BOSTON UNIVERSITY

CS 561: Data Systems Architectures

Class 20

Correlation-Aware Partitioning for Joins

Zichen Zhu

https://bu-disc.github.io/CS561/

Join in Relational Databases

Enroll

ClassID	StudentID
cs561	0000011
cs561	3078002
…	…
0000011	0000011

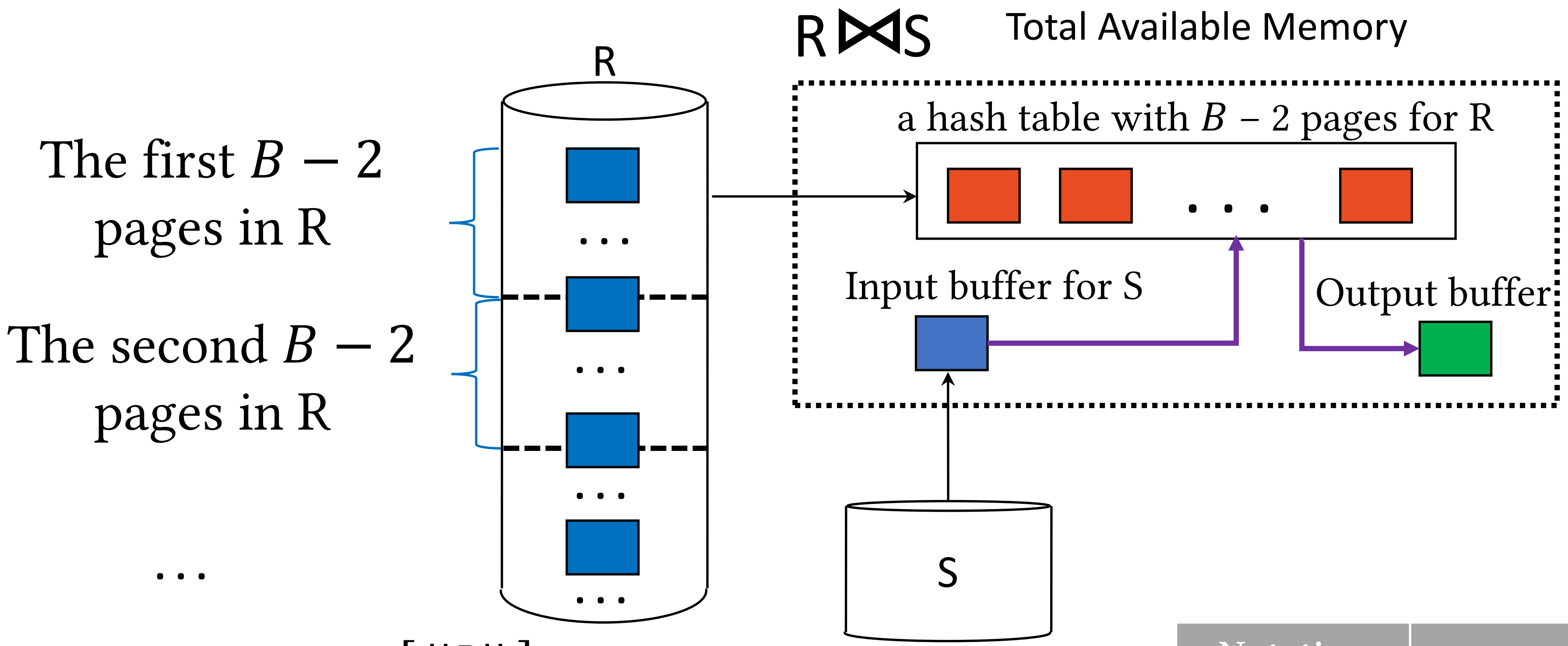
```
Select *
From Student, Enroll
Where Student.StudentID = Enroll.StudentID
```

Student

StudentID	YOB	…	Gender
0000001	1970/01/02	…	M
0000002	1966/03/02	…	F
…	…	…	…
6534702	2000/10/02	…	M

BOSTON UNIVERSITY

Block Nested Loop Join (assuming $||S|| > ||R||$)



Join cost (I/Os): $\left\lceil \frac{||R||}{B-2} \right\rceil \cdot ||S|| + ||R||$

Notation	Meaning
$\lvert\lvert R \rvert\rvert$ ($\lvert\lvert S \rvert\rvert$)	The number of pages of relation R (S)
B	Buffer size (in pages)

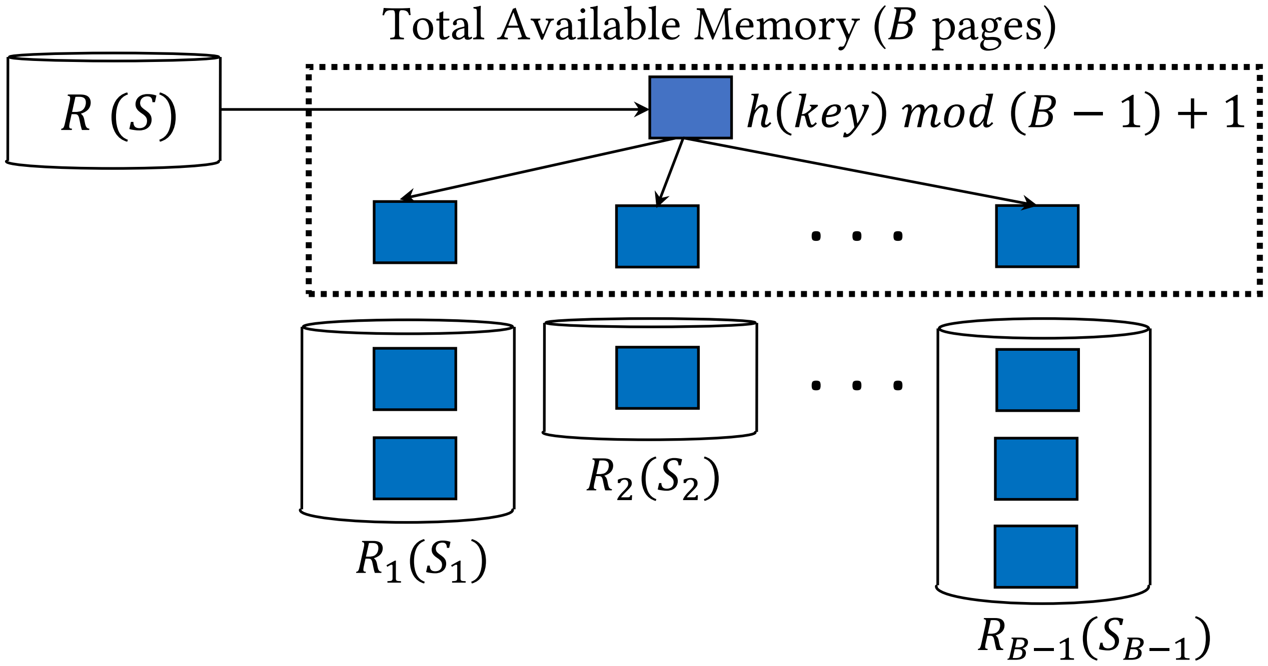
If B is large, the minimum #I/O is $||S|| + ||R||$ when $||R|| \leq B-2$

Grace Hash Join

$R \bowtie S$

$2 \times (||S|| + ||R||)$

Partitioning both R and S



Assuming $||R_j|| \leq B-2$

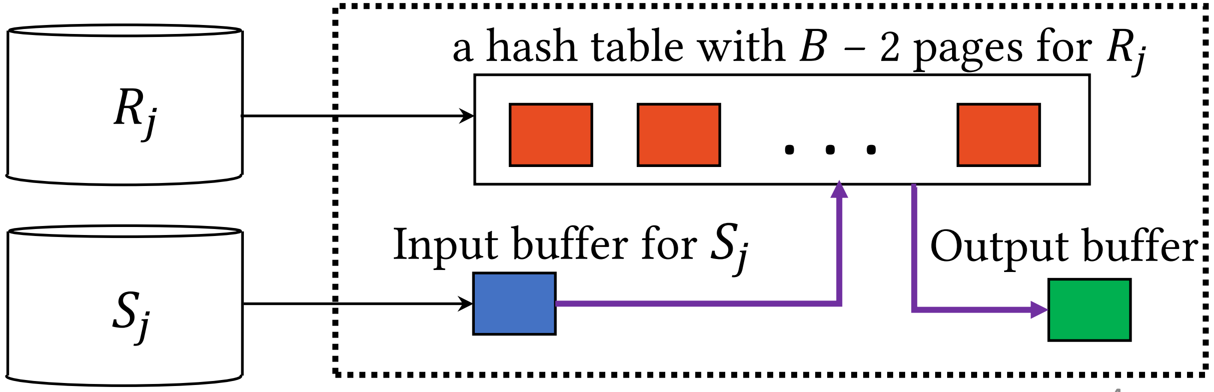
$$\sum_{j=1}^{B-1}(||S_j|| + ||R_j||) = ||S|| + ||R||$$

Totally, the #I/Os for Grace Hash Join is

$$3 \cdot (||S|| + ||R||)$$

Probing

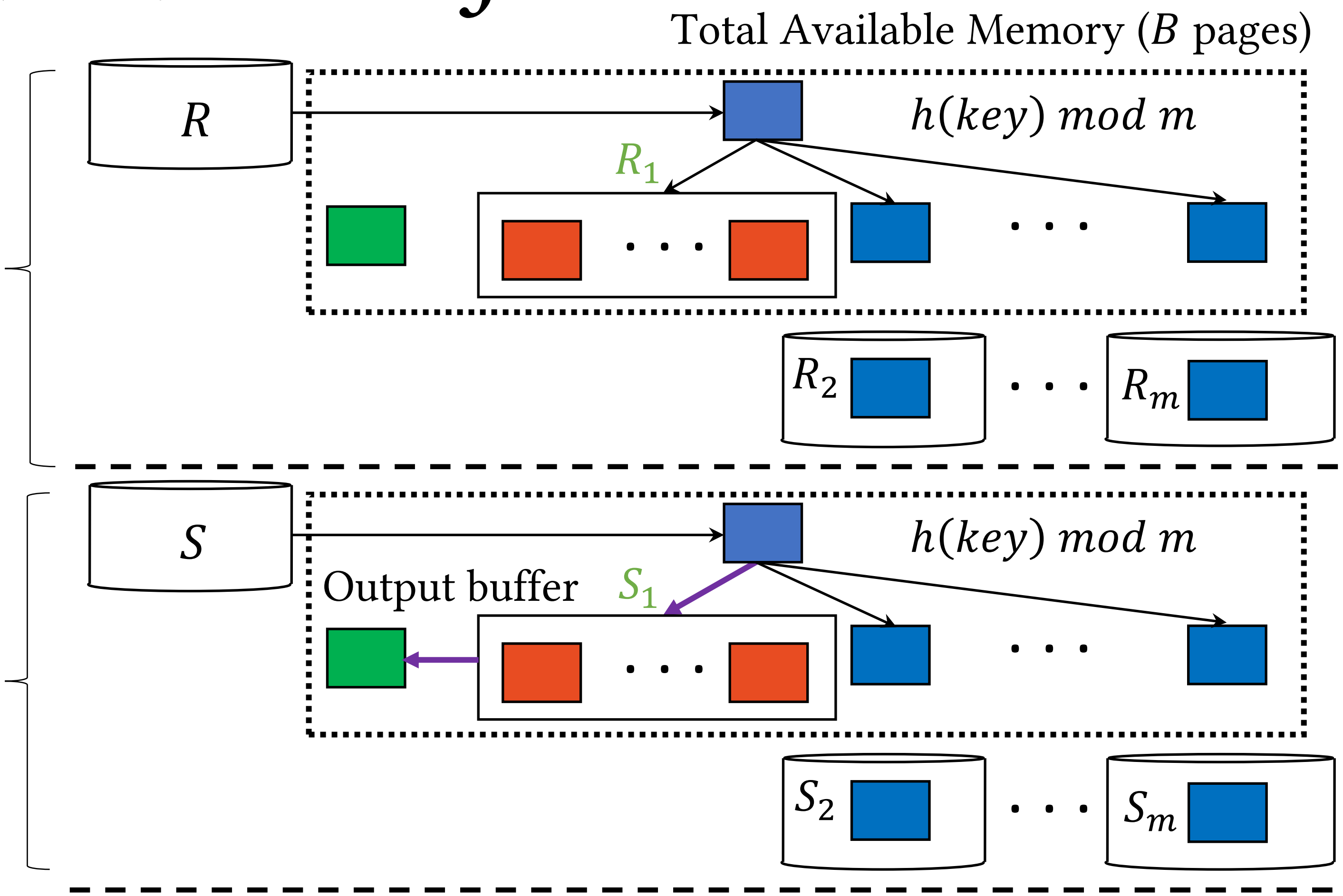
$R_j \bowtie S_j$ for each pair (R_j, S_j): $\left\lceil \frac{||R_j||}{B-2} \right\rceil \cdot ||S_j|| + ||R_j||$



State-of-the-art: Hybrid Hash Join

Partitioning R $2 \cdot ||R|| - ||R_1||$

Partitioning S $2 \times ||S|| - ||S_1||$



Probing $$\sum_{j=2}^{m}\left(\left\lceil\frac{||R_j||}{B-2}\right\rceil \cdot ||S_j|| + ||R_j||\right) = \sum_{j=2}^{m}(||S_j|| + ||R_j||)$$

Assuming $||R_j|| \leq B - 2$

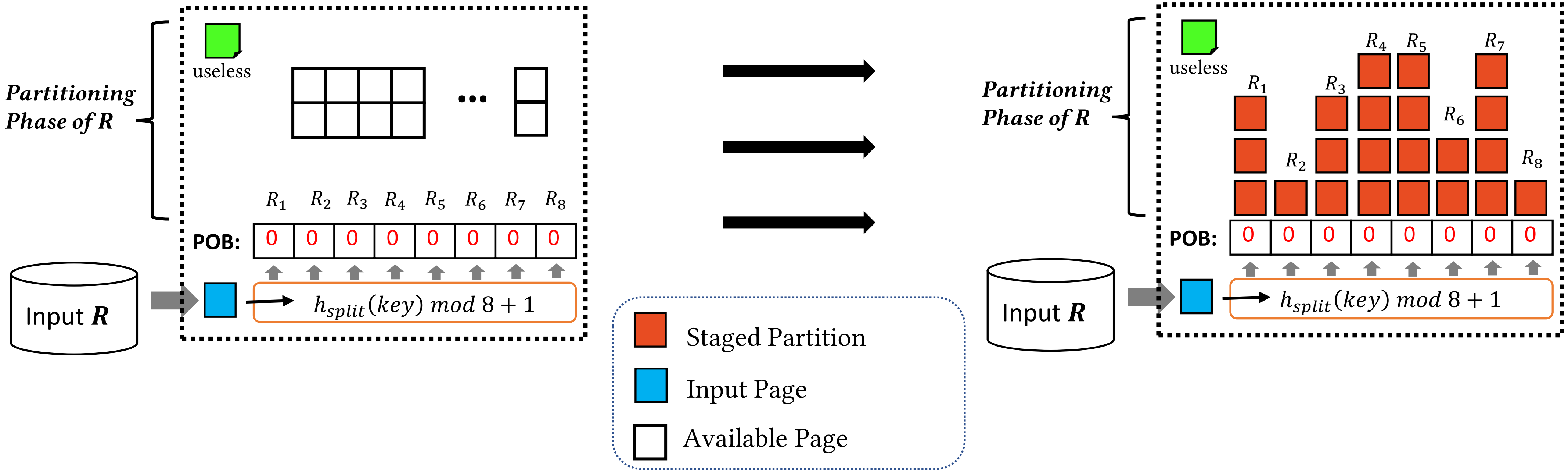
Dynamic Hybrid Hash Join (DHH)

State of the art DBs (e.g., PostgreSQL and AsterixDB) use DHH to decide which partitions are *staged.*

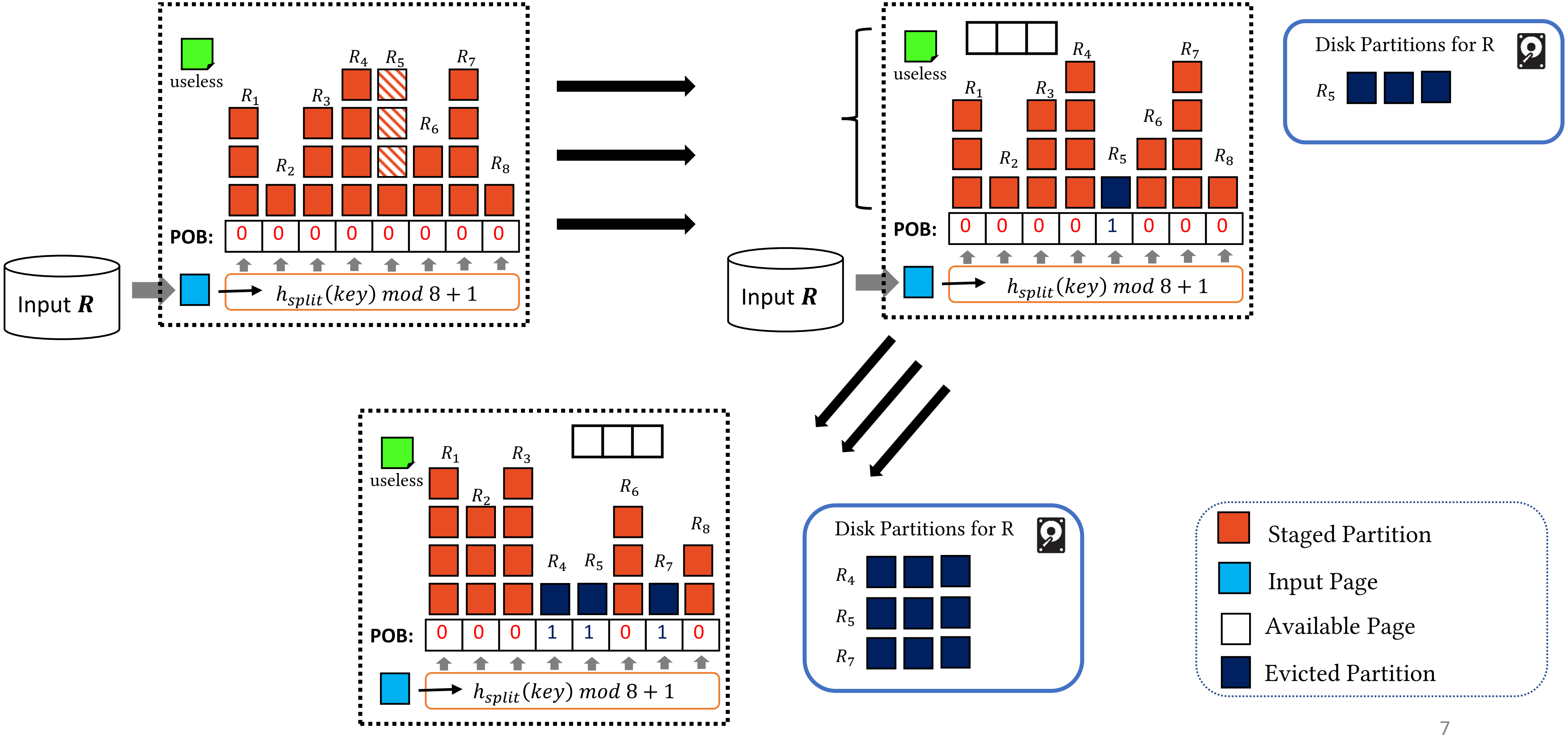
AsterixDB PostgreSQL

Example: Partitioning R (m = 8)

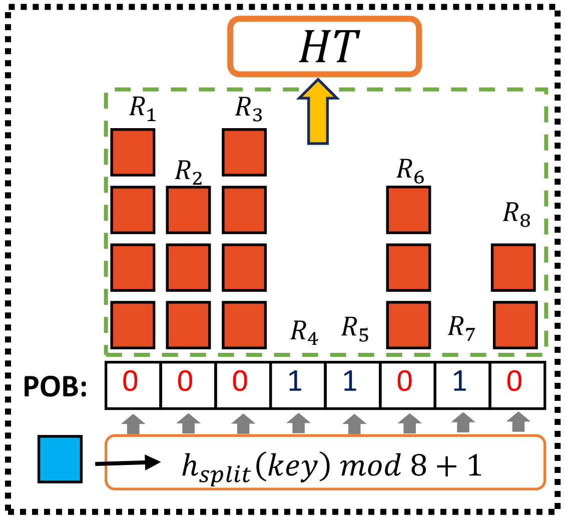
No available pages to use!

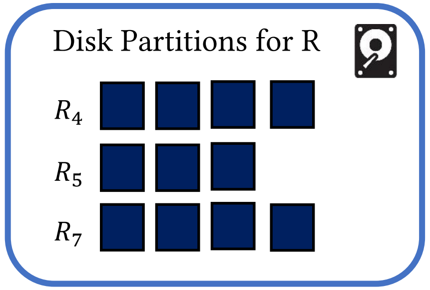


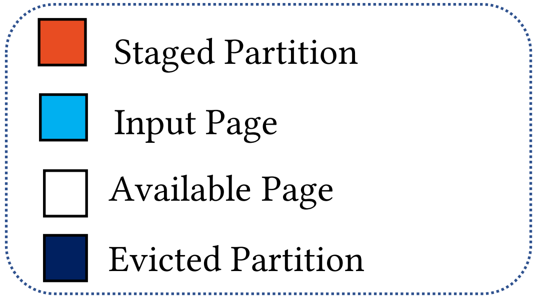
Partition R: Suppose we choose R_5 to evict



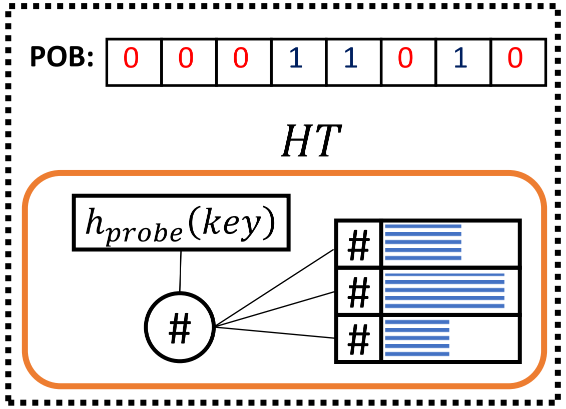
Partition R: Building a Hash Table (HT)





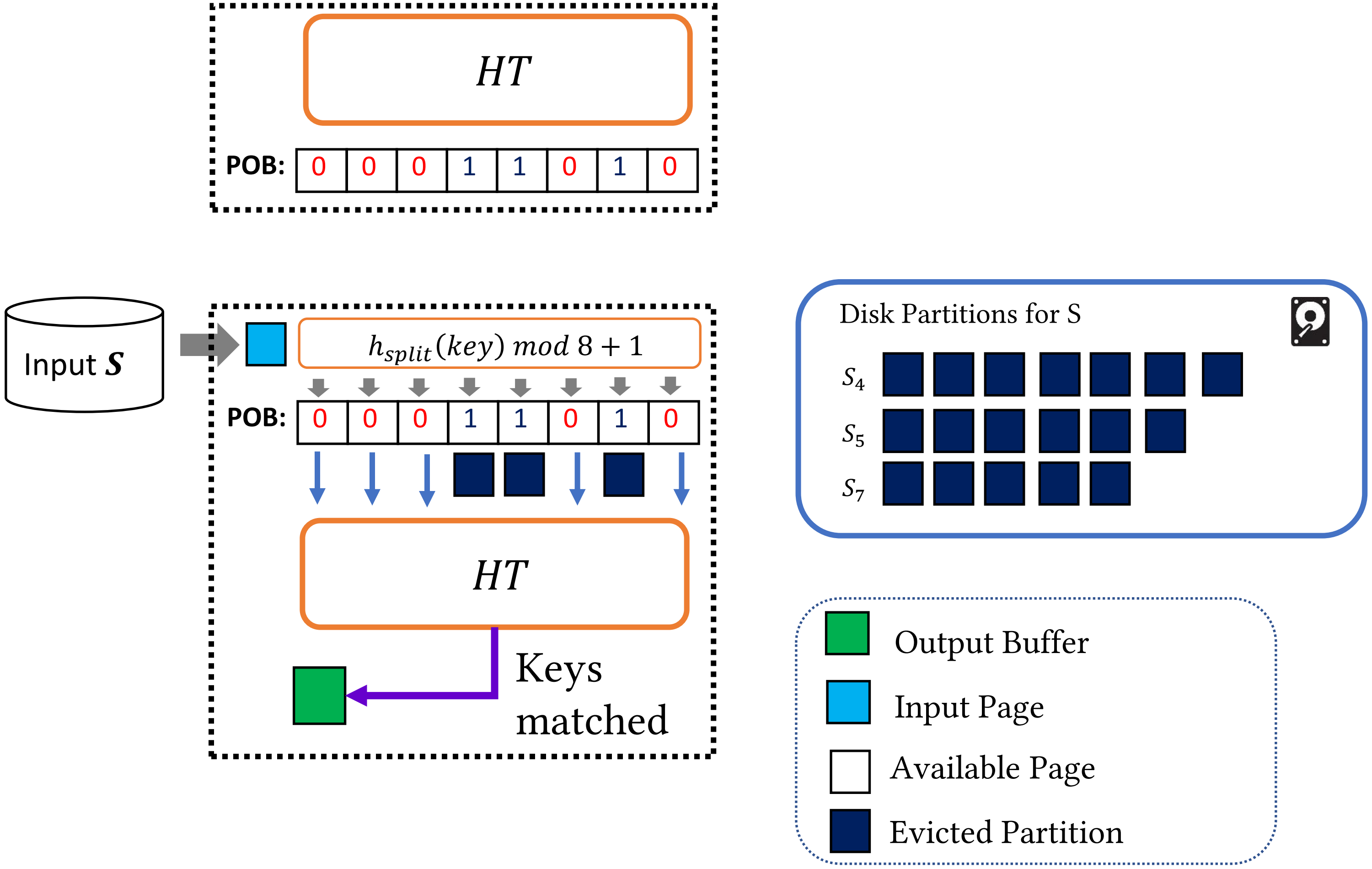


The final memory state after partitioning R:



I/O cost: $||R|| + ||R_4|| + ||R_5|| + ||R_7||$

Partition S and Probe



I/O cost (partitioning S):

$$||S|| + ||S_4|| + ||S_5|| + ||S_7||$$

I/O cost (partitioning R):

$$||R|| + ||R_4|| + ||R_5|| + ||R_7||$$

I/O cost (probing):

$$\sum_{j\in\{4,5,7\}}^{m} \left(\left\lceil \frac{||R_j||}{B-2} \right\rceil \cdot ||S_j|| + ||R_j|| \right)$$

In total (assuming $||R_j|| \leq B - 2$):

$$||R|| + ||S|| + \sum_{j\in\{4,5,7\}}^{m} 2 \cdot \left(||S_j|| + ||R_j||\right)$$

DHH Bridges between BNLJ and GHJ

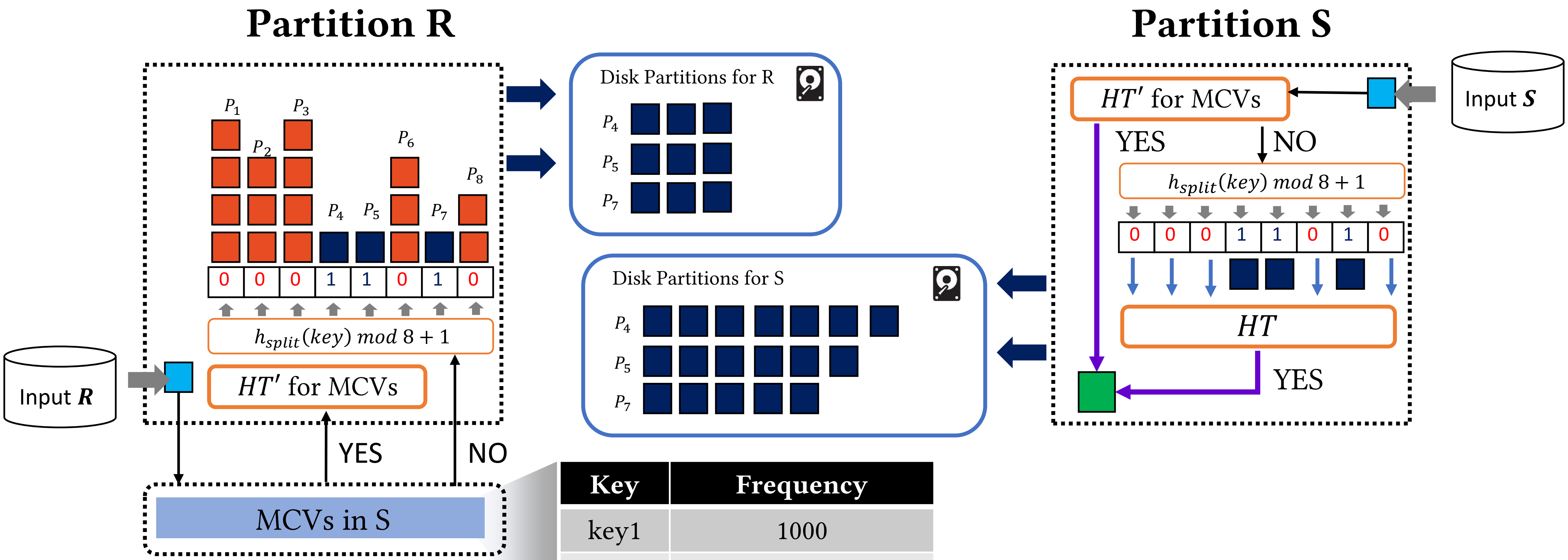
Method	I/O cost
BNLJ	$\|\|R\|\| + \|\|S\|\|$ when $\|\|R\|\| \leq B - 2$
GHJ	$3 \cdot (\|\|R\|\| + \|\|S\|\|)$ when $\|\|R_j\|\| \leq B - 2$
DHH	$\|\|R\|\| + \|\|S\|\| + 2 \cdot \sum_{j \in J}(\|\|R_j\|\| + \|\|S_j\|\|)$ when $\|\|R_j\|\| \leq B - 2$

J represents the ids of partitions that are spilled to the disk

Can we do better?

Skew Optimization: Stage Most-Common-Values (MCVs) to reduce $||S_j||$

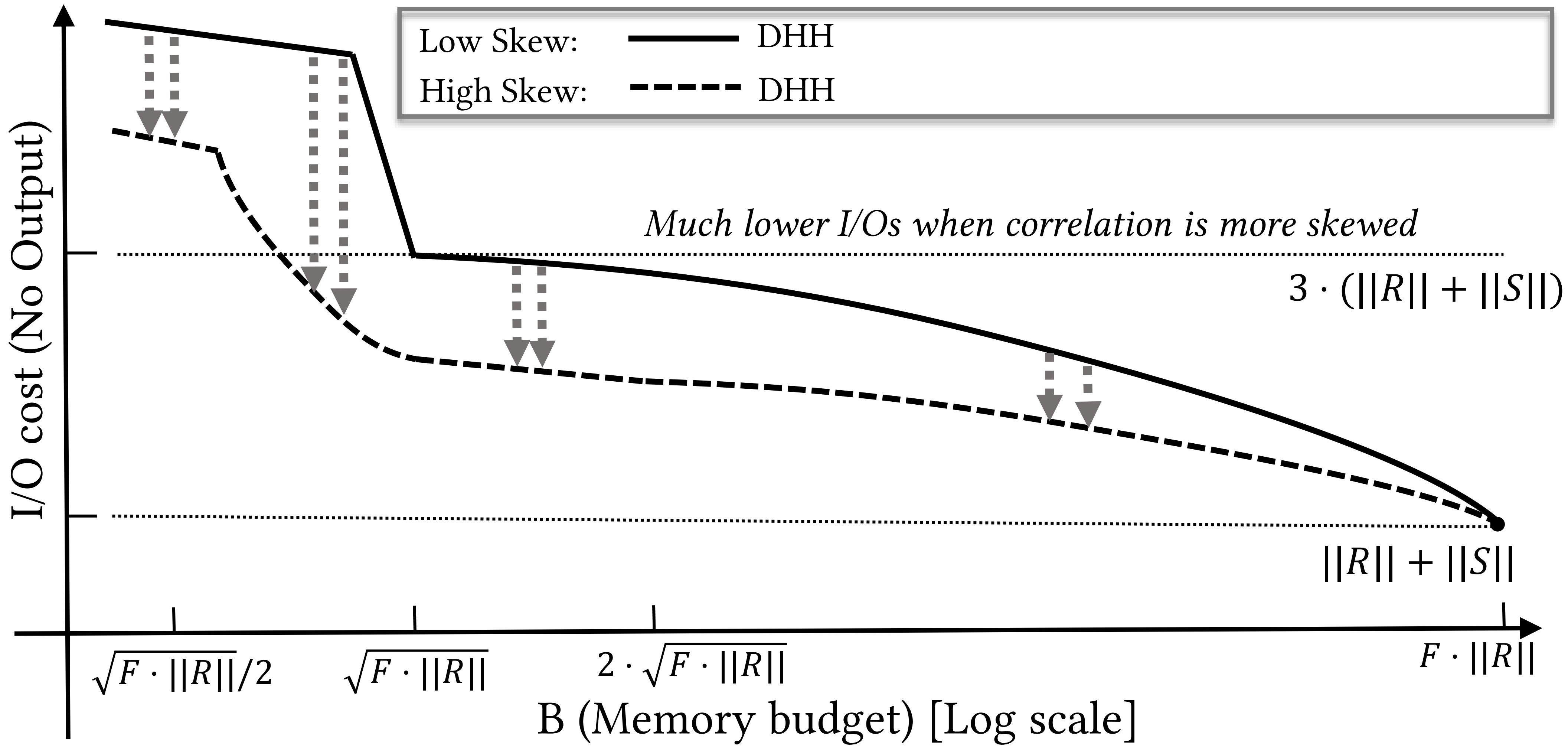
Skew Optimization in DHH



Key	Frequency
key1	1000
…	…
keyn	500

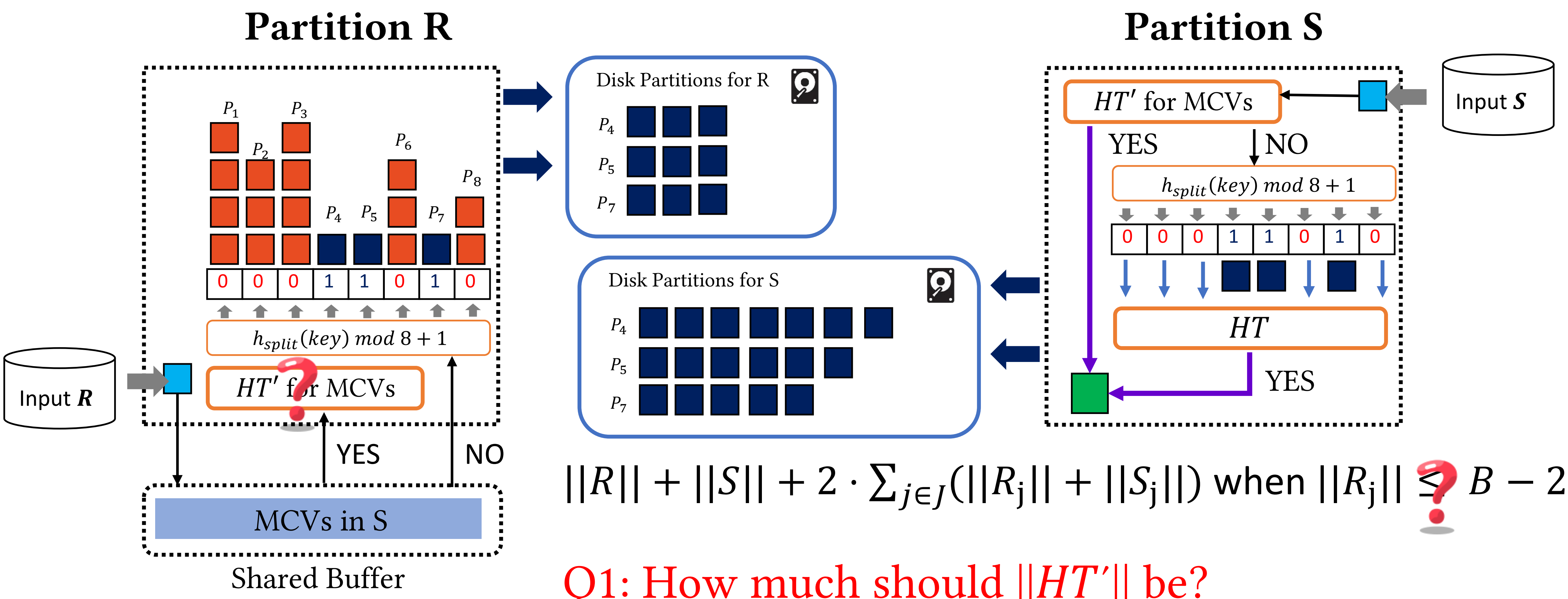
Correlation Table (CT)

Skew Optimization in DHH



Skew optimization reduces the number of I/Os when the matching exhibits skew

Can we do better?

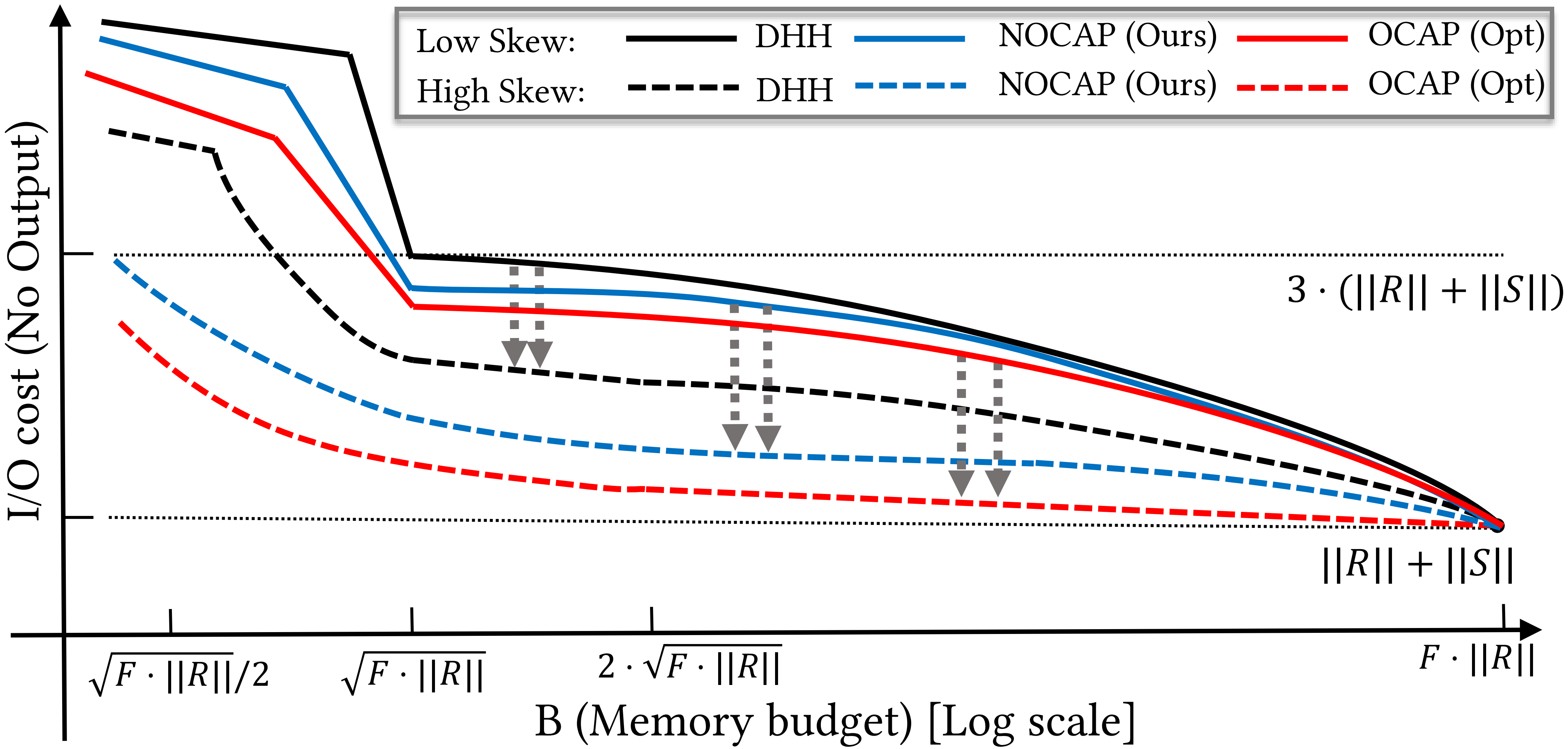


$||R|| + ||S|| + 2 \cdot \sum_{j \in J}(||R_j|| + ||S_j||)$ when $||R_j|| \leq B - 2$

Q1: How much should $||HT'||$ be?

Q2: What if $||R_j|| > B - 2$?

DHH v.s. Instance-Optimal Join (OCAP)



A good partitioning algorithm should be skew-aware and adaptive to the given memory

Modeling the Join Cost of DHH

Recall DHH Join Cost: $||R|| + ||S|| + \sum_{j \in J}\left(\left(\left\lceil\frac{||R_j||}{B-2}\right\rceil + 1\right) \cdot ||S_j|| + 2 \cdot ||R_j||\right)$

J represents the ids of partitions that are spilled to the disk

Define a $n \times (m+1)$ Boolean matrix P to represent the partitioning assignment

Notation	Meaning
$n\ (n_R)$	The number of tuples in relation R
m	The number of partitions on disk
$P = \begin{bmatrix} 0 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix}_{n \times (m+1)}$	A Boolean matrix P where $P_{i,j} = 1$ represents the i^{th} record belongs to the j^{th} partition
R_1	A partition cached in memory

$$\underset{P,m}{arg\ \min} \sum_{j=2}^{m+1}\left(\left(\left\lceil\frac{||R_j||}{B-2}\right\rceil + 1\right) \cdot ||S_j|| + 2 \cdot ||R_j||\right)$$

$$\text{s.t.}\ \ \forall i \in [n],\ \sum_{j=1}^{m+1} P_{i,j} = 1$$

$$||R_1|| +\ m + 2 \le B$$

$$P_{i,j} \in \{0,1\}, \forall i \in [n], \forall j \in [m+1]$$

Integer Programming

$$\underset{P,m}{arg\ \min} \sum_{j=2}^{m+1} \left(\left(\left\lceil \frac{||R_j||}{B-2} \right\rceil + 1 \right) \cdot ||S_j|| + 2 \cdot ||R_j|| \right)$$

s.t. $\forall i \in [n],\ \sum_{j=1}^{m+1} P_{i,j} = 1$

$$||R_1|| + m + 2 \leq B$$

$$P_{i,j} \in \{0,1\}, \forall i \in [n], \forall j \in [m+1]$$

Index i	Frequency in S
1	1
…	…
n-1	77
n	100

Correlation Table (CT)

$$||R_j|| = \sum_{i=1}^{n} P_{i,j} / b_R$$

$$||S_j|| = \sum_{i=1}^{n} P_{i,j} \cdot CT[i] / b_S$$

Instance-Optimal Join (Optimal Correlation-Aware Partitioning)

Input: n, B, b_R, b_S, CT

Output: P, m

Exponential searching space to enumerate all possible partitions!

Three Properties of P_{opt} to Reduce Complexity

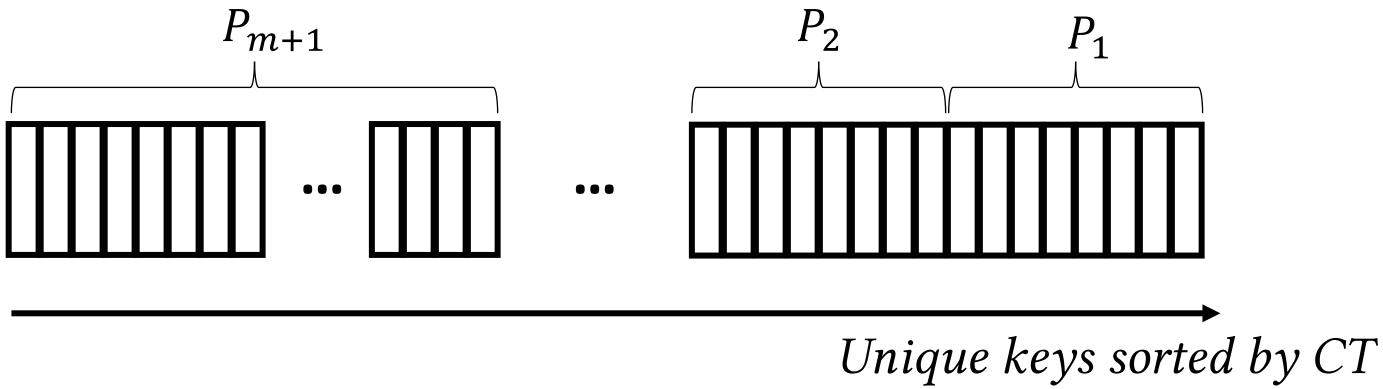
Consecutiveness $O(B^{n+1}) \Rightarrow O(B^2 \cdot n^2)$

Monotonicity $O(B^2 \cdot n^2) \Rightarrow O(n^2 \cdot B \cdot \log B)$

Divisibility $O(n^2 \cdot B \cdot \log B) \Rightarrow O(n^2 \cdot \log B / B)$

Consecutiveness

Theorem 1 Given an arbitrary sorted CT array, there is an optimal partitioning $P_{opt} = \langle P_1, P_2, \ldots, P_{m+1} \rangle$ where for any $i_1 \leq i_2$, if $i_1 \in P_j$ and $i_2 \in P_j$, we have $i \in P_j$ for any $i \in [i_1, i_2]$.



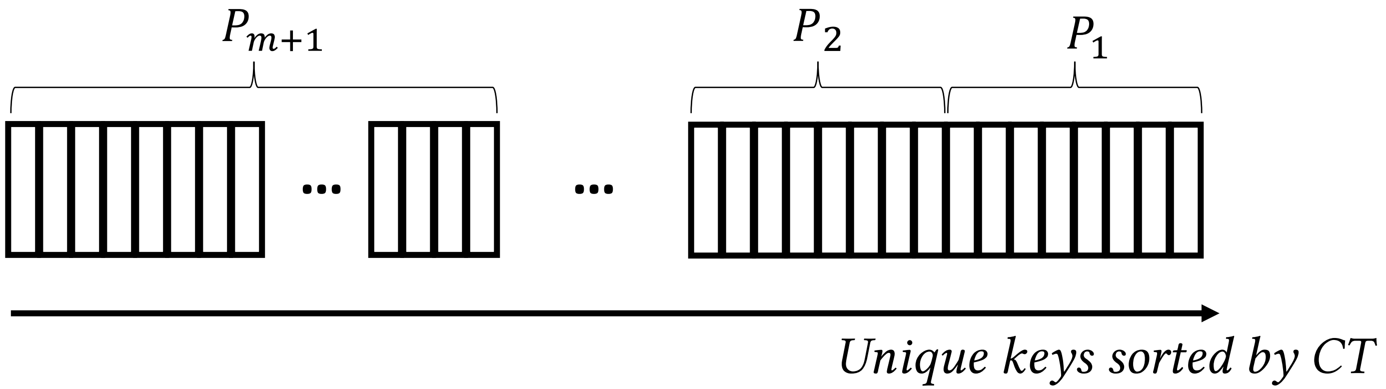
Index i	Frequency in S
1	1
…	…
n-1	77
n	100

Correlation Table (CT)

Monotonicity

Theorem 2 Given an arbitrary sorted CT array, there is an optimal partitioning $P_{opt} = \langle P_1, P_2, \dots, P_{m+1} \rangle$ **from Theorem 1** where $\left\lceil \frac{||R_{m+1}||}{B-2} \right\rceil \geq \left\lceil \frac{||R_m||}{B-2} \right\rceil \geq \cdots \geq \left\lceil \frac{||R_2||}{B-2} \right\rceil \geq \left\lceil \frac{||R_1||}{B-2} \right\rceil$.

R_j is a group of records from relation R while P_j is a group of keys



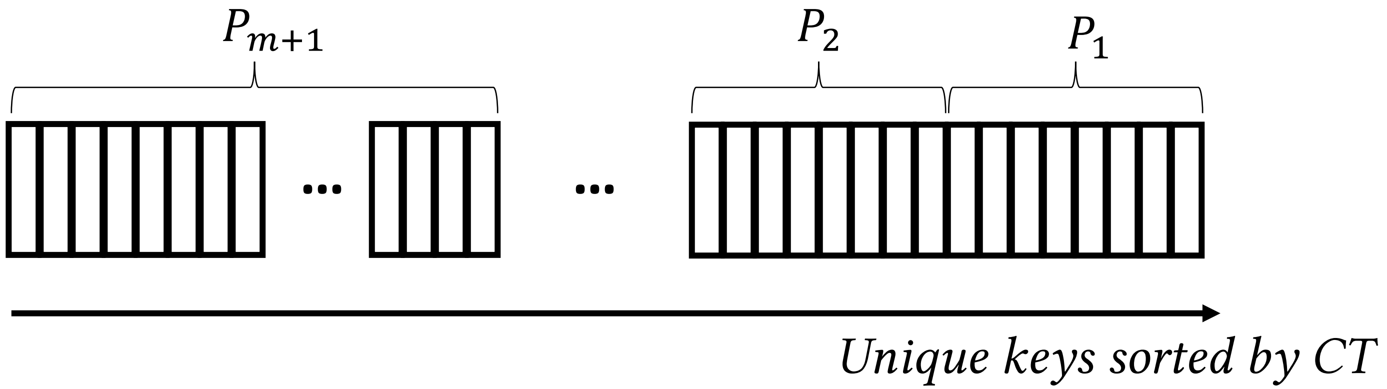
Index i	Frequency in S
1	1
…	…
n-1	77
n	100

Correlation Table (CT)

Divisibility

Theorem 3 Given an arbitrary sorted CT array, there is an optimal partitioning $P_{opt} = \langle P_1, P_2, \ldots, P_{m+1} \rangle$ **from Theorem 2** where $||R_j||$ is divisible by $B-2$ for $j \in [2, m]$.

$\left\lceil \frac{||R_{m+1}||}{B-2} \right\rceil \geq \left\lceil \frac{||R_m||}{B-2} \right\rceil \geq \cdots \geq \left\lceil \frac{||R_2||}{B-2} \right\rceil \geq \left\lceil \frac{||R_1||}{B-2} \right\rceil$ from Theorem 2



Index i	Frequency in S
1	1
…	…
n-1	77
n	100

Correlation Table (CT)

Practical Challenges for OCAP

1. *We cannot have the whole CT in practice*

Index i	**Frequency in S**
1	1
…	…
10M	1000

2. *Partitioning assignment also occupies memory*

$$P = \begin{bmatrix} 0 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix}_{n\times(m+1)}$$

Inspirations from OCAP

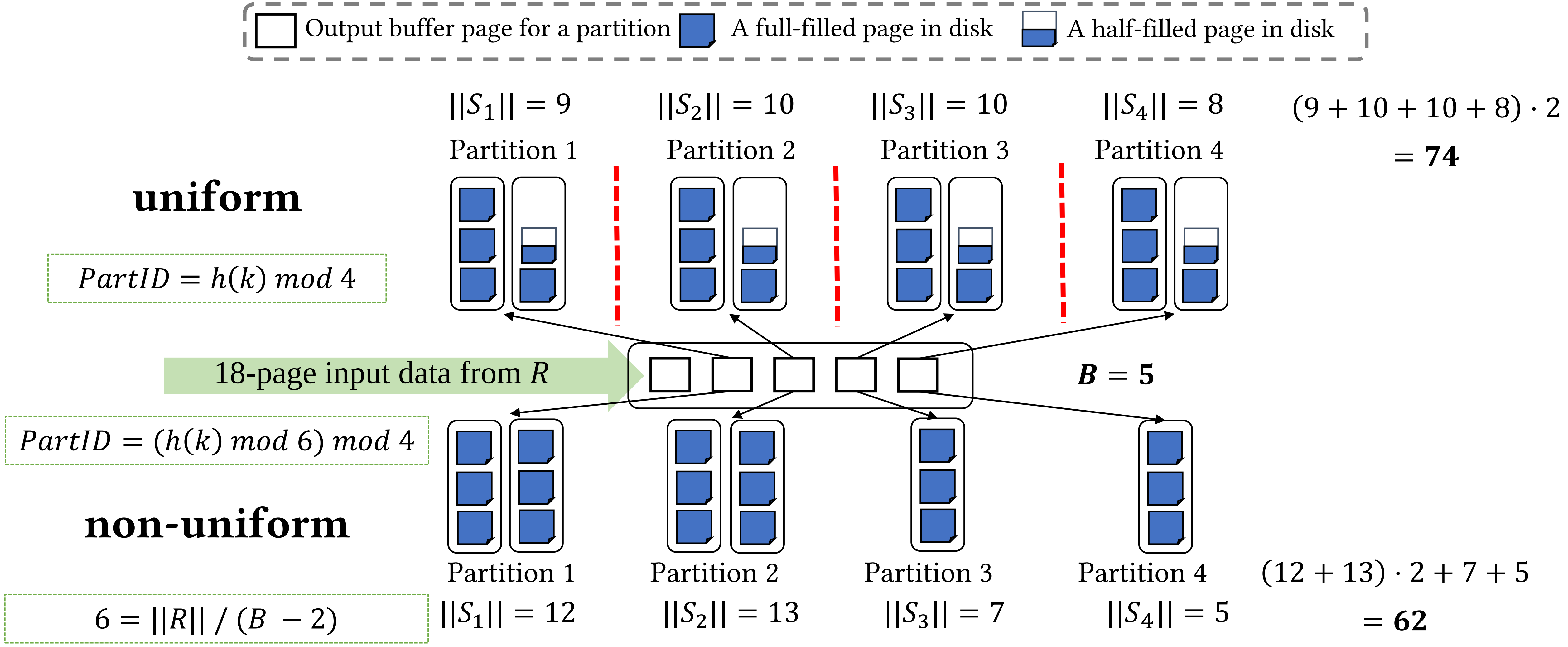
Consecutiveness and Monotonicity: $\lceil\frac{||R_{m+1}||}{B-2}\rceil \geq \cdots \geq \lceil\frac{||R_1||}{B-2}\rceil$ *for sorted CT*

⇒ We can prioritize MCVs in *two* ways: build an in-memory hash table (if B is large) or assign them into a small partition on disk (if B is small)

Divisibility: On-disk partitions should be mostly divisible by $B-2$

⇒ We should ensure on-disk partitions fulfill $z \cdot (B-2)$ pages ($z \in Z^{+}$)

Divisibility when Partitioning Data on Disk



Prioritizing MCVs with Constrained Memory

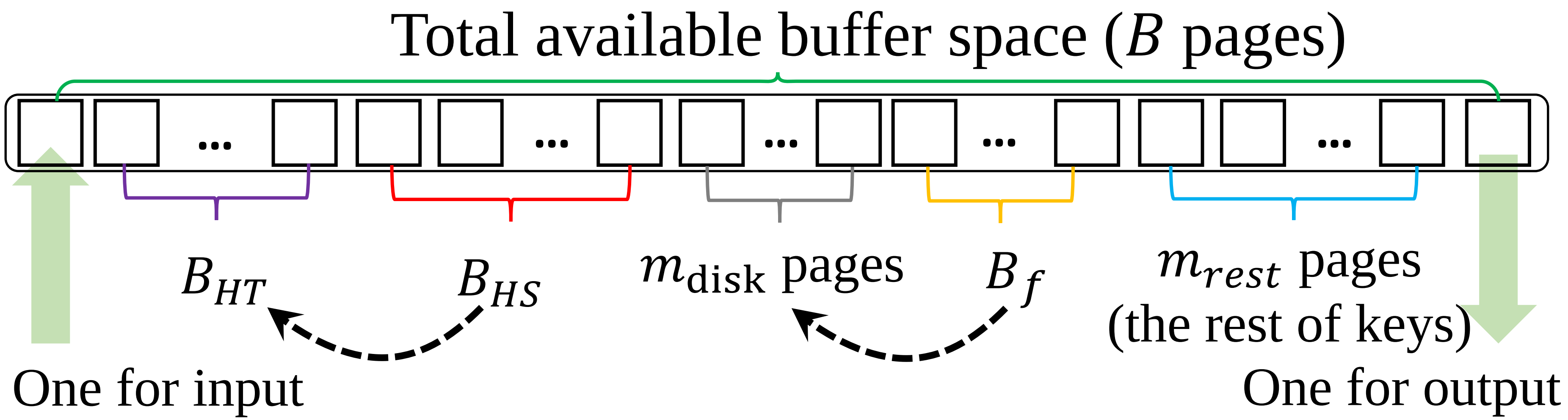
Index i	Frequency in S
1	1
…	…
n-k+1	73
n-k+2	73
…	…
n-1	100
n	100

We only have top-k frequent keys (MCVs)

Run augmented DHH with memory budget m_{rest}

Run OCAP with some memory budget $m = B - 2 - m_{rest}$

A hash set for keys with high CT	HS
A hash table to store the whole record of the hash set	HT
A hash map for keys to be partitioned	f
Several write-buffer pages for partitioning	m_{disk}

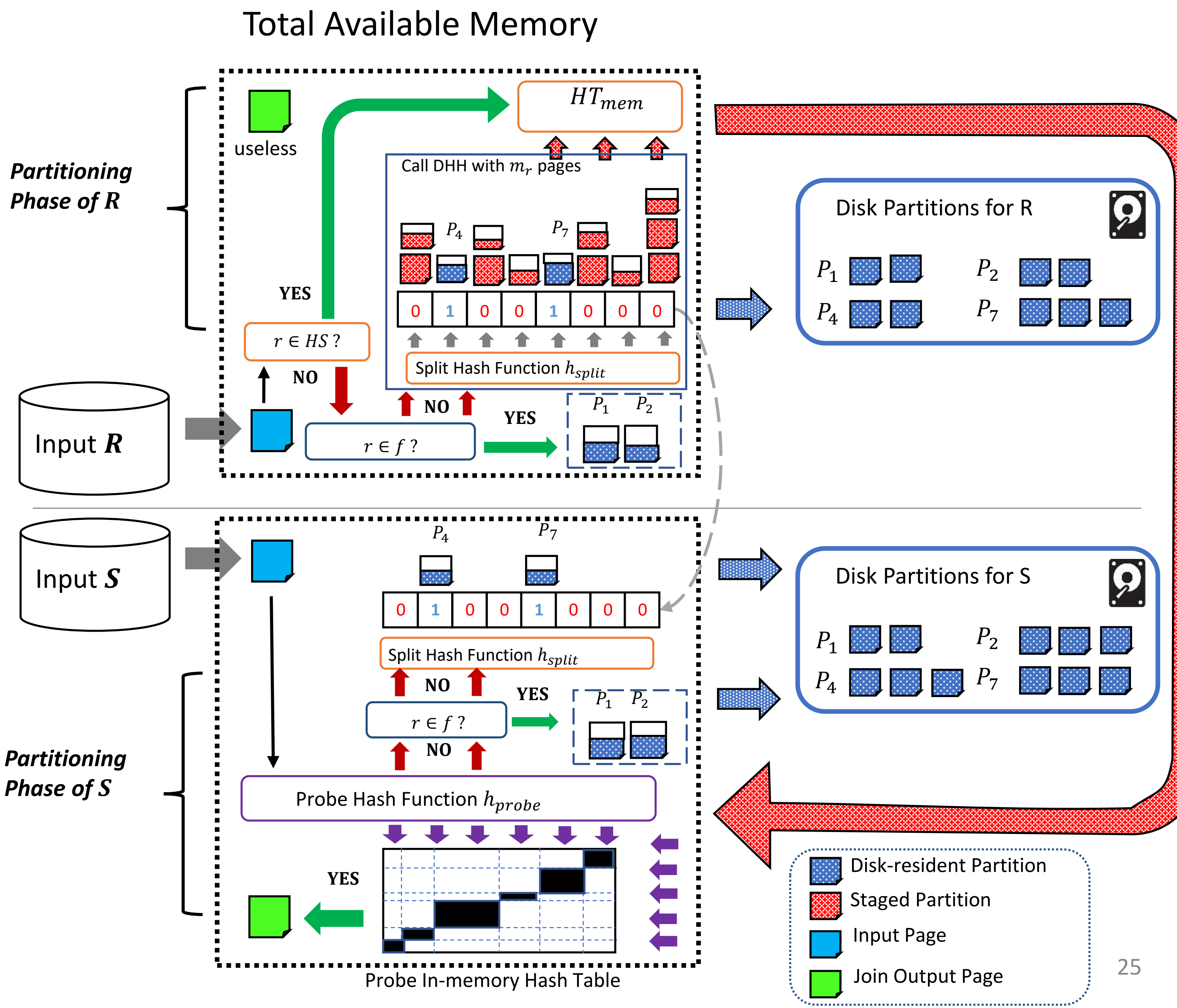


NOCAP

Partitioning Workflow:

***OCAP** for top-k' frequent keys*

***DHH** to partition the rest*



Experiment Setup

Storage: PCIe P4510 SSD

Measured read/write symmetry:

random_write_latency/sequential_read_latency = 3.3

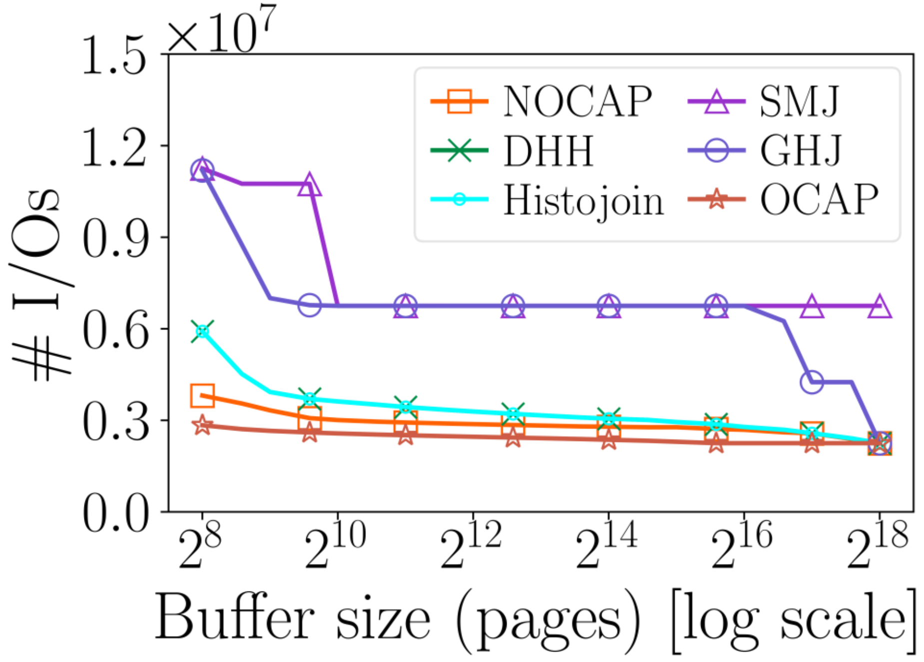
sequential_write_latency/sequential_read_latency = 3.2

PK-FK join input size: 1M #records join with 8M #records

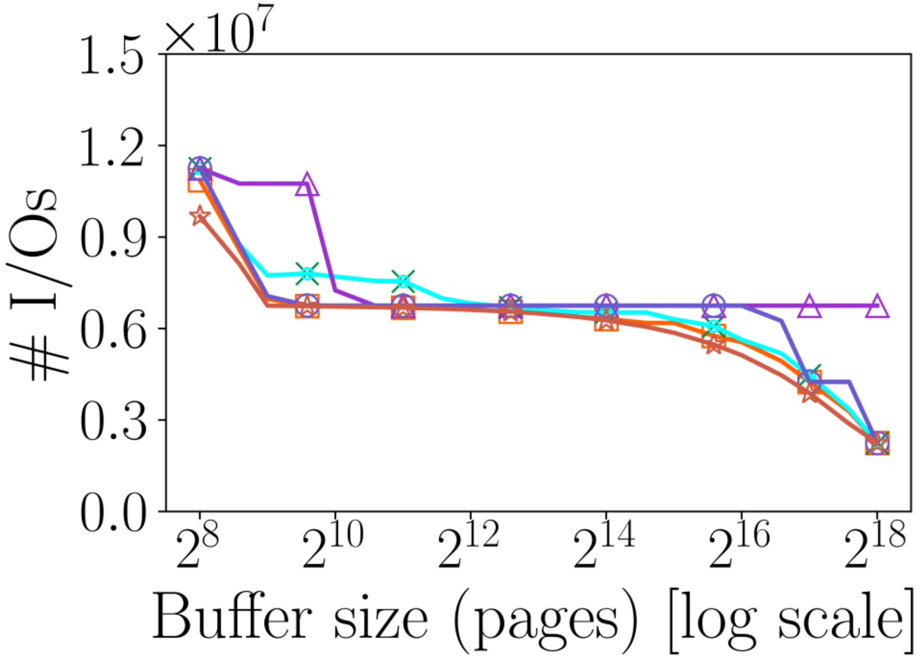
Record size: 1KB per record

Page size: 4KB

Selected Experimental Results



Zipfian (α = 1.3)

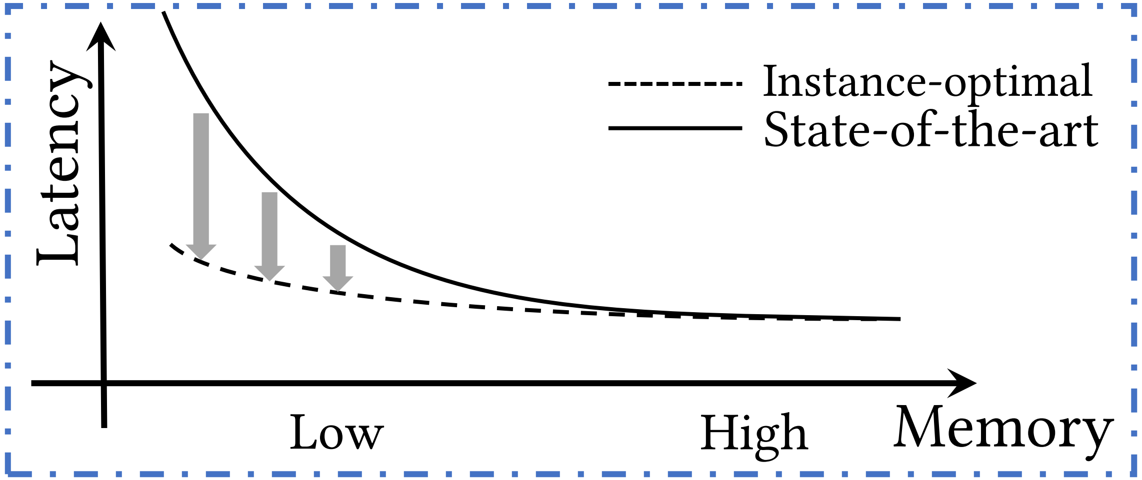


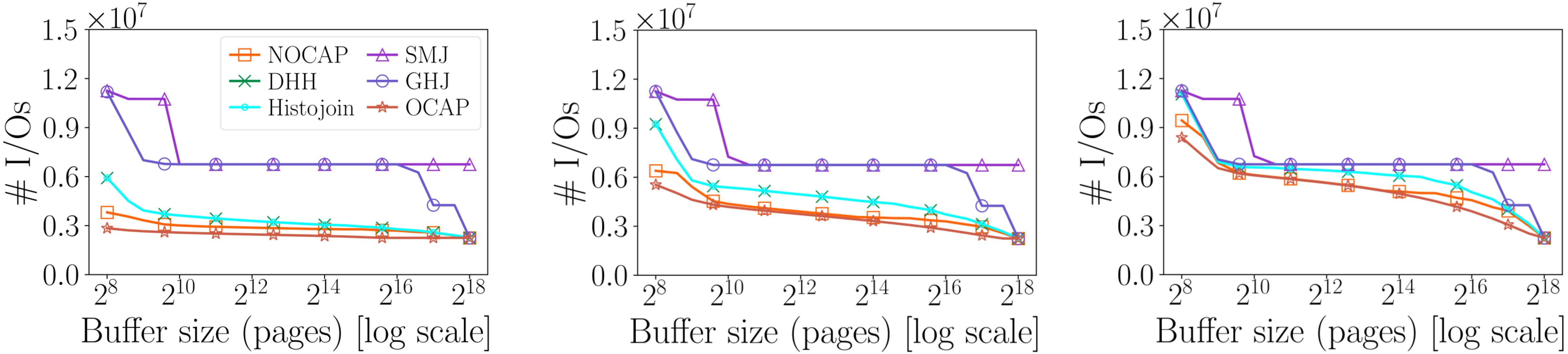
Uniform

Correlation-aware joins (**DHH, Histojoin, and NOCAP**) can **adaptively** reduce I/O cost when it comes to a **skew** distribution.

Note: OCAP only represents a lower bound, not a practical algorithm

Varying skew



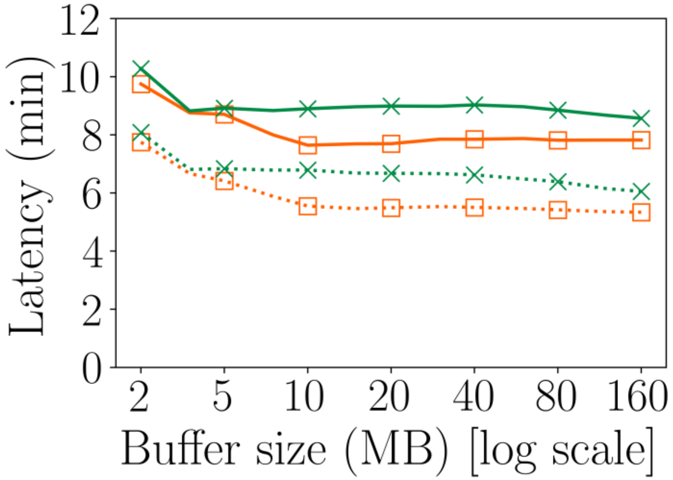


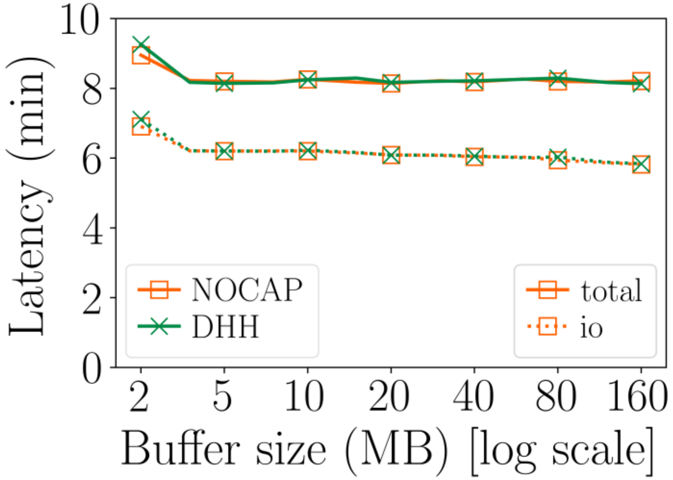
Zipfian (α = 1.3) Zipfian (α = 1.0) Zipfian (α = 0.7)

While DHH helps reduce #I/Os, **NOCAP** can better exploit the correlation skew to **achieve even lower I/O cost**.

Other datasets (JCC-H and JOB)

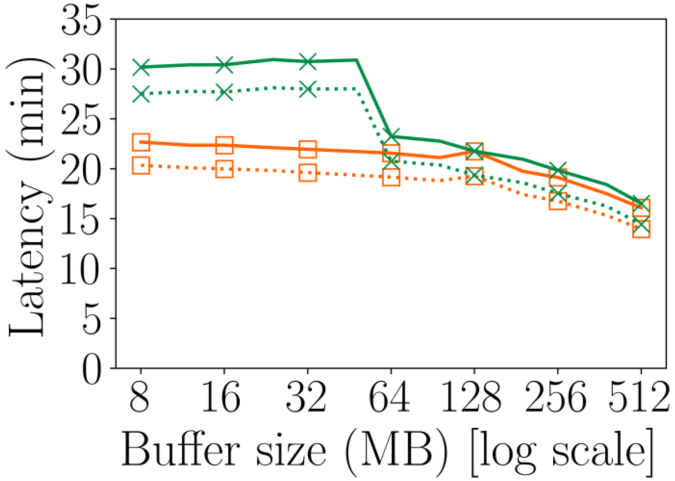
JCC-H SF=10
(Tuned Skew)
with Revised Q12

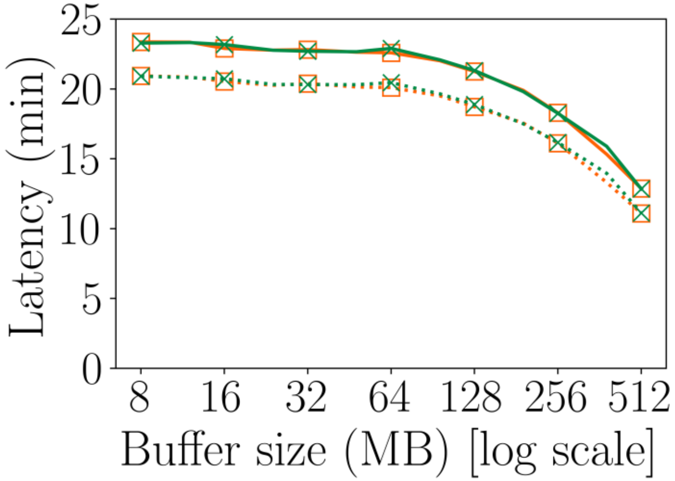




JCC-H SF=10
(Original Skew)
with Revised Q12

JOB
(cast_info ⋈ title)



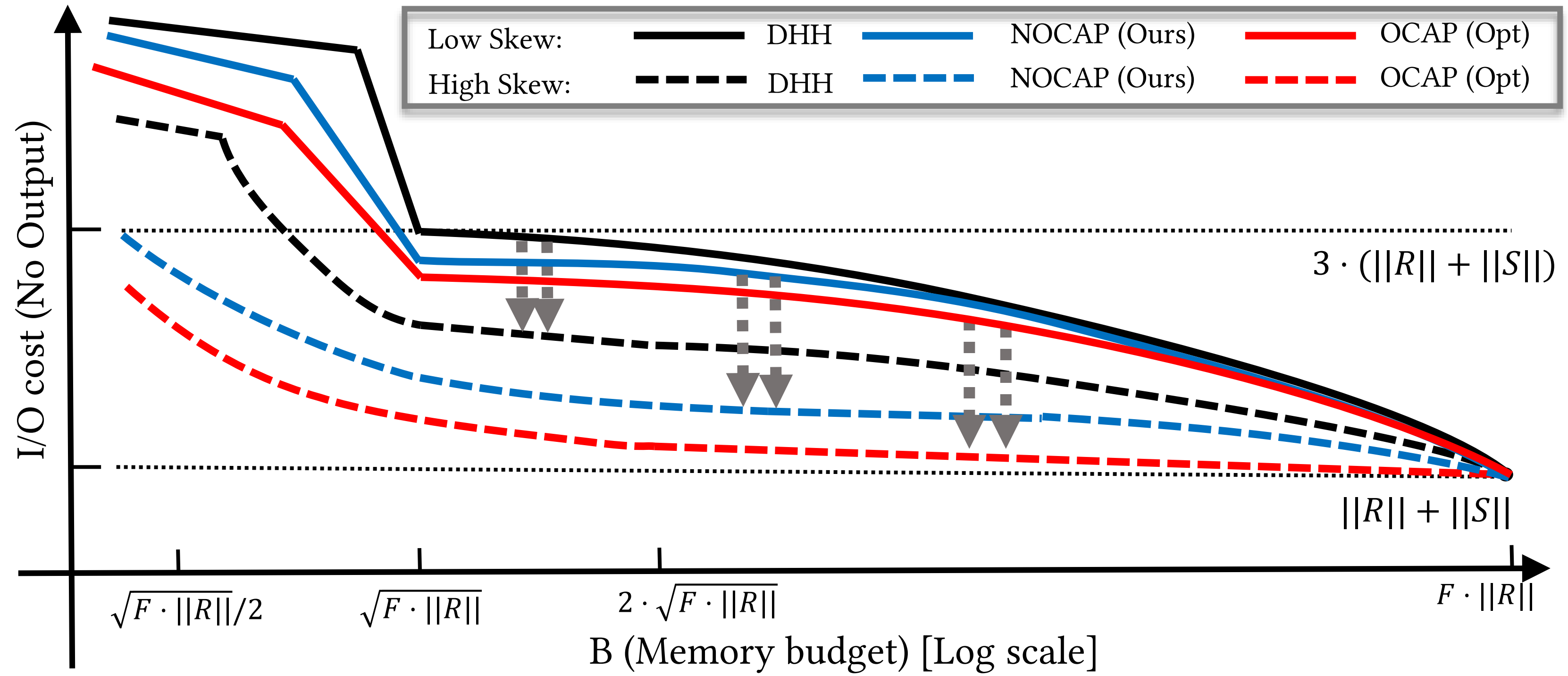


JOB
(cast_info ⋈name)

While DHH occasionally performs as close as NOCAP,
NOCAP is more adaptive when the workload varies.

Summary of NOCAP

NOCAP join outperforms DHH by up to 30%, and the textbook GHJ by up to 4X. Even for uniform distribution, NOCAP outperforms DHH by up to 10%!



BOSTON UNIVERSITY

CS 561: Data Systems Architectures

Class 20

Correlation-Aware Partitioning for Joins

Zichen Zhu

https://bu-disc.github.io/CS561/