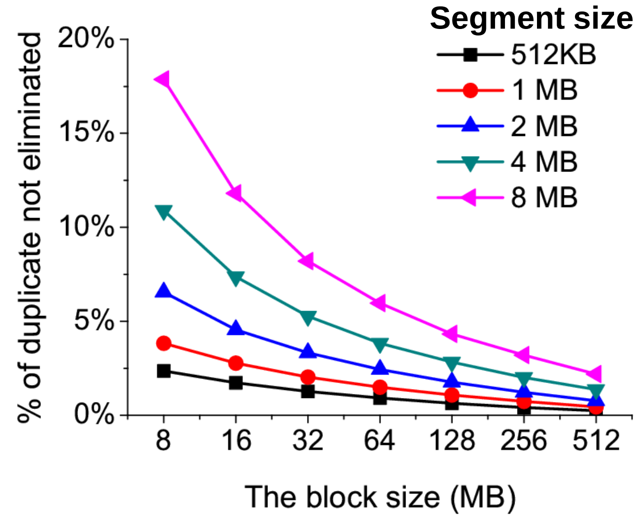
- Interplay of similarity and locality algorithms.
  - Quantitative analysis of our similarity and locality algorithms.
- Comparison of state-of-the-art work.
  - Locality approach: ChunkStash-HDD.
  - Similarity approach: Extreme Binning.
- Four datasets.

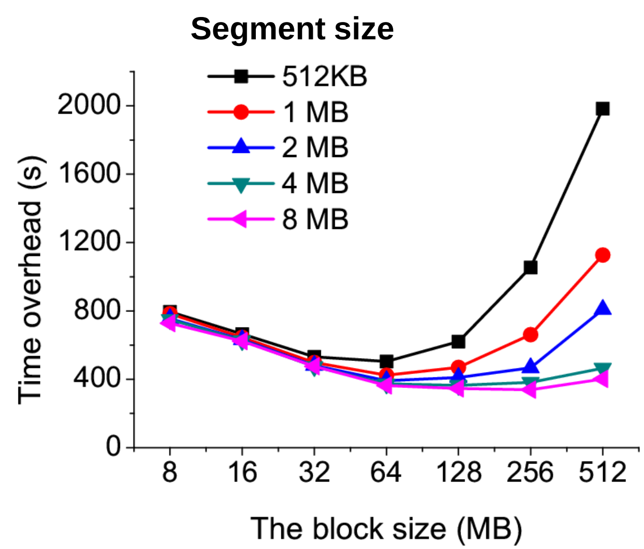
Small files

Feature	*One-set*	*Inc-set*	*Linux-set*	*Full-set*
Locality	*Weak*	*Weak*	*Strong*	*Strong*
Similarity	*Weak*	*Strong*	*Strong*	*Strong*

Interplay of Similarity and Locality

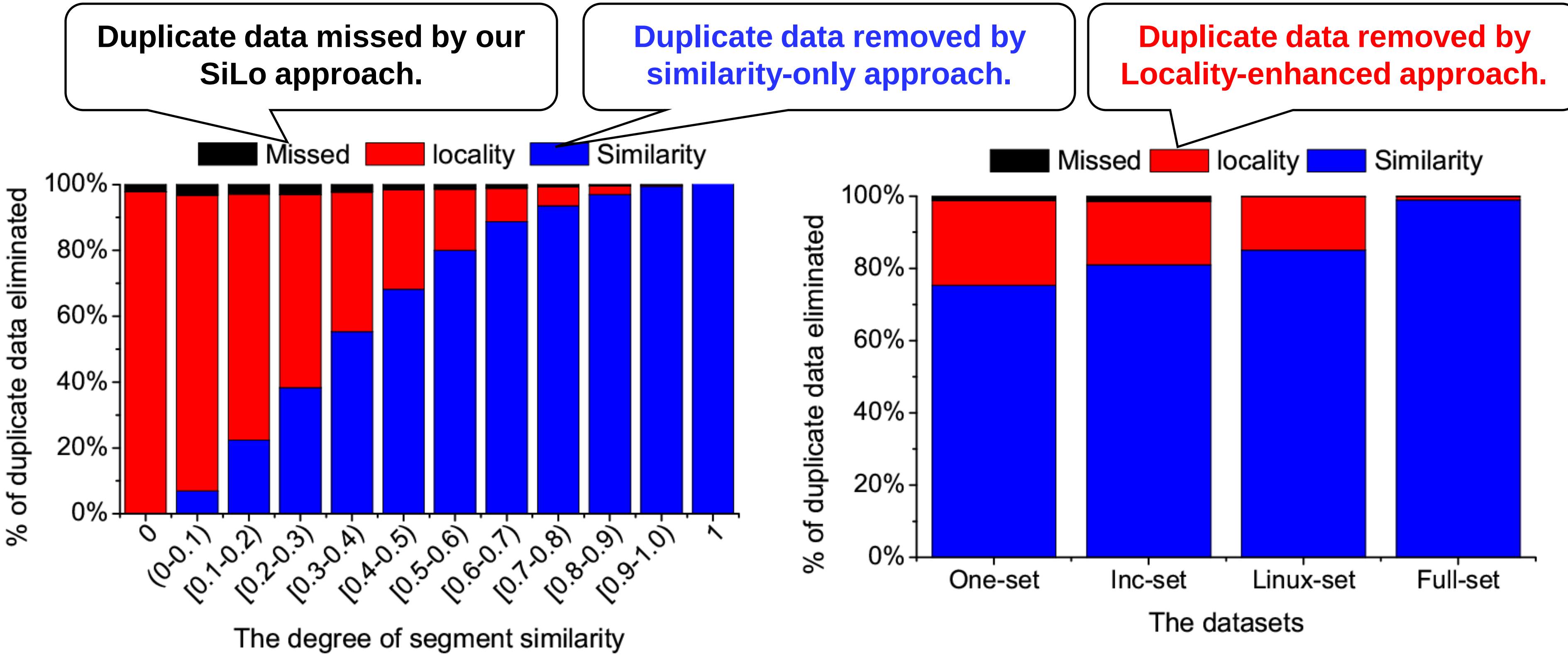






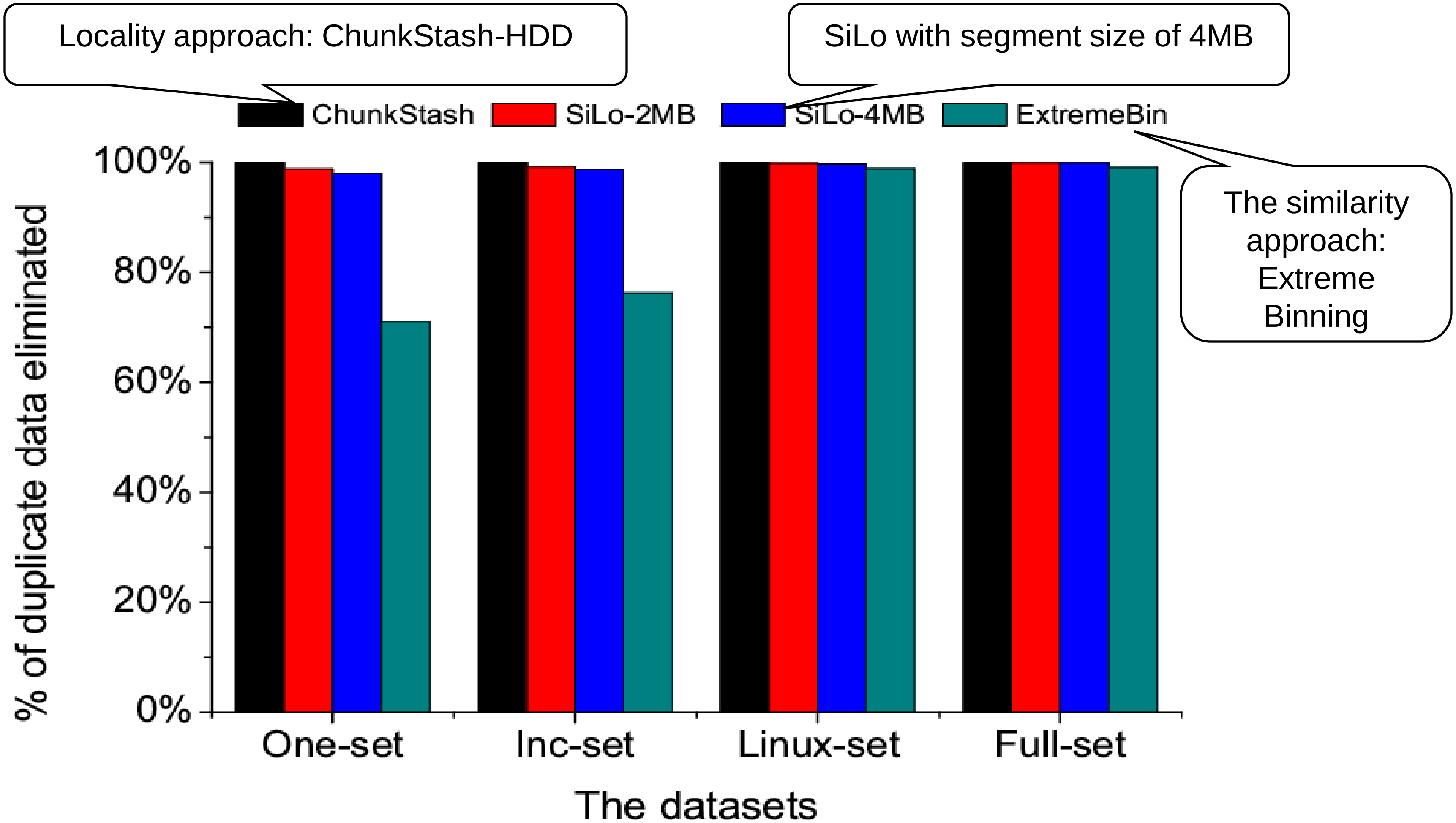
- **Percentage of duplicate data eliminated and Time overhead of SiLo deduplication as a function of block size and segment size.**
  - **The larger the block size is, the more locality can be retained.**
  - **The smaller the segment size is, the more similarity can be exposed.**

Locality Enhancement Evaluation



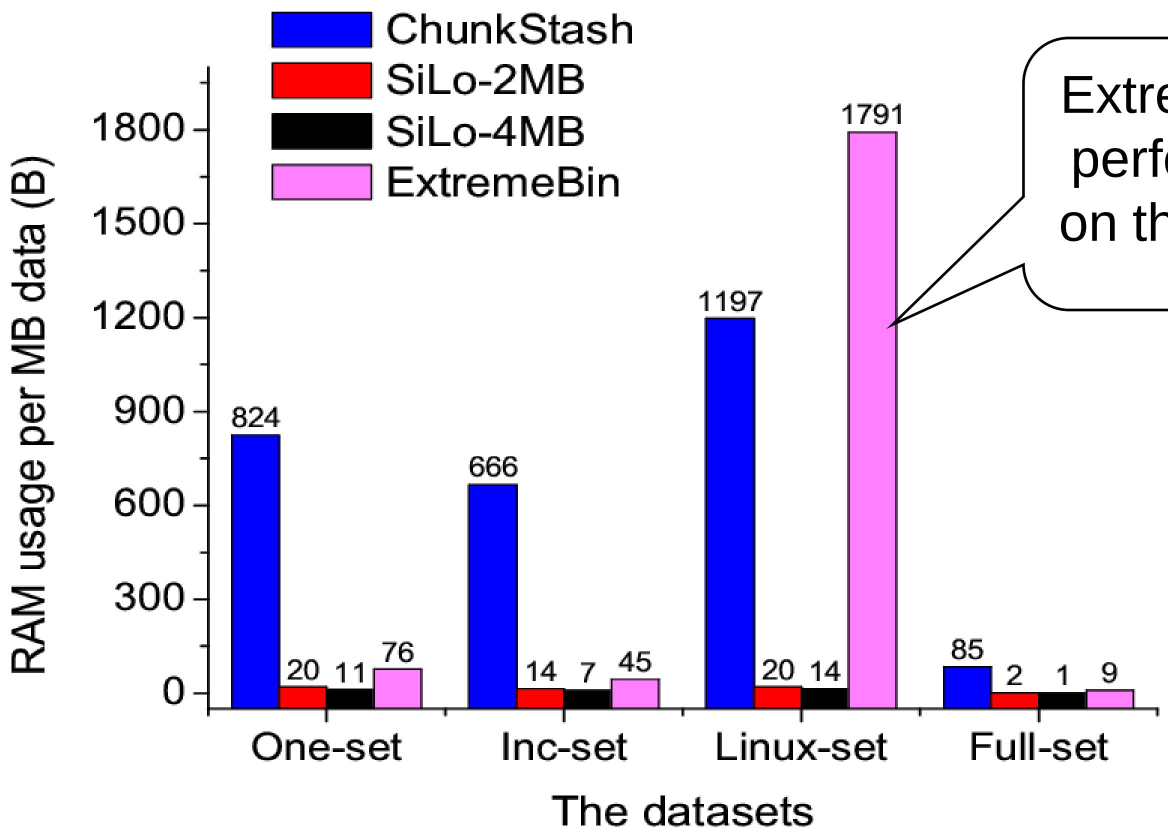
The full Exploitation of locality jointly with similarity can remove almost all of the redundant data missed by the similarity detection.

Duplicate Elimination



SiLo achieves near-exact duplicate elimination under all workloads.

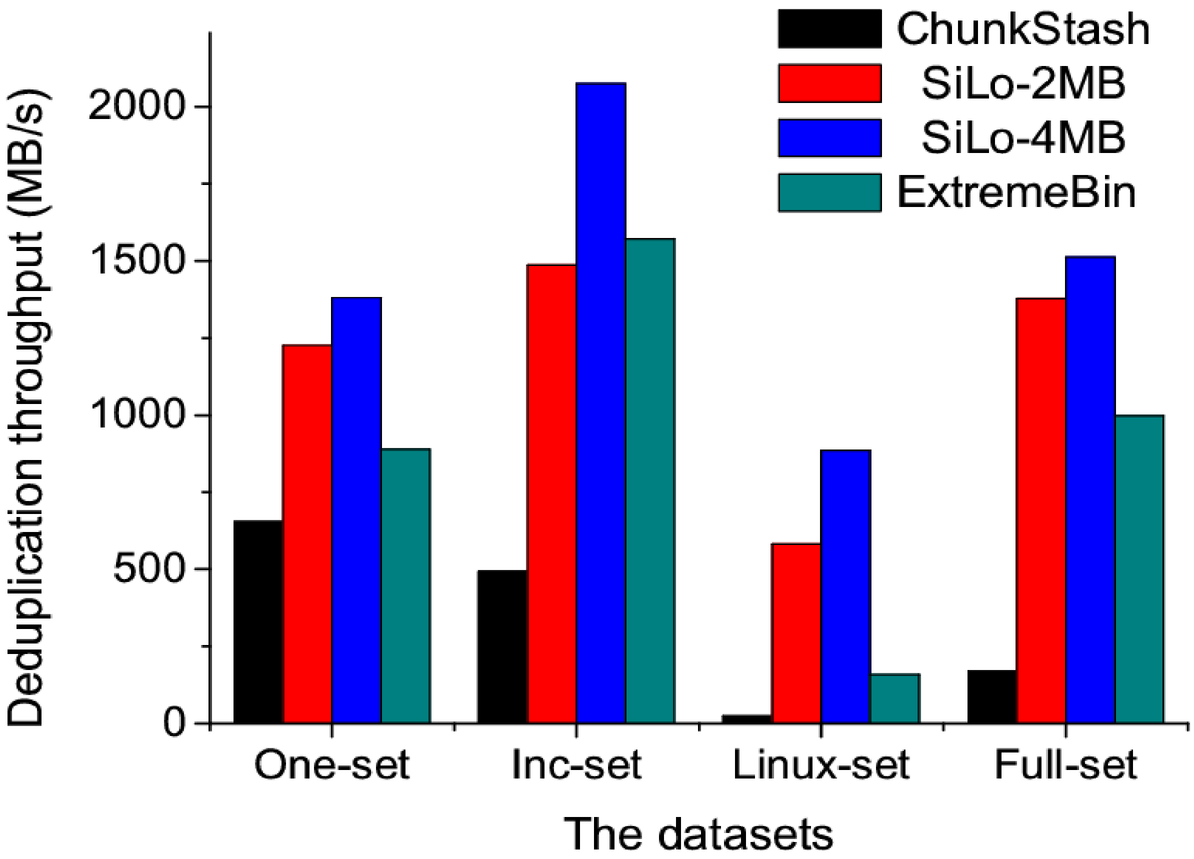
RAM Usage for Indexing



Extreme Binning performs poorly on the Linux-set.

SiLo consumes a RAM capacity that is only 1/41~1/60 and 1/3~1/90 respectively of that consumed by ChunkStash and Extreme Binning.

Deduplication Throughput



Our evaluations on deduplication throughput suggest that SiLo outperforms ChunkStash by a factor of about 3 and Extreme Binning by a factor of about 1.5.

Summary

- **SiLo, a near-exact deduplication system.**
  - effectively and complementarily exploits similarity and locality
  - achieve high duplicate elimination and throughput at extremely low RAM overheads.
- **Combination of similarity and locality.**
  - SiLo proposes a new similarity algorithm that groups many small strongly correlated files into a segment and segments a large file to better expose and exploit their similarity characteristics.
  - SiLo proposes an effective locality approach that captures more similar and duplicate data missed by the probabilistic similarity detection and also improve the deduplication throughput.
- **Our experimental evaluation of SiLo.**
  - Quantitative analysis and demonstration of our similarity and locality algorithms.
  - SiLo system consistently and significantly outperforms two existing state-of-the-art systems.

Thank You !

Questions?

Acknowledgements

973 Program of China under Grant No. 2011CB302301,
(“863” Program) of China under Grant No.2009AA01A401 and 2009AA01A402,
NSFC No.61025008,60933002,60873028,60703046,
Changjiang innovative group of Education of China No. IRT0725,
Fundamental Research Funds for the central universities, HUST, under grant 2010MS043,
The US NSF under Grants NSF-IIS-0916859, NSF-CCF-0937993 and NSF-CNS-1016609.