HotStorage'18

BIBIM: A Prototype Multi-Partition Aware Heterogeneous New Memory

Gyuyoung Park [1], ***Miryeong Kwon*** [1], *Pratyush Mahapatra* [2], *Michael Swift* [2], *and Myoungsoo Jung* [1]

Yonsei University

Computer Architecture and MEmory Systems Lab





CAMELab.org
Computer Architecture and Memory Systems Laboratory



THE UNIVERSITY of WISCONSIN MADISON

1. Demands for Hybrid Memory

2. Preview of BIBIM

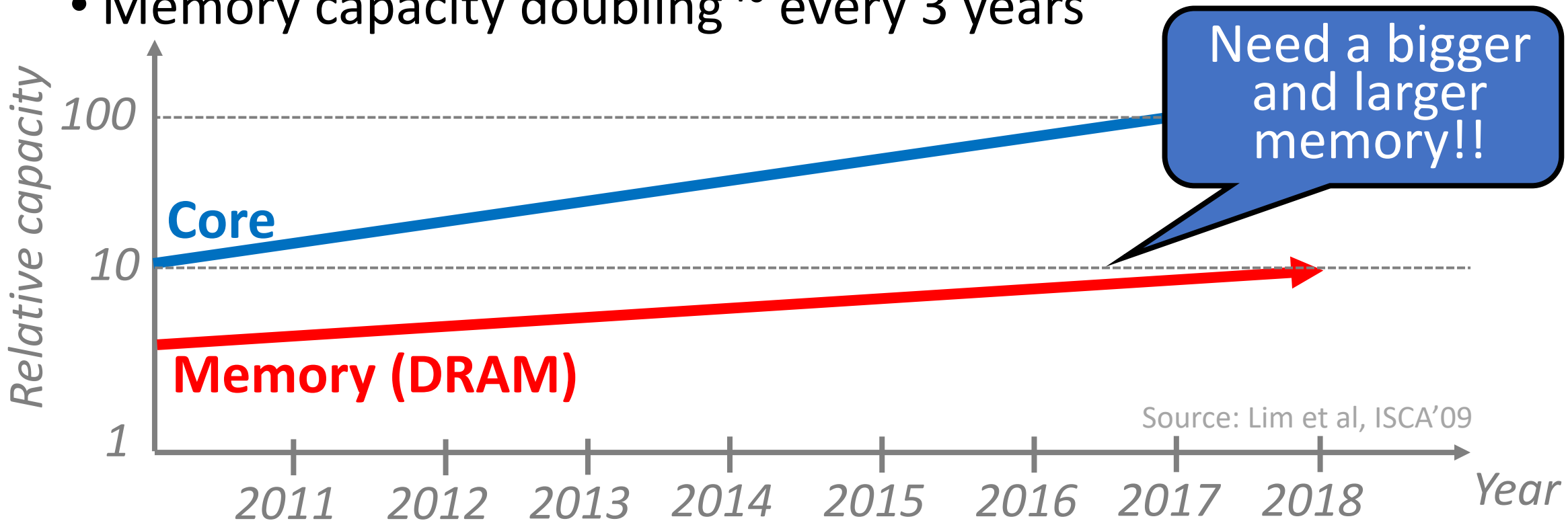
3. Design Considerations of BIBIM

4. Example of BIBIM Operation

5. Evaluation

Increasing Demands for Bigger Memory

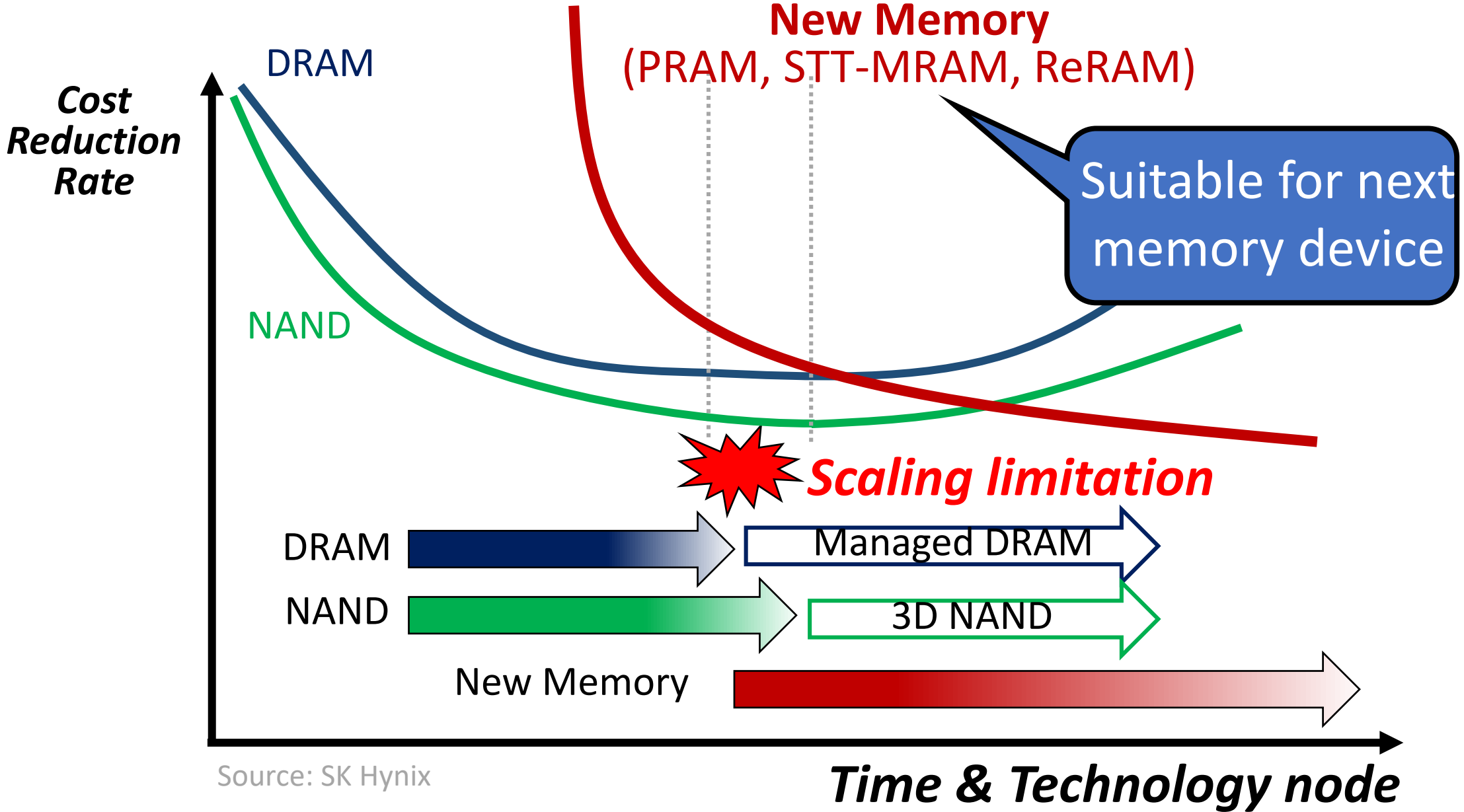
- More cores require more memory capacity
  - However, there is memory capacity gap
  - Core count doubling ~ every 2 years
  - Memory capacity doubling ~ every 3 years



Moreover, there are many data-intensive applications

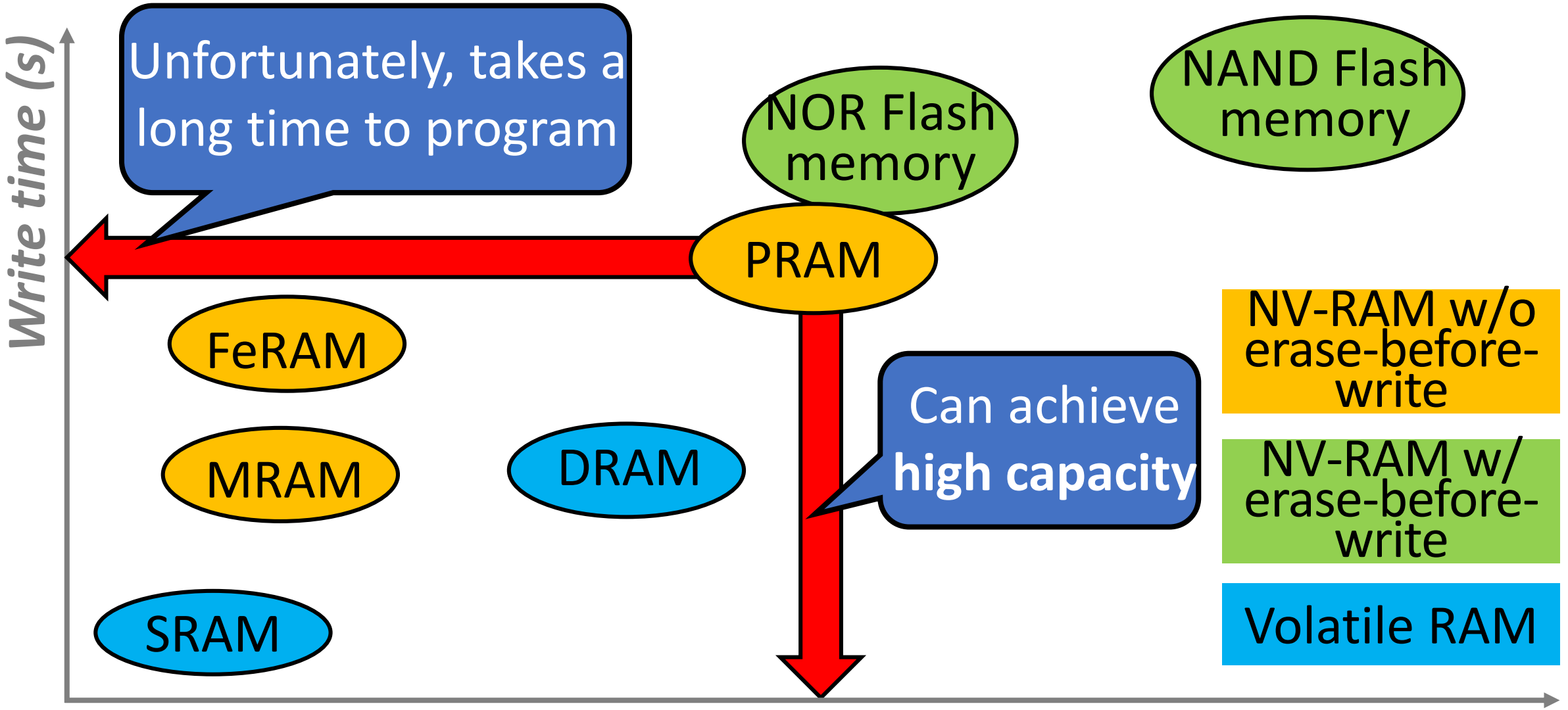
Ex) Memory caching (MEMCACHED, Redis), In-memory database (DynamoDB, HANA)

Comparison of Technology Scaling



Source: SK Hynix

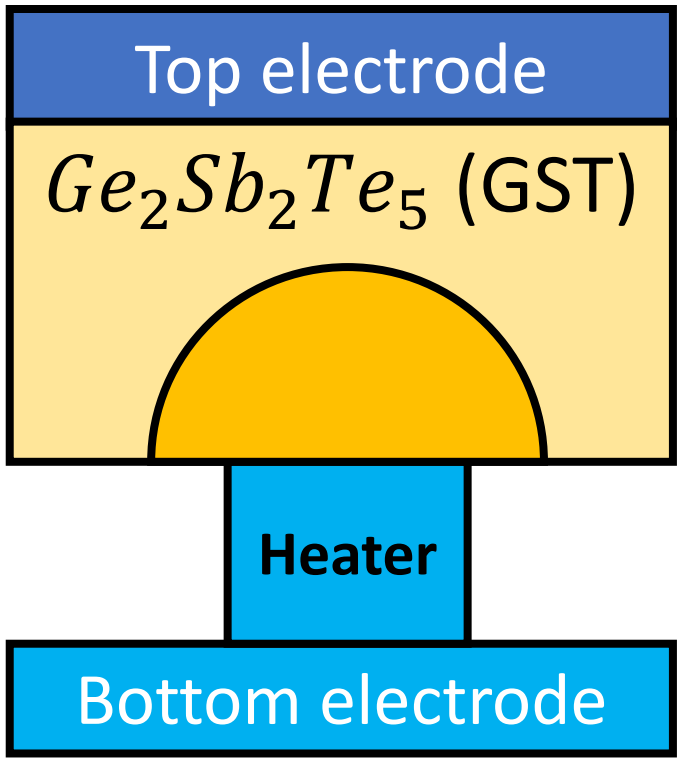
Comparison of Programming Latency



Courtesy: Motoyuki Ooishi



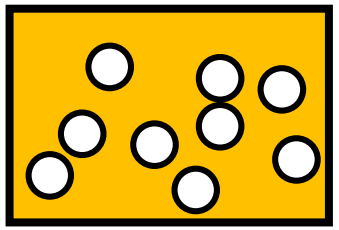
Why does PRAM Show Long Write Latency? How Significant?



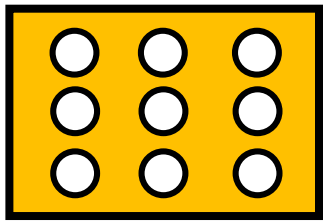
PRAM cell

PRAM exploits the unique behavior of chalcogenide glass (GST). By switching status of GST, it can store data

0 → 1

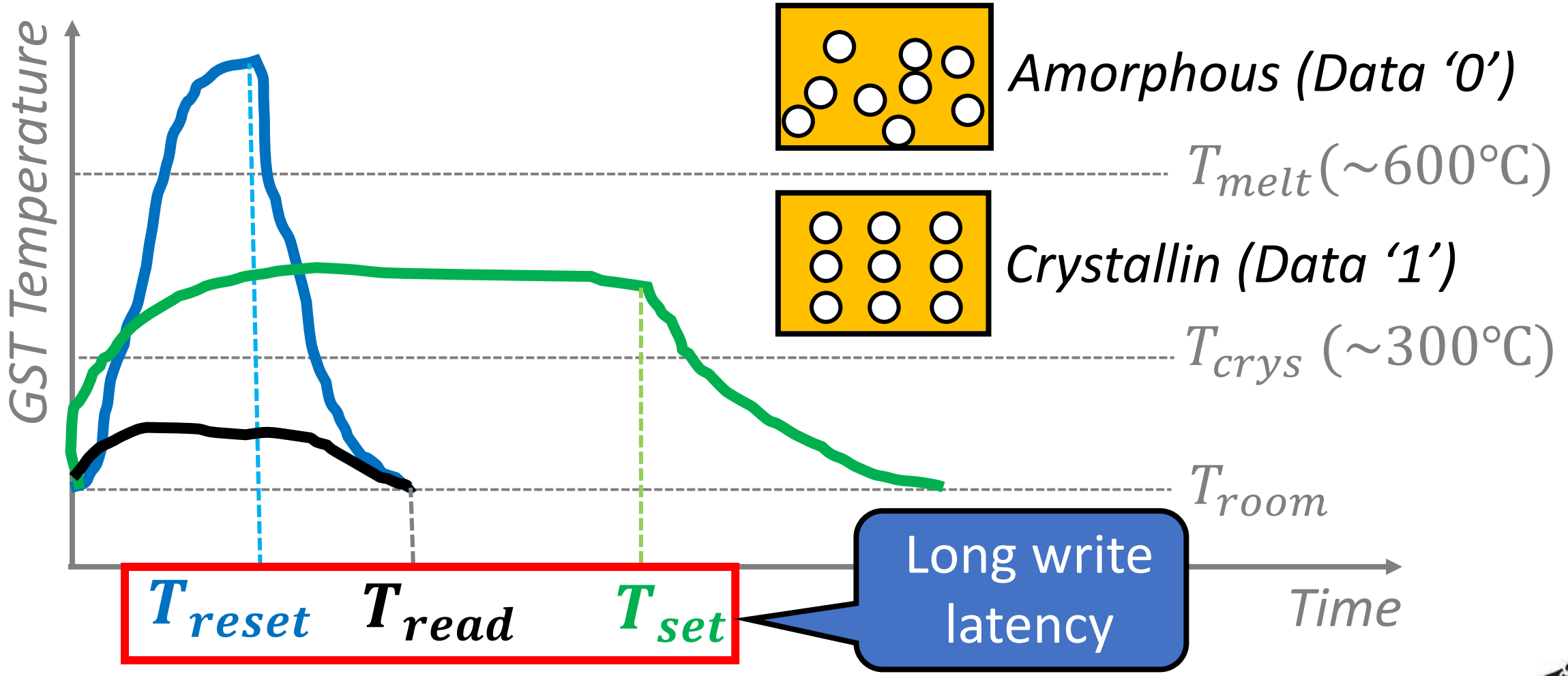


1 → 0



Programming Method of PRAM

GST status is changed by heating the material



Moreover, compared with DRAM, PRAM has **asymmetric** latency

General Assumptions of PRAM Latency

Architecting Phase Change Memory as a Scalable DRAM Alternative

Benjamin C. Lee† Engin Ipek†

†Computer Architecture Group
Microsoft Research
Redmond, WA
{blee, ipek, dburger}@microsoft.com

ABSTRACT

Memory scaling is in jeopardy as charge storage and sensing mechanisms become less reliable for prevalent memory technologies, such as DRAM. In contrast, phase change memory (PCM) storage relies on scalable current and thermal mechanisms. To exploit PCM's scalability as a DRAM alternative, PCM must be architected to address relatively long latencies, high energy writes, and finite endurance.

ISCA'09

HPCA'18

Crash Consistency in Encrypted Non-Volatile Main Memory Systems

Sihang Liu[1], Aasheesh Kolli[2,3], Jinglei Ren[4], and Samira Khan[1]

[1]University of Virginia [2]VMware Research [3]Pennsylvania State University [4]Microsoft Research

Abstract

Non-Volatile Main Memory (NVMM) systems provide high performance by directly manipulating persistent data in-memory, but require crash consistency *support to recover data*

DRAM [12]. These new memory technologies are blurring the difference between storage and memory, making it possible to store and manipulate persistent data in-place *in memory*. Such systems with *non-volatile main memory (NVMM)*, also referred to as *persistent memory* systems improve the performance of

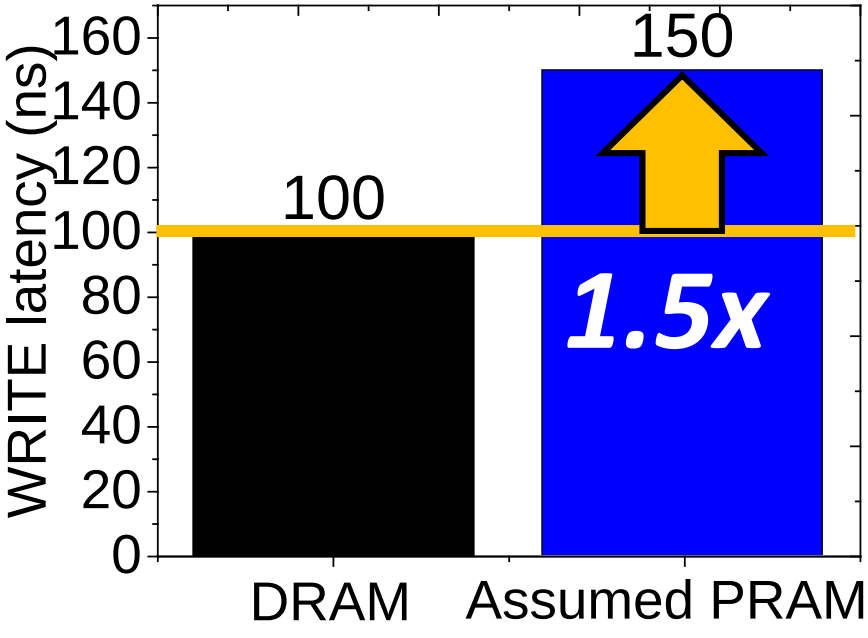
Accelerating Write by Exploiting PCM Asymmetries

...g Zhu
Engineering
aine
@maine.edu

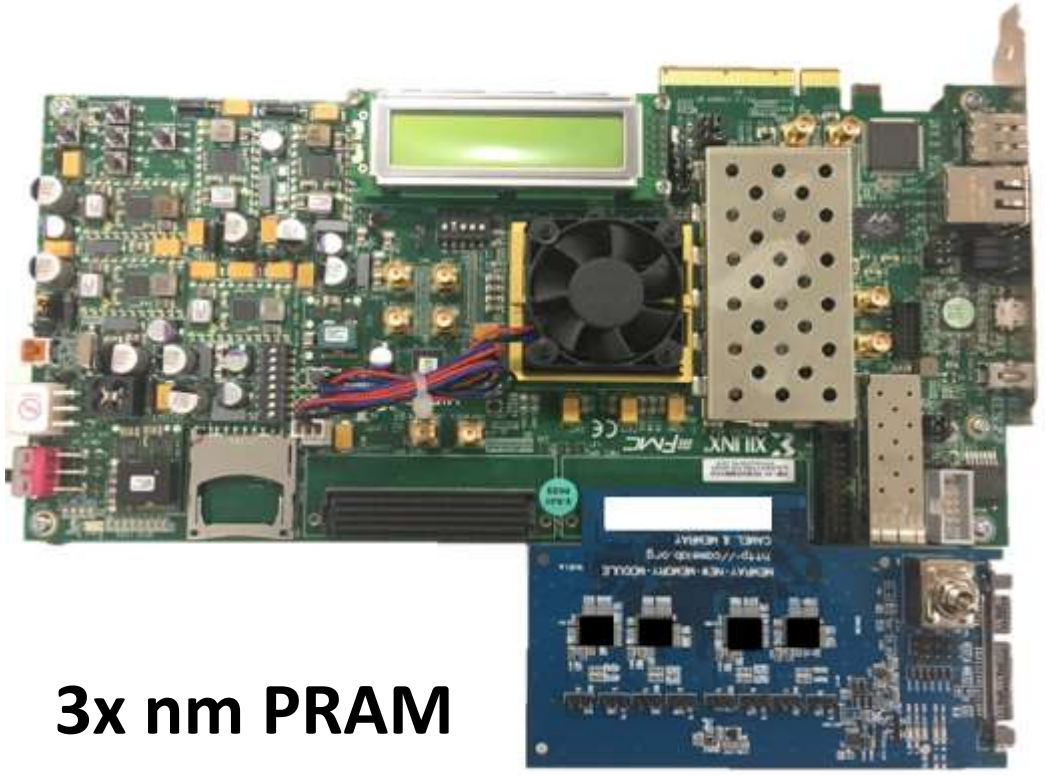
...be simultaneously written to a bank. *N* is referred to as a *unit* in this paper. Accordingly, writing a cache line of ...ytes requires multiple serially executed write units, which ...s down the overall write performance dramatically. Beside

HPCA'13

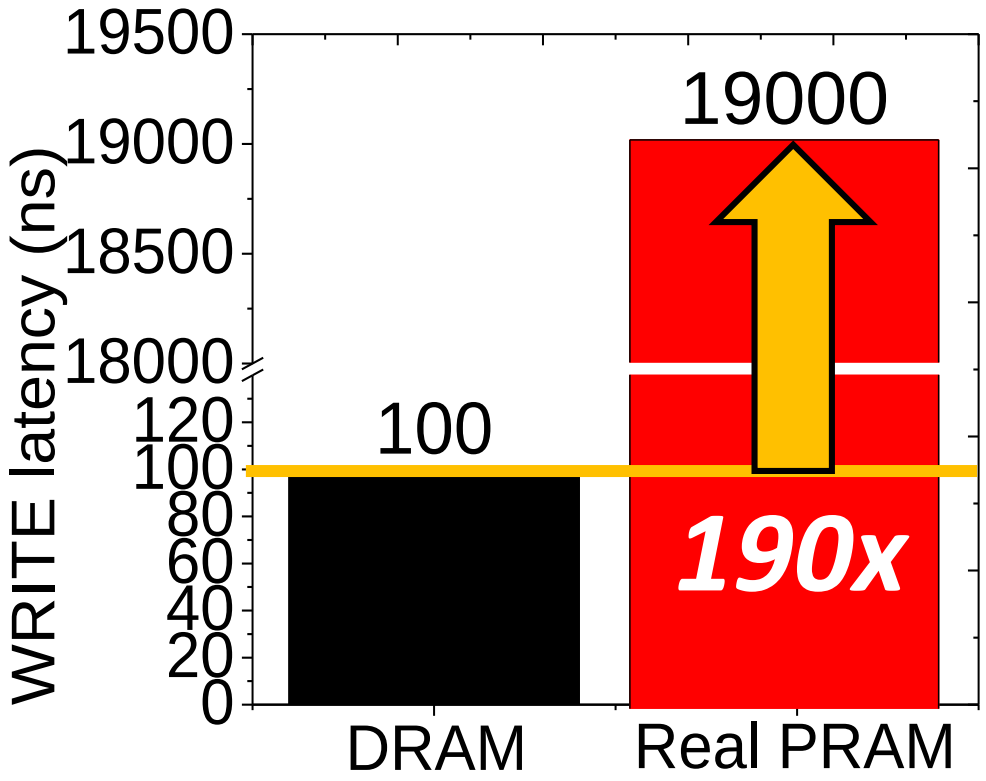
Many previous works assume PRAM's write latency as **similar to or slightly worse** than DRAM (**1.5x**)



PRAM Latency Measurement



3x nm PRAM



Our performance measurement on **real** 3x nm PRAM exhibits **expensive** write latency than DRAM (**190x**)



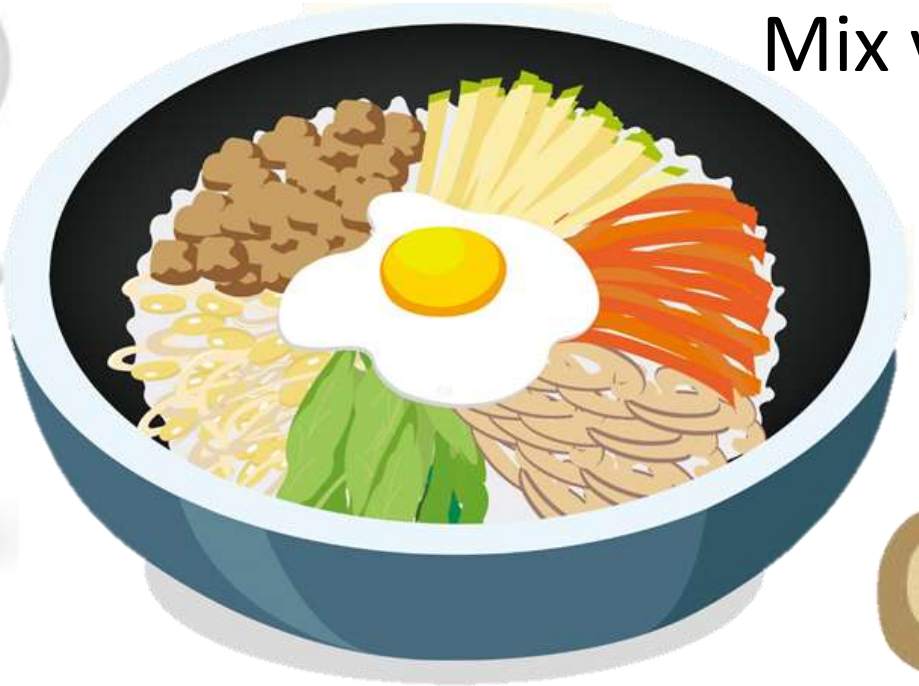
Then, How Can PRAM's Long Write Latency Be Mitigated?

Let's borrow the concept of ***Bibim***

*Bibim means "**Mix**" in Korean*

Mix various ingredients for **better taste**





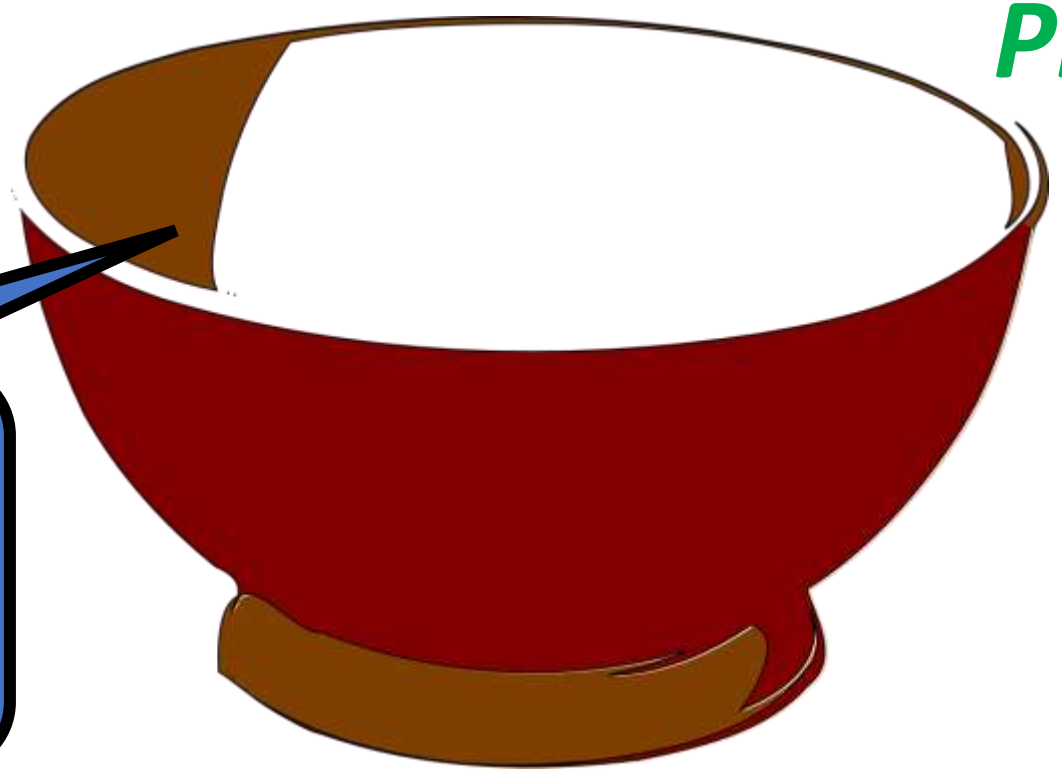


Hybrid Memory Can Help Us

Put fast **DRAM** and slow **PRAM** together



DRAM



PRAM



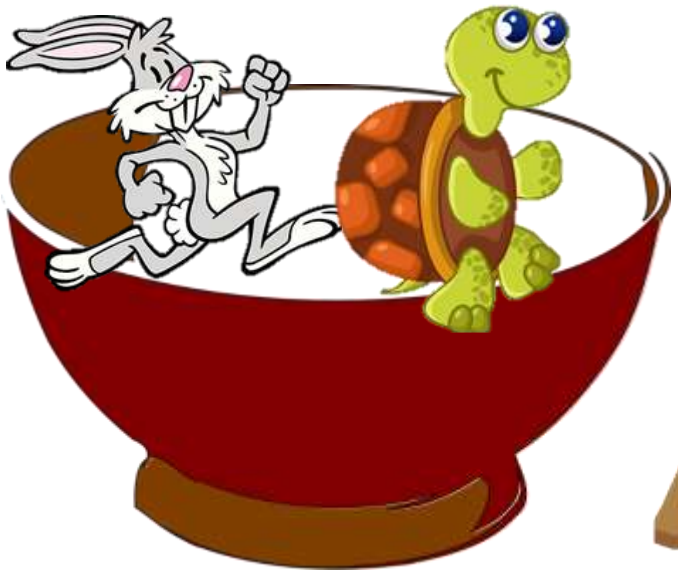
NOTE) I'm just used as a write-only **inclusive cache** of slow PRAM



BTW, How Can Hybrid Memory Be Used in Real System?

We design new memory controller, ***BIBIM***, for non-volatile hybrid memory







1. Demands for Hybrid Memory

2. Preview of BIBIM

3. Design Considerations of BIBIM

4. Example of BIBIM Operation

5. Evaluation

Demo (Track and Field)

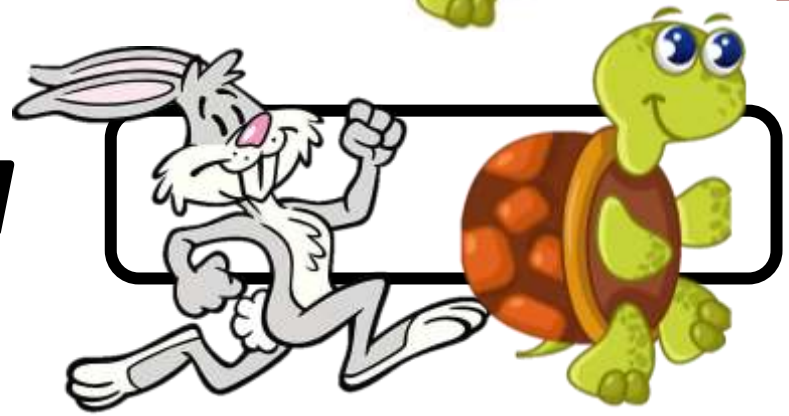
DRAM



PRAM



BIBIM



Performance comparison (memory access with synthetic, read-write inter-mixed trace)

Demo – Slow Version

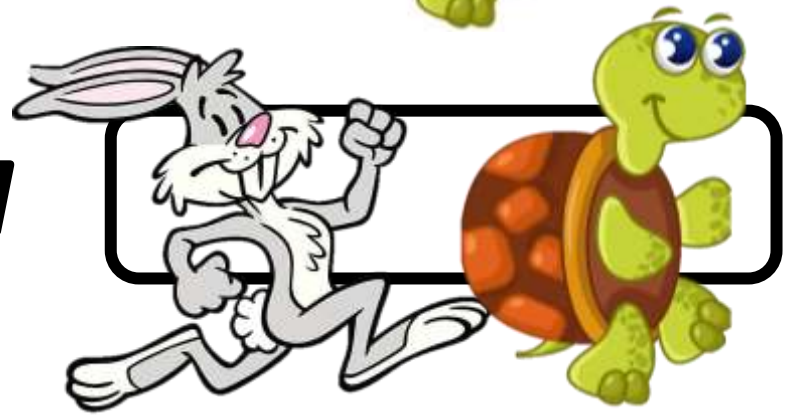
DRAM



PRAM



BIBIM



```
D:¥Projects¥PRAM¥Hotstorage 18¥¥video>python bibim_test.py ycsb dram
```

```
D:¥Projects¥PRAM¥Hotstorage 18¥¥video>python bibim_test.py ycsb pram
```

```
D:¥Projects¥PRAM¥Hotstorage 18¥¥video>python bibim_test.py ycsb bibim
```



ent
mory
sts

ory

Still PRAM servicing memory requests



1. Demands for Hybrid Memory

2. Preview of BIBIM

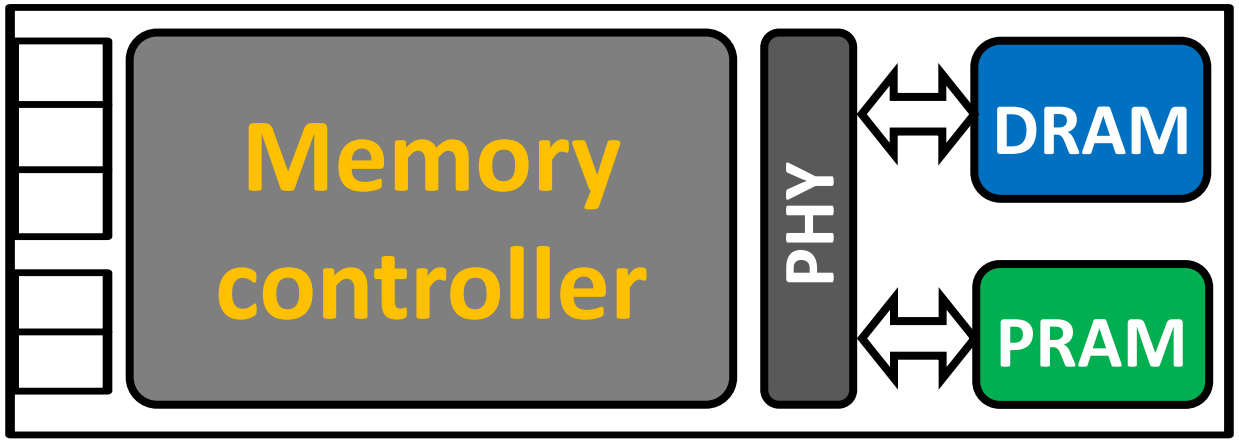
3. Design Considerations of BIBIM

4. Example of BIBIM Operation

5. Evaluation



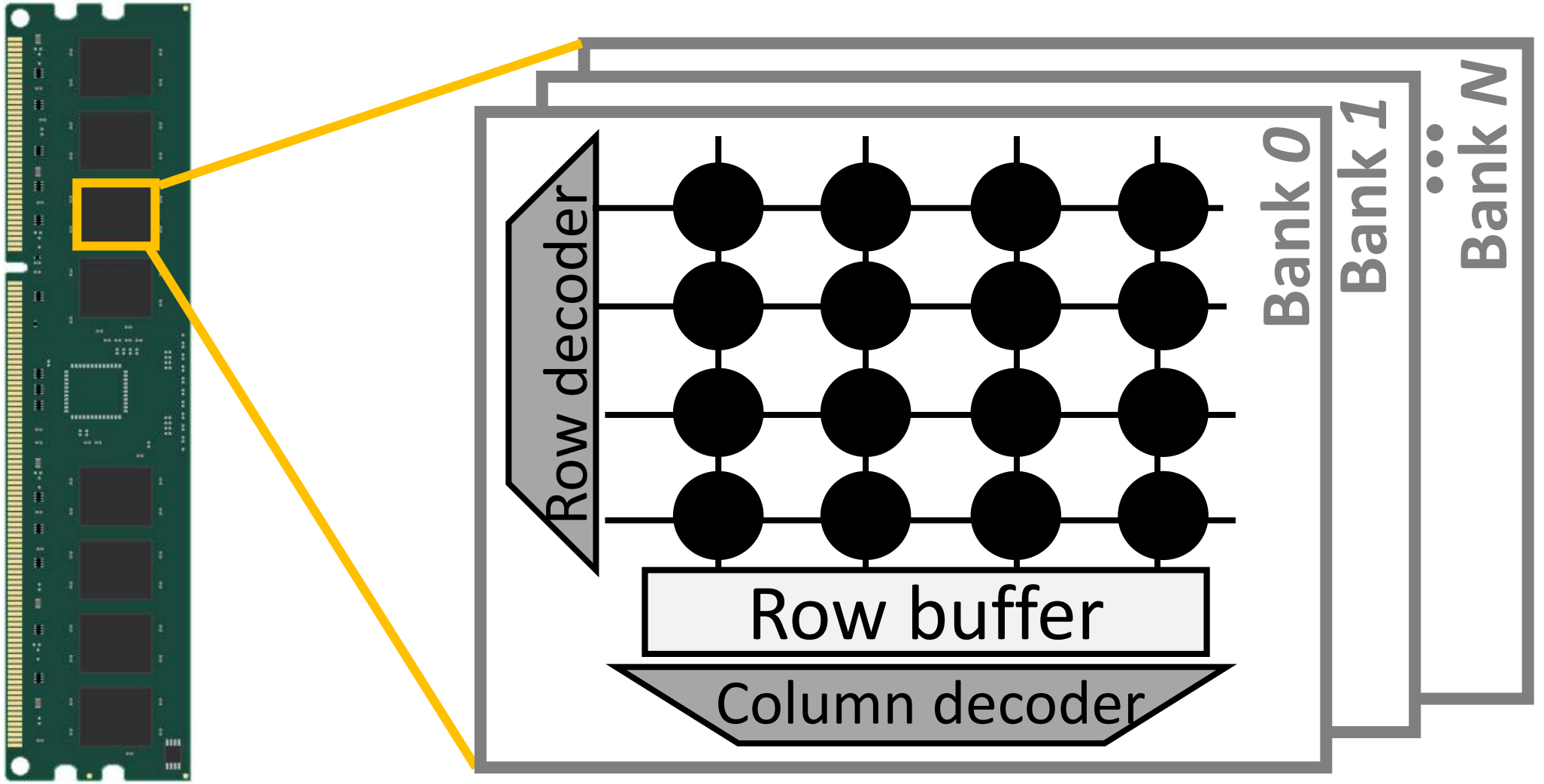
What Are The Considerations To Design Hybrid Memory (DRAM+PRAM) Controller?



To get insights of controller design, let's understand the details of DRAM and PRAM

DRAM's Multi-bank Architecture

- **Multiple banks** to serve multiple memory requests in parallel
- **Single row buffer** within a bank





Does PRAM have the Same Internal Architecture with DRAM?



Challenge1: PRAM's write latency is long

➔PRAM employs multiple row buffers

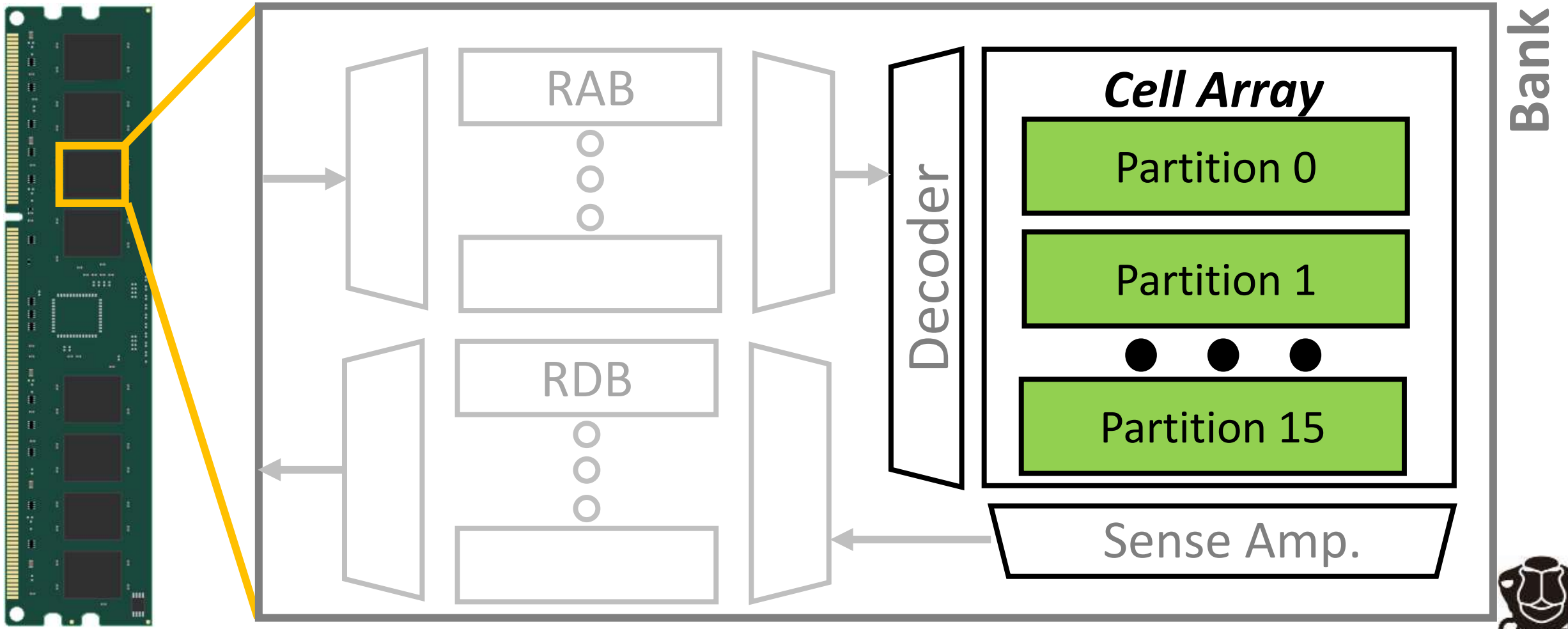
Challenge2: PRAM's asymmetric latency incurs lots of **bank conflicts**

➔PRAM uses multi-partition architecture

CAMEL

PRAM's Multi-partition Architecture

- **Multiple partitions within the bank** for partition-level parallelism
- **Multiple row buffers** to mitigate long write latency





Can A Conventional DRAM Controller Be Aware of Multi-Partition? (Inside of A Bank)



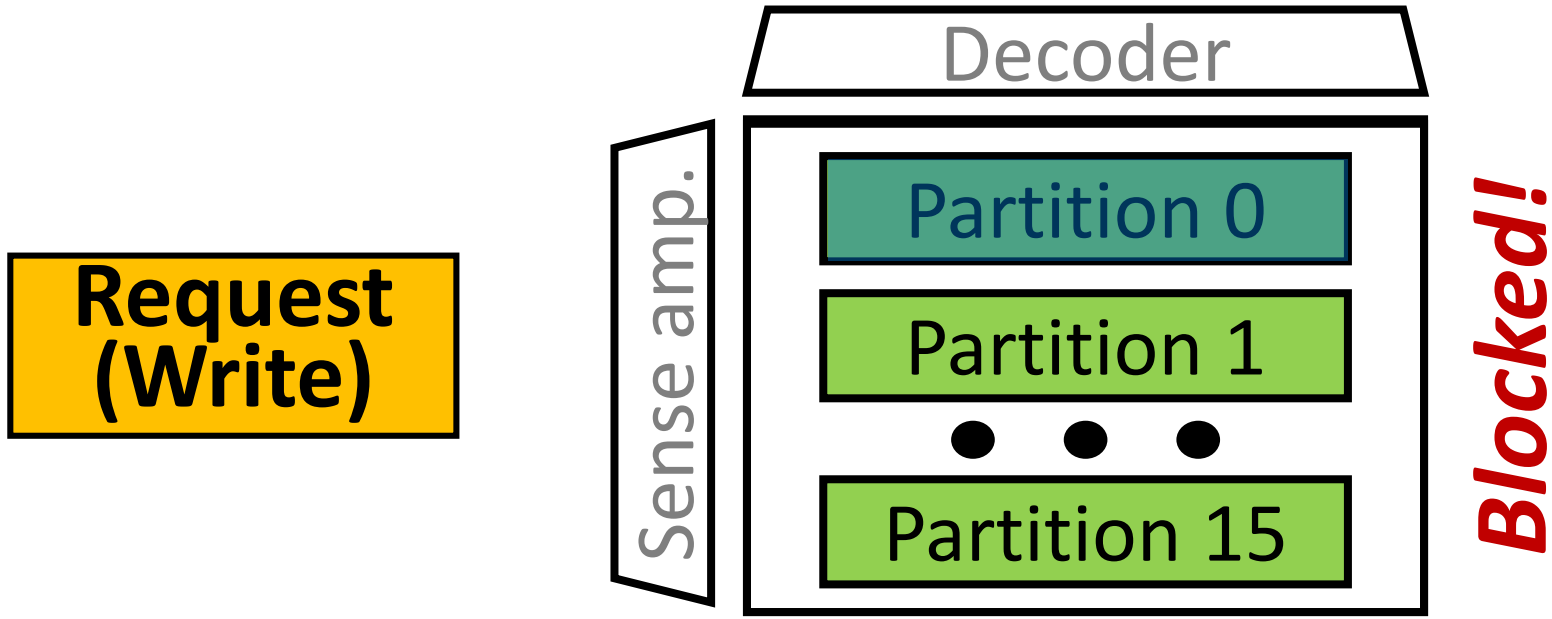
(Revisited) Conventional DRAM scheduler just utilizes **bank-level parallelism**. Cannot see inside of bank!

∴ **Partition-level parallelism** should be supported



How Memory Requests Can Be Scheduled By Exploiting Multi-Partition Architecture?

Limitation of PRAM: In PRAM design, WRITE request blocks whole PRAM bank.





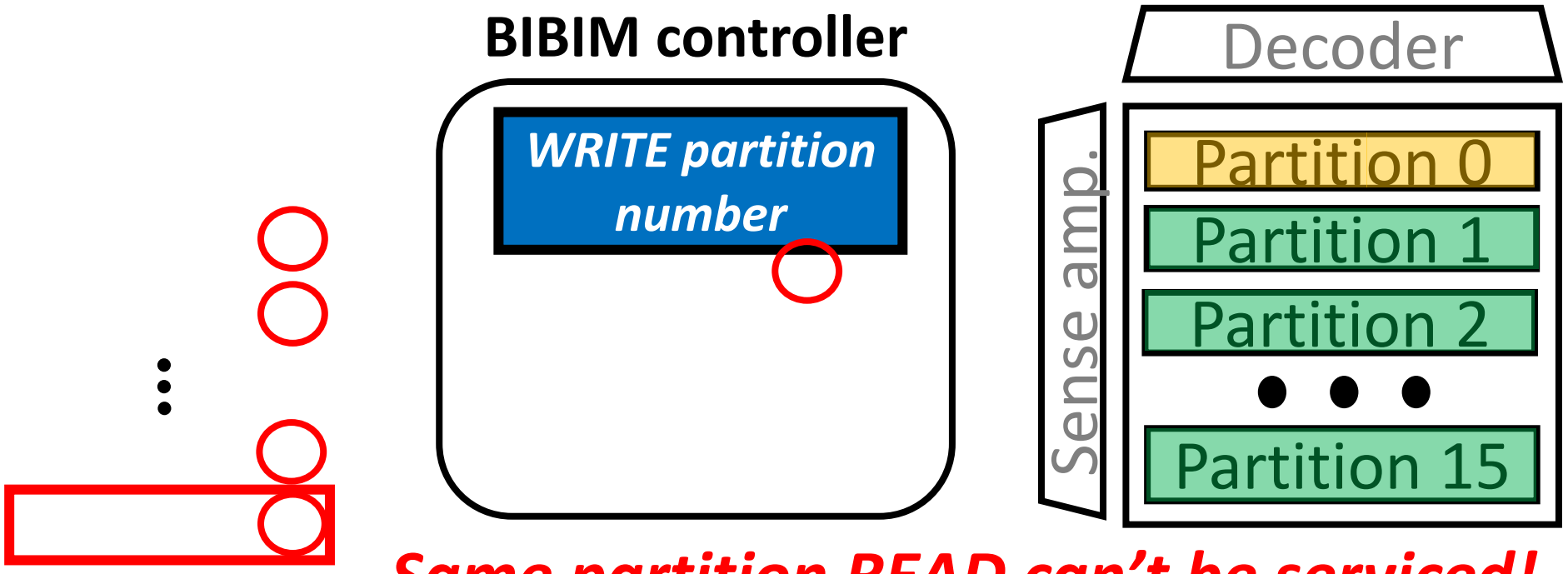
How Memory Requests Can Be Scheduled By Exploiting Multi-Partition Architecture?



Key insight: Although WRITE cannot be serviced, READs can be serviced if partition number is different

Non-Blocking Read Service (NBRS)

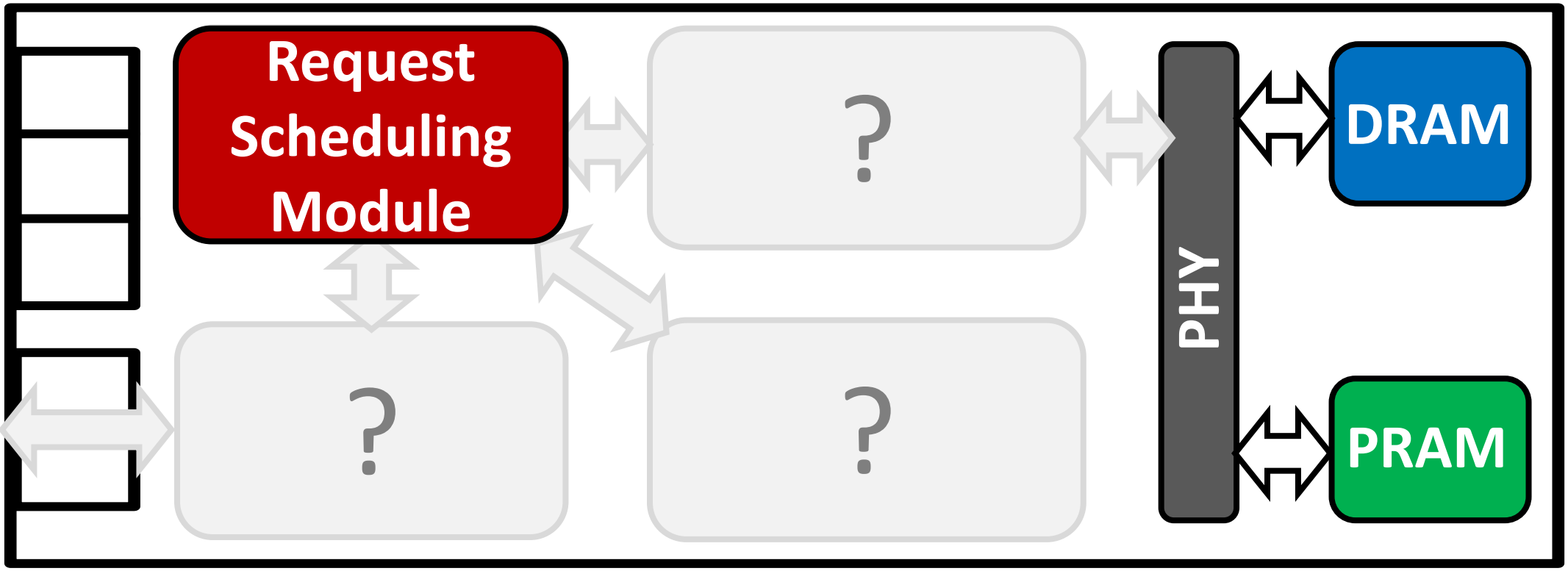
Solution: Add a ***register*** to store 'partition number of WRITE' and compare it with partition number of incoming READ request



Same partition READ can't be serviced!



BIBIM Design1: Scheduling Support Module for PRAM-aware New Scheduling Scheme





Now Requests Are Scheduled. Then, How Then Can It Serve to Hybrid Memory?



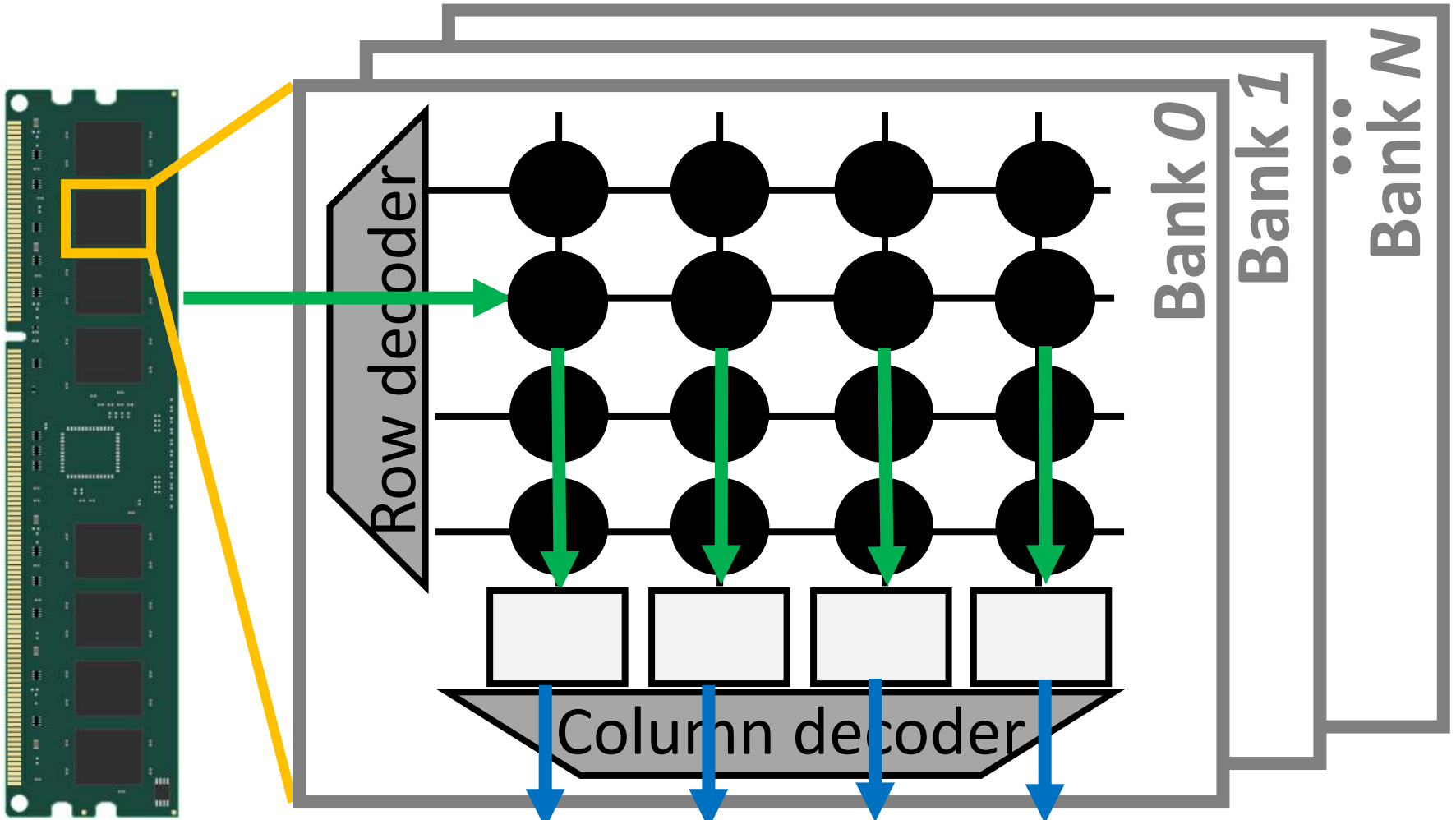
JEDEC

Firstly, as is generally known, LPDDR2 is JEDEC standard low-power memory interface (used for DRAM)

CAMEL

DRAM's Timing (LPDDR2 by JEDEC)

1) Activation: activate target row & write that data to row buffer
2) Read/Write: accessing row buffer with column address
3) Precharge: charge half-voltage of bit-line





Does PRAM have the Same Memory Interface (LPDDR2) with DRAM?

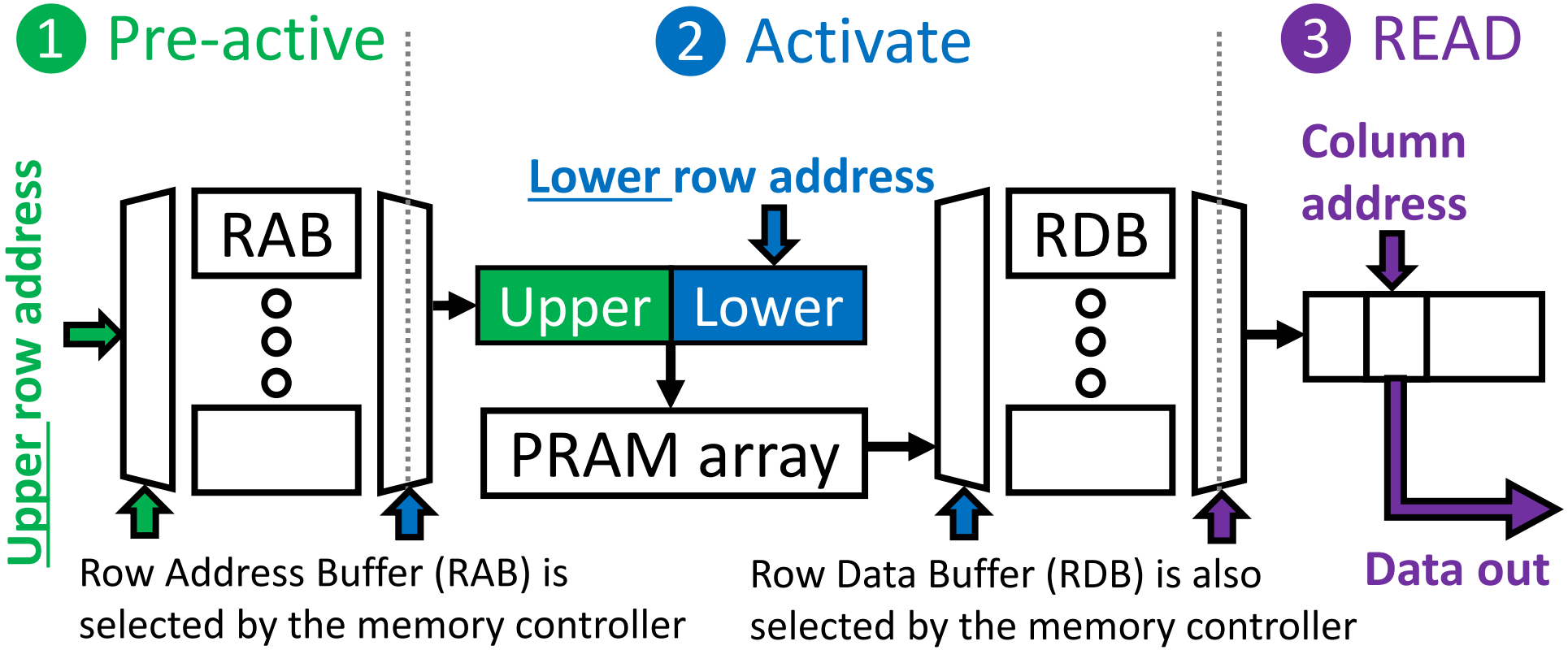


(Revisited) PRAM has a different architecture with DRAM such as **Multiple row buffers** and **More larger capacity**

∴ Different interface is required

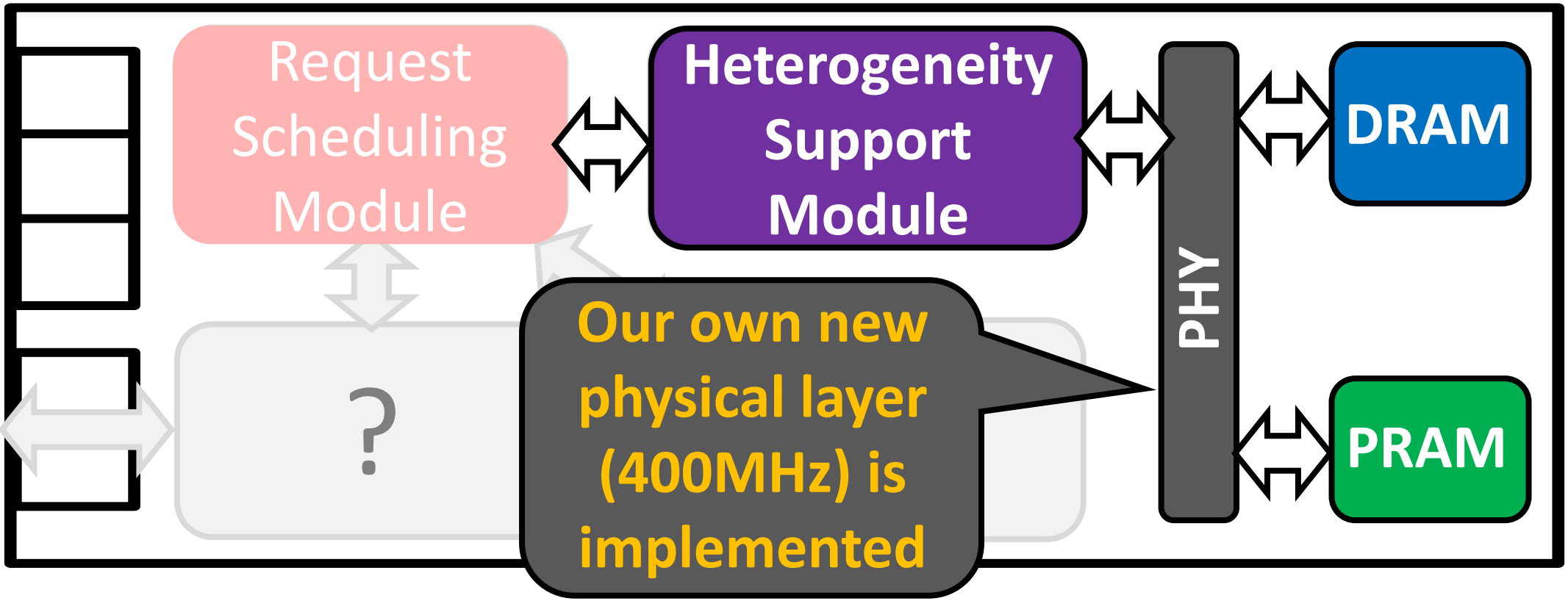
PRAM's Timing

- PRAM requires different timing model from DRAM
  - NVM memory space is much larger than a DRAM
- 3-Phase addressing (LPDDR2-NVM by JEDEC)





BIBIM Design2: Heterogeneity Support Module for both LPDDR2 & LPDDR2-NVM





Don’t Forget DRAM is For Cache. Then, How Caching Can Be Supported?

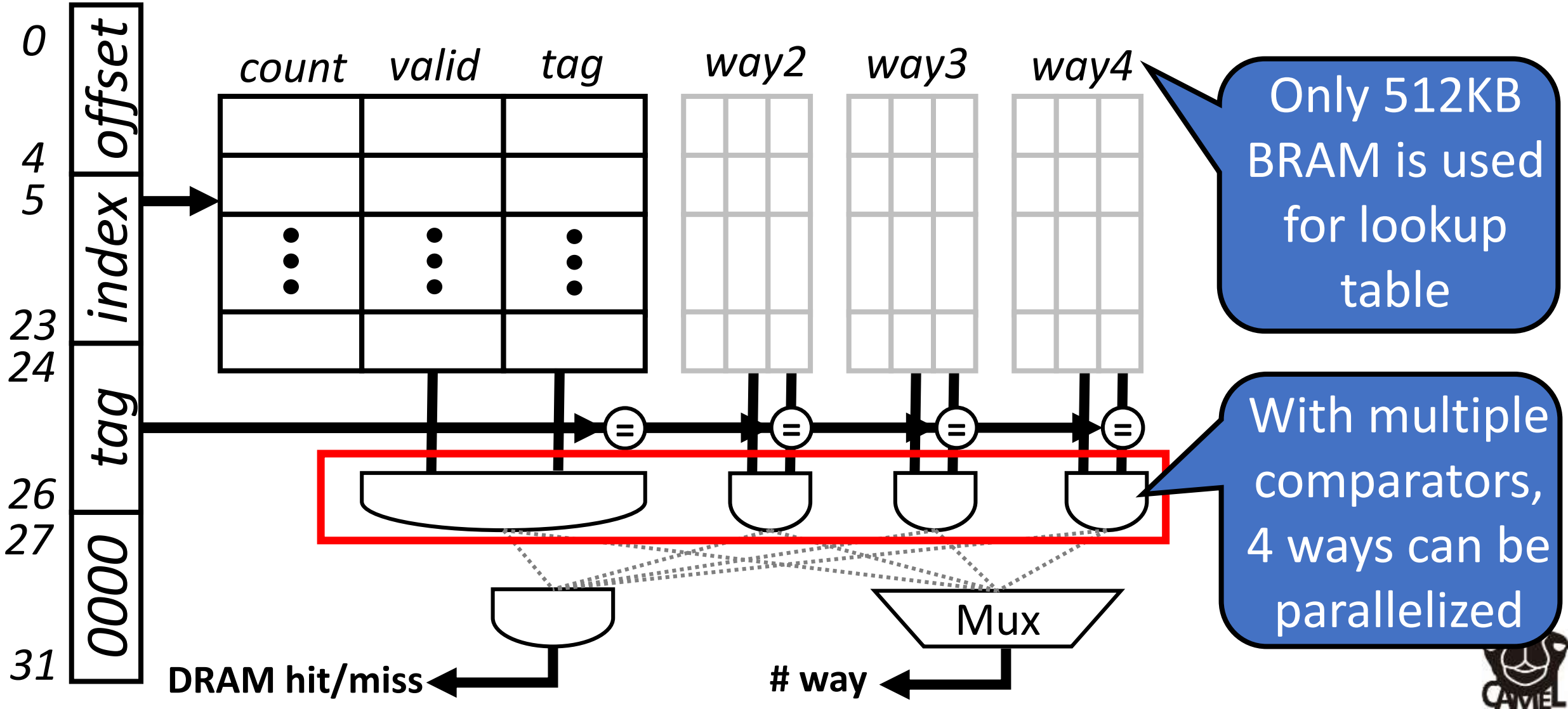


Solution: Keep which data exist in DRAM (caching info) in **lookup table**.

Moreover, like conventional cache, controller should have algorithms such as DRAM dataline update, eviction, and find empty dataline.

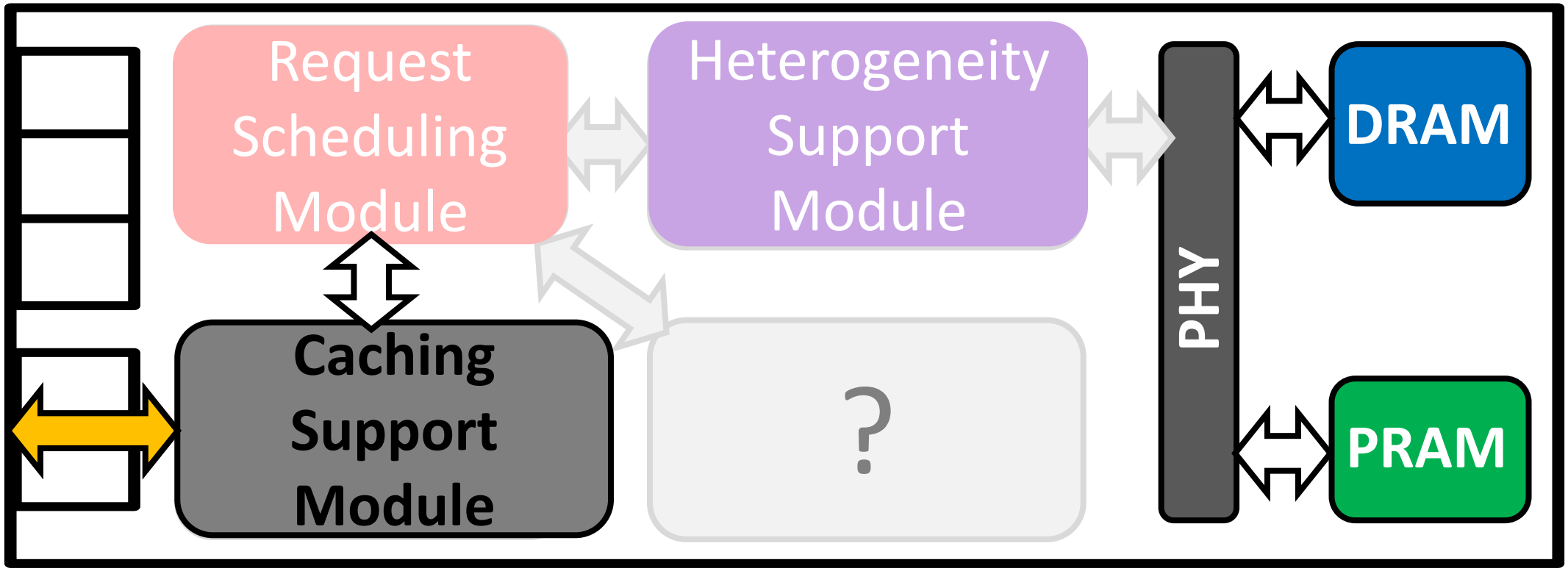
Hardware Support for Lookup Table

Lookup table do not include data value, includes address information





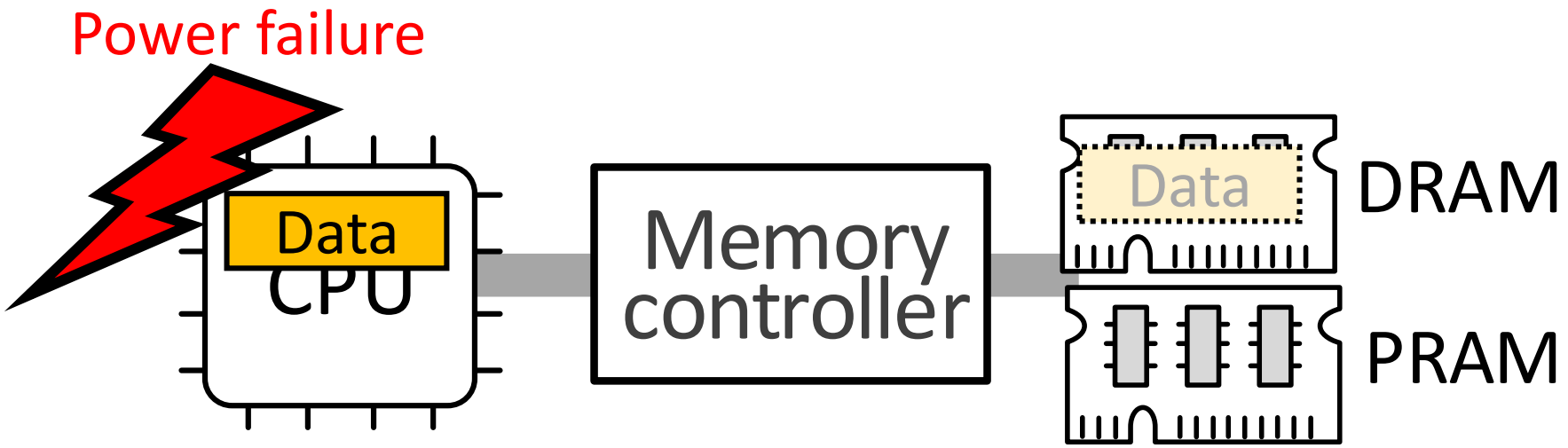
BIBIM Design3: Caching Support Module for Use DRAM As Inclusive Cache of PRAM





BTW, How Non-Volatility of PRAM Can Be Maintained Although DRAM Is Integrated? (Hybrid)

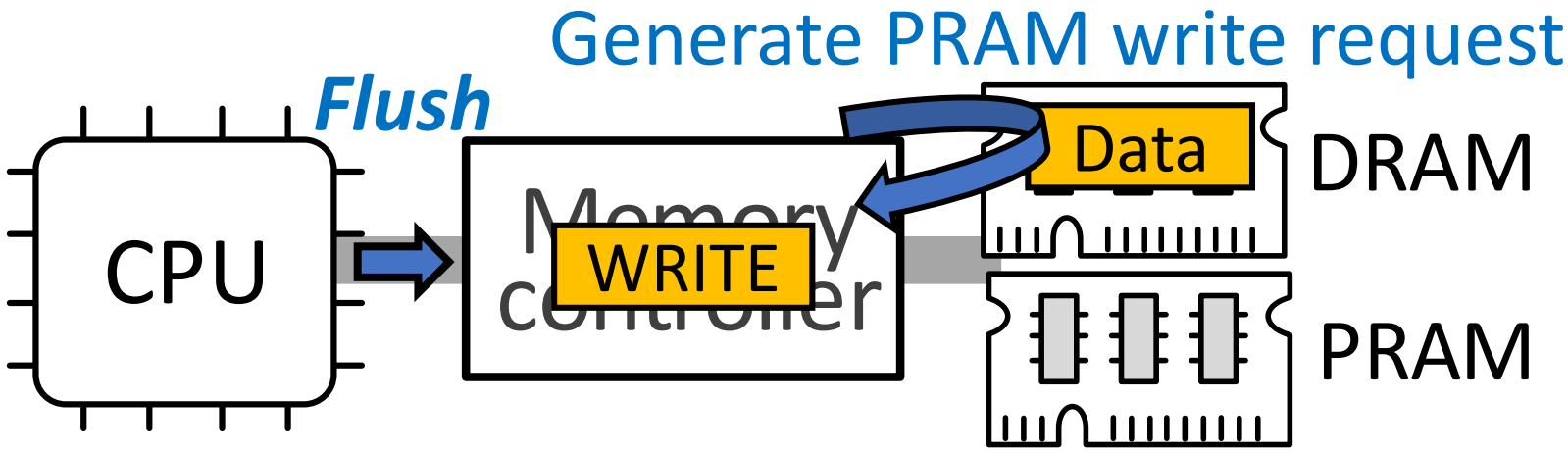
Challenge of hybrid memory: Data in DRAM will disappear when there is a power failure



FLUSH Operation

Solution: Provide ***Flush*** operation which moves DRAM data to PRAM. Memory controller generates 'PRAM write' request corresponding to the target DRAM row.

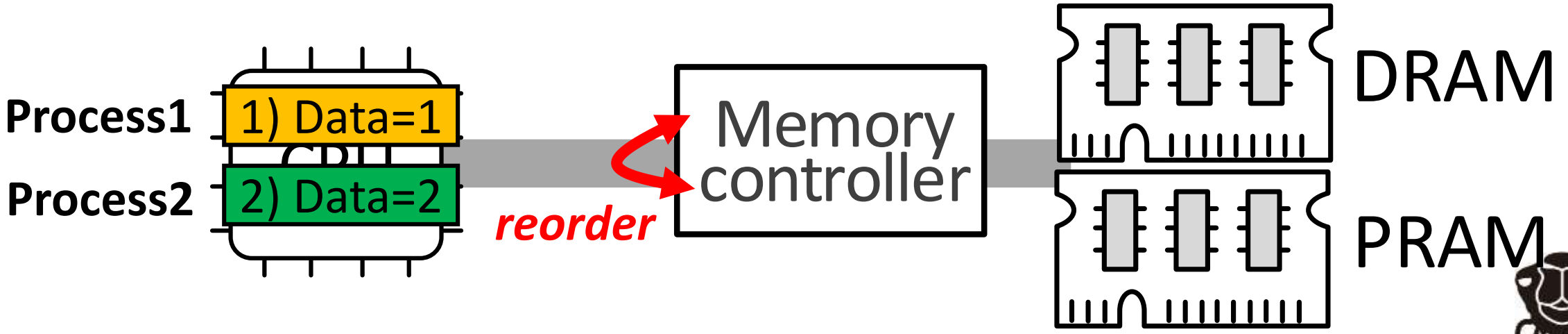
NOTE) 'PRAM write' will be stored in command queue which exists in memory controller. And DRAM dataline is invalidated.





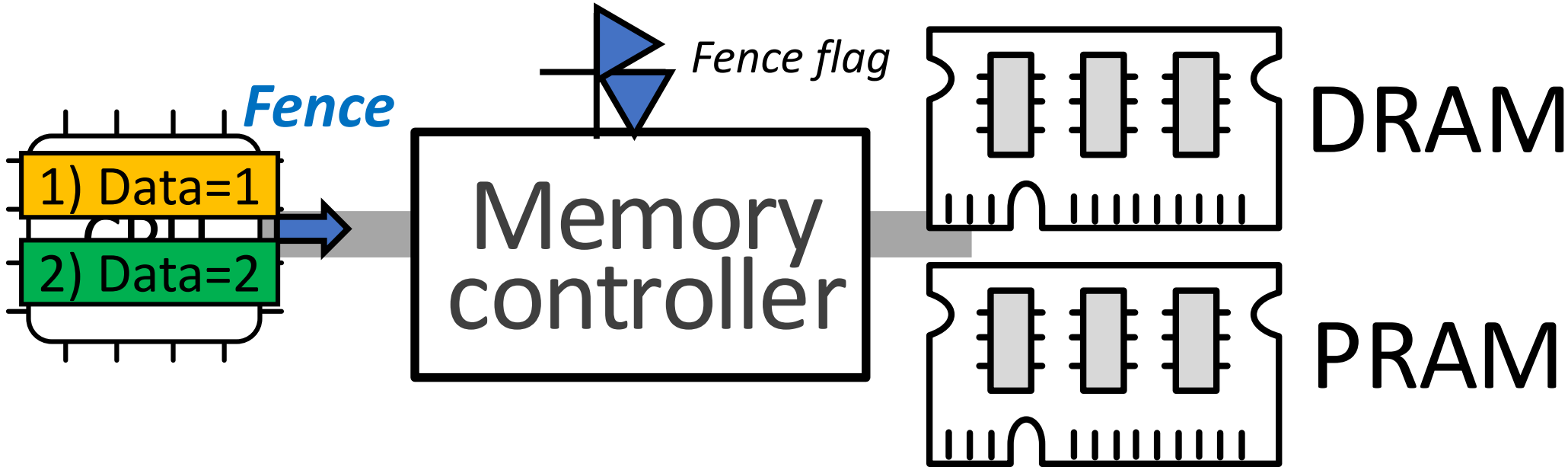
Okay, Data Delivery Is Guaranteed. Is It Good Enough?

Challenge of flush: User believes data has the latest value. But, the memory controller can reorder the order of memory request



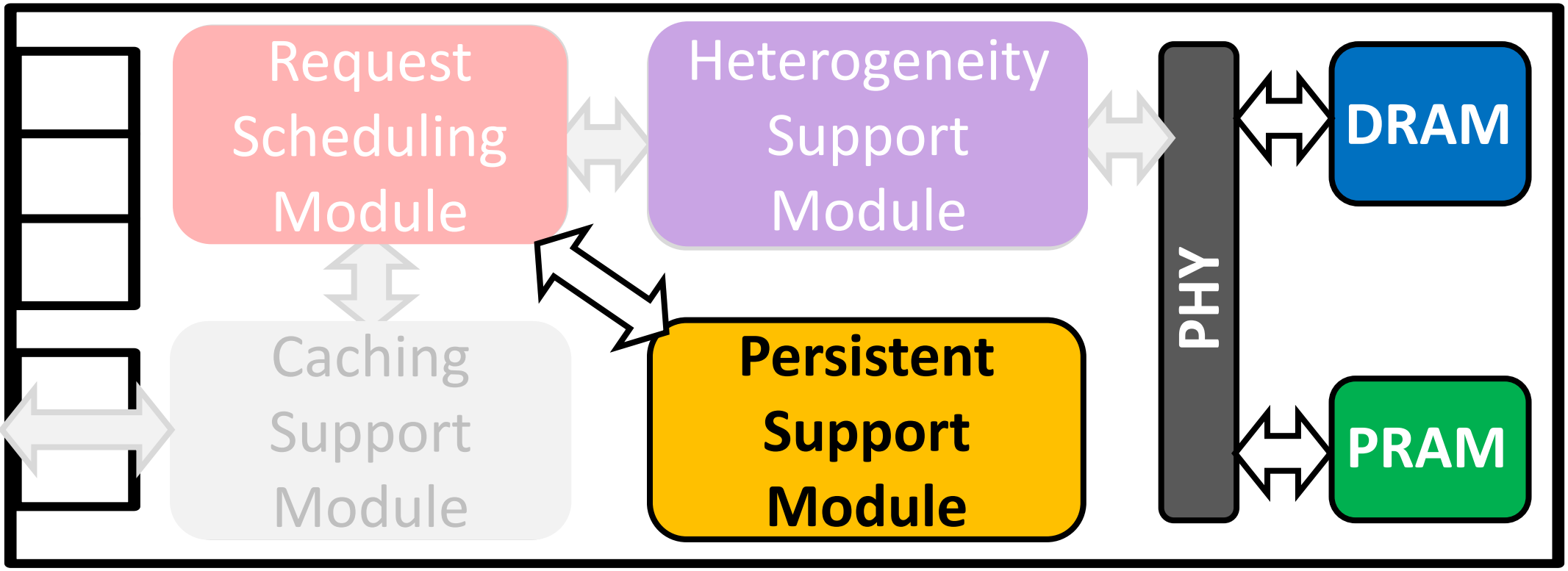
FENCE Operation

Solution: Provide ***Fence*** operation to enforce data delivery order of memory requests. The memory controller can simply add '*fence flag*' to check fenced or not





BIBIM Design4: Persistent Support Module to Guarantee Data Delivery & Delivery Order



1. Demands for Hybrid Memory

2. Preview of BIBIM

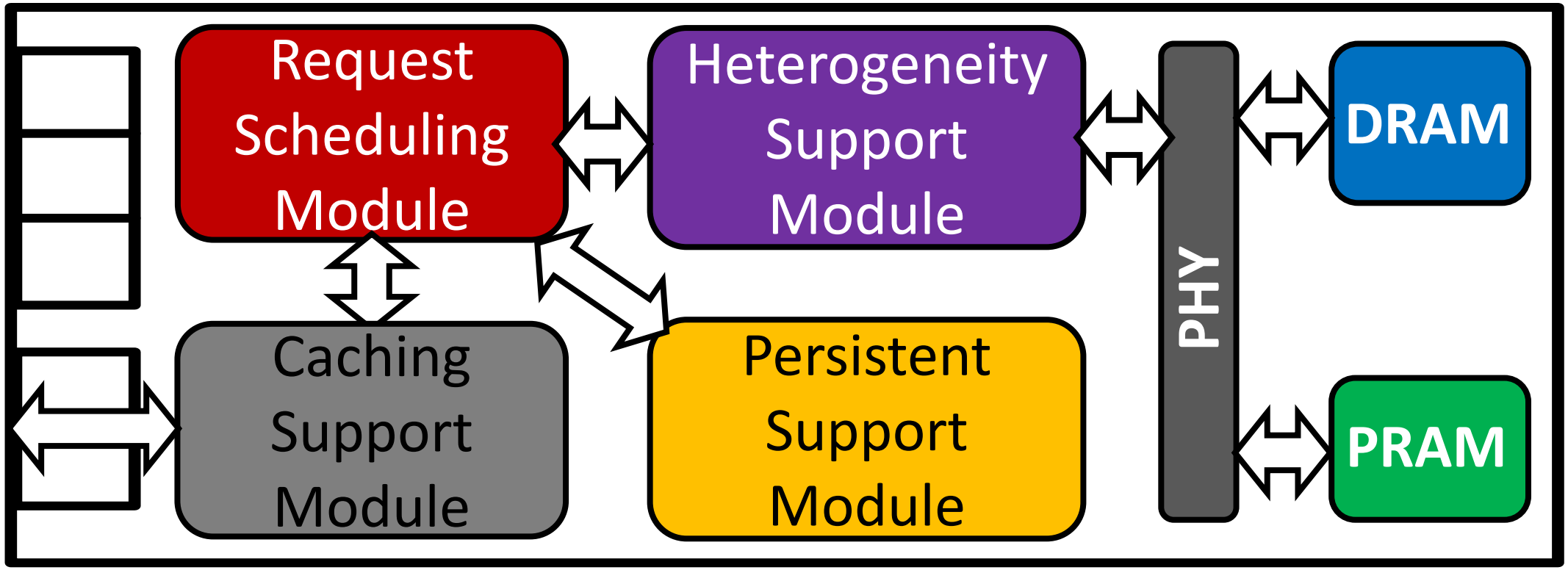
3. Design Considerations of BIBIM

4. Example of BIBIM Operation

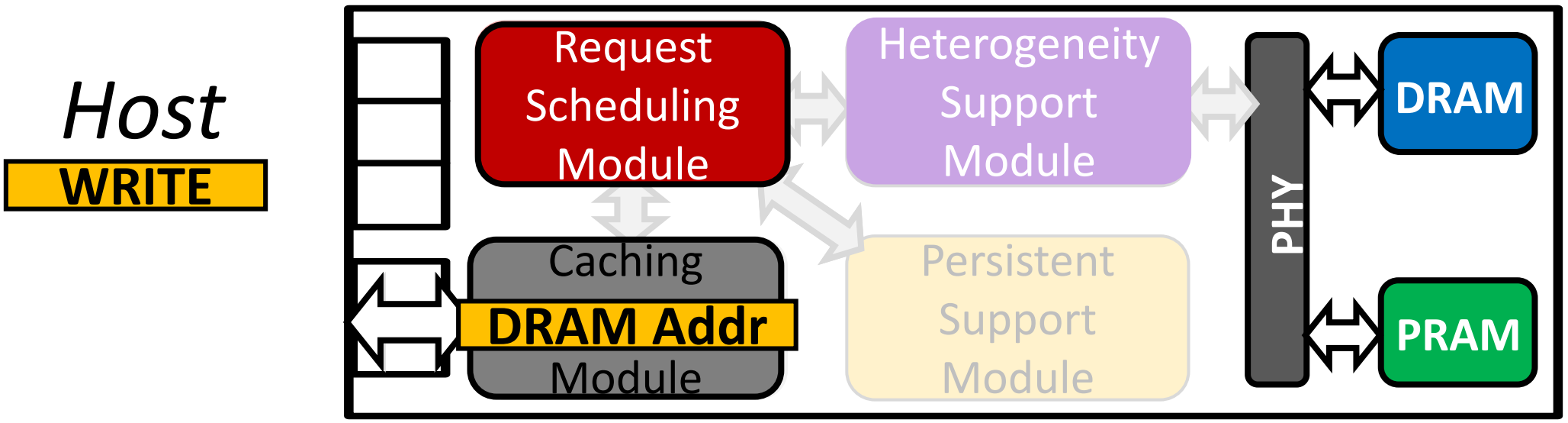
5. Evaluation



We Designed Four Modules in BIBIM. BTW, How They Work Together?



Specific Example: Evict DRAM Dataline

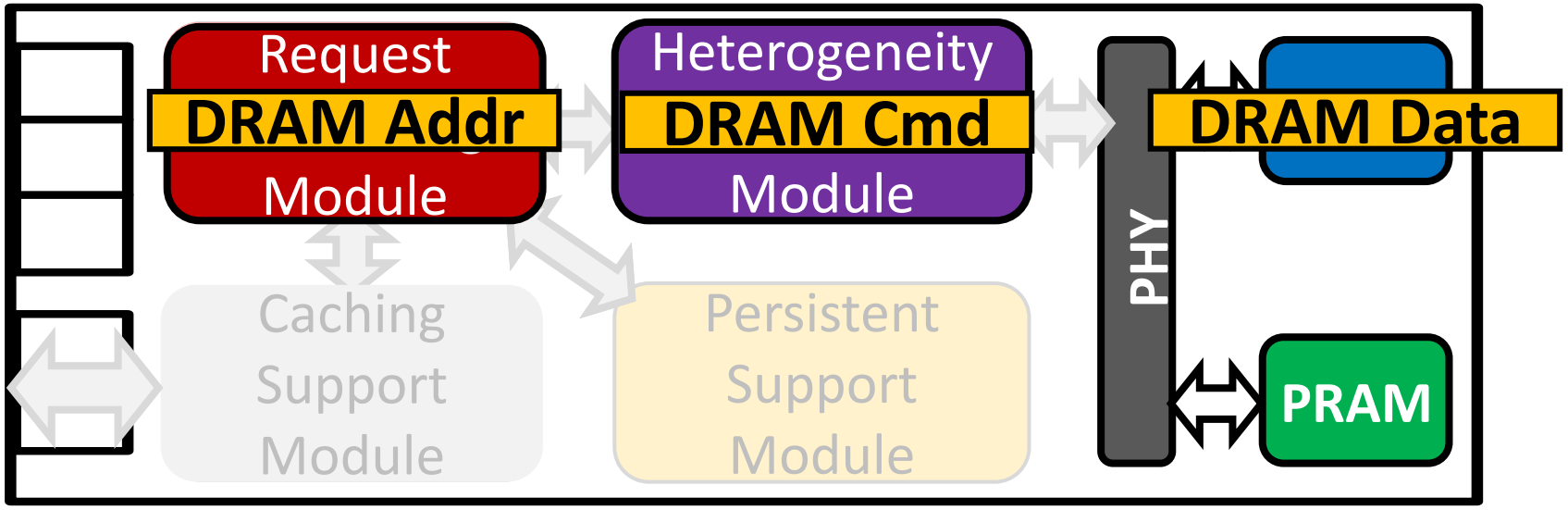


[Host] Sends WRITE request to the hybrid memory

[CACHING] Check DRAM dataline hit or not

[CACHING] (If it is a miss) Find victim DRAM dataline based on LRU policy, and return victim DRAM address to the SCHEDULING module

Specific Example: Evict DRAM Dataline

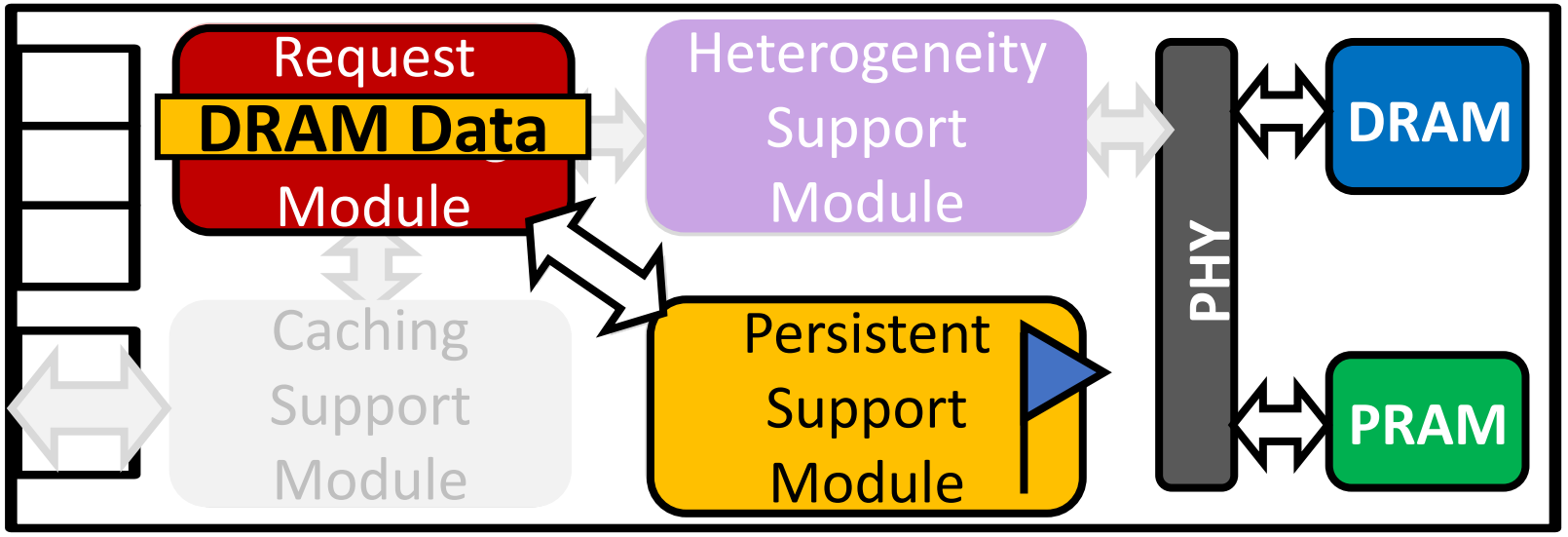


[SCHEUDLING] Sends DRAM address to the HETEROGENEITY module

[HETEROGENITY] Generate DRAM commands

[DRAM] Return DRAM dataline to the SCHEDULING module

Specific Example: Evict DRAM Dataline



[SCHEUDLING] Ask fence flag is set or not to the PERSISTENT module

[PERSISTENT] Return as ‘fence flag is set’

[SCHEUDLING] Do not serve PRAM WRITE (Eviction of victim DRAM dataline)

1. Demands for Hybrid Memory

2. Preview of BIBIM

3. Design Considerations of BIBIM

4. Example of BIBIM Operation

5. Evaluation

Persistent Memory Workloads

WHISPER: Wisconsin-**H**P Labs **S**uite for **P**ersistence



Wisconsin-**H**P Labs **S**uite for **P**ersistence **(WHISPER)**

Benchmark	Brief description
echo	Scalable, multi-version key-value store
ycsb	H-store like DB. Undo logs for consistency
tpcc	H-store like DB. Undo logs for consistency
ctree	Micro-benchmarks for simulations
hashmap	Micro-benchmarks for simulations
vacation	Online travel reservation system

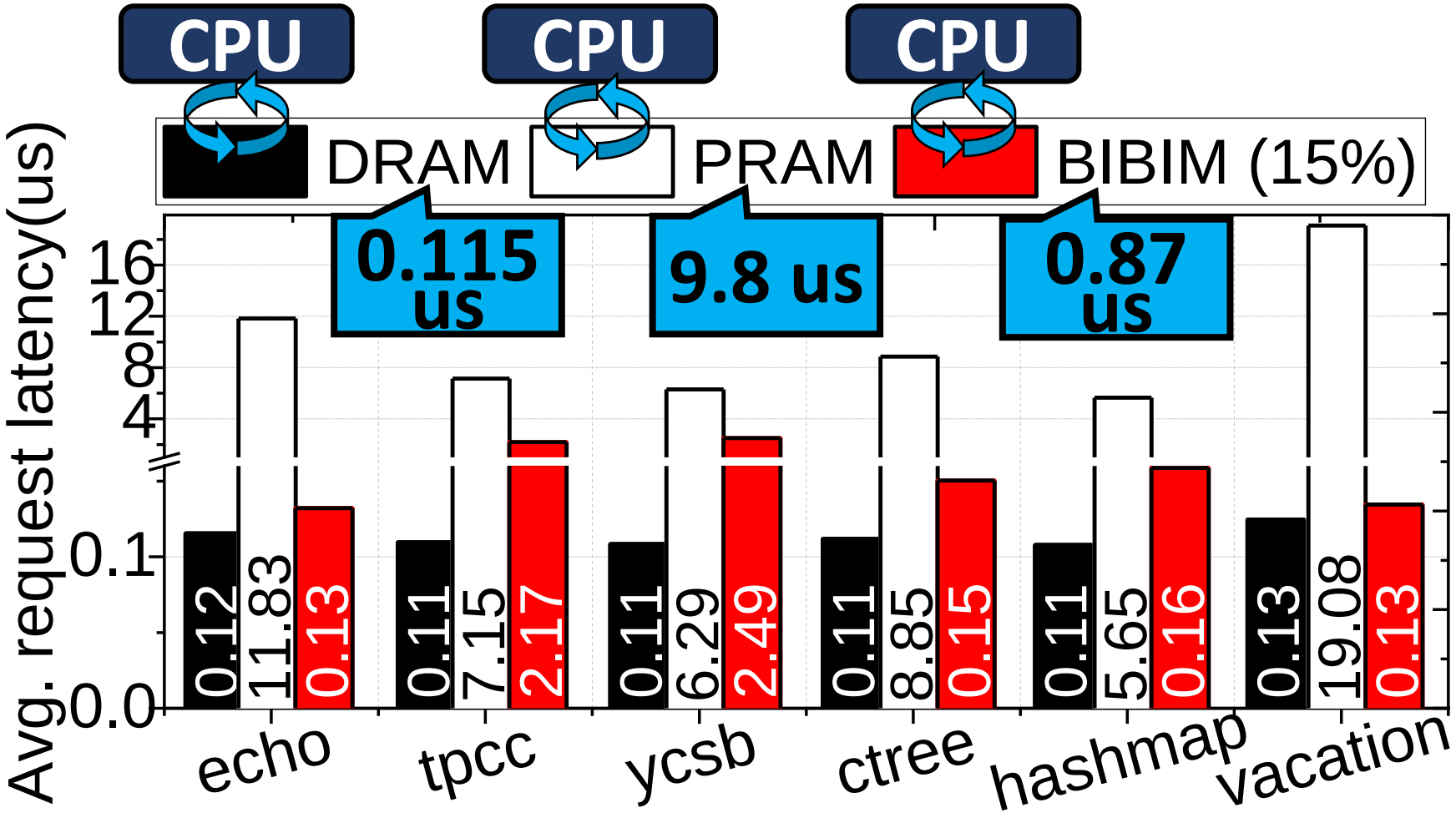
Write-intensive (82% of total requests)

Latency Comparisons

(*DRAM-only* vs. *PRAM-only* vs. *Hybrid memory w/BIBIM*)



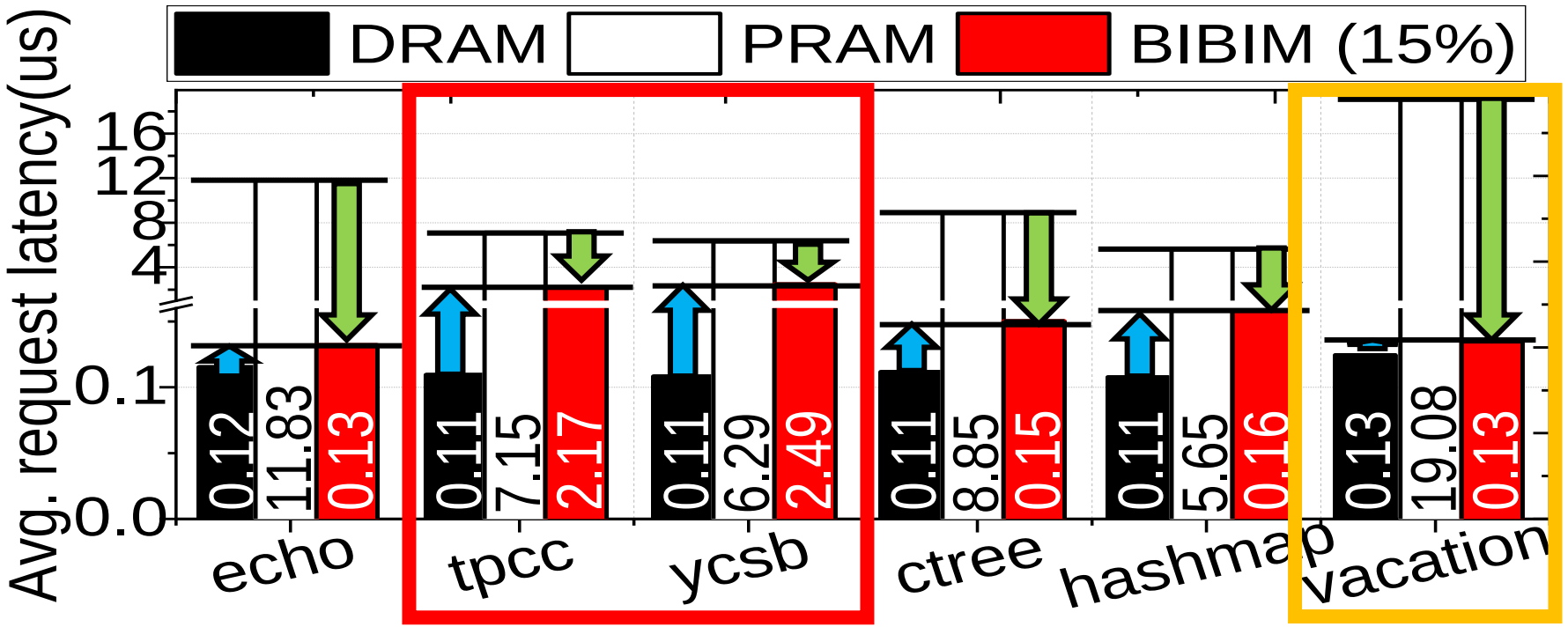
Average Request Latency



Lower is better!

DRAM > BIBIM > PRAM

Average Request Latency



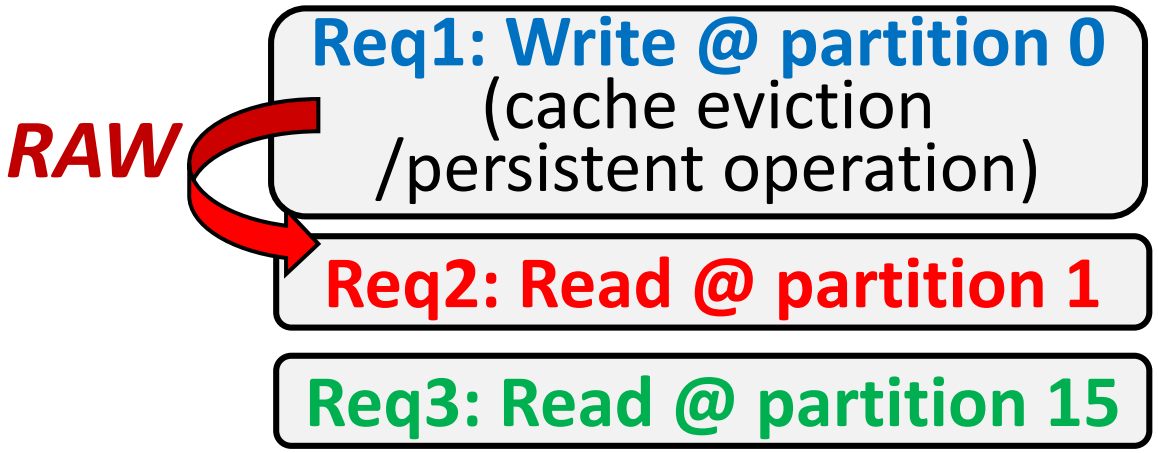
- BIBIM reduces the long latency than PRAM by 87%
- BIBIM only degrades the perf. than DRAM by 44%
- Best performance benefits: *vacation*; many cache hits (high write fraction)
- Small performance benefits: *tpcc, ycsb*; persistent operation (*flush*) generates many PRAM writes

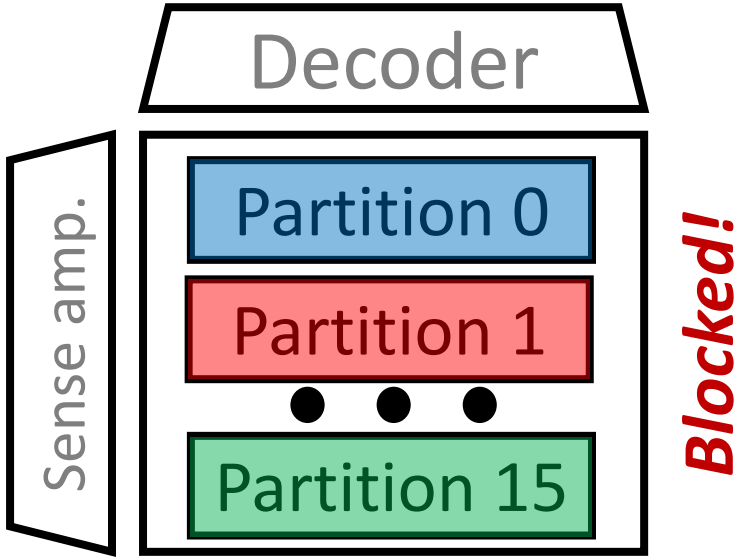
Non-blocking Read Service (NBRS) Analysis

Tested on PRAM-only (*PRAM*)

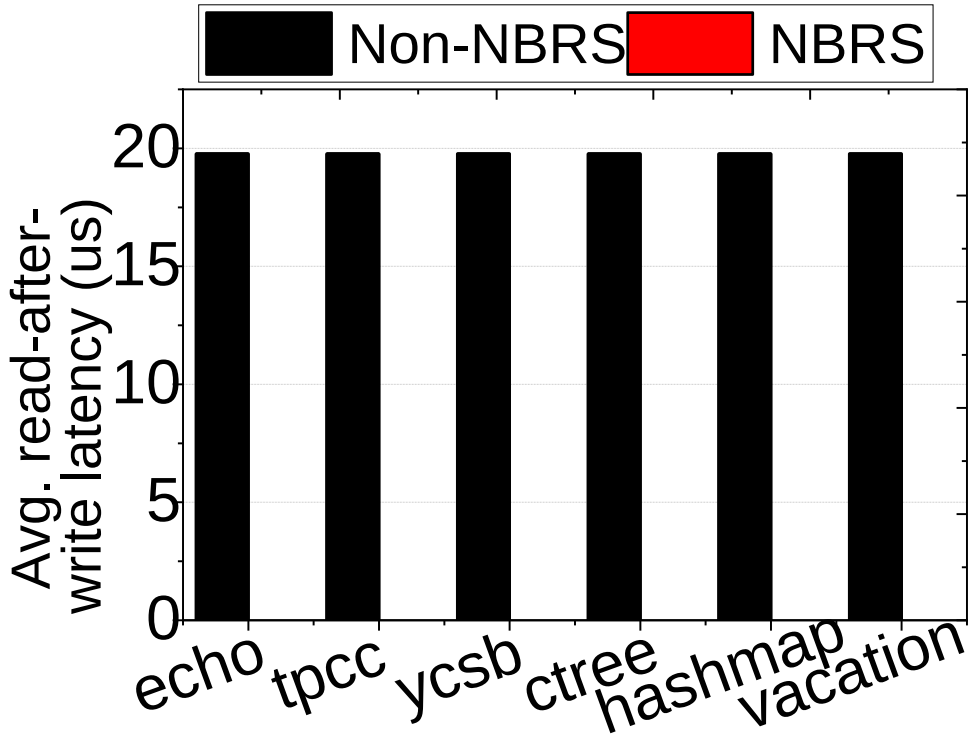
RAW Latency of Non-NBRS PRAM

- Read-after-write (*RAW*)





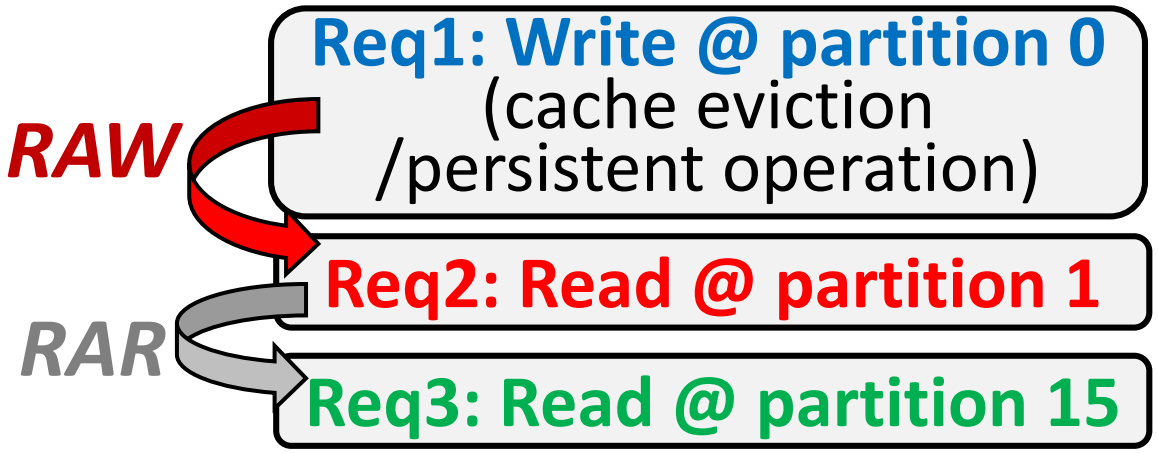
Non-NBRS PRAM bank

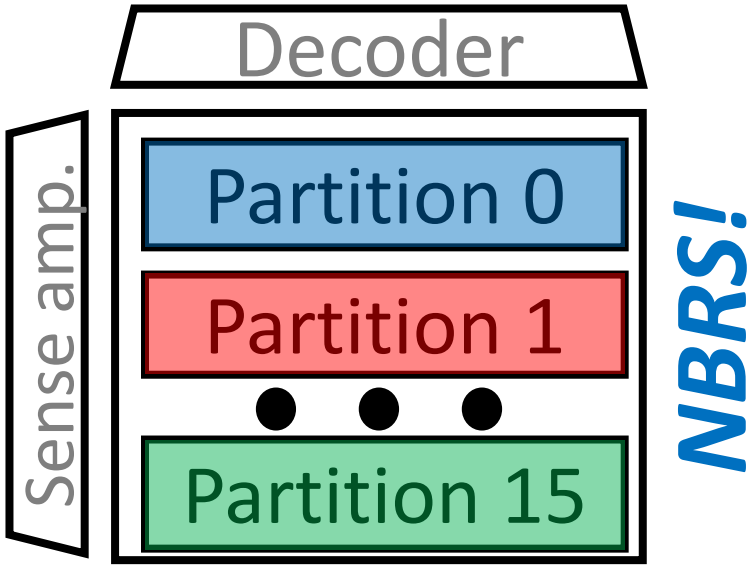


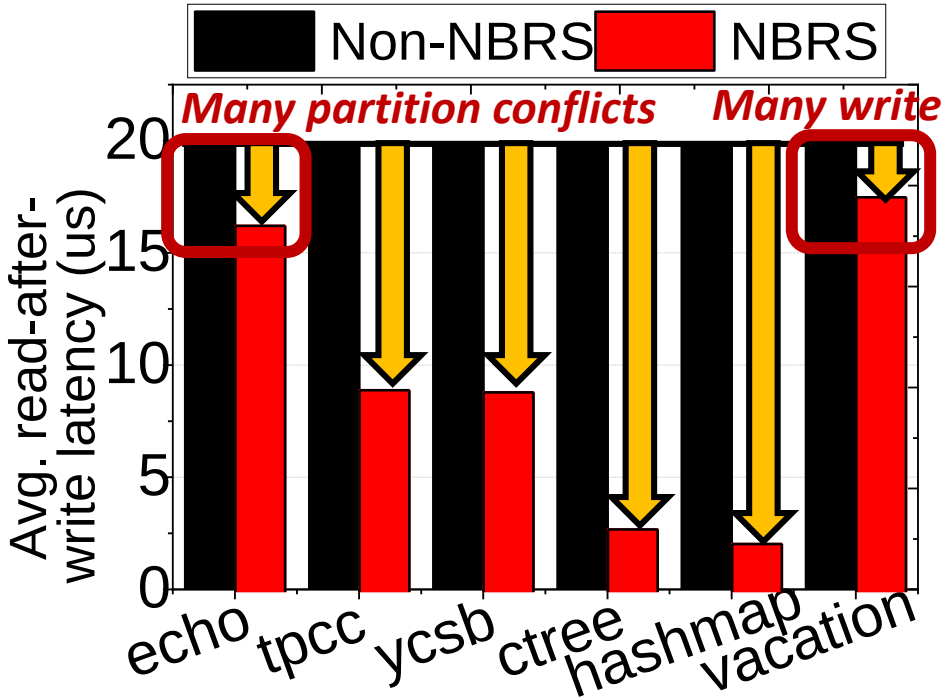
RAW latency of Non-NBRS:
19.8 us on average.
(≈ PRAM write latency)

RAW Latency of NBRS PRAM

- Read-after-write (*RAW*)







RAW latency of NBRS:
52% reduced
on average.

Demo – Normal Version

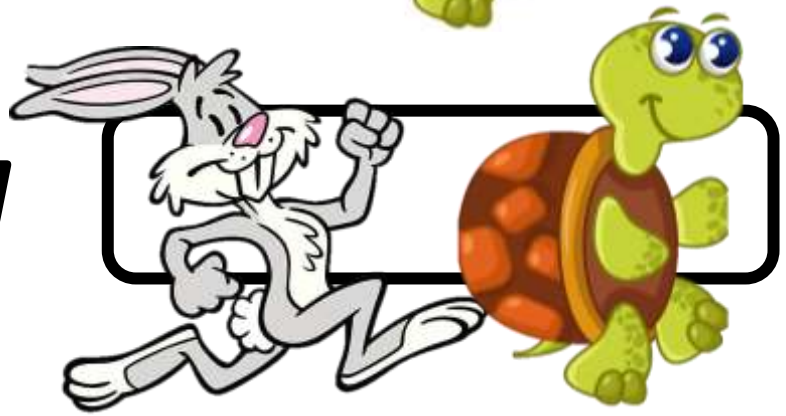
DRAM



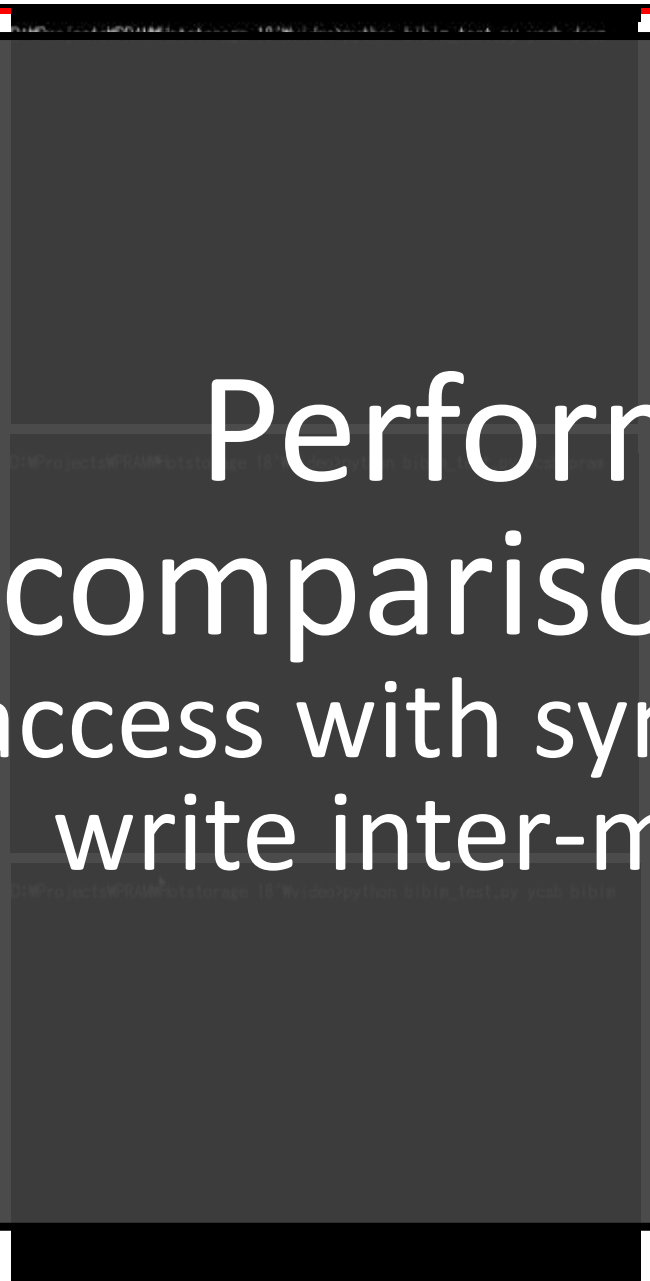
PRAM



BIBIM

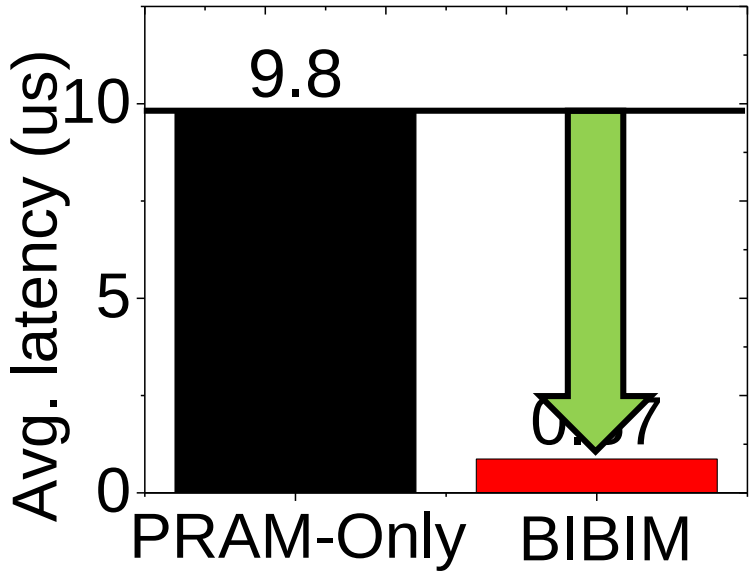


Performance comparison (memory access with synthetic, read-write inter-mixed trace)



Conclusion

BIBIM can achieve both high capacity and short latency



Our main contributions:

1) Multi-partition aware scheduler (NBRS)
2) New physical layer (PHY)
3) Persistent support for hybrid memory