FGDEFRAG: A Fine-Grained Defragmentation Approach to Improve Restore Performance

Yujuan Tan, Jian Wen, Zhichao Yan, Hong Jiang, Witawas Srisa-an, Baiping Wang, Hao Luo

CHONGQING UNIVERSITY

UNIVERSITY OF TEXAS ARLINGTON

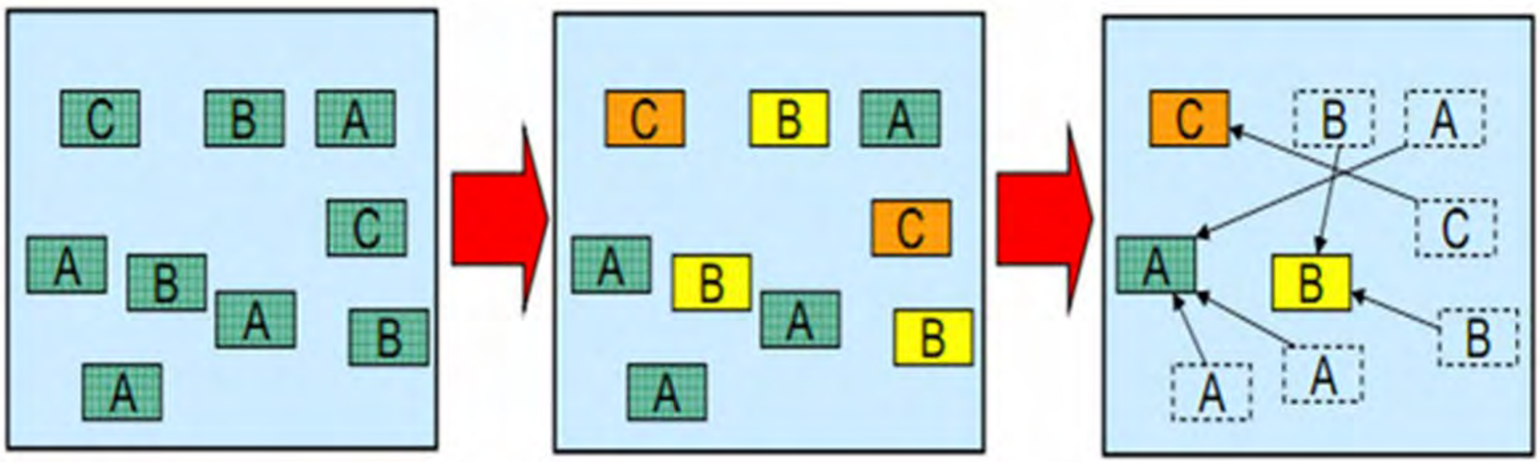
UNIVERSITY OF NEBRASKA-LINCOLN

Outline

- Background and Motivation
- FGDEFRAG Design
- Experimental Evaluation
- Conclusion

Data Deduplication

widely used in backup systems



1. Data elements are analyzed to determine a unique signature for each

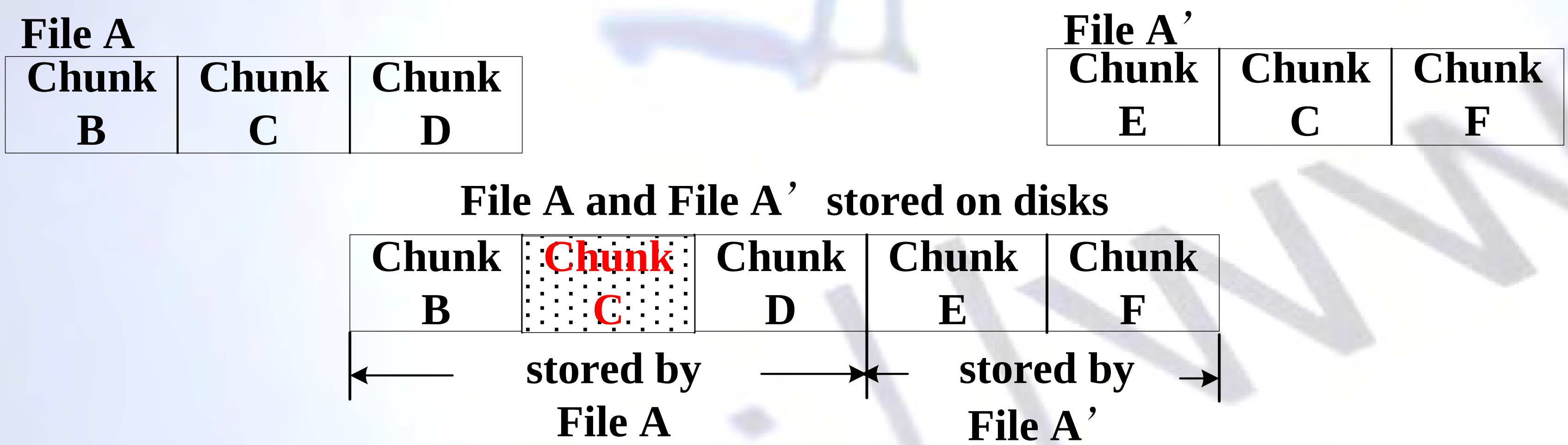
2. Signature values are compared to identify all duplicates

3. Duplicate data elements are eliminated and are replaced with pointers to the existing reference element

High compression ratio 10x~100x

Data Fragmentation

The removal of redundant chunks makes the logically adjacent data chunks be scattered in different places on disks, transforming the retrieval operations from sequential to random.



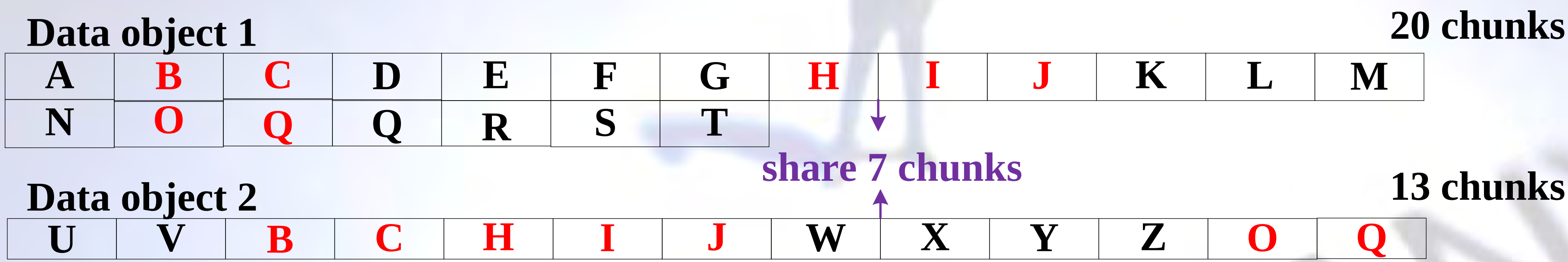
We call a chunk such as chunk *C* as fragmented data of file *A'*

This fragmentation problem results in excessive disk seeks and leads to poor restore performance

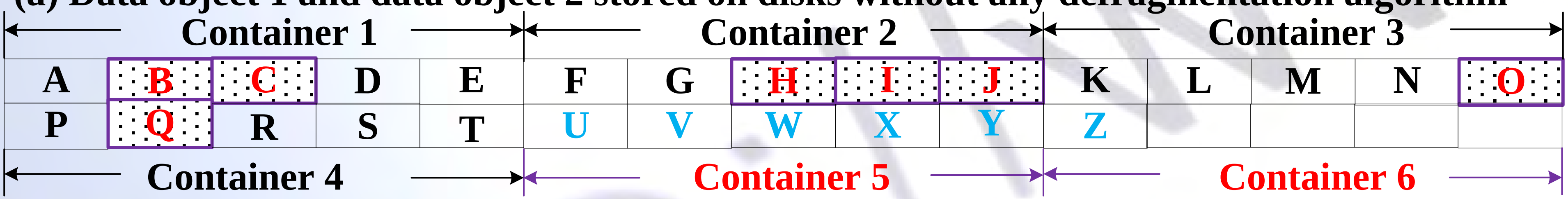
Existing Defragmentation Approaches

HAR, CAP, CBR for backup workloads.

iDedupe for primary storage systems



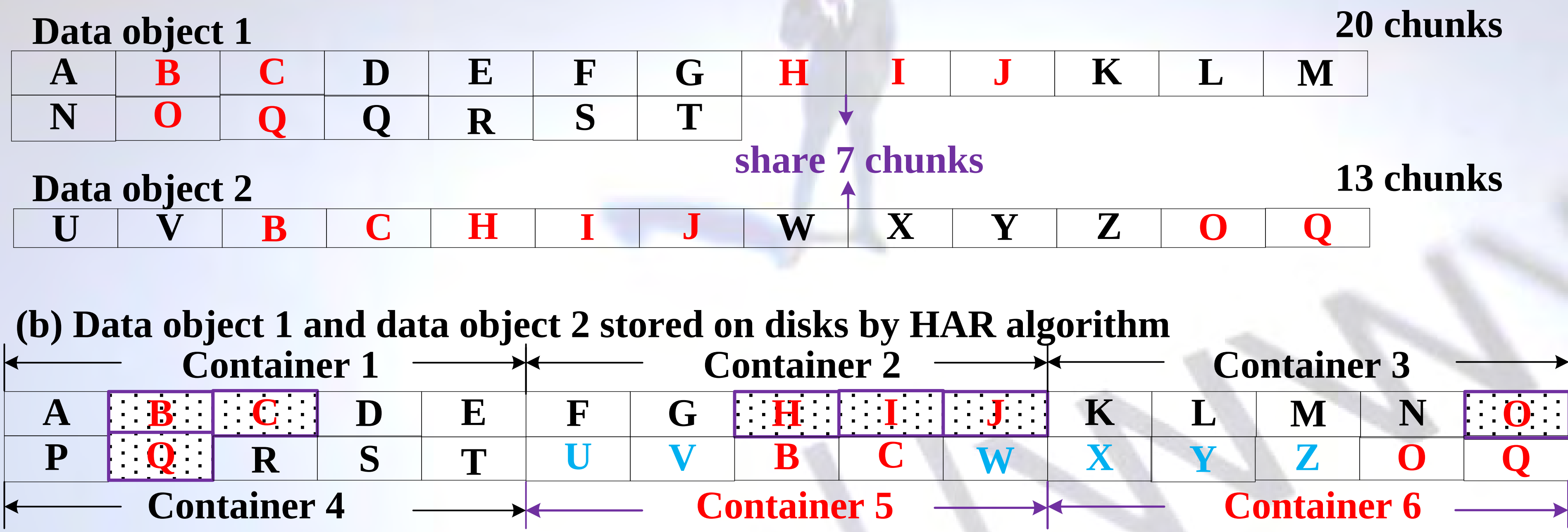
(a) Data object 1 and data object 2 stored on disks without any defragmentation algorithm



All the chunks are stored in fixed-size containers of five chunks each on disks.

Existing Defragmentation Approaches(1)

- HAR: published in USENIX ATC 2015



Sparse Container:

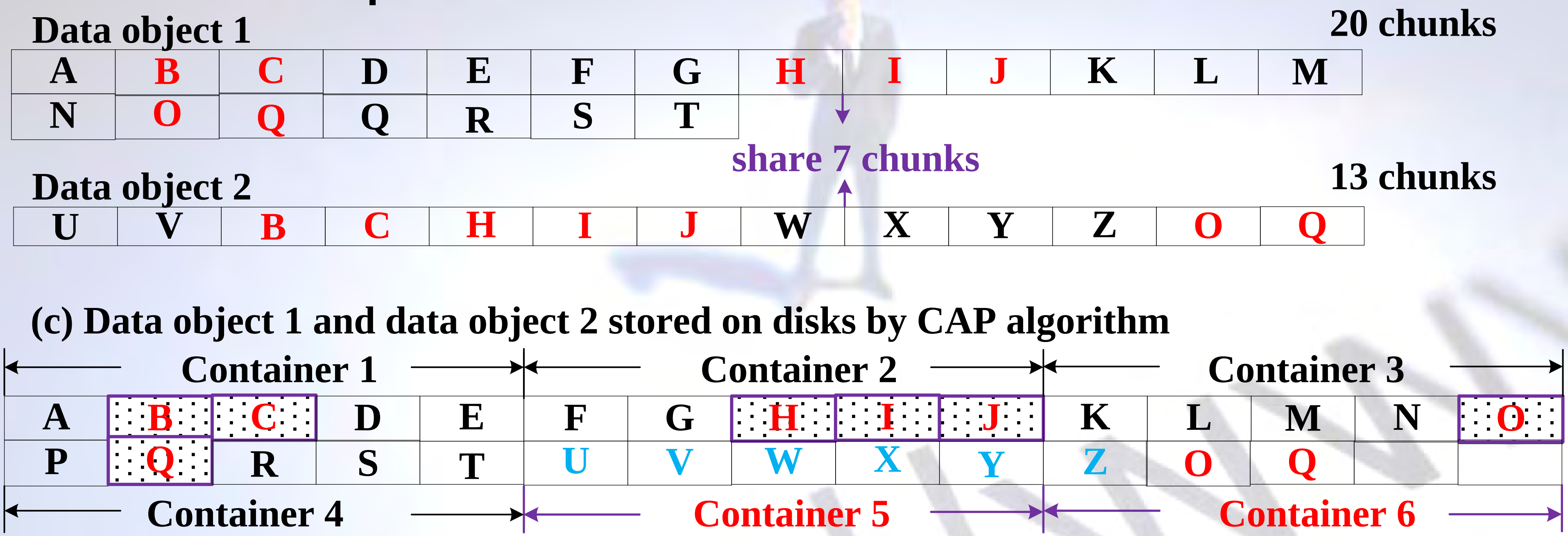
The percentage of the referenced chunks <   50%

Fragmental Containers： Container 1, 3 and 4

Fragmental Chunks: B, C, O and Q

Existing Defragmentation Approaches(2)

- CAP: published in USENIX FAST 2013



Select top N referenced containers---according to the number of referenced valid chunks in each container---as non fragmental containers

If N=2, fragmental containers: Container 3 and 4

fragmental Chunks: O and Q

Existing Defragmentation Approaches

- A common, fundamental assumption

1. Each read operation involves a large fixed number of contiguous chunks

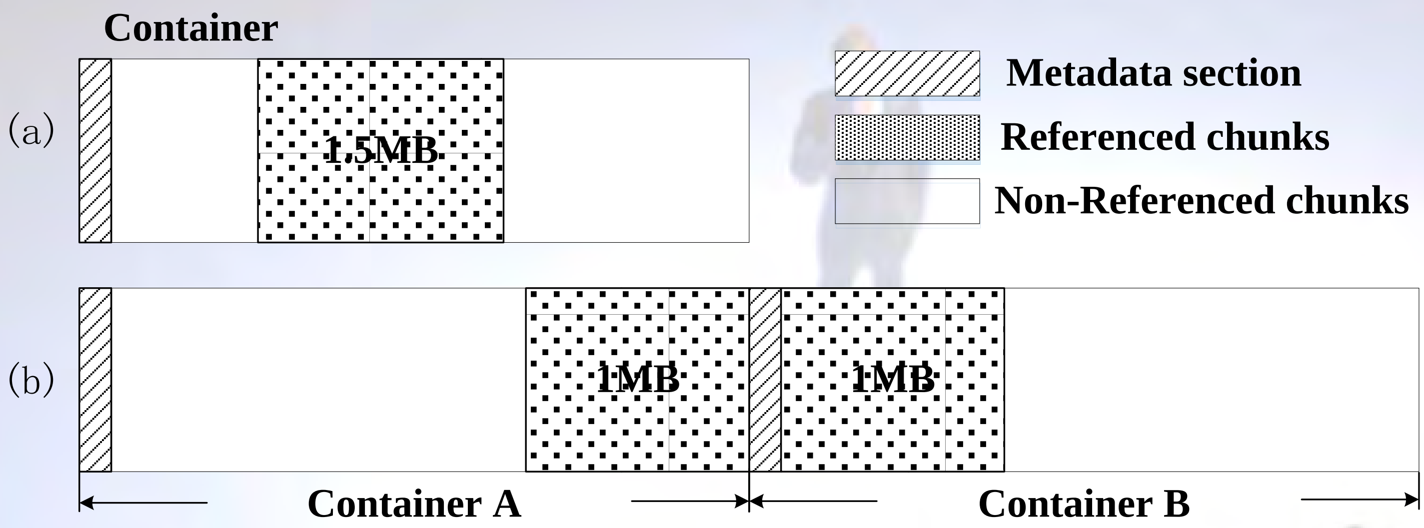
2. The disk seek time is sufficiently amortized for each read operation, and the read performance is determined by the percentage of referenced chunks per read

- Problem:

1. The identification of fragmented data is restricted within a fixed-size read window

2. Causing many false positive detections

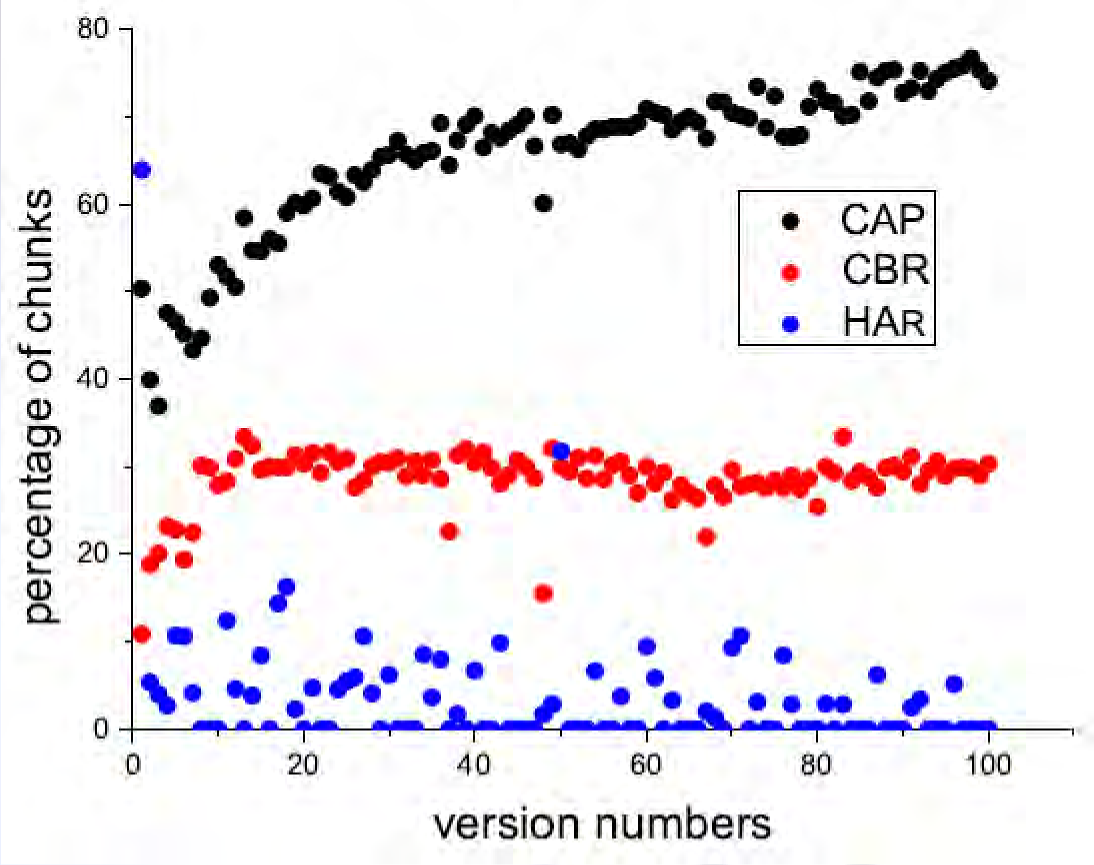
False Positive Detection



(a) A group of referenced chunks stored sufficiently close to one another fails to meet the preset percentage threshold .

(b) A group of referenced chunks that meets the threshold but are split into two neighboring read windows

False Positive Detection



Percentages of data chunks falsely identified by CAP(average 65.3%, maximum 77%), CBR (average 28.7%, maximum 40%), and HAR(average 3.7%, maximum 64%).

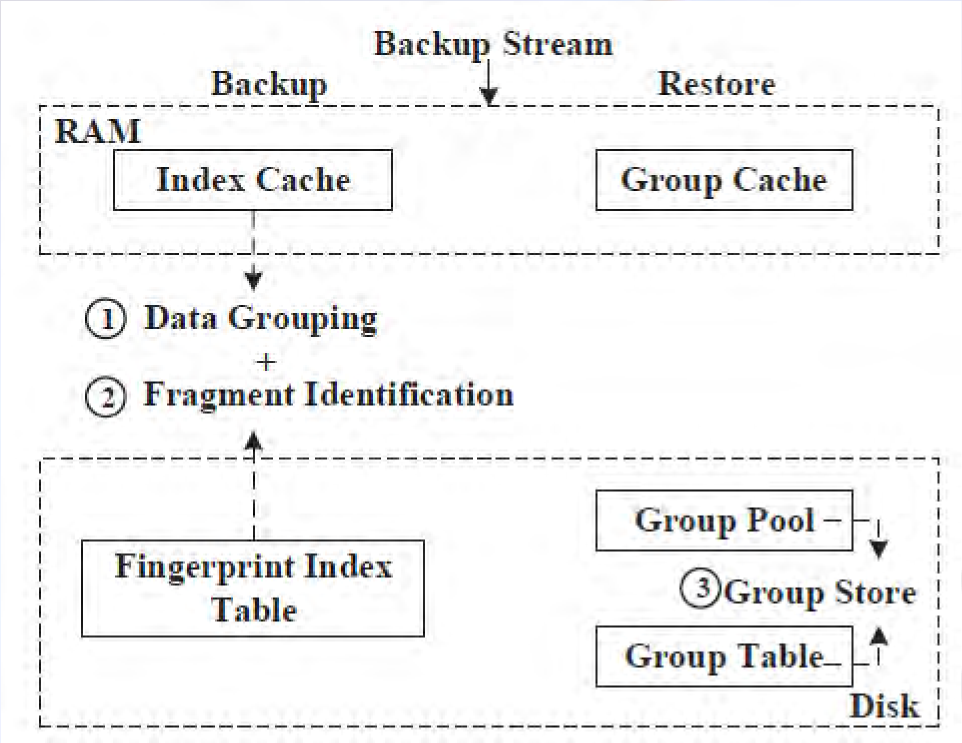
Outline

- Background and Motivation
- FGDEFRAG Design
- Experimental Evaluation
- Conclusion

FGDEFRAG Design

- Uses variable-sized and adaptively located data regions.
- The data regions are based on address affinity, instead of the fixed-size regions.
- Uses the adaptively located data regions to identify and remove fragmented data.
- Uses the adaptively located data regions to atomically read data during data restores.

FGDEFRAG Architecture



Three key functional modules:

Data Grouping, Fragment Identification, Group Store

Data Grouping

(a) The original sequence of the redundant chunks in the segment

A 1001	C 1003	I 1054	D 1006	B 1002	F 1009	G 1010	H 1052	K 1056	O 1015	Q 1017	P 1016	J 1055
R 1018	E 1007	L 1057	M 1059	N 1061								

(b) The sorted list of the redundant chunks in the segment

A 1001	B 1002	C 1003	D 1006	E 1007	F 1009	G 1010	H 1052	I 1054	J 1055	K 1056	L 1057	M 1059
N 1061	O 1081	P 1082	Q 1083	R 1084								

Chunk address

(c) The logical groups in the segment

A 1001	B 1002	C 1003			D 1006	E 1007		F 1009	G 1010

Logical group 1

H 1052		I 1054	J 1055	K 1056	L 1057		M 1059		N 1061

Logical group 2

O 1081	P 1082	Q 1083	R 1084

Logical group 3

Grouping Gap: the amount of non-referenced data between two referenced chunks takes the disk a time equal to or greater than its disk seek time to transfer

Fragment Identification

$$\frac{x}{t + \frac{x+y}{B}} \geq B \cdot \frac{1}{N}$$

- *B* the disk bandwidth, *t* the disk seek time, *N* a non-zero positive integer, *x* the total size of the referenced chunks, and *y* the total size of the non-referenced chunks in the group
- The left side of this inequality expression represents the valid read bandwidth of reading all the referenced data
- The right side of the inequality expression represents the *bandwidth threshold*, a given fraction of the full disk bandwidth *B*.

A group is considered a fragmental group and its referenced chunks regarded as fragmental chunks if the valid read bandwidth is smaller than the bandwidth threshold.

Outline

- Background and Motivation
- FGDEFRAG Design
- Experimental Evaluation
- Conclusion

Performance Evaluation

- **Baseline defragmentation approaches**

HAR(+OPT), CAP(+Assembly Area), CBR (+LFK) , Non-Defragmentation approaches(+LRU or +OPT), FGDEFRAG(+LRU or +OPT)

- **Performance metrics**

Deduplication ratio : the amount of data removed divided by the total amount of data in the backup stream

Restore performance

- **Workload：The public archive datasets**

MAC snapshots：Mac OS X Snow Leopard server

Fslhome dataset：students' home directories from a shared network file system

Workload Characteristics

Dataset name	MAC snaphsots	fslhome
#of versions	100	11
Total size	6.36TB	3.4TB
Unique size	6.29GB	400GB

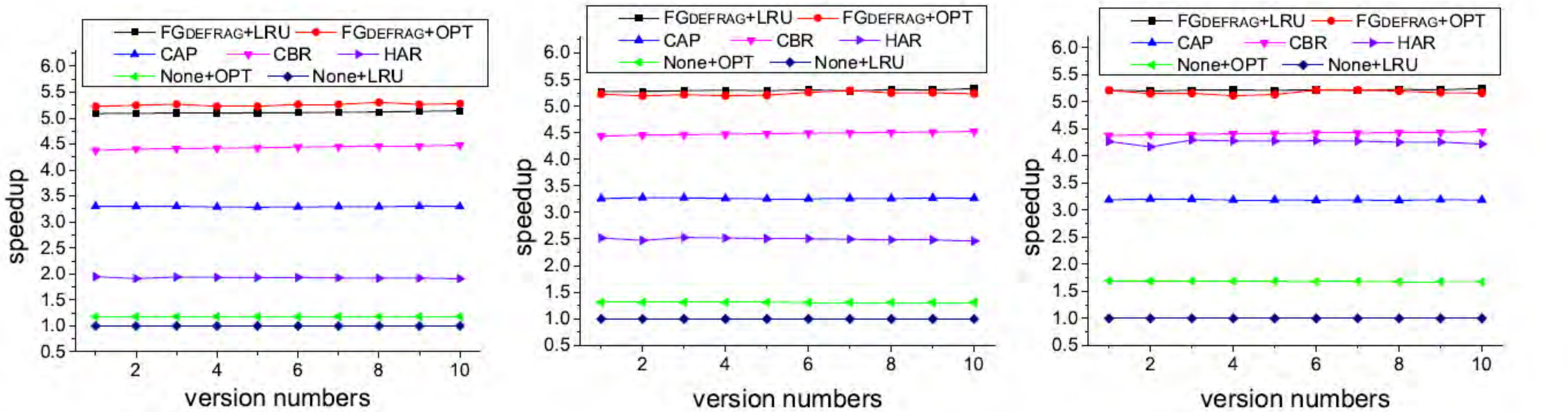
Deduplication Ratio

	Deduplication Ratio (percentage)		Rewritten Data (GB)	
	MAC	fslhome	MAC	fslhome
FGDEFRAG	96.4	91.5	163	112
CAP	90	85	542	380.8
CBR	95.3	90	231	175
HAR	98	93	50	88.2
None	99	93.5	0	0

FGDEFRAG rewrites 70% and 29.4% less data than CAP and CBR for the MAC snapshots dataset, 70.6% and 36% less data than CAP and CBR for the Fslhome dataset.

HAR identifies the fragmental chunks a whole backup stream globally. It misses identifying some local fragmental chunks, and thus rewrites less redundant chunks to disks

Restore Performance

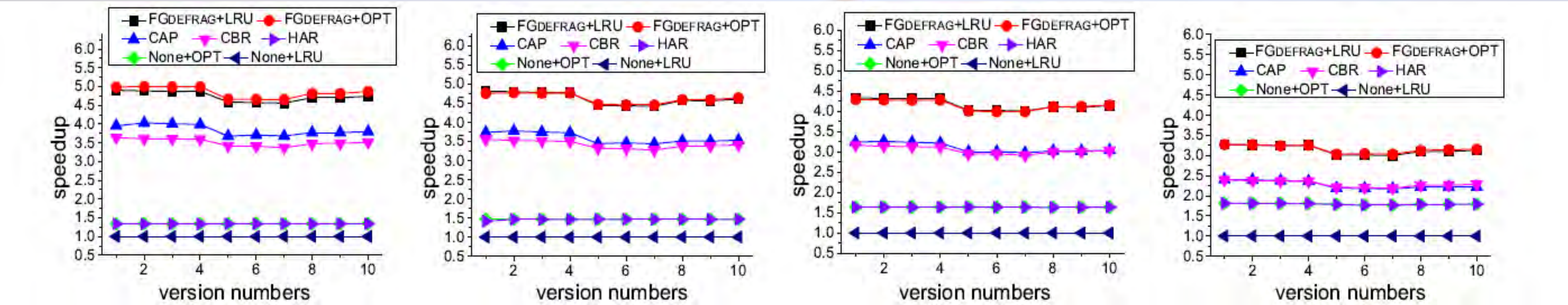


(a) Cache size: 512MB (b) Cache size: 1GB (c) Cache size: 2GB

Fig. 6: Comparison between FGDEFRAG and the baseline approaches in restore performance with the last 10 versions of the MAC snapshots dataset. Speedup represents the restore performance normalized by that of the None+LRU approach.

FGDEFRAGE outperforms CAP, CBR and HAR by 60%, 20% and 176% when the cache size is 512MB; 63%, 19% and 116% when the cache size is 1GB, and 62%, 19.6% and 23% when the cache size is 2GB.

Restore Performance



(a) Cache size: 512MB (b) Cache size: 1GB (c) Cache size: 2GB (d) Cache size: 4GB

Fig. 7: Comparison between FGDEFRAG and the baseline approaches in restore performance with the last 10 versions of the Fslhome dataset. Speedup represents the restore performance normalized by that of the None+LRU approach.

- FGDEFRAG outperforms CAP, CBR and HAR by 27%, 38% and 262% with a 512MB cache; 30%, 37% and 217% with a 1GB cache; 35%, 38% and 159% with a 2GB cache; and 43%, 39%,and 76% with a 4GB cache.

Sensitive study

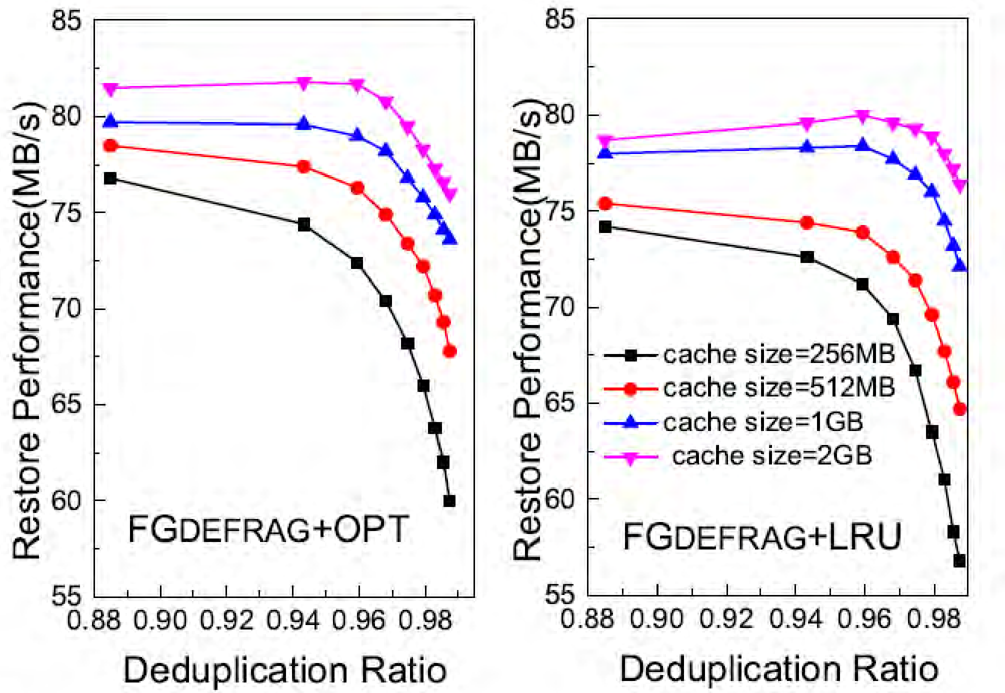


Fig. 8: Sensitivity of the restore performance and deduplication ratio to the value of the bandwidth threshold factor N. The restore performance is the average value of the last 20 versions of the MAC snapshots dataset. The 9 data points on each curve represent the corresponding restore performance and deduplication ratio for each of the 9 N values, 2, 4, 6, 8 10, 12, 14, 16, 18, from left to right.

The deduplication ratio increases with *N*, while the restore performance decreases significantly as *N* increases.

To properly trade off between deduplication ratio and restore performance, we need to select appropriate values of *N* for different datasets.

Outline

- Background and Motivation
- FGDEFRAG Design
- Experimental Evaluation
- Conclusion

Conclusion

- Analyzing the existing defragmentation approaches
- Proposing FGDEFRAG, a new defragmentation approach that uses variable-sized and adaptively located groups to identify and remove fragmentation.
- Our experimental results show that FGDEFRAG outperforms CAP, CBR and HAR in restore performance by 27% to 63%, 19% to 39%, 23% to 262%.
- FGDEFRAG also outperforms CAP and CBR but slightly underperforms HAR, because HAR identifies the fragmental chunks globally but at the expense of missed detection of some local fragmental chunks。