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

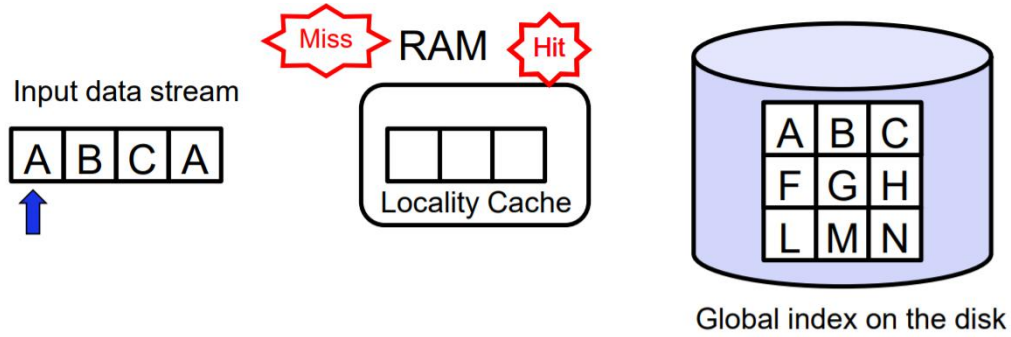
Wen Xia, Hong Jiang, Dan Feng, and Yu Hua

slide made by wgl

Background

- **Full Index Deduplication**
  - Dataset of 800TB & Average chunk size 8KB -> 2TB of fingerprints
  - Access to on-disk index is 1000 times slower than RAM
- **Goals**
  - Make full use of RAM, putting the hot fingerprints into RAM

Background



➢Locality Based Deduplication (ChunkStash)

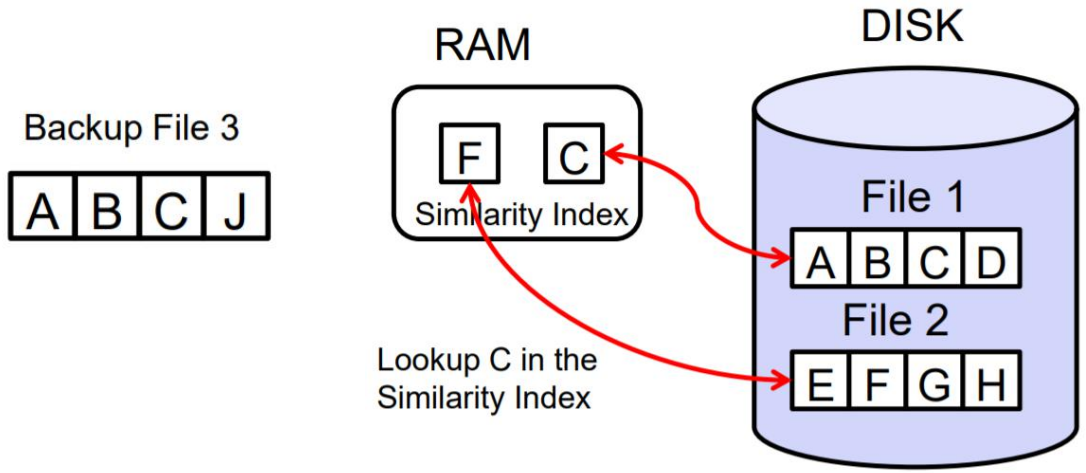
- File appear in same order throughout multiple backups
- Storing the chunks in the order
- Preserving the locality in the RAM

➢Similarity Based Deduplication (ExtremeBin)

- Exploits the similarity among files
- Puts similar files in a database

➢Goals

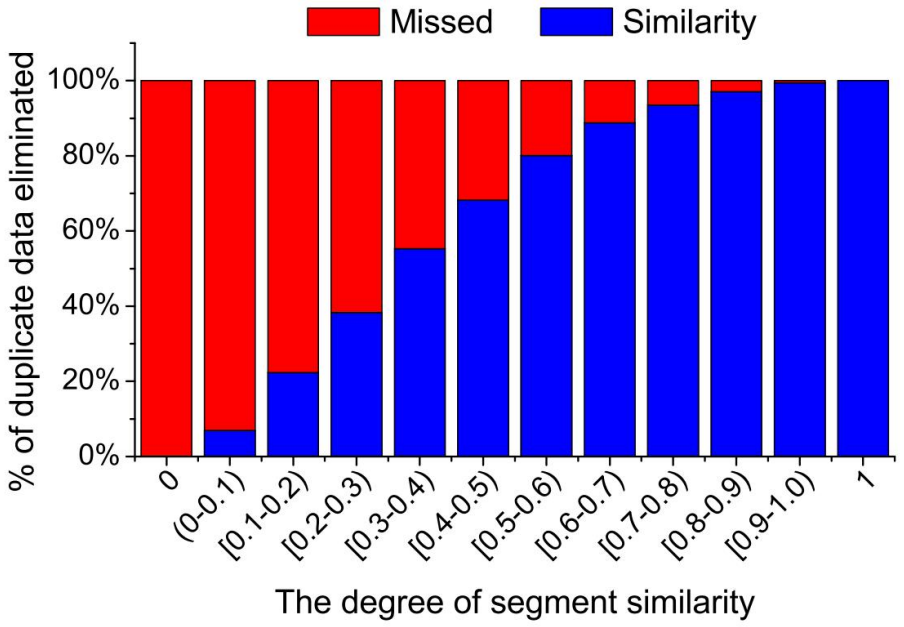
- Maxmize deduplication ratio
- Minimize the RAM usage



Motivation

- **Locality & Similarity Problem**
  - Backup stream may lack locality
  - Similarity among files may be either lacking or weak

- **Idea**
  - Combining Similarity and Locality

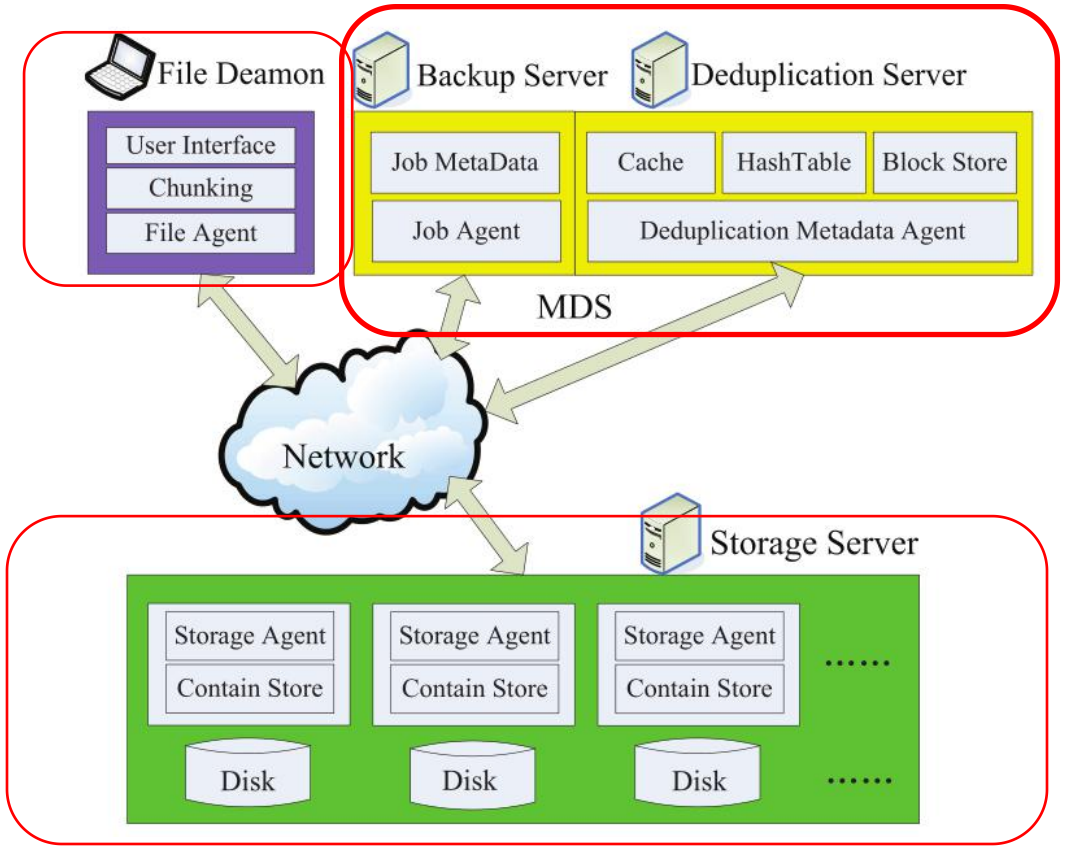


Motivation

- **Analysis of Datasets**
  - Small Files (≤64KB) : ≤20% of the total space, ≥80% of the number of files
  - Large Files (≥2MB) : ≤20% of the number of files, ≥80% of the total space
- **Problem**
  - Small Files -> Large chunk-lookup index
  - Large Files -> Less similarity will appear
- **Idea**
  - Grouping small files
  - Segmenting large files

Architecture

➢ **SiLo**

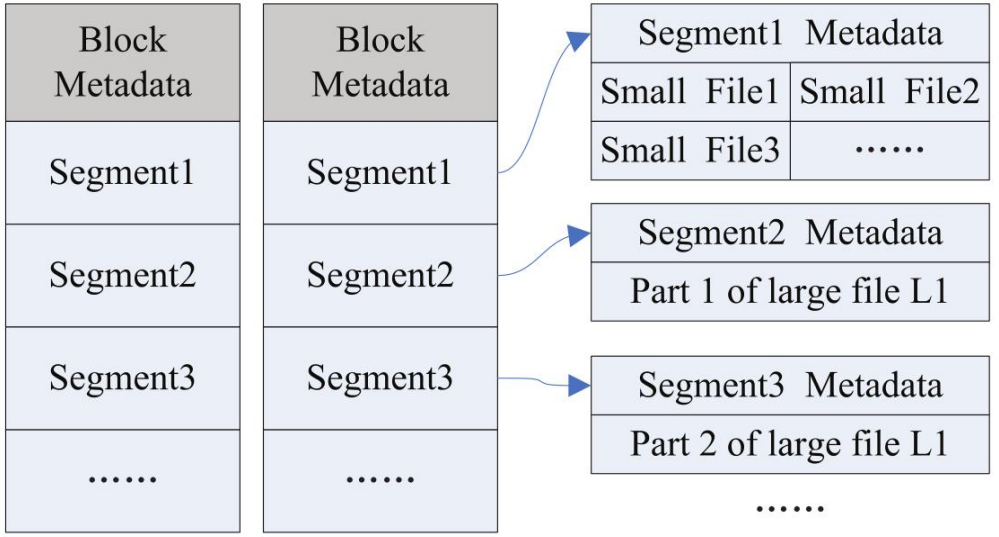


- ➢ File Deamon(FD)
  - User chunking
- ➢ Deduplication Server(DS)
  - Fingerprint & Deduplication
- ➢ Backup Server(BS)
  - Metadata database
- ➢ Storage Server(SS)
  - Repository for backed-up data

Design

➢ Similarity Algorithm

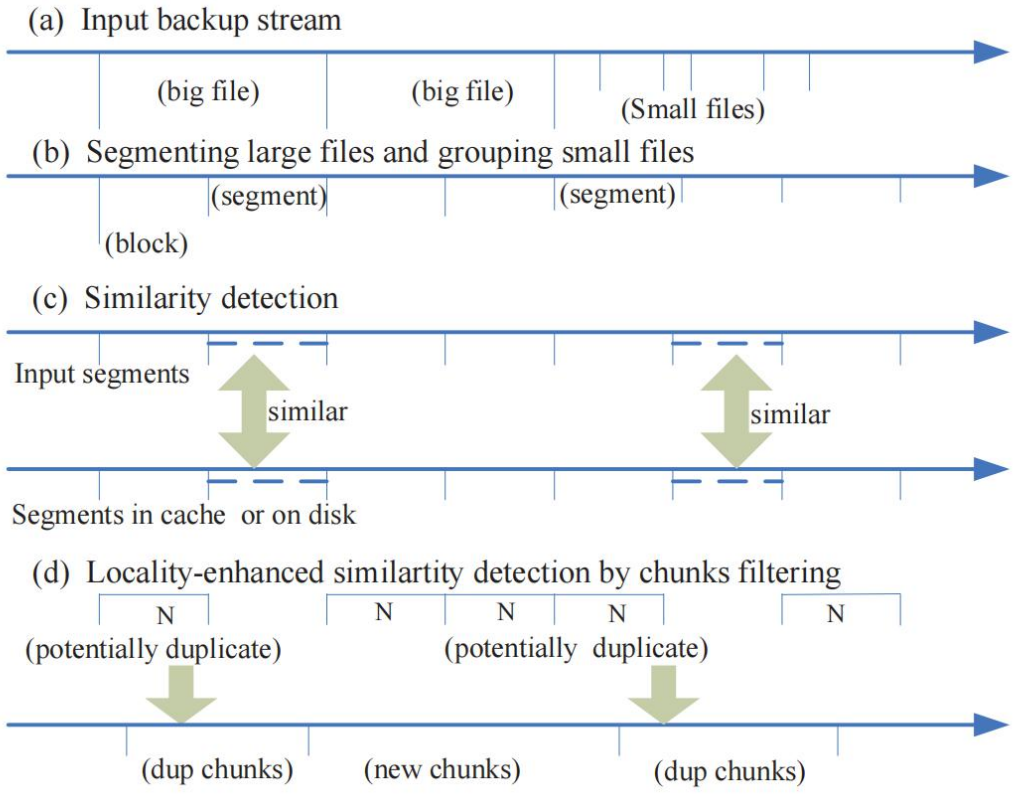
- Segment for Similarity
  - Grouping correlated small files
  - Segmenting large files
- Block for Locality
  - Grouping contiguous segments
  - Prepare for Locality Algorithm



Design

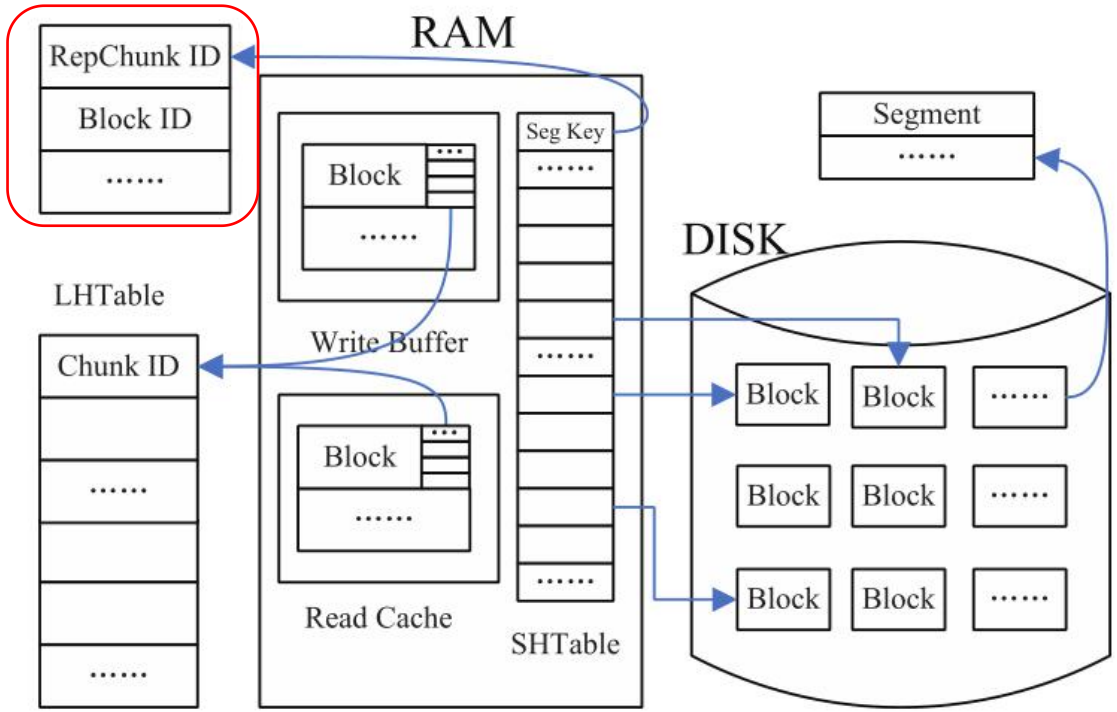
- **Locality Algorithm**
  - (a) Input backup stream
  - (b) Grouping small files & Segmenting large files
  - (c) Similarity detection among segments
  - (d) Segment Similar leads to Block Similar

  - Block -> Contiguous segments -> Locality
  - More duplicate data missed by similarity detection



Design

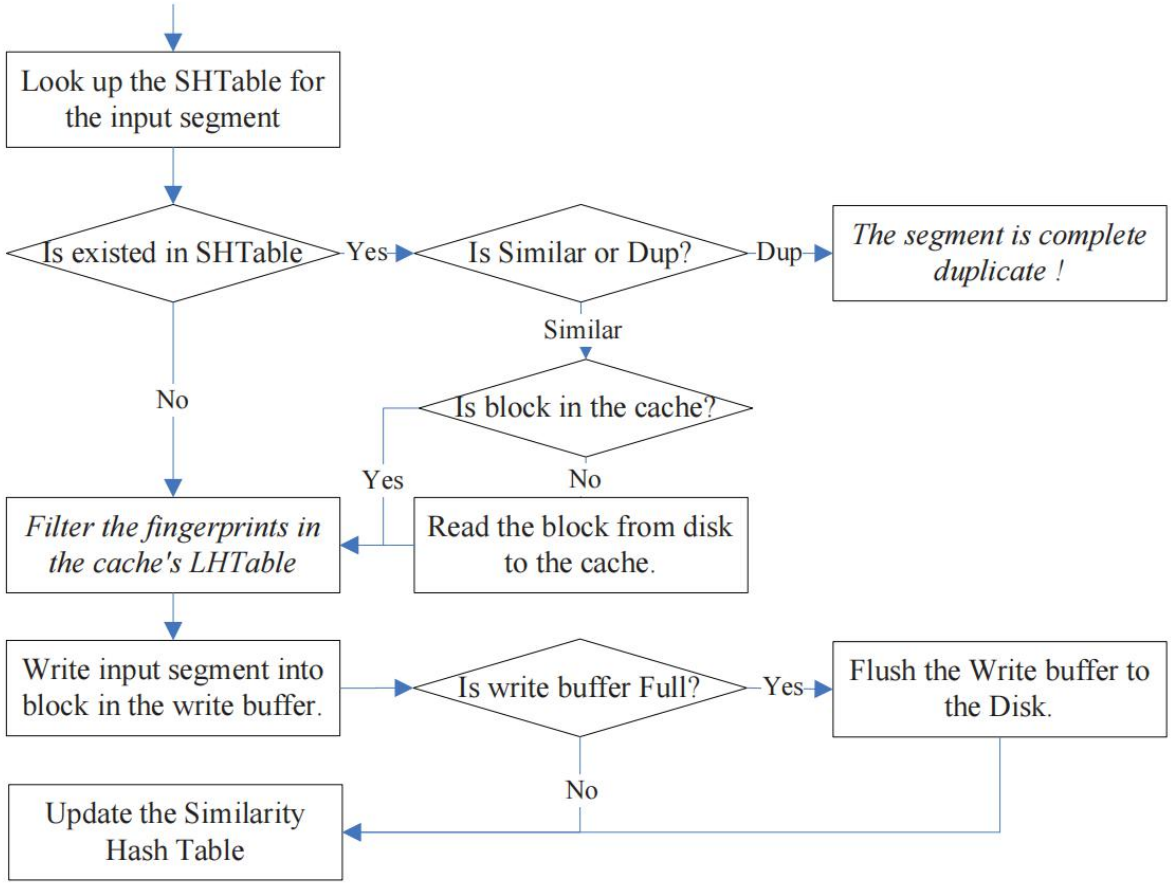
➢ Data structures of DS



- ➢ Segment: exploit similarity
  - Representative FP
  - Block ID
- ➢ Block: locality layout of segments
  - Contiguous segments
  - LHTable: Locality chunk-index
- ➢ SHTable: Similarity Detection
  - All Segment similarity index
- ➢ Write Buffer & Read Cache

Design

➢ Workflow



➢① Chunking & Segmenting

➢② Similarity detections among segments

➢③ Load similar segment's Block

➢④ Fingerprint Filter

➢⑤ Write Block

➢⑥ Update SHTable

Evaluation

Dedupe factor: Totalsize /(Totalsize - Dedupsize)

Feature	One-set	Inc-set	Linux	Full-set
Total size	530GB	251GB	101 GB	2.51TB
Total files	3.5M	0.59M	8.8M	11.3M
Total chunks	51.7M	29.4M	16.9M	417.6M
Avg.chunk size	10KB	8KB	5.9KB	6.5KB
Dedupe factor	1.7	2.7	19	25
Locality	weak	weak	strong	strong
Similarity	weak	strong	strong	strong

➢Hardware

- A quad-core CPU running at 2.4GHz
- 4GB RAM, 2 gigabit network interface cards
- 2 * 500GB 7200rpm hard disks

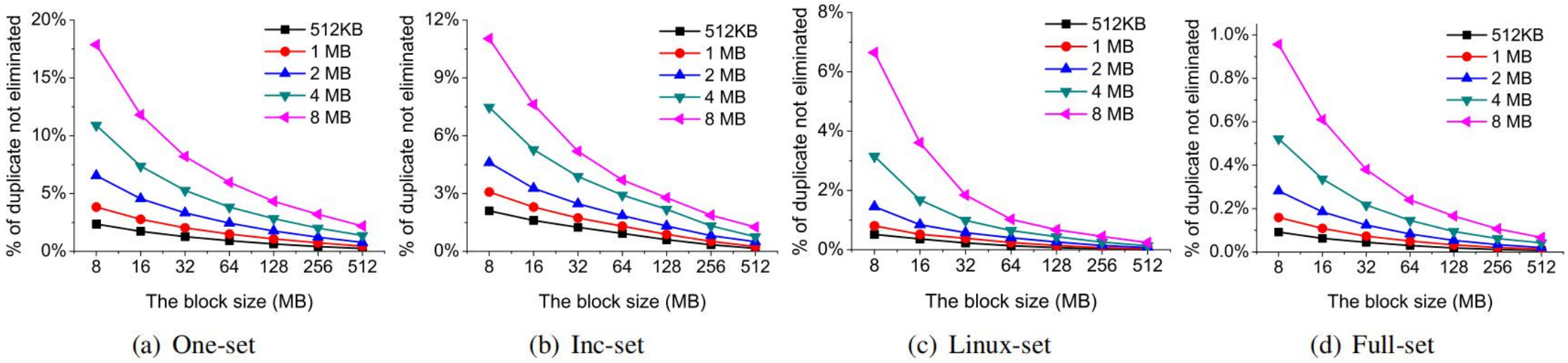
➢DataSets

- One-set: Only one full backup -> weak locality and weak similarity.
- Inc-set: Initial full backups and 391 incremental backups -> strong similarity but weak locality.
- Linux-set: 900 versions from version 1.1.13 to 2.6.33 -> small files
- Full-set: 380 full backups -> strong locality and strong similarity

Evaluation

➢Block size and Segment size

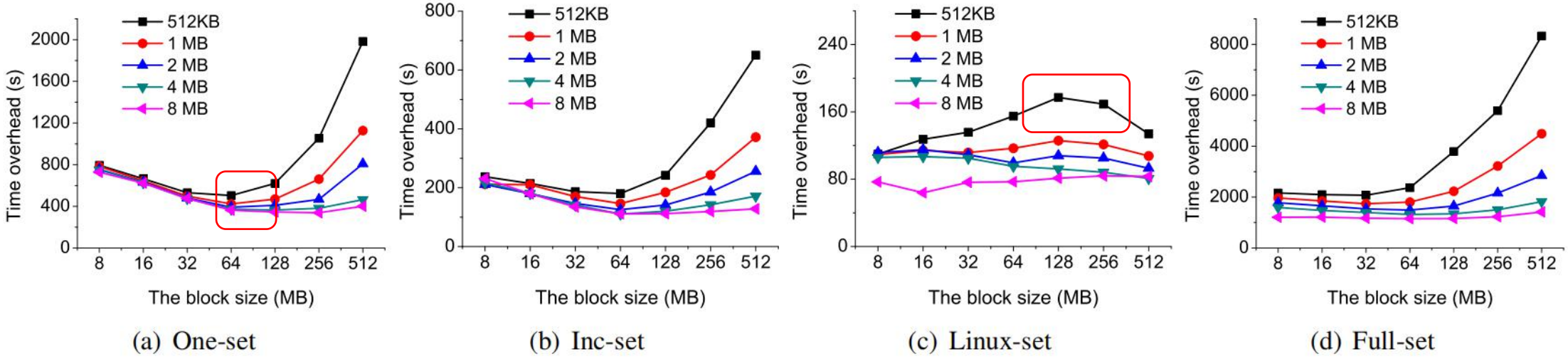
percentage of duplicate data not eliminated



➢Duplicate elimination performance increases with block size but decreases with segment size

Evaluation

- **Time Overhead**
- Tradeoff between deduplication and time overhead

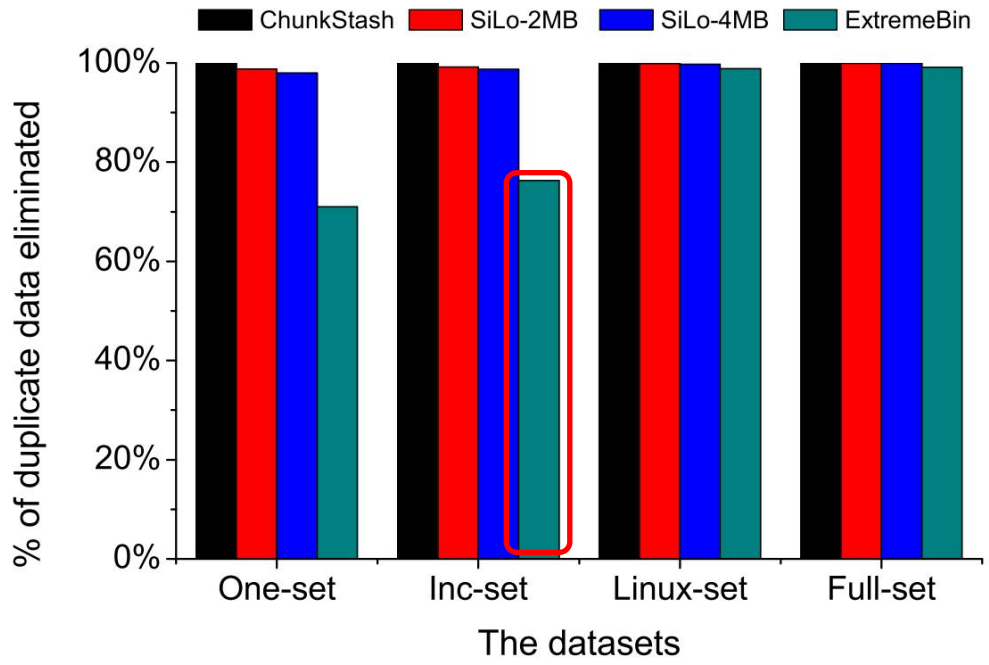


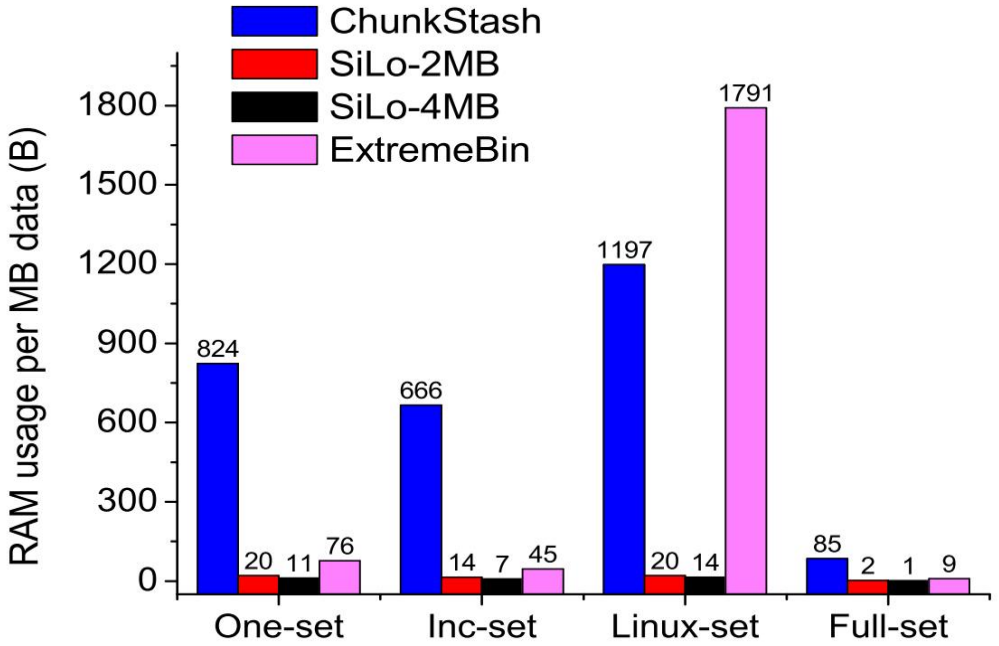
- Smaller segment size results in more frequent similarity detections -> disk access
- Larger block size may result in more unrelated segments being read in

Evaluation

Block size: 256MB,
SiLo-2MB: Segment size 2MB
SiLo-4MB: Segment size 4MB

➢Comparative Evaluation of SiLo





➢Duplicate elimination (Locality / Similarity)

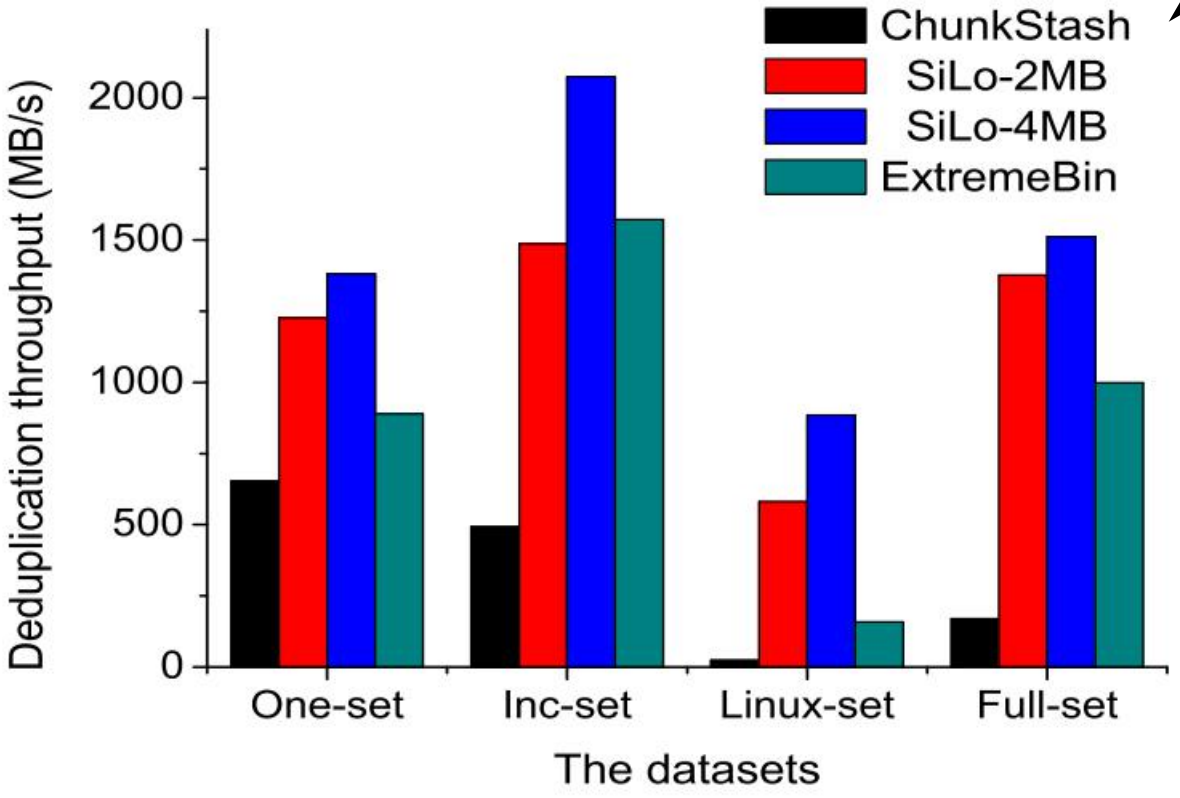
- ChunkStash -> exact deduplication 100%
- ExtremeBin -> Based on dataset similarity

➢RAM usage

- ChunkStash -> High RAM usage
- ExtremeBin -> Small Files lead to large index

Evaluation

➢Deduplication throughput



➢Throughput

- ChunkStash -> frequent accesses to on-disk index
- ExtremeBin -> one disk per access file
- SiLo-2MB -> one disk access per segment
- SiLo-4MB -> large segment size leads to less similarity detection index

Conclusion

- **SiLo**
  - Grouping small correlated files and segmenting large files
  - Segment for Similarity & Block for Locality
  - Combination of similarity and locality
- **Evaluation**
  - Tradeoff of Block size and Segment size
  - Duplicate elimination performance & Time Overhead
  - Comparison : Duplicate elimination & RAM usage & throughput

Comments

➢Strengthens

- The idea of combing similarity and locality is interesting
- Similarity and locality are effectively preserved through the design of segment and block

➢Weaknesses

- The calculation algorithm of segment's fingerprint is very important, but it is not involved in the paper
- Read cache’s size is not mentioned in the paper
- Dataset problem

Append

- **Analysis of dataset**