2DCC: Cache Compression in Two Dimensions

Amin Ghasemazar, Mohammad Ewais, Prashant Nair, Mieszko Lis

The University of British Columbia

DATE 2020

a place of mind
THE UNIVERSITY OF BRITISH COLUMBIA

Natural Sciences and Engineering Research Council of Canada
Conseil de recherches en sciences naturelles et en génie du Canada

Canada

Executive summary

Observation: leveraging redundancy only **within** or **across datalines** leads to a significant loss in compression opportunities

Problem: how to takes advantage of both types of redundancy?

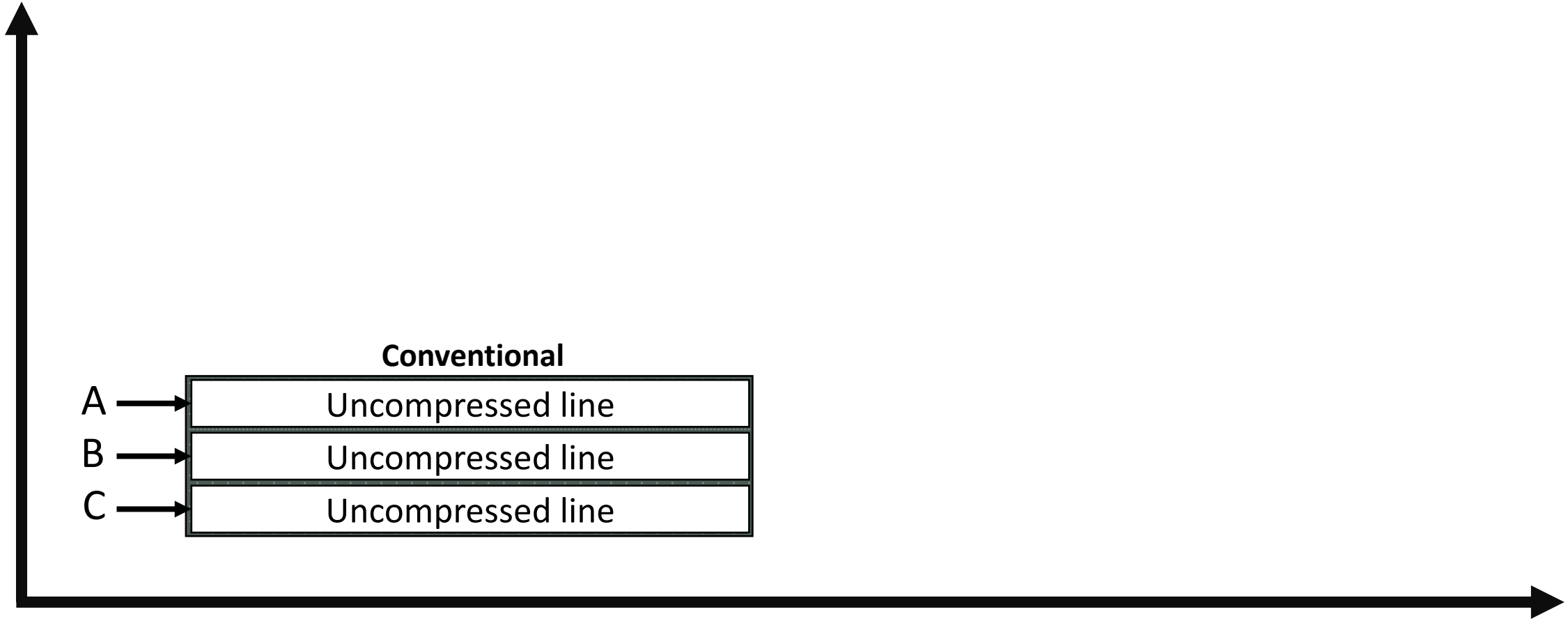
Key Idea: simple technique that enables compressing working sets that

- contain **either** type of redundancy
- contain **both** types of redundancy

Results: 2.12× geomean compression

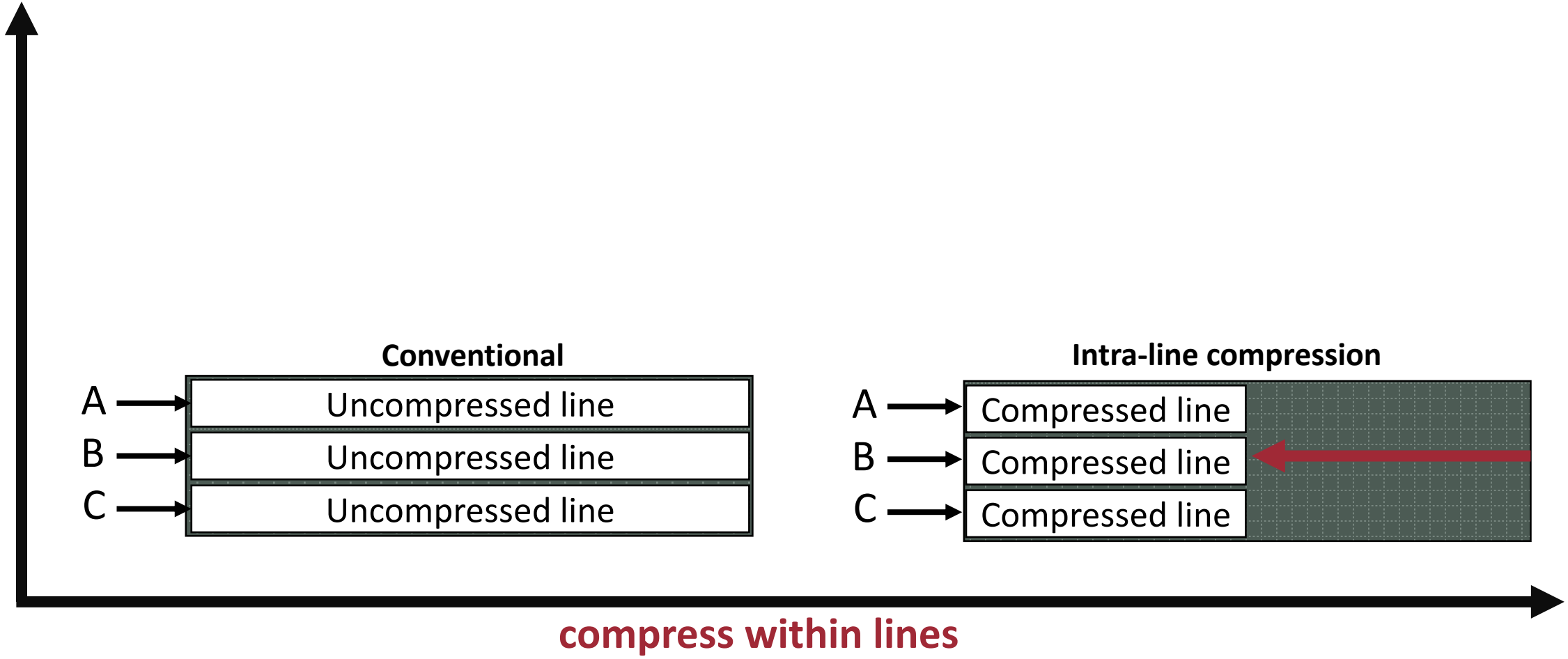
Compressed caches: overview

Uncompressed cache: store a block for each tag



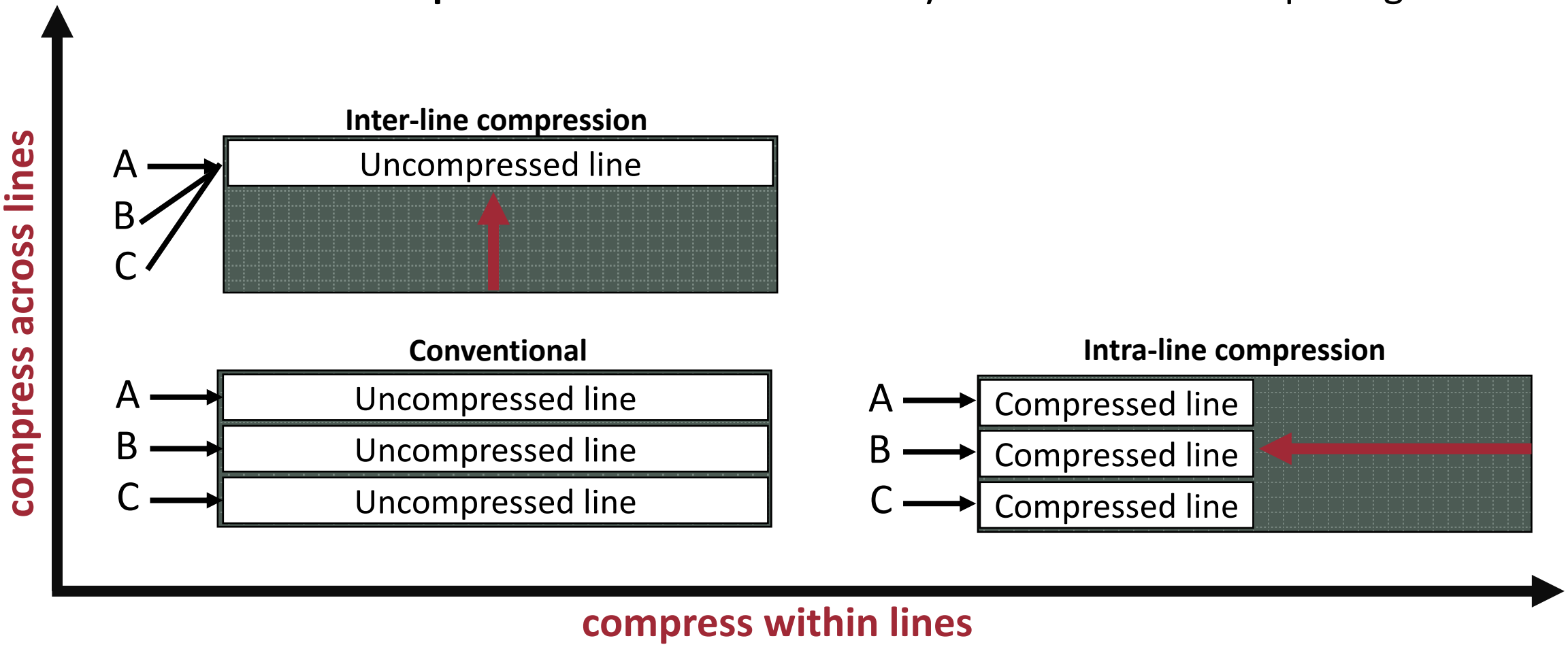
Compressed caches: overview

Compress within lines: store multiple smaller blocks



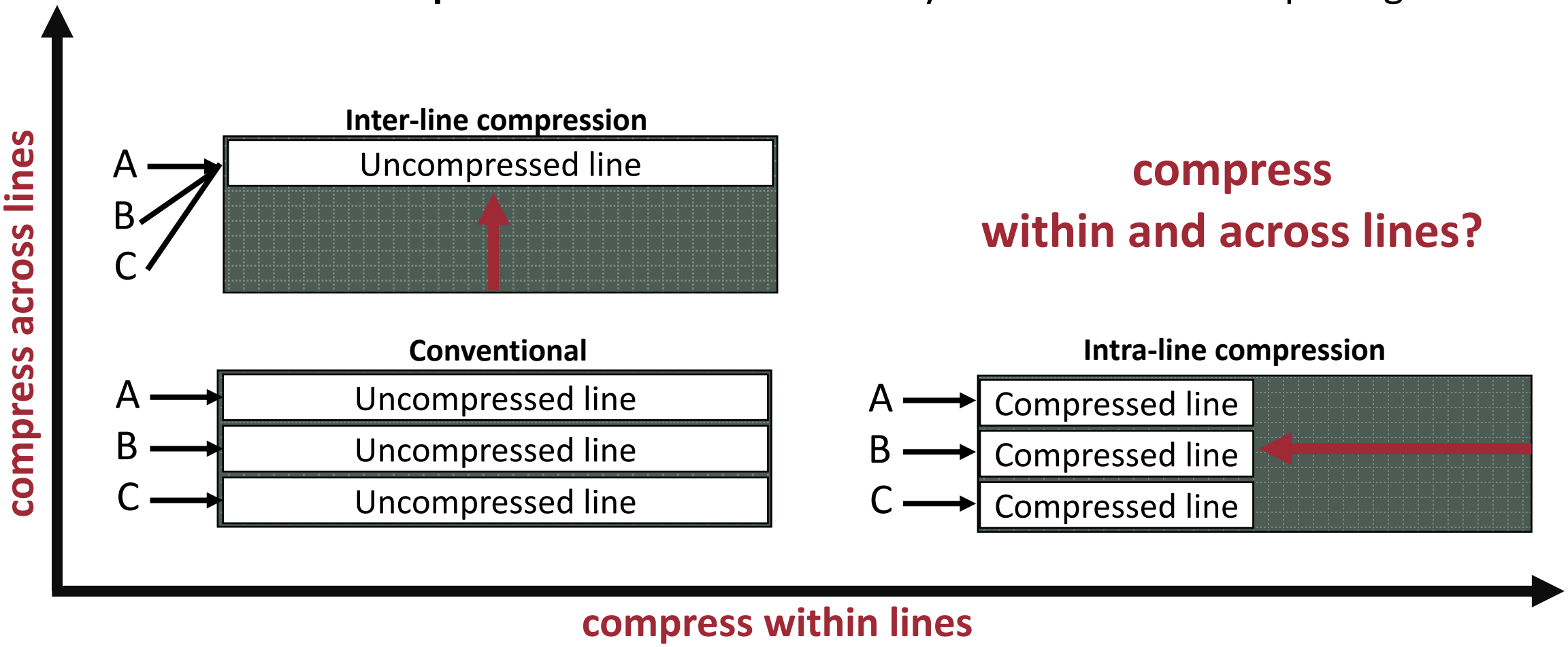
Compressed caches: overview

Compress within lines: store multiple smaller blocks
Compress across lines: store only one block for multiple tags



Compressed caches: overview

Compress within lines: store multiple smaller blocks
Compress across lines: store only one block for multiple tags



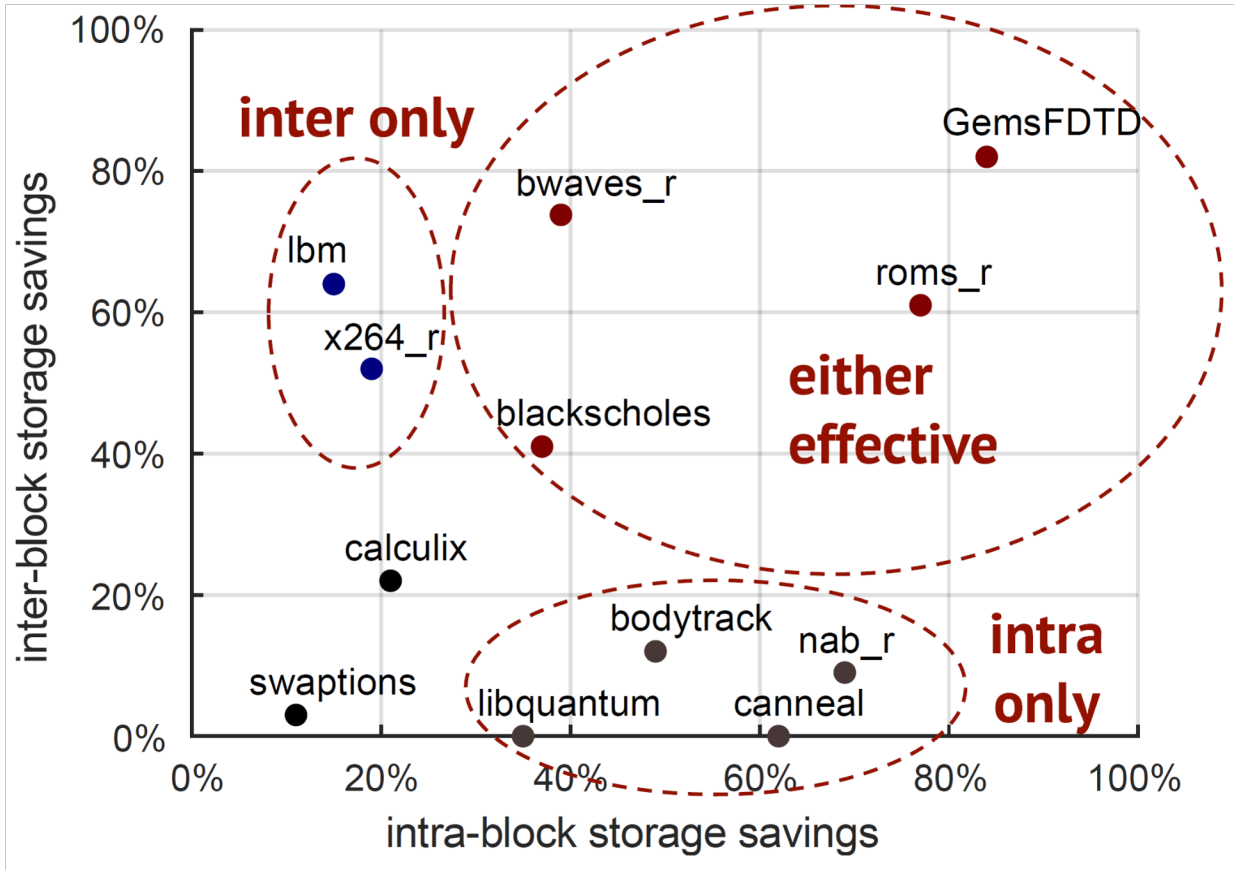
Motivation

- Does real workloads exhibit both type of redundancy patterns?

Motivation

Workloads are compressible using:

- **Inter only**
- **Intra only**
- **Either** methods

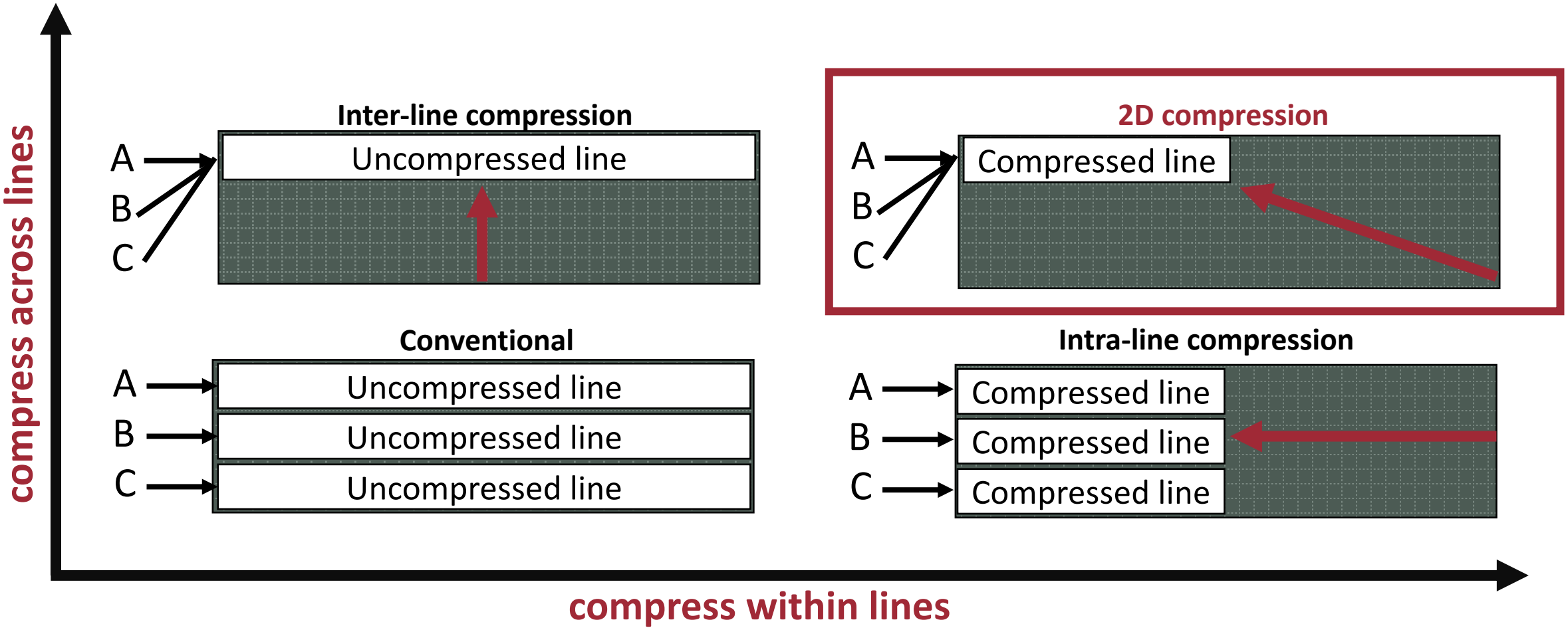


(more is better)

D. A. Huffman, "A method for the construction of minimum-redundancy codes," *IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

Goal: compress within and across cachelines

Compress within and across lines: store smaller block for multiple tags



Goal: compress within and across cachelines

Compress within and across lines: store smaller block for multiple tags

Cache Compression in two dimensions

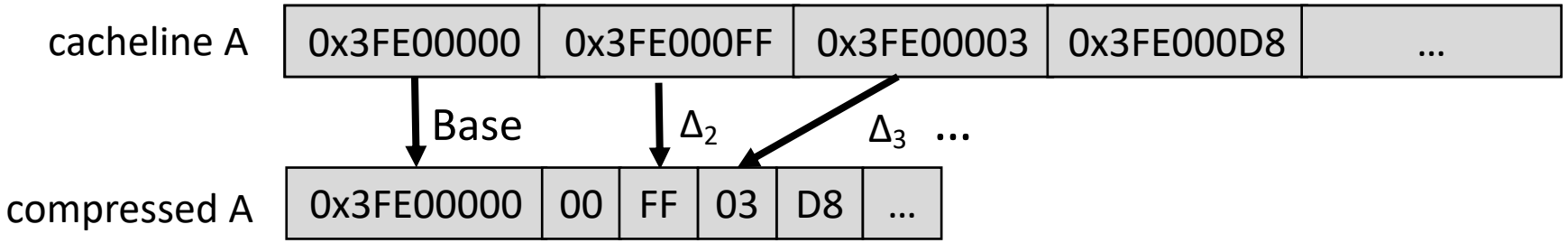
2DCC

Research questions

- Q1: How to compress **across** and **within** datalines ?
- Q2: How to find **identical cachelines** quickly ?
- Q3: How to store **variable-sized cachelines** ?
- Q4: How to efficiently **utilize** the storage structures ?

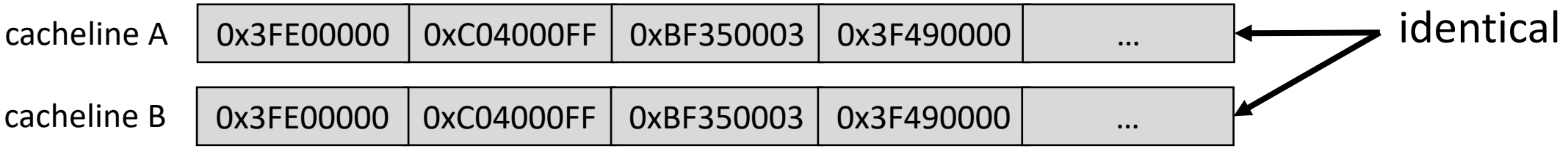
Compression within and across cachelines

intra-line compression



[1]

inter-line compression



[2]

[1] G. Pekhimenko et al., Base-delta-immediate compression: practical data compression for on-chip caches, PACT ’12.
[2] Y. Tian et al., Last-level cache deduplication, ICS ’14.

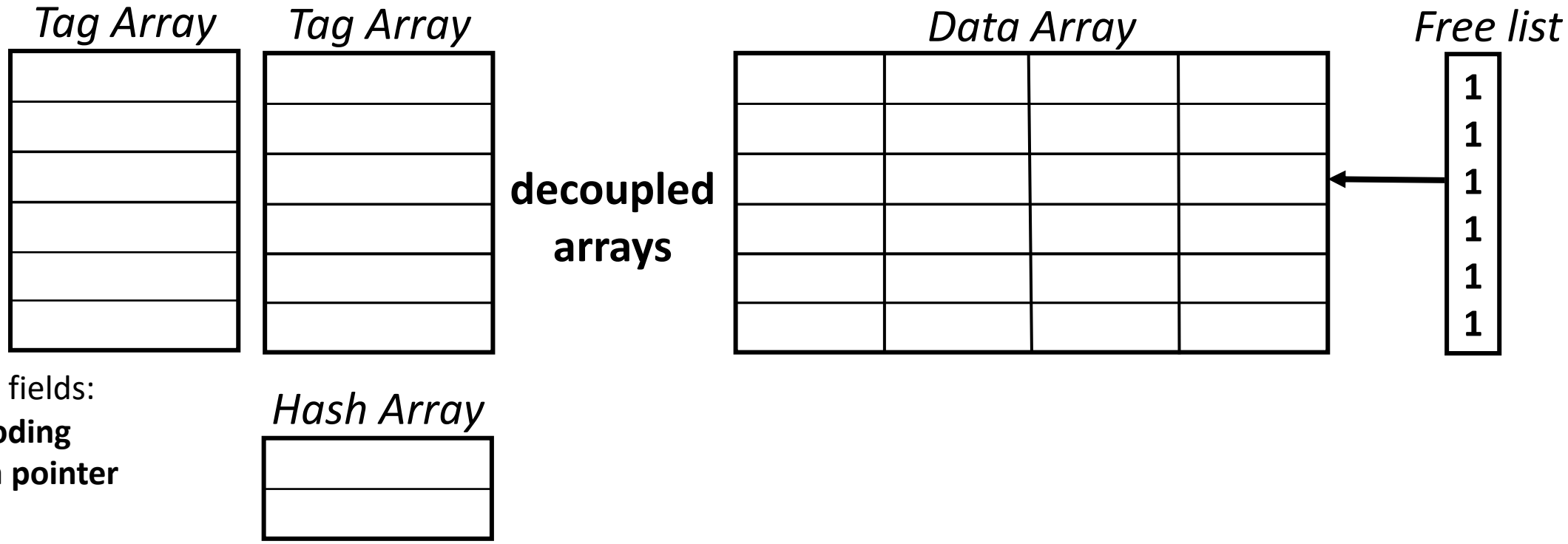
2DCC

Architecture

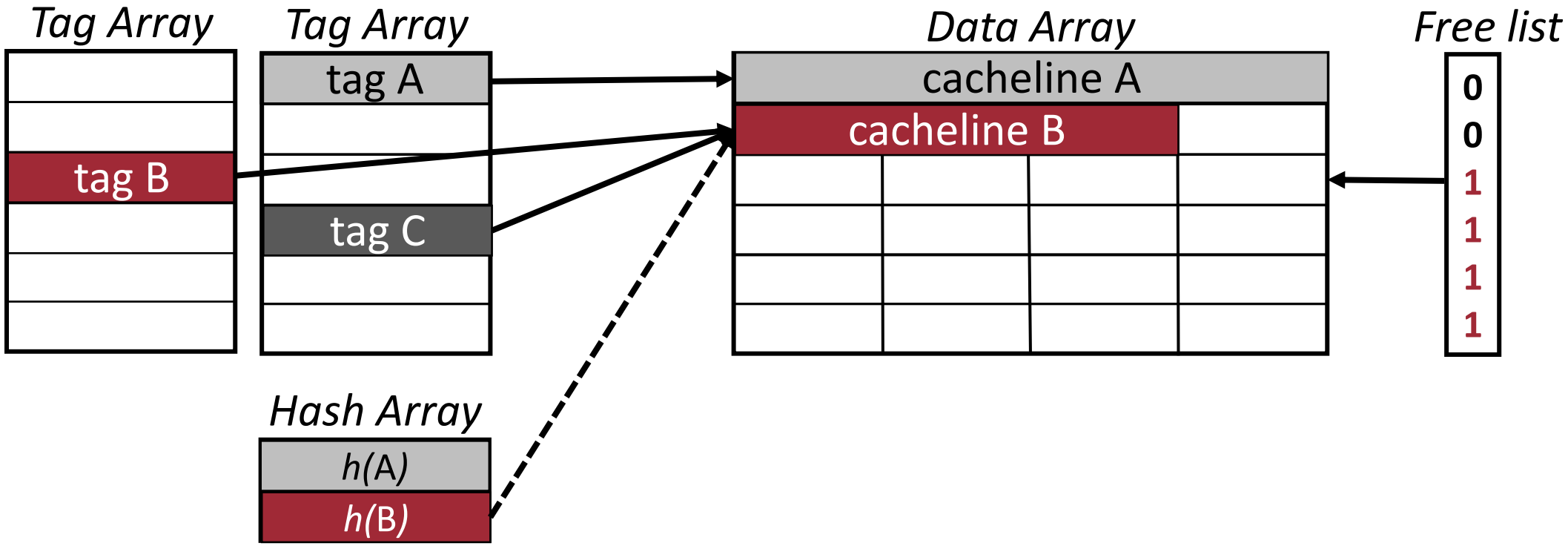
2DCC: challenges

- Detecting duplicate blocks quickly ➔ **Block content hashing**
- Storing variable-sized blocks ➔ **segmenting the data array**
- Avoiding fragmentation ➔ **Free list, Replacement policies**

2DCC: storage structures



2DCC: storage structures



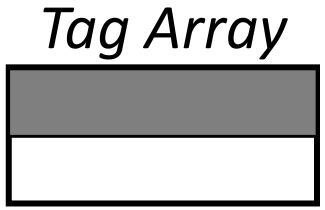
2DCC: challenges (continued)

- Decoupling cache structures → cache may be limited by tag or data storage

→ **Replacement policies per cache structures**

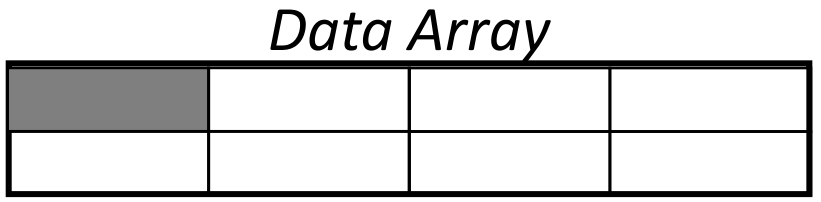
2DCC: replacement policies

Tag Array: retaining addresses likely to be accessed in the future



➔ **LRU**

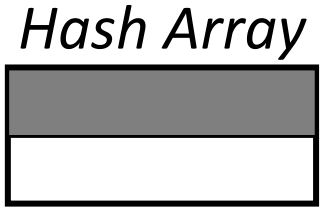
Data Array: enabling the tag array to store more



➔
1) **available free set ➔ insert block**
2) **randomly find partly occupied set with enough space**
3) **chose the set with least number of evictions**

Hash Array: identifying

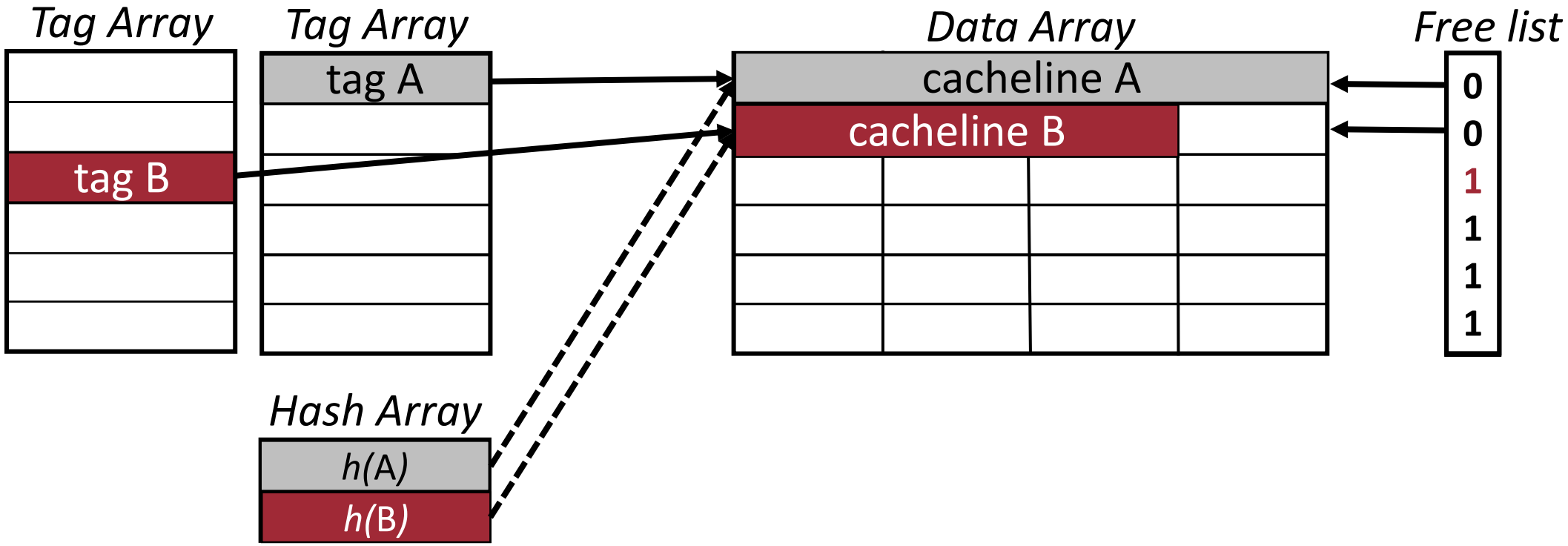
1) cached blocks whose contents are likely to reappear
2) incoming blocks whose contents are likely to reappear



➔ **LRU**

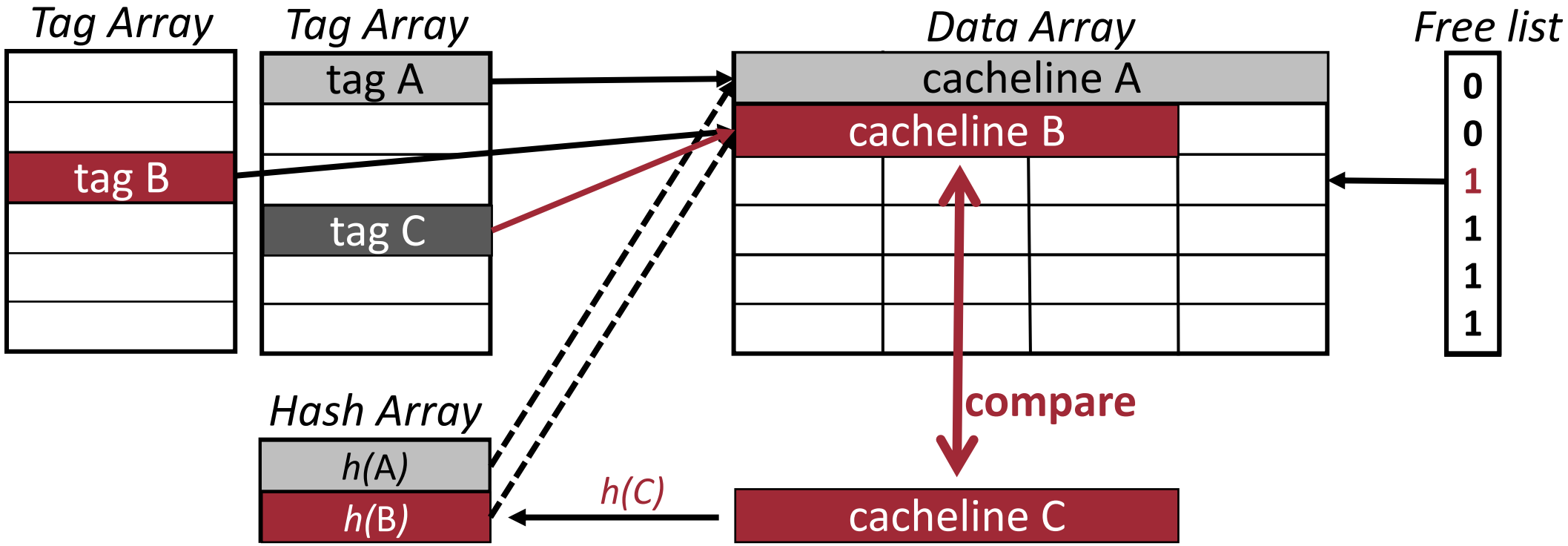
2DCC: operation example

Available **free set**



2DCC: operation example

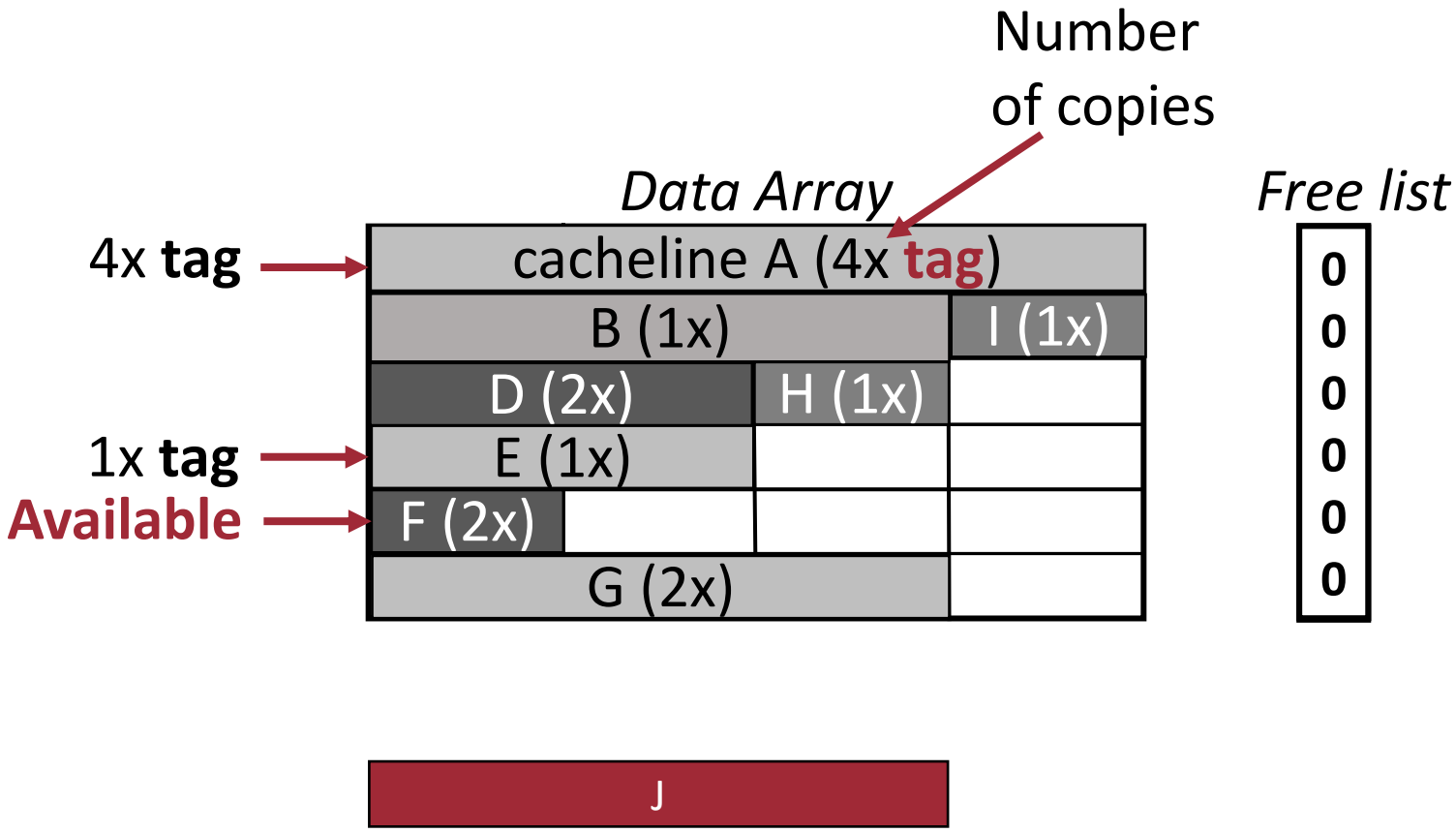
Available **free set**



2DCC: operation example

~~Available **free set**~~

Randomly find **partly occupied set** with **enough space**

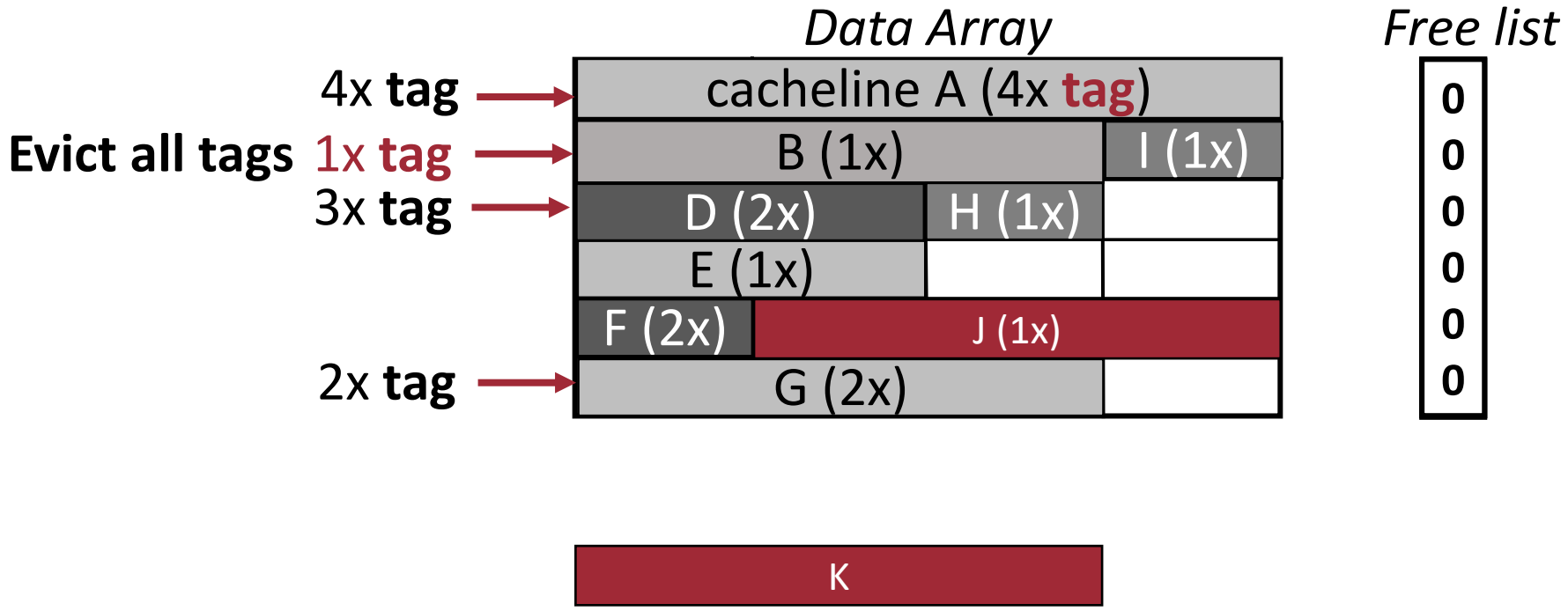


2DCC: operation example

~~Available **free set**~~

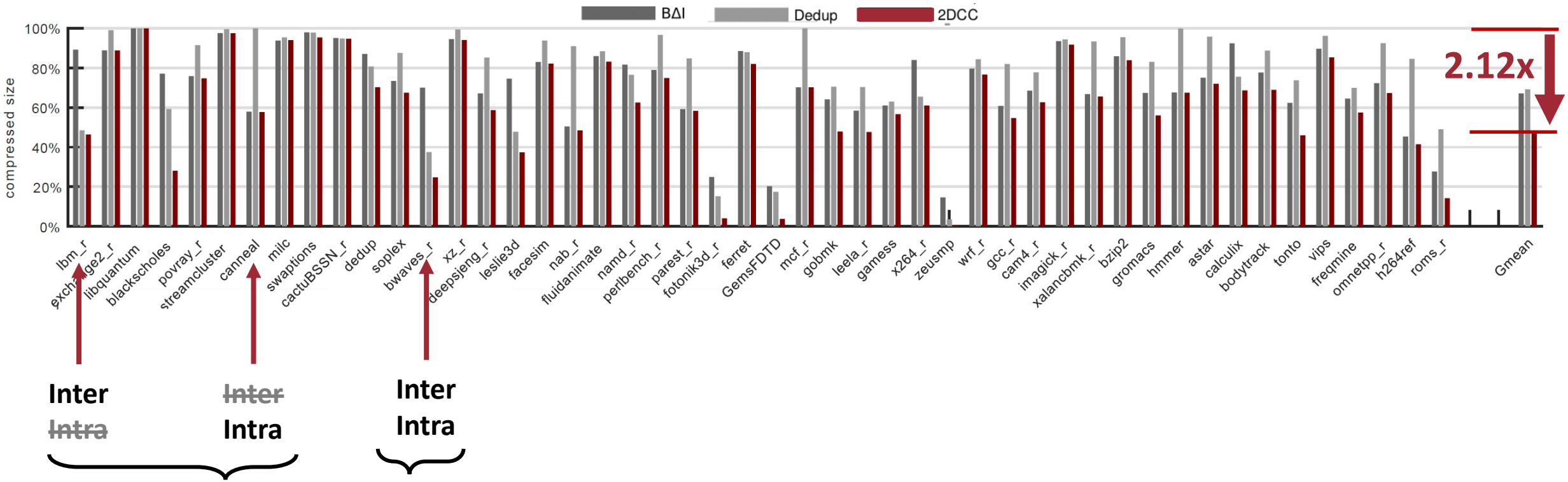
Randomly find **partly occupied set** with ~~**enough space**~~

chose the set with **least number of evictions**



2DCC
Results

Results: compression



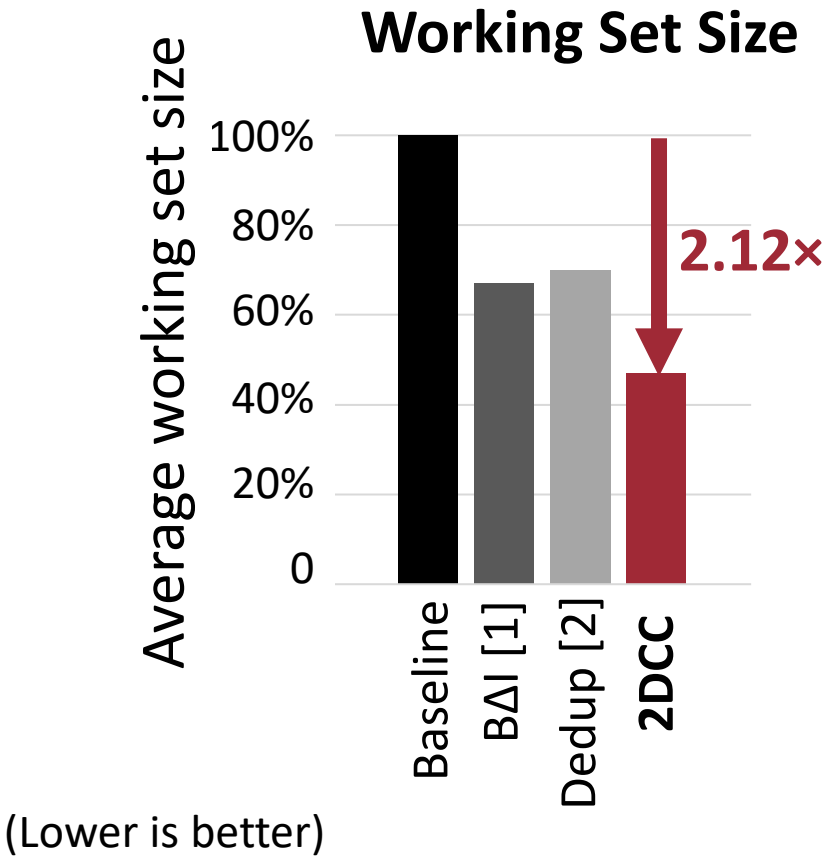
Inter
~~**Intra**~~

~~**Inter**~~
Intra

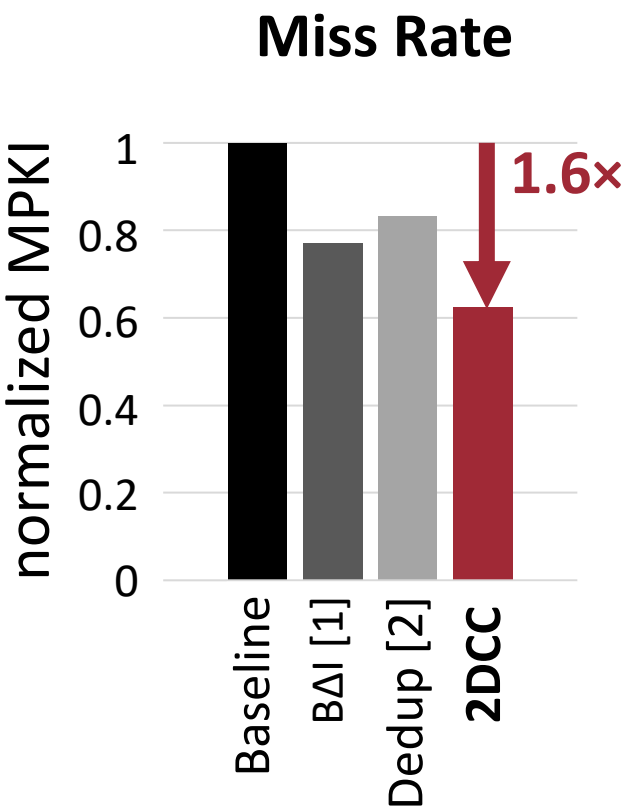
Inter
Intra

2DCC: **Selecting** / **Combining** methods

Results: compression and performance



(Lower is better)



Iso-silicon 1MB

Results reported for SPEC CPU'06, SPEC CPU'17, PARSEC

Compression 2.12×

[1] G. Pekhimenko et al., Base-delta-immediate compression: practical data compression for on-chip caches, PACT '12.

[2] Y. Tian et al., Last-level cache deduplication, ICS '14.

Results: compression breakdown

2DCC compression benefit comes from: