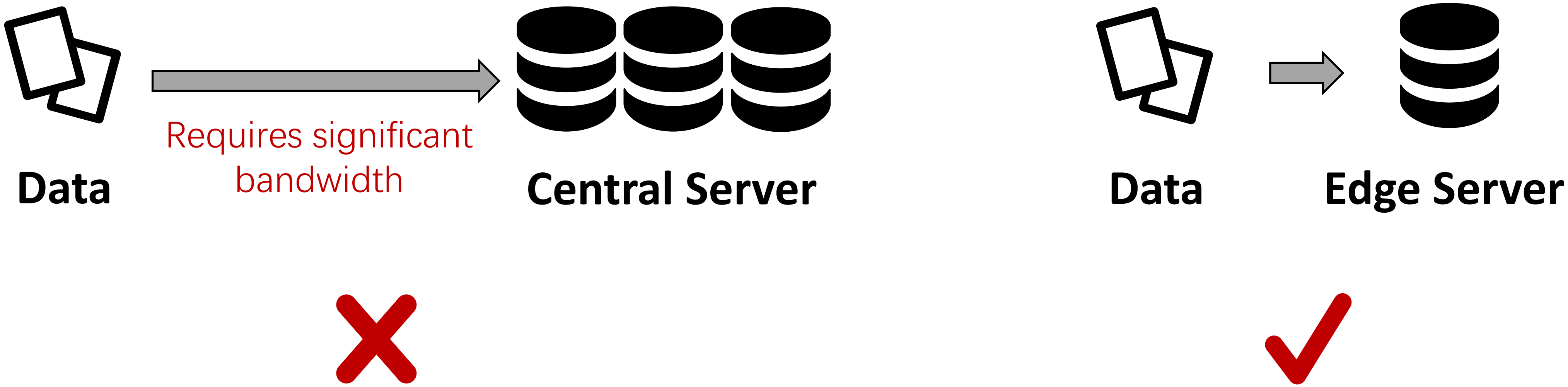
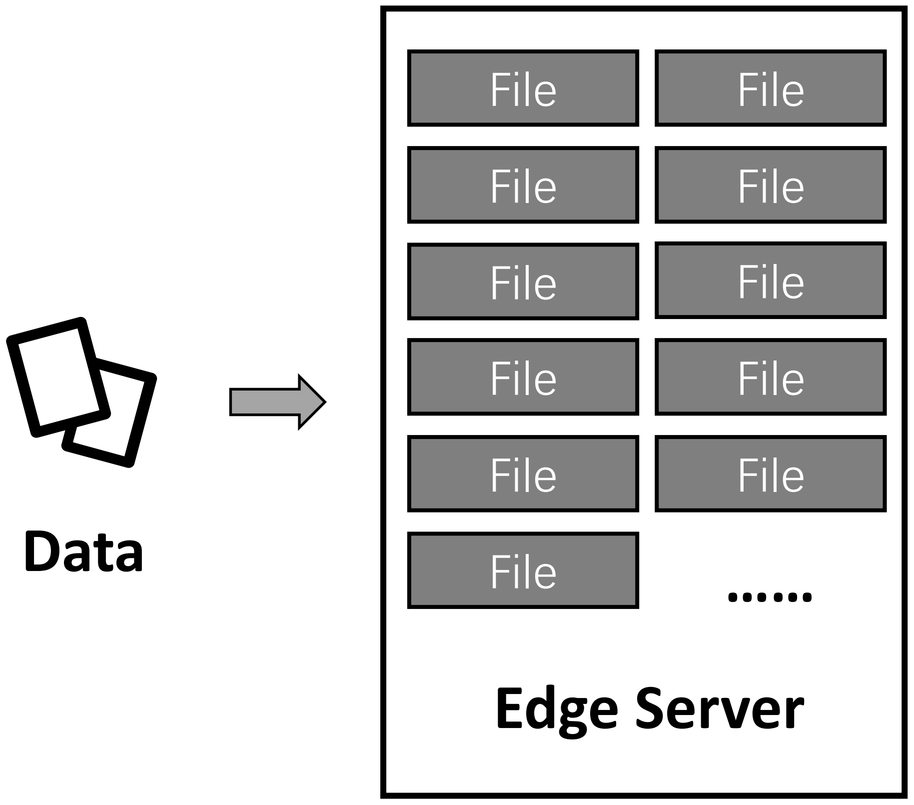
LOFS: A Lightweight Online File Storage Strategy for Effective Data Deduplication at Network Edge

IEEE T-PDS 21

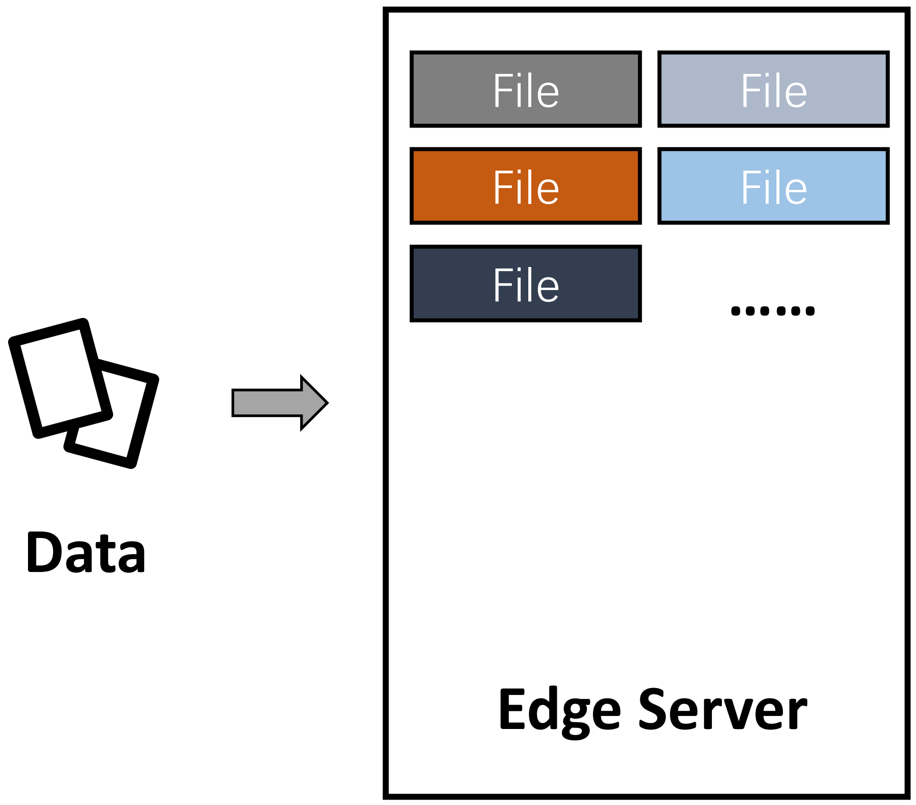
Background


Data
Requires significant bandwidth
Central Server
Data
Edge Server

Background

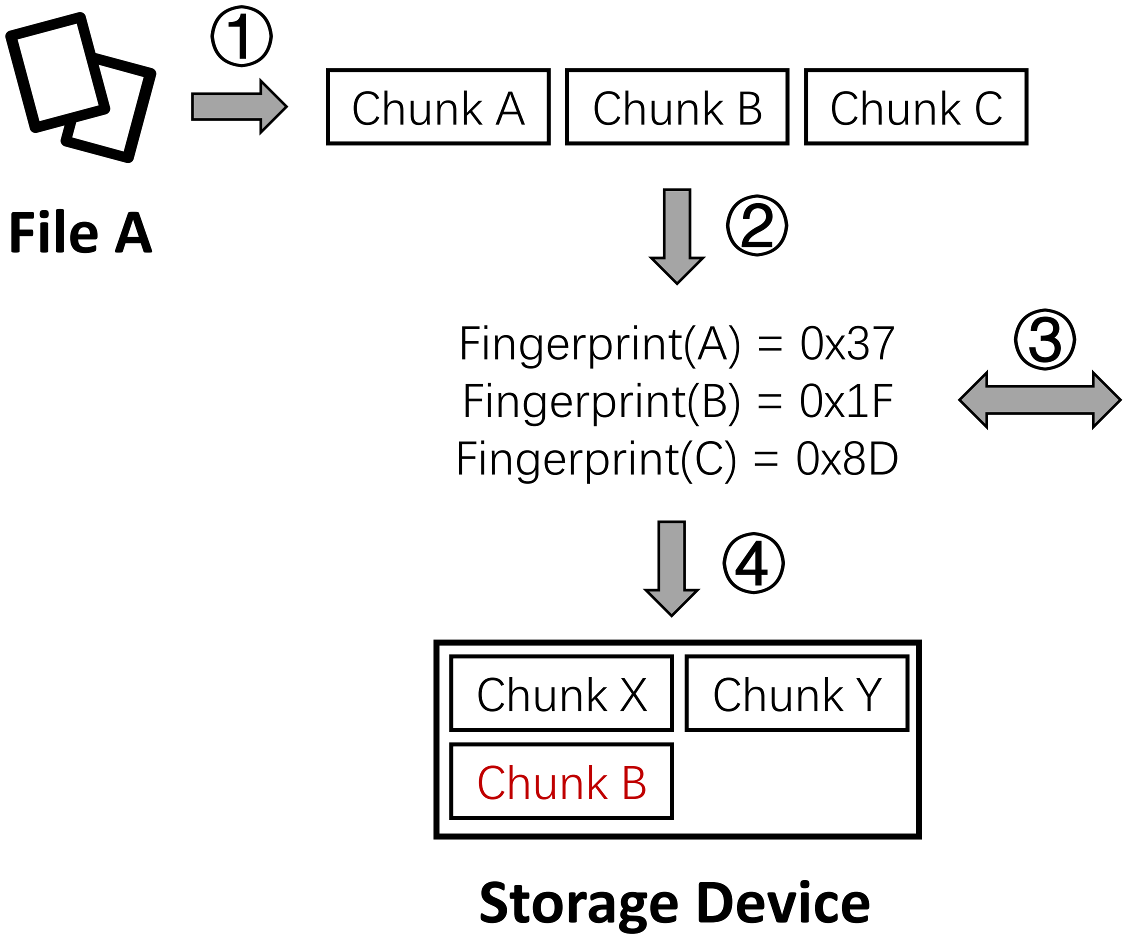


Massive duplicated data



Limited low redundancy data

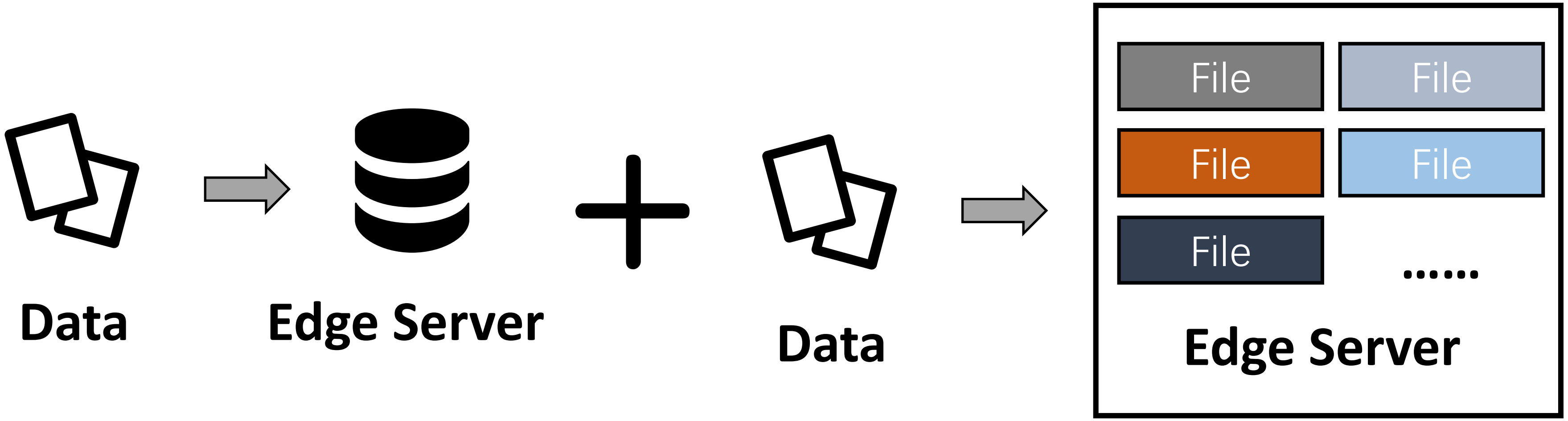
Background



Fingerprint	ChunkID
0x37	Chunk X
0x8D	Chunk Y
0x1F	Chunk B

Fingerprint Index

Background

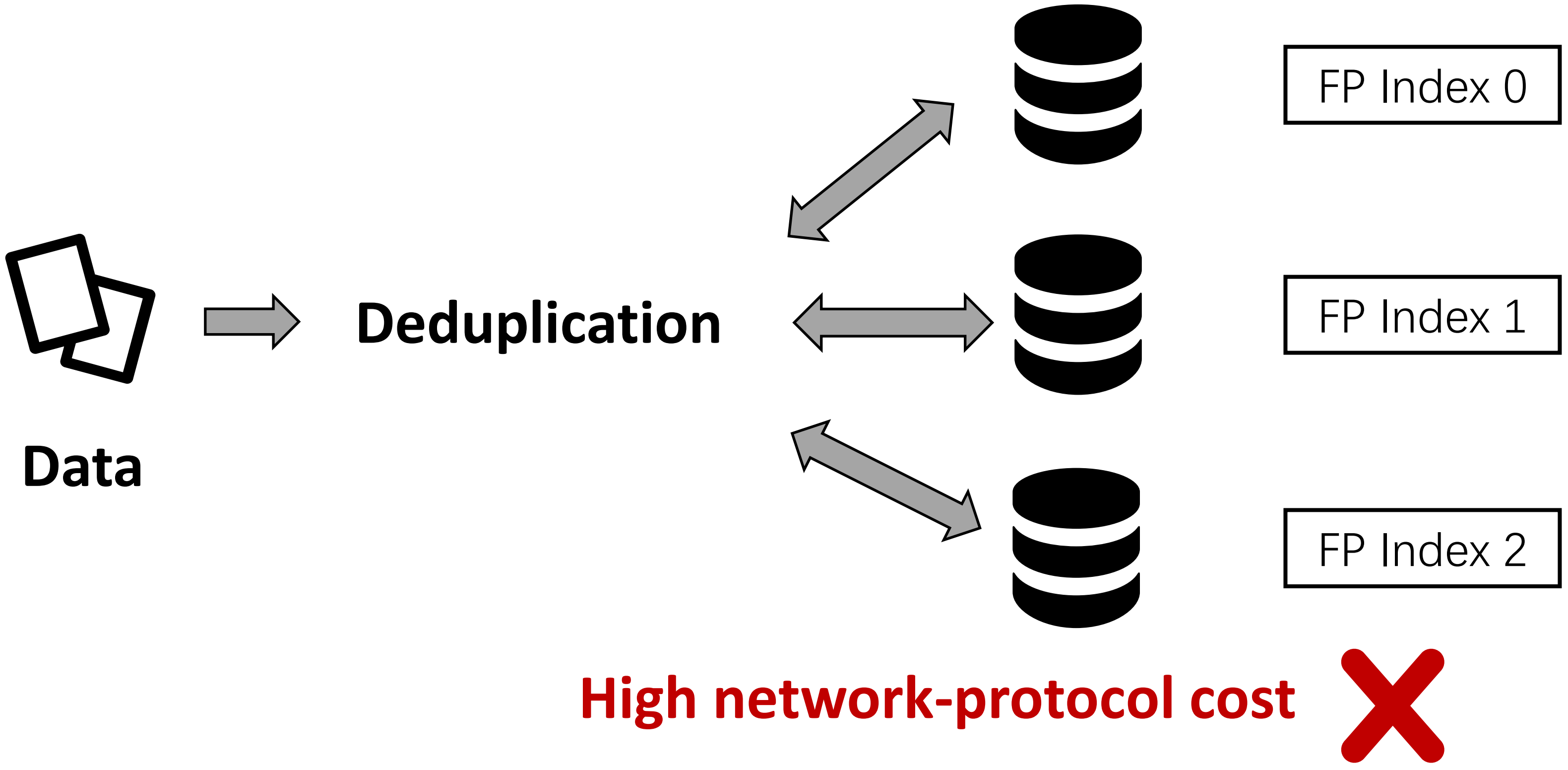


Easy and efficient

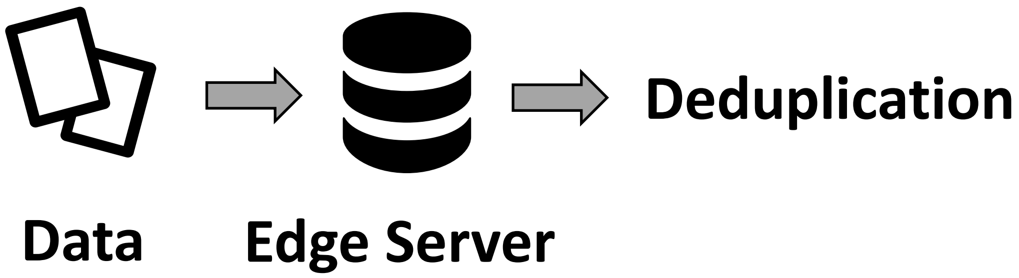
Easy and efficient

Have some problems

Background


Data
Deduplication
FP Index 0
FP Index 1
FP Index 2
High network-protocol cost

Background

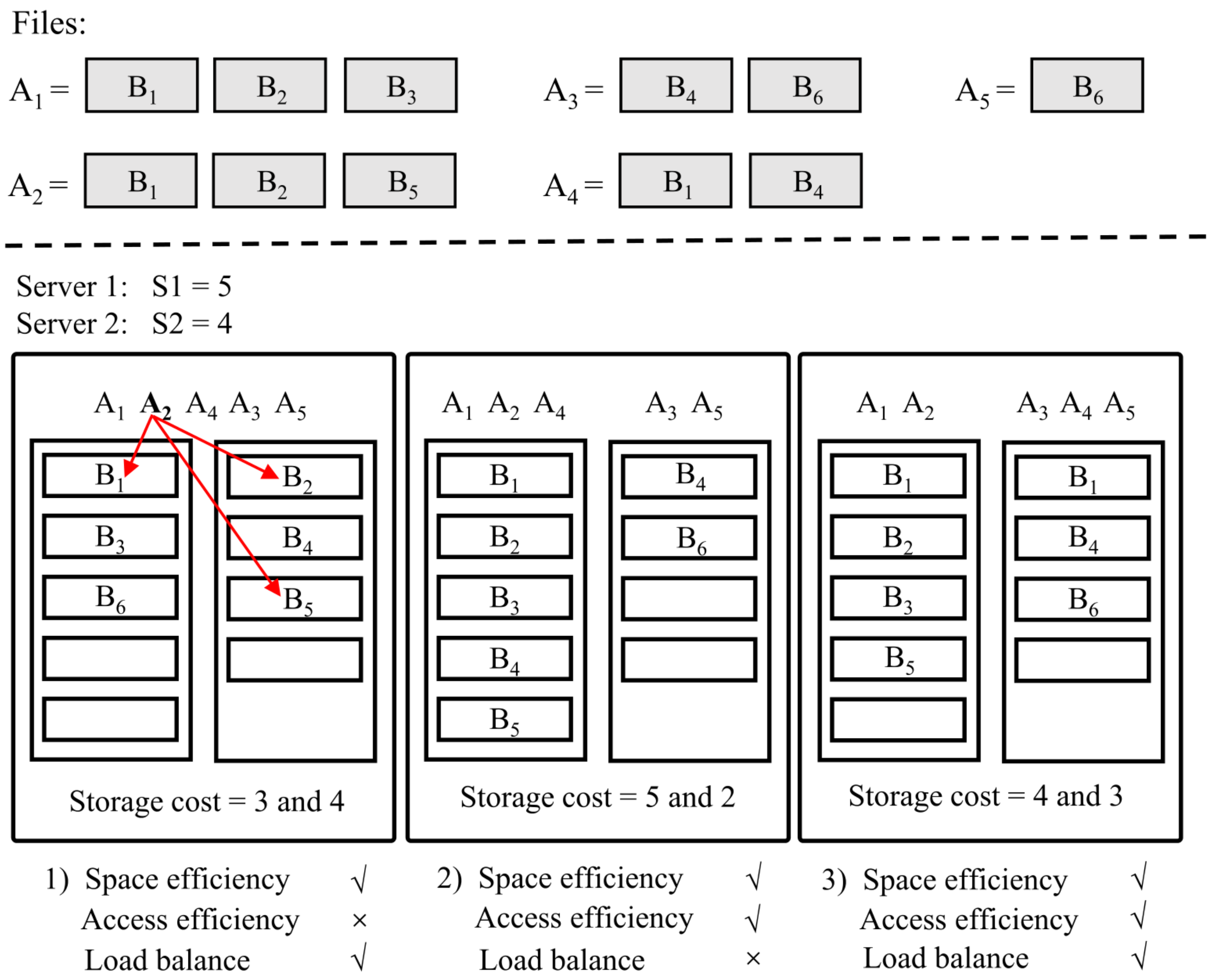


Post-process deduplication

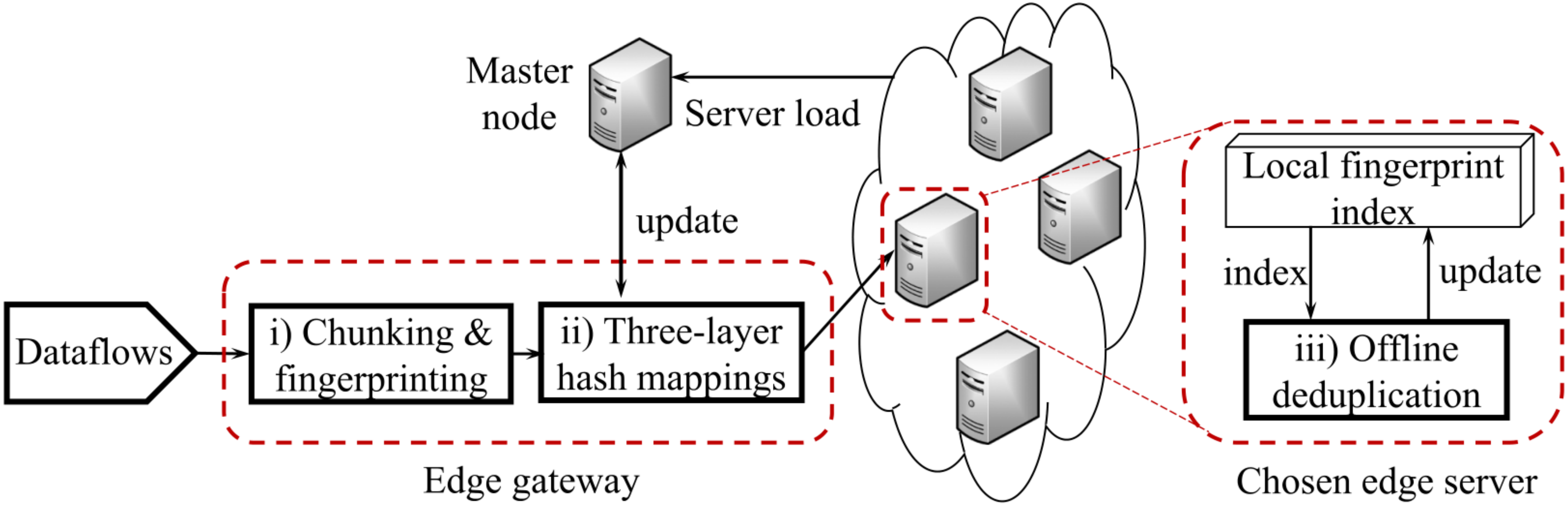
Background

3 Requirements:

- **Space efficiency**
- **Access efficiency**
- **Load balance**



The system architecture of the LOFS strategy

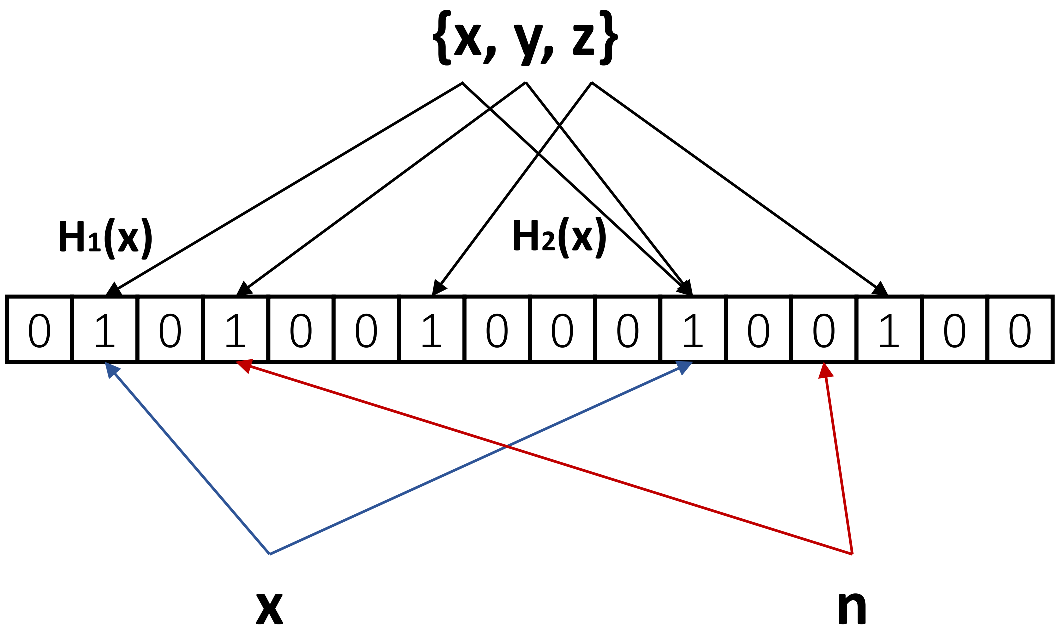


The system architecture of the LOFS strategy

Data → **LOFS** → **Which server should be stored in**

Three-Layer architecture
Bloom Filter-Based File Sketch
LSH-Based Similarity Mining
Capacity-Aware LSH Tablespace Division

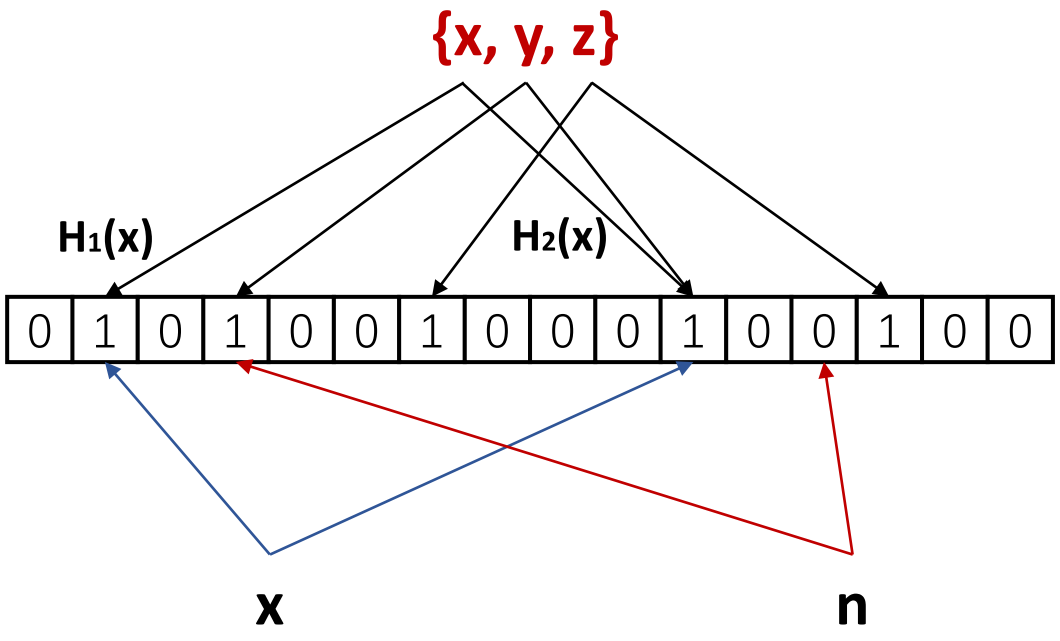
First Layer: Bloom Filter-Based File Sketch



- Similar chunks are likely to have multiple identical hash values
- The similarity between two chunks can be represented in the Bloom filter

*Hamming distances：
“001100011010”
“001101000110”
Ham Dis = 4

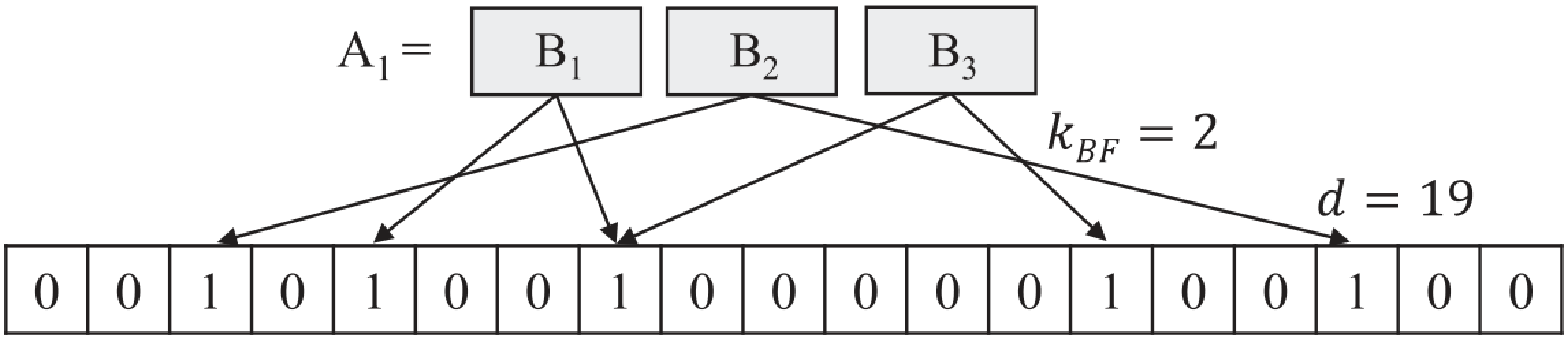
First Layer: Bloom Filter-Based File Sketch



2 challenges：

- Unable to construct the sample space before all the files in A have arrived.
- Even if the sample space can be constructed, the sequential comparison of the partitioned chunks with the sample space will still cause nontrivial computation overhead.

First Layer: Bloom Filter-Based File Sketch



Bloom Filter in LOFS：

- **Give up sample space**
- **Sketch each file by mapping the chunks into a fixed-length bit vector.**

File ⇨ **d-bit file sketch**

Second Layer: LSH-Based Similarity Mining

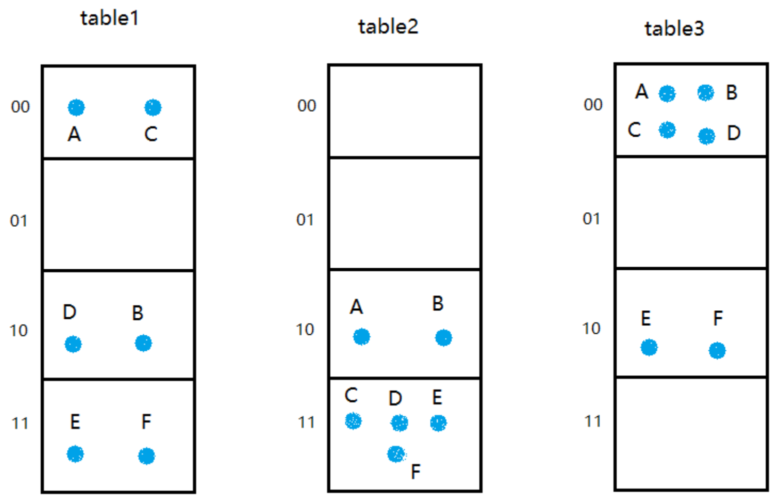
The basis of location-sensitive (LSH) hashing:

- **After transforming two adjacent data points in the original data space by the same mapping or projection, the probability that these two data points are still adjacent in the new data space is high, while the probability that non-adjacent data points are mapped to the same bucket is small.**

Second Layer: LSH-Based Similarity Mining

A = 10001000
B = 11001000
C = 10001100
D = 11001100
E = 11111100
F = 11111110

H1 = 2nd digit
H2 = 4th digit
H3 = 1st digit
H4 = 6th digit
H5 = 3rd digit
H6 = 8st digit



X = 11111111
Check:
Bucket11 in table1
Bucket11 in table2
Bucket11 in table3
Point: C D E F

Second Layer: LSH-Based Similarity Mining

LSH has 2 important parameters:

- **Num of hash func;**
- **Num of hash table.**

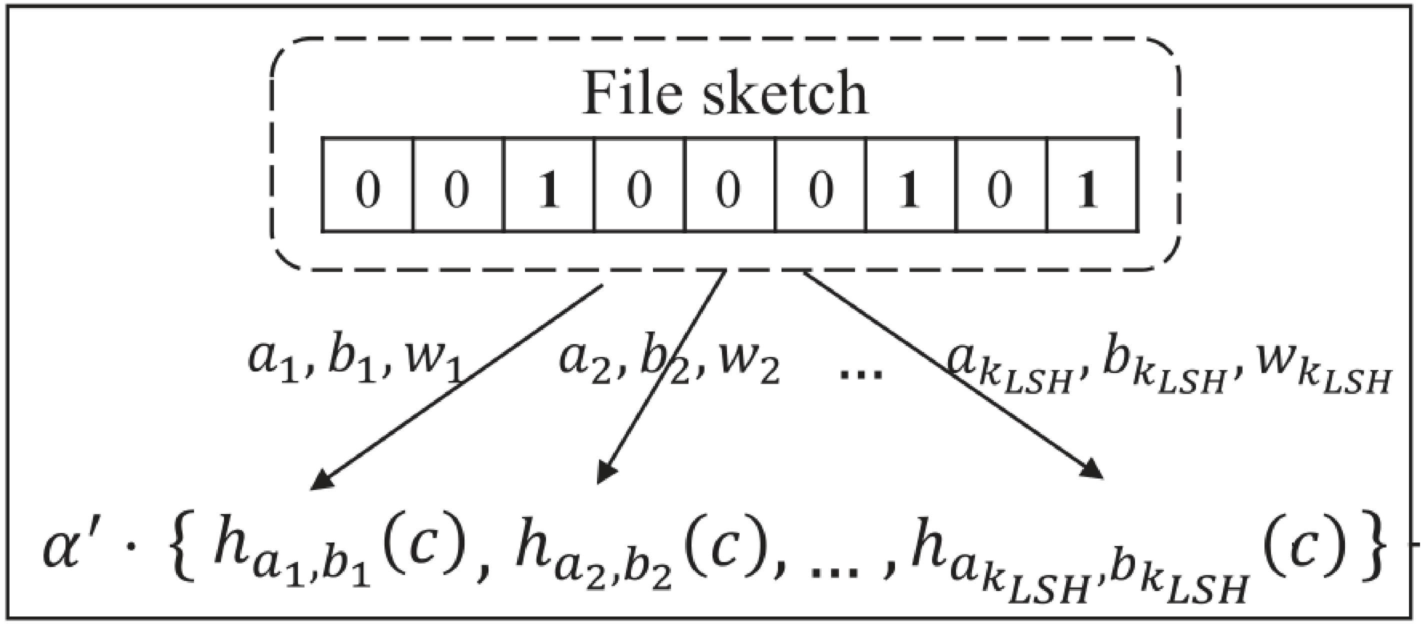
This Leads to a challenge:

- **Similarity between sketches cannot be unified in different hash tables.**

LSH in LOFS:

- **Give up sample space**
- **Just uses 1 hash table.**

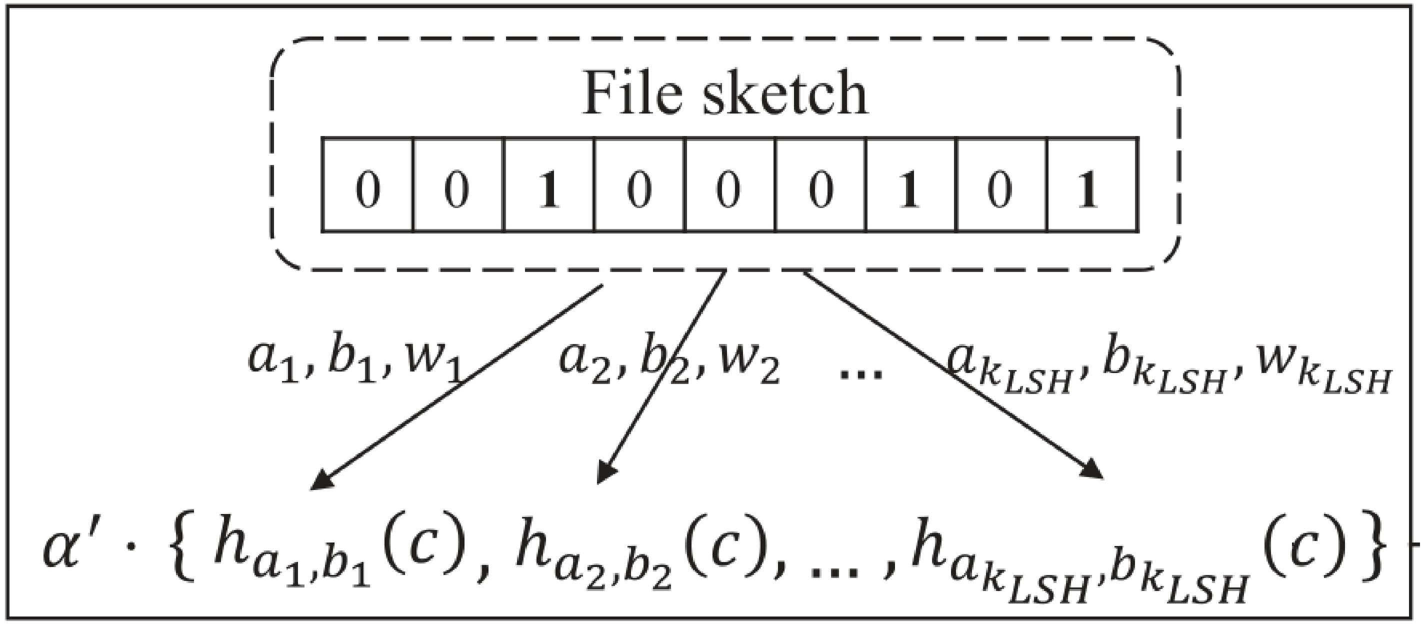
Second Layer: LSH-Based Similarity Mining



$$h_{a,b}(c) = \left\lfloor \frac{\alpha \cdot c + b}{w} \right\rfloor$$

α : D-dimensional random vector following the Cauchy distribution;
b : A real number chosen uniformly from the range [0,w);
w : A large constant.

Second Layer: LSH-Based Similarity Mining



$$h_{a,b}(c) = \left\lfloor \frac{\alpha \cdot c + b}{w} \right\rfloor$$

α' : K_{LSH}-dimensional random vector following the standard Cauchy distribution.

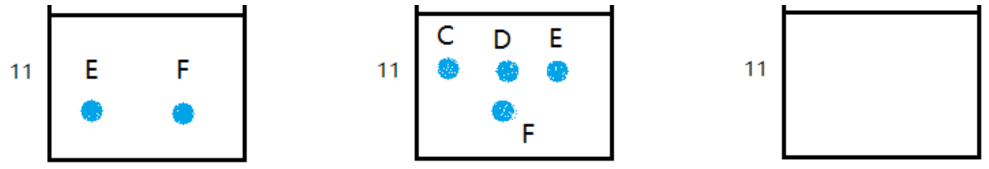
d-bit file sketch ⇨ a projection point

Third Layer: Capacity-Aware LSH Tablespace Division

Which files to put on one servers?

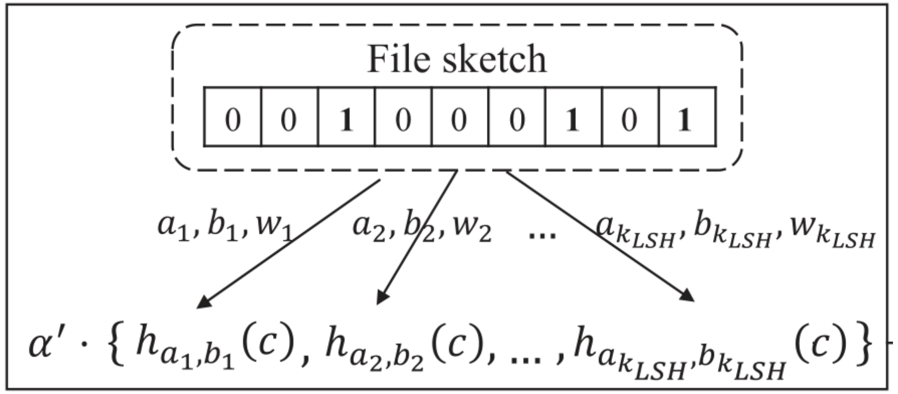
- **Files with similar LSH values**

How many files to put on one server?

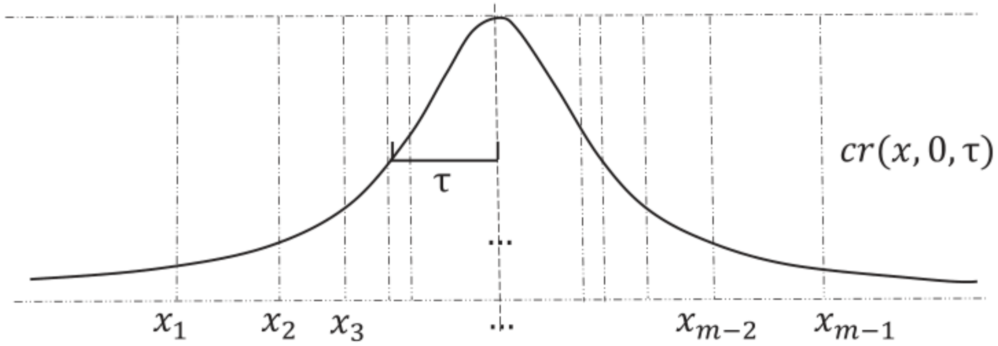


Load balance

Third Layer: Capacity-Aware LSH Tablespace Division



$$h_{a,b}(c) = \left\lfloor \frac{\alpha \cdot c + b}{w} \right\rfloor$$



Review:

α : D-dimensional random vector following the Cauchy distribution;

α‘ : K_{LSH}-dimensional random vector following the standard Cauchy distribution.

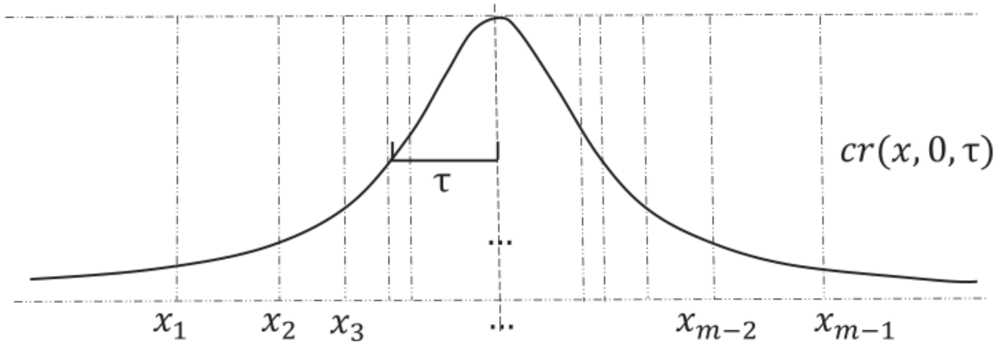
Also following Cauchy distribution!!

Third Layer: Capacity-Aware LSH Tablespace Division

Now we can know the Probability Distribution Function and Cumulative Distribution Function:

$$cr(x, 0, \frac{\sum_{i=1}^{n} ||c_i||_1}{n}) = \frac{n}{\pi \sum_{i=1}^{n} ||c_i||_1} \frac{1}{1 + (\frac{nx}{\sum_{i=1}^{n} ||c_i||_1})^2}$$

$$CR\left(x, 0, \frac{\sum_{i=1}^{n} ||c_i||_1}{n}\right) = \frac{1}{\pi} arctan\left(\frac{nx}{\sum_{i=1}^{n} ||c_i||_1}\right) + \frac{1}{2}$$

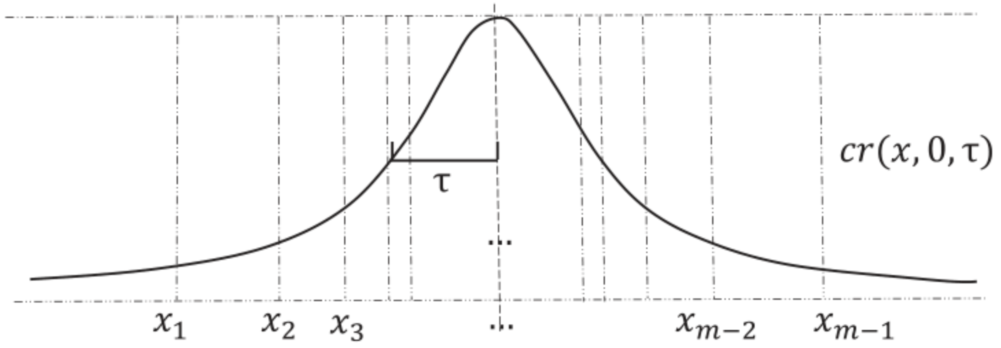


Third Layer: Capacity-Aware LSH Tablespace Division

Physical space size =
number of files * average file size
How to get average file size?

- Use mean value of a set of |x|

More "1" would appear in its file sketch, leading to a large absolute value of the projected points.

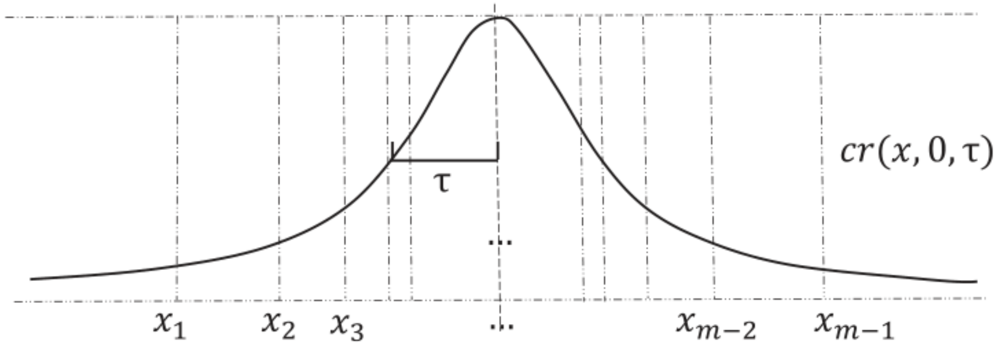


Third Layer: Capacity-Aware LSH Tablespace Division

Let $|\overline{X_j}| = Avg(|X_{j-1}|, |X_{j-1}+1|, \ldots, |X_j|)$
Then storage capability =
$|\overline{X_j}| * (CR(X_j) - CR(X_{j-1}))$

$$\frac{[CR(x_j, 0, \tau) - CR(x_{j-1}, 0, \tau)]\overline{|x_j|}}{|x|} = \frac{C_j}{\sum C_j}.$$

$$x_j = \tau \times tan\left(\pi(CR(x_{j-1}, 0, \tau) + \frac{C_j}{\sum C_j}\frac{\overline{|x|}}{\overline{|x_j|}} - \frac{1}{2})\right)$$



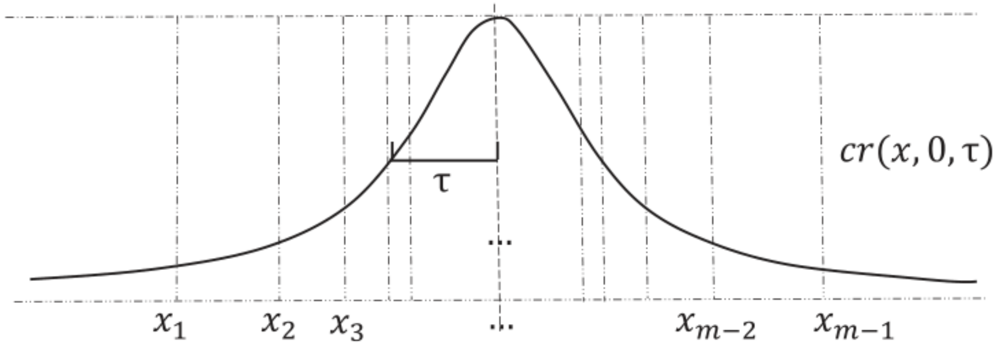
C_j: Storage capacity of server j

Third Layer: Capacity-Aware LSH Tablespace Division

$$cr(x, 0, \frac{\sum_{i=1}^{n} ||c_i||_1}{n}) = \frac{n}{\pi \sum_{i=1}^{n} ||c_i||_1} \frac{1}{1 + (\frac{nx}{\sum_{i=1}^{n} ||c_i||_1})^2}$$

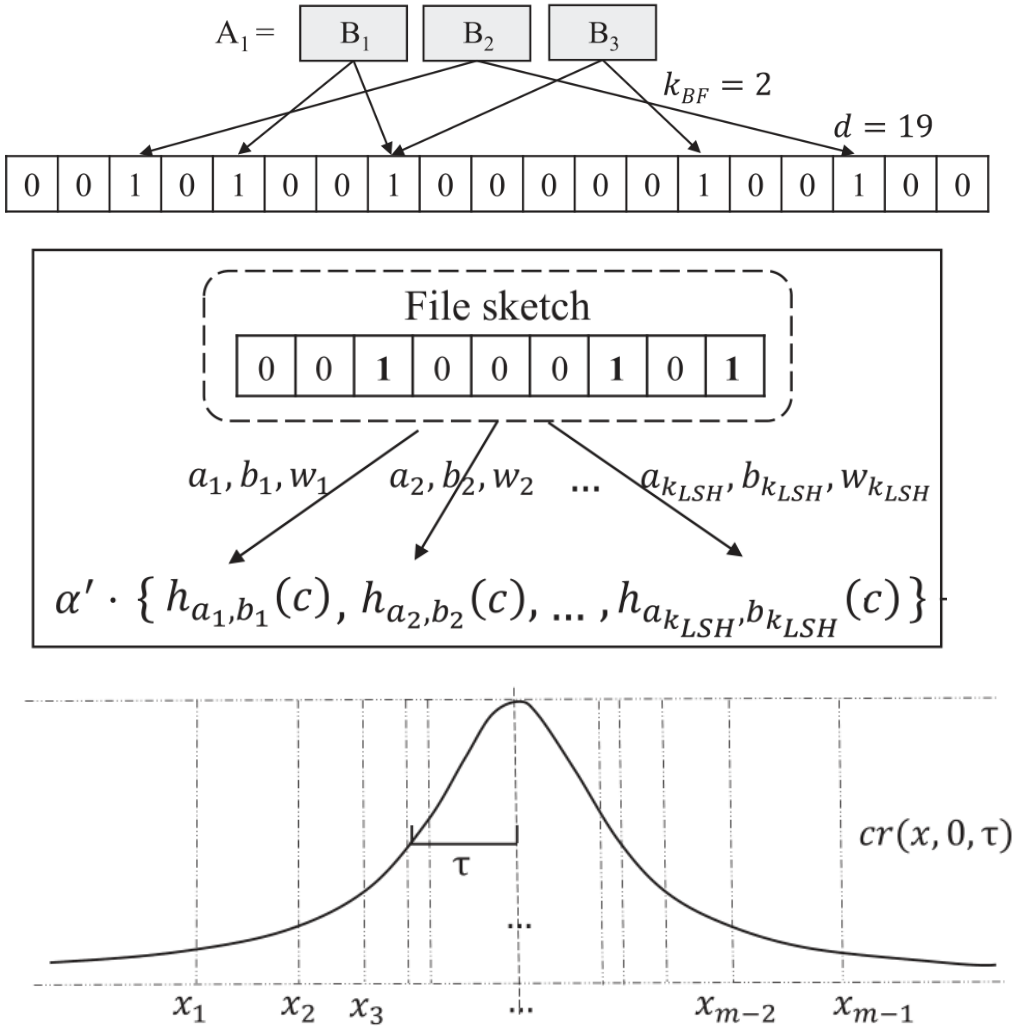
2 trade-off：

- **Files with the same projection point must be assigned to the same edge server;**
- **The parameters of the Cauchy distribution do not change in real time.**



Analysis

Time complexity:
1st layer: O(K_{BF} * β)
2nd layer: O(K_{LSH} * d)
3rd layer: O(1)
Total: O(K_{BF} * β + K_{LSH} * d) → O(β)



Analysis

The Probability of File-Level Collisions Between Sketches:

- **For two different files, the probability that they have the same sketch is negligible.**

It is easy to understand when the sketch is large and has many hash functions.

This proves that the Bloom filter-based design preserves the features of the file very well.

Analysis

The Probability of File-Level Collisions Between Sketches:

- **For any three file sketches q, c1, c2, if $||q - c1||_1 = d1$, $||q - c2||_1 = d2$, and $d1 \leq d2$, then:**

$$p(|h(q) - h(c_1)| \leq \delta) \geq p(|h(q) - h(c_2)| \leq \delta)$$

This proves that the projection points retain similarity after sketches being performed LSH-based mapping, because the positions of the projection points of similar sketches are still adjacent to each other.

***Hamming distances：**
"001100011010"
"001101000110"
Ham Dis = 4

$$h_{a,b}(c) = \left\lfloor \frac{\alpha \cdot c + b}{w} \right\rfloor$$

Analysis

The Probability of File-Level Collisions Between Sketches:

- **For any two file sketches q, c,** $p(|h(q) - h(c)| = \mathbf{d})$ **monotonically decreases in terms of** $\mathbf{d}$**.**

$$h_{a,b}(c) = \left\lfloor \frac{\alpha \cdot c + b}{w} \right\rfloor$$

Evaluation

Datasets:

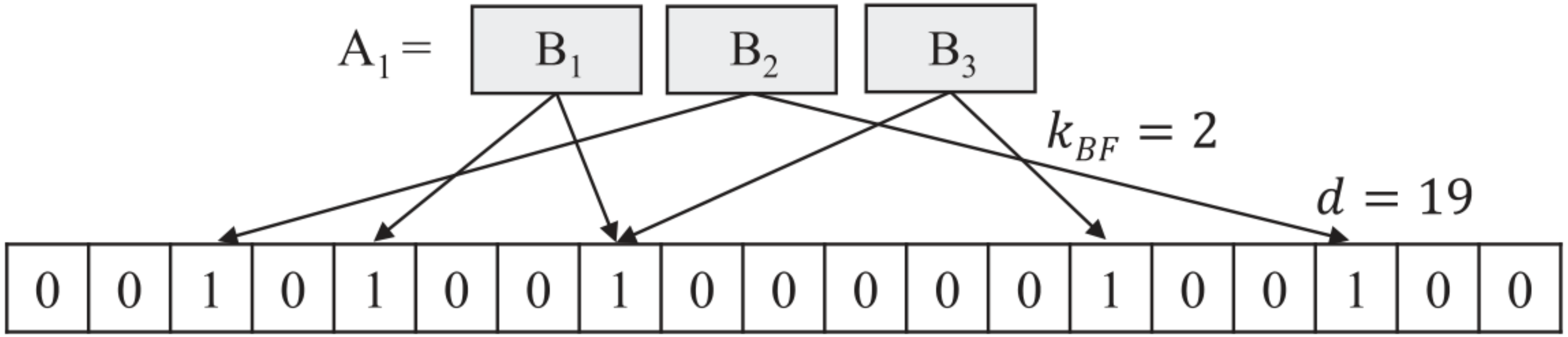
Dataset	GitHub1	GitHub2
Total Data Size	4.00 GB	685.36 MB
Average File Size	9.69 KB	7.37 KB
Max. File Size	8.59 MB	2.49 MB
Min. File Size	1 B	1 B
No. File	432,484	95,179
Average Chunk Size	3.53 KB	3.14 KB
Global Dedup Ratio	88.98%	72.0%

Distinction：

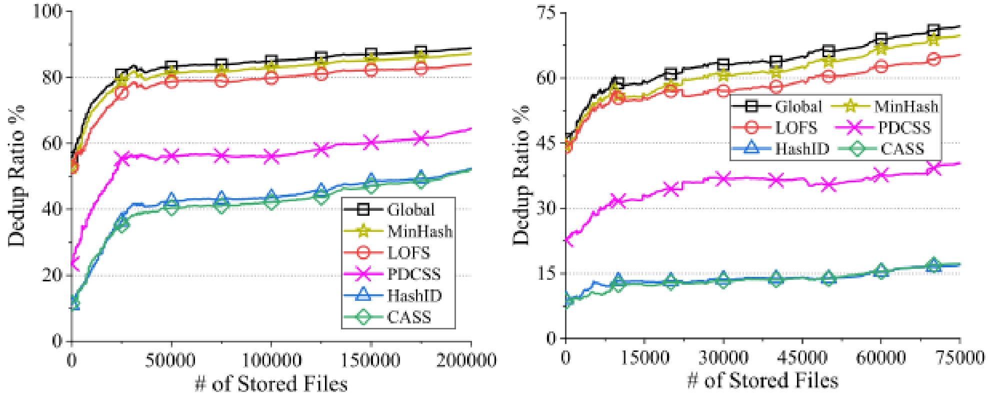
- **A has fewer projects but more versions**
- **B has more projects but fewer versions**

Evaluation

Datasets	GitHub1					GitHub2				
Length of sketch (d)	500	1000	2000	4000	8000	500	1000	2000	4000	8000
Max # of collisions	5	3	2	2	2	3	3	3	2	2
Total # of collisions	2,491	594	136	28	4	2,224	549	133	22	2
Collision ratio	8.31%	1.98%	0.45%	0.093%	0.013%	9.47%	2.37%	0.57%	0.094%	0.017%

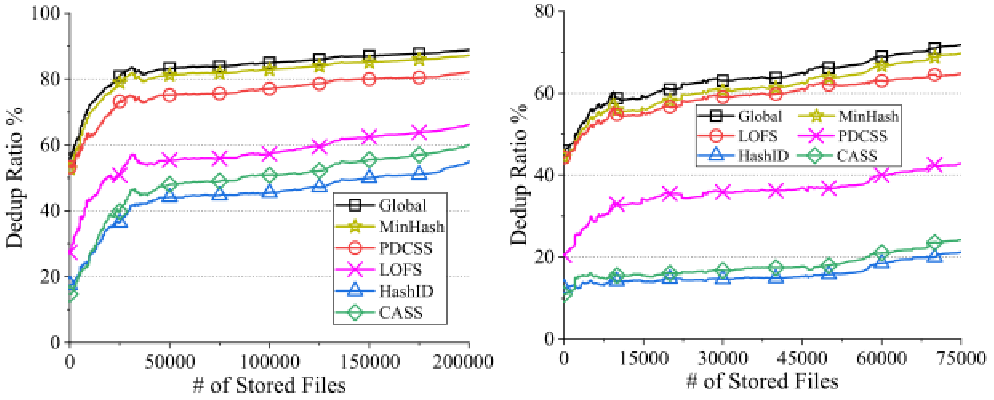


Evaluation



(a) The dedup ratio for GitHub1. (b) The dedup ratio for GitHub2.

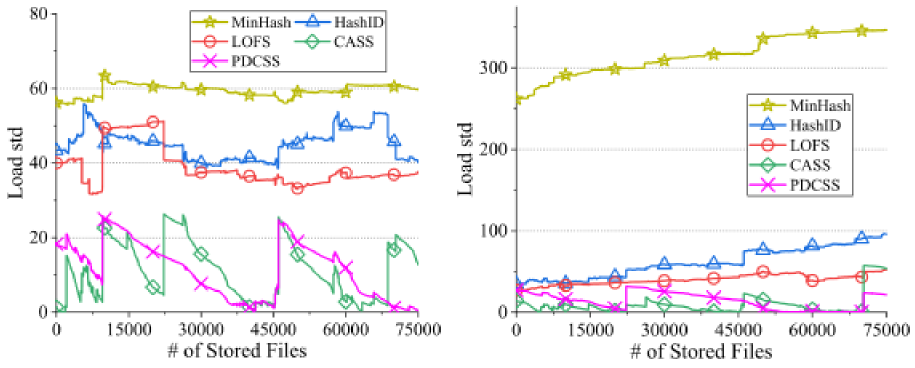
Fig. 7. The dedup ratio with uniform server capacities.



(a) The dedup ratio for GitHub1. (b) The dedup ratio for GitHub2.

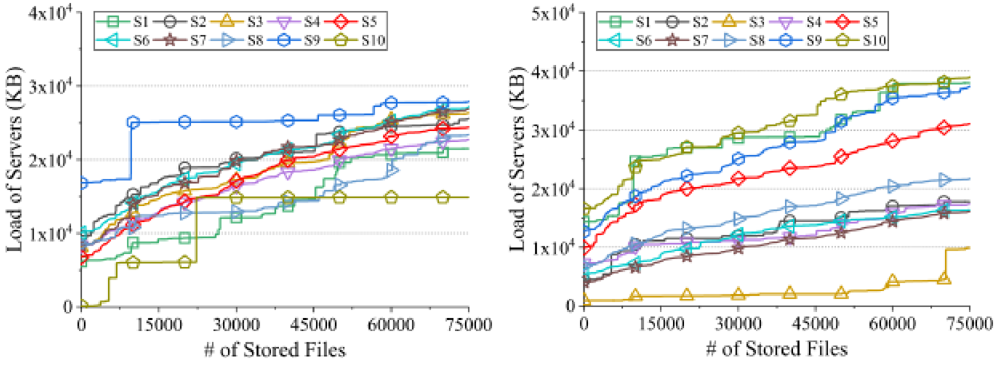
Fig. 8. The dedup ratio with heterogeneous server capacities.

Evaluation



(a) The load std with uniform server capacities.
(b) The load std with heterogeneous server capacities.

Fig. 9. The load std with uniform or heterogeneous capacities.

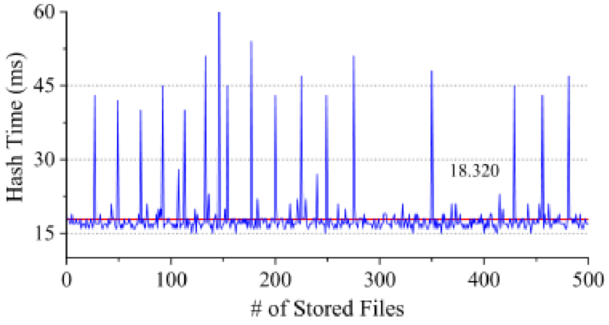


(a) The server load of LOFS with uniform server capacities.
(b) The server load of LOFS with heterogeneous server capacities.

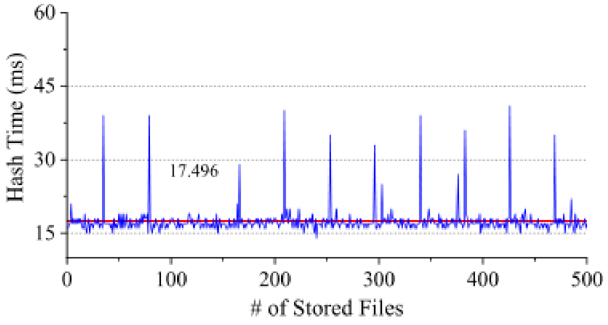
Fig. 10. The server load of LOFS with uniform or heterogeneous capacities.

6, 2.5, 0.8, 1.6, 3.2, 2, 1.9, 2.5, 4.5, 5

Evaluation



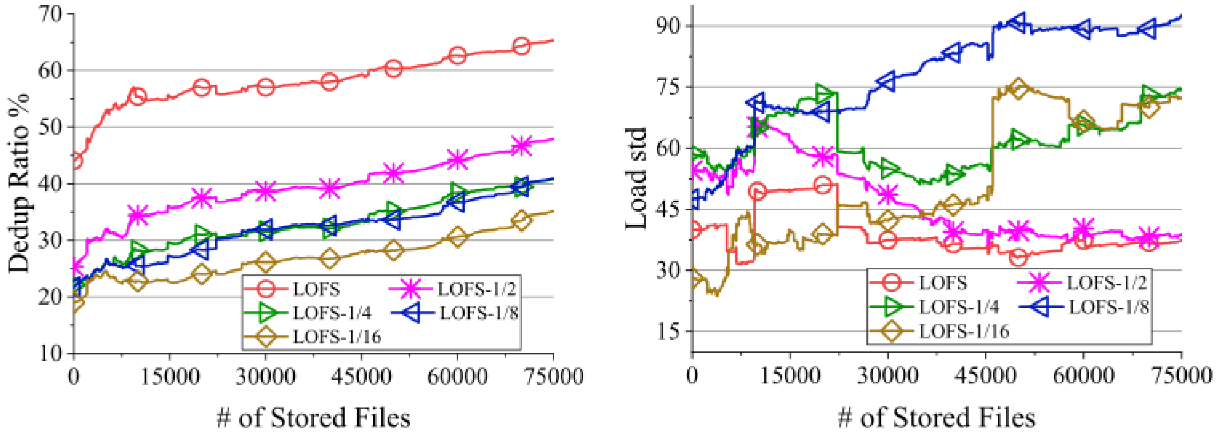
(a) The hash time of LOFS for GitHub1.



(b) The hash time of LOFS for GitHub2.

Fig. 11. The hash time of LOFS.

Evaluation

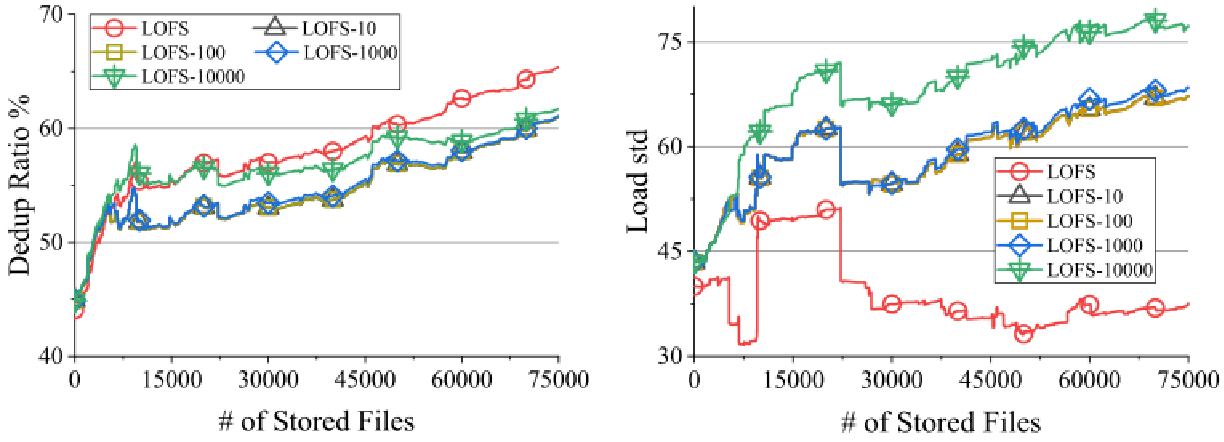


(a) The dedup ratio with different sample ratios.

(b) The load std with different sample ratios.

Fig. 12. The performance of LOFS with different sample ratios.

Evaluation



(a) The dedup ratio with different update frequencies.

(b) The load std with different update frequencies.

Fig. 13. The performance of LOFS with different update frequencies of tablespace partition.

Evaluation

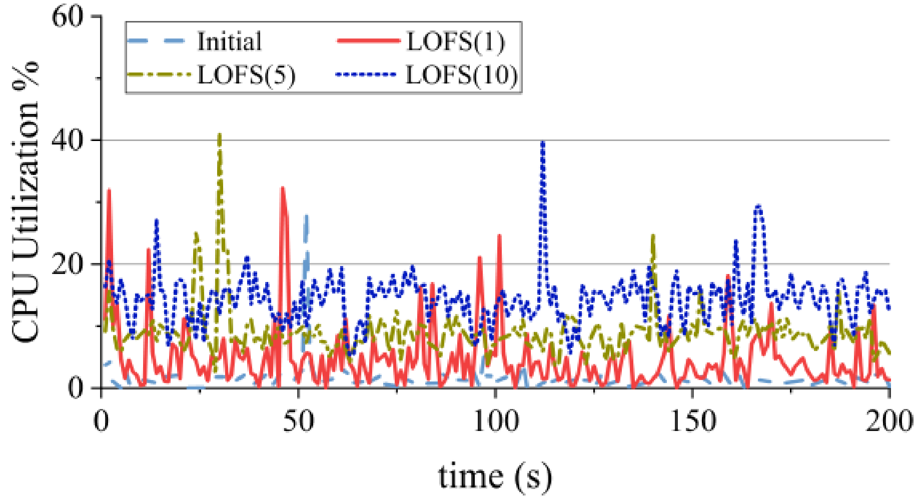


Fig. 14. The CPU utilization with different file arrival rates.

DISCUSSION

- **Application Scenarios**
- **Data Reliability**
- **Transmission Overhead**
- **Data Migration**