

COMPUTER ARCHITECTURE AND SYSTEMS LAB

Genie Cache: Non-blocking Miss Handling and Replacement in Page-Table-based DRAM Cache

Youngin Kim and William J. Song
School of Electrical and Electronic Engineering, Yonsei University

Background: More Memory Bandwidth and Capacity

- Cutting-edge applications such as **AI and HPC** demand increasingly **larger memory bandwidth and capacity**.

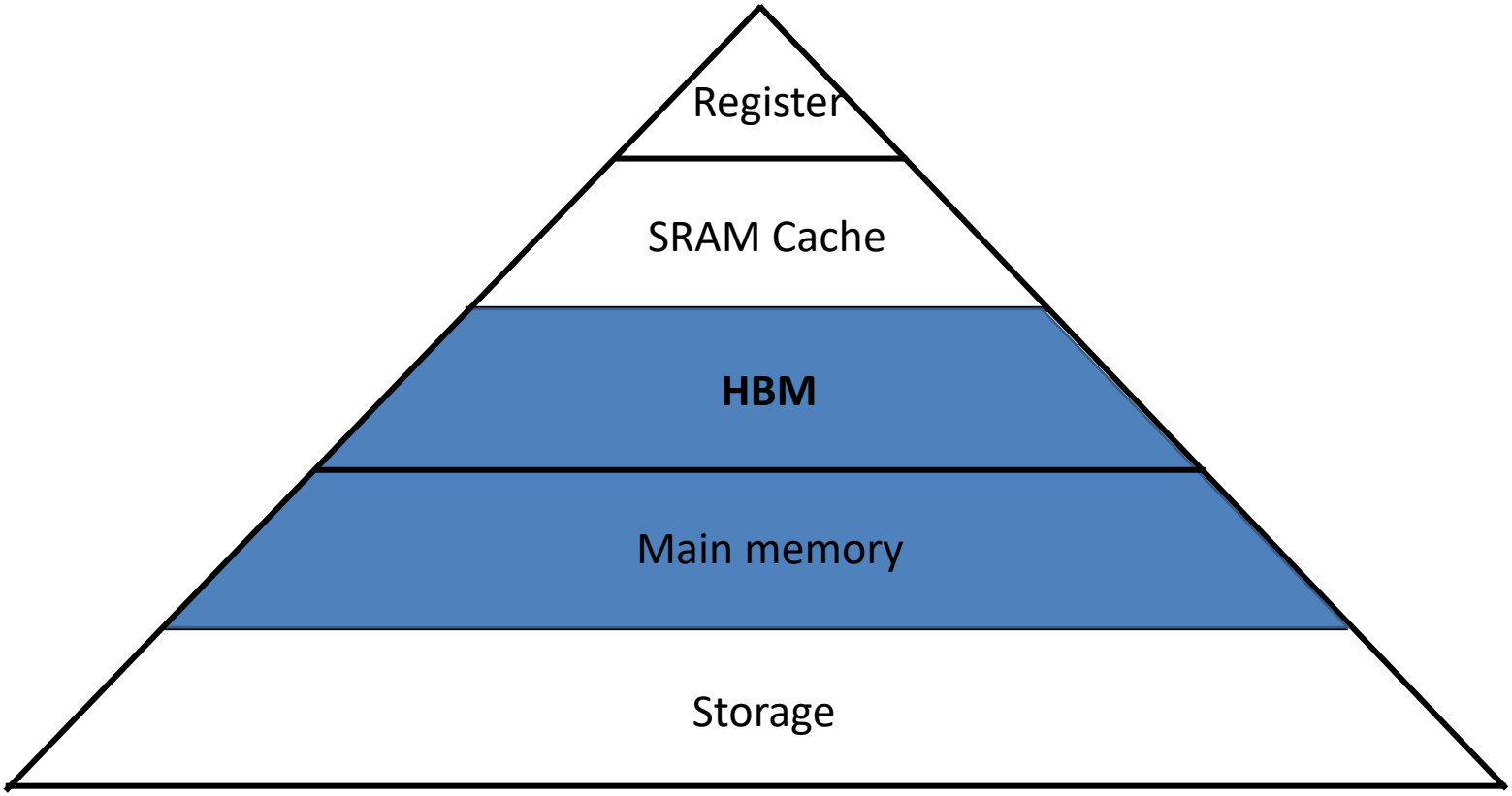


- Some of Intel's processors incorporate **H**igh-**B**andwidth **M**emory (HBM) on the package **to meet the bandwidth demand**.

- Image credit (Intel)

- However, HBM only provides a few tens of GB of capacity, which is **not enough for a server-class system.**

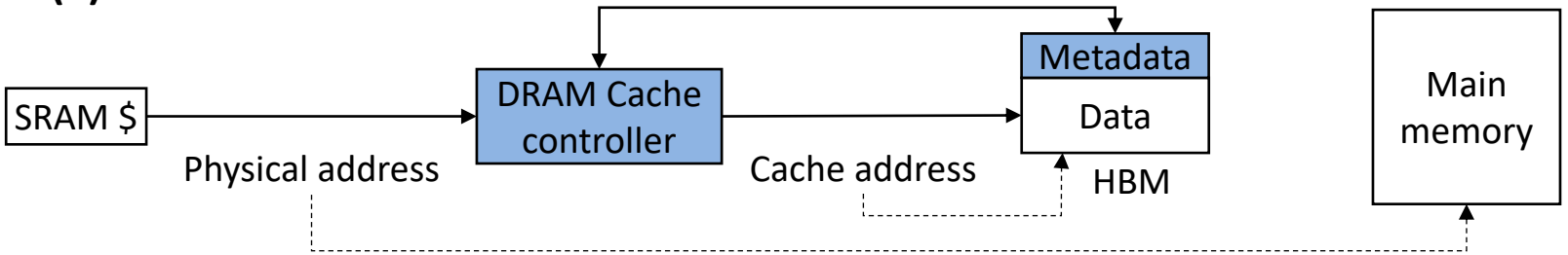
Background: Heterogeneous Memory System



- Therefore, a system still needs large and slow conventional memory along with small and fast HBM.
- Industry and academia have explored architecting HBM as **a DRAM cache of main memory**.

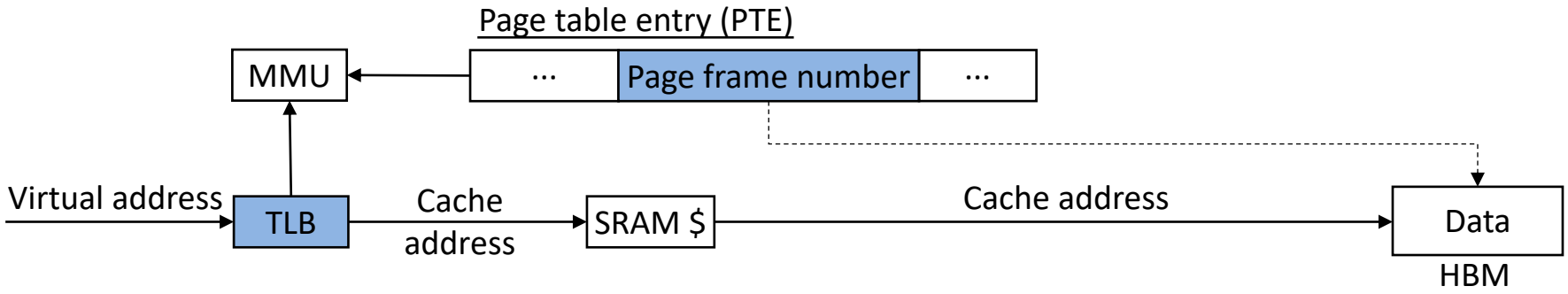
Background: DRAM Cache (DC) Designs

(1) HW-based DRAM cache



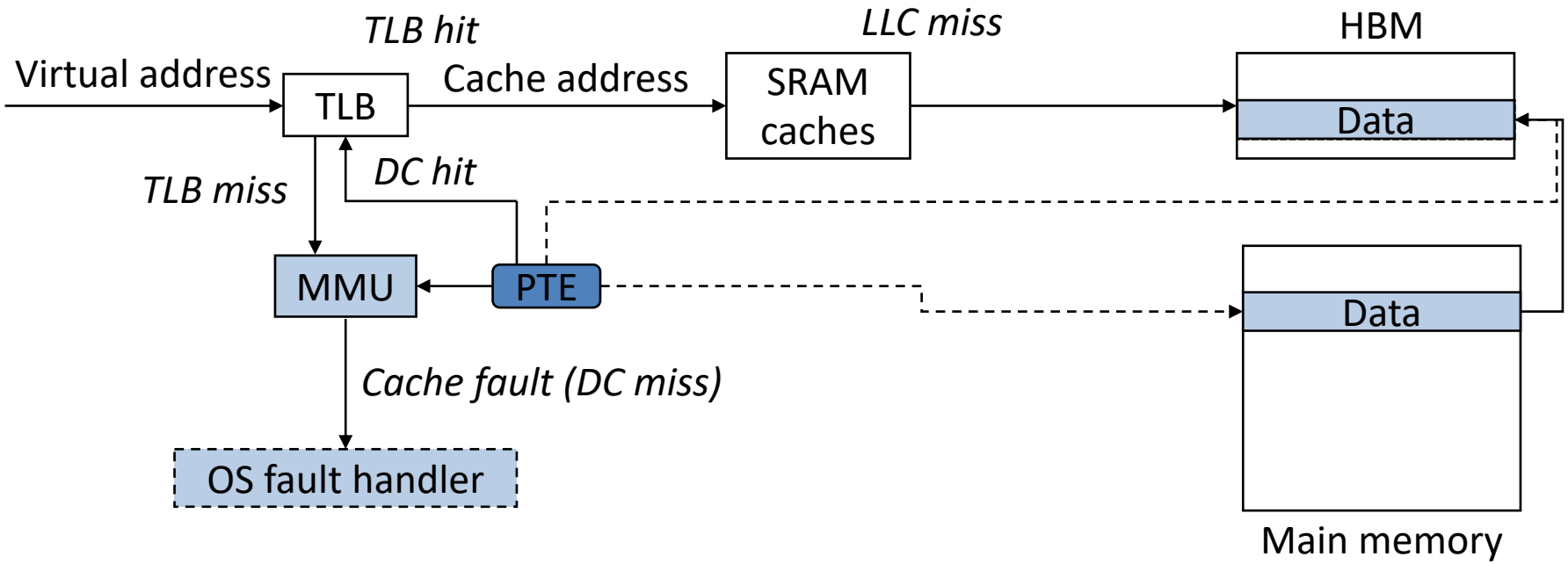
- HW-based cache is **software-transparent** and requires HW **DC controller**.
- It incurs **physical-to-cache** translation overhead.

(2) Page-Table-based (PT-based) DRAM cache

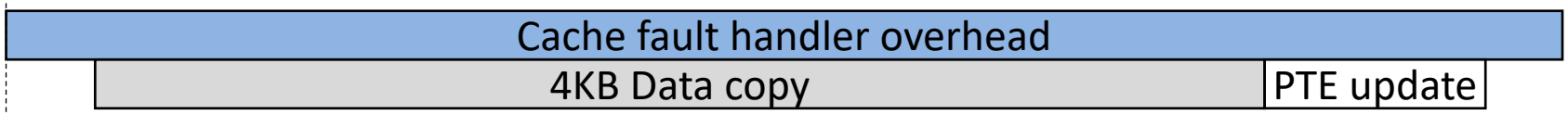


- PT-based cache supports **virtual-to-cache** direct translation and requires **little HW modification**.
- However, PT-based cache suffers from significant **miss handling overhead**.

Why is The Miss Handling Overhead So Big?

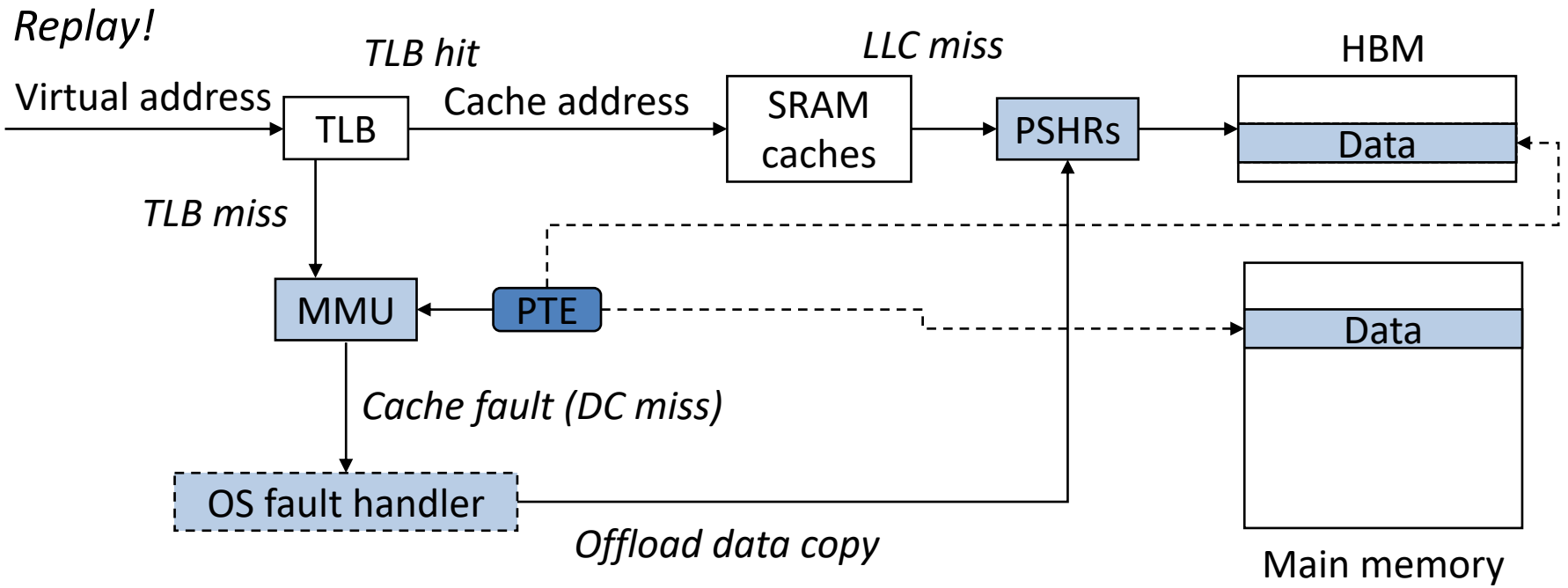


Fault handling overhead

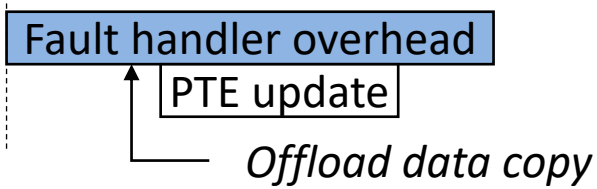


- The cache fault handler **consumes CPU cycles** substantially for copying the data and modifying the metadata**.**

HW Offloading of Data Copy (NOMAD - HPCA'23)



Fault handling overhead



- NOMAD dramatically reduces cache **fault handling latency** by **offloading data copy and verification** tasks **.**

Is NOMAD Tolerant Enough of Cache Misses?

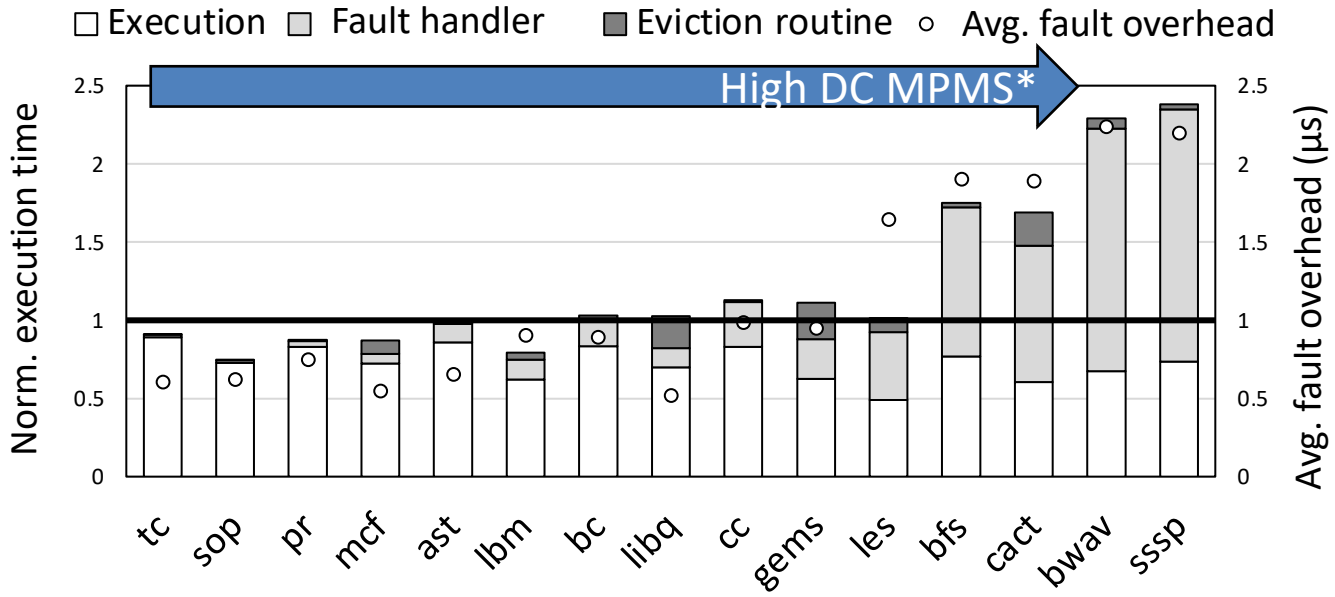
Observation#1: Cache Miss Handling Overhead

- NOMAD still relies on **the SW fault handler** for the cache miss handling, and the fault handler **requires kernel synchronization** to handle duplicated faults from different cores.
- Therefore, average fault handling overhead significantly increases on **concurrent cache misses** from multiple cores.

Observation#2: Replacement Overhead

- PT-based designs leverage **a deferred (or batched) TLB shootdown** technique that evicts multiple pages and raises a single TLB shootdown at the end.
- This approach effectively hides TLB shootdown overhead, but if there are too **many dirty pages** among the victims, TLB flush is deferred too much, and **applications that access victims are blocked** by page faults during the batch eviction process.

Is NOMAD Tolerant Enough of Cache Misses?

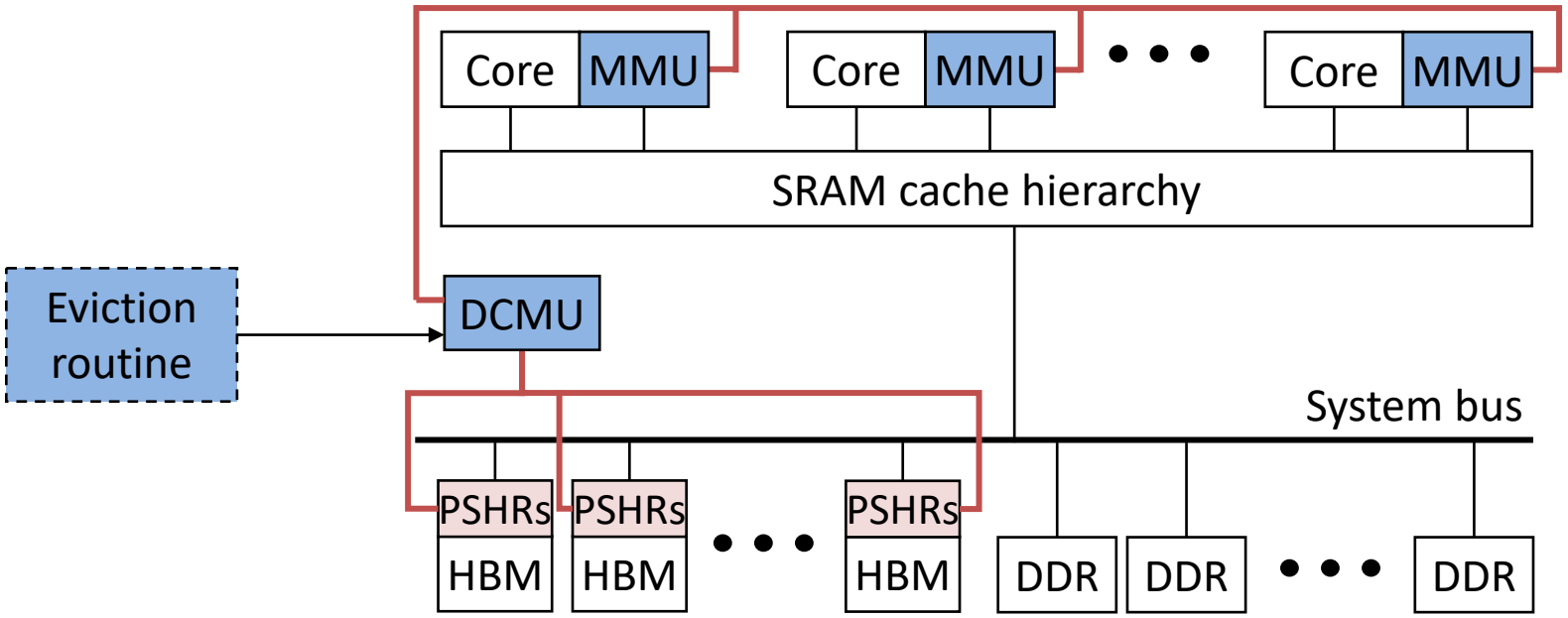


*DC MPMS: DRAM cache misses per microsecond

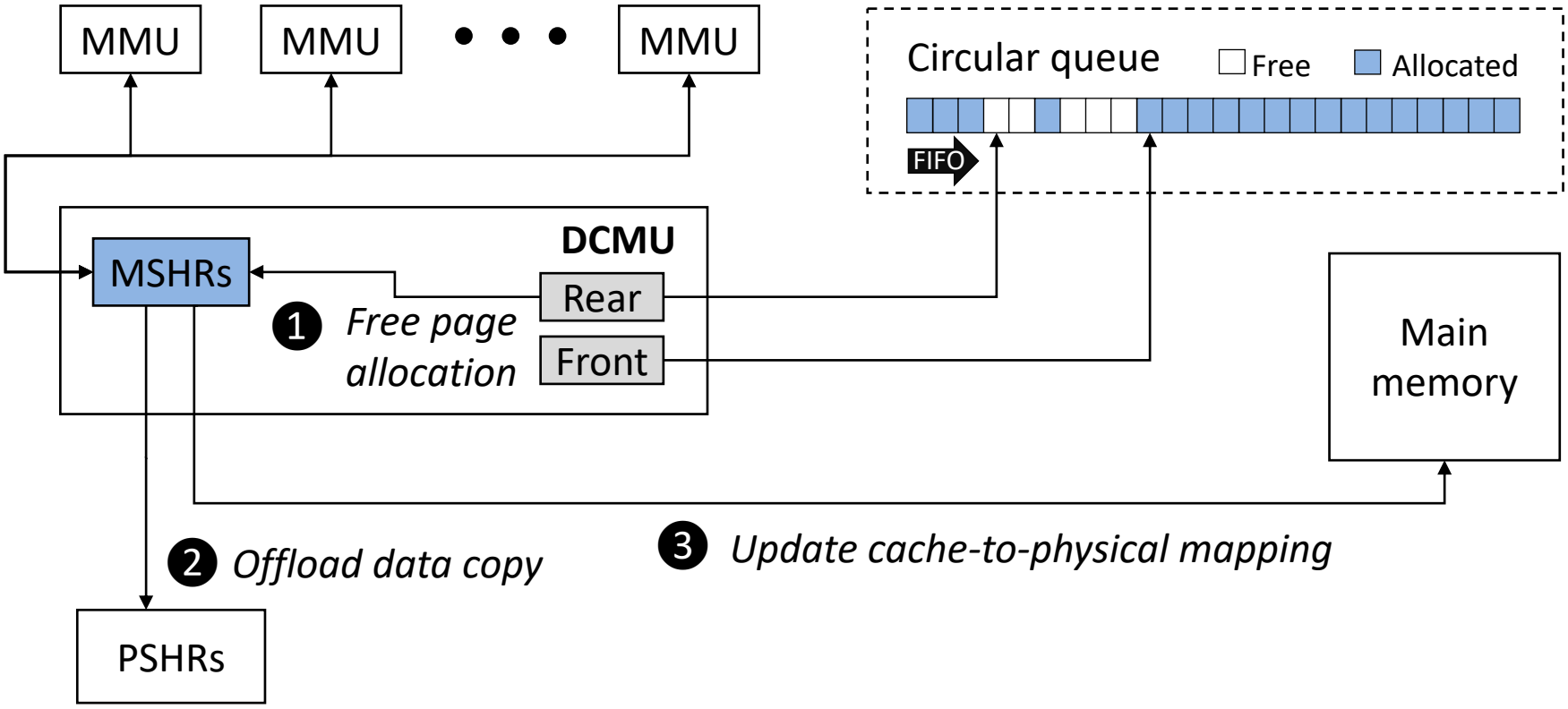
- The **higher the DC MPMS**, the **longer the cache fault overhead** (dots) due to the kernel synchronization overhead. As a result, NOMAD underperforms the baseline for workloads with high MPMS.
- In addition, for workloads with **high dirty page ratios**, NOMAD spends a considerable amount of time on cache eviction.

Genie Cache: System-Level Overview

- Optimization#1: **HW-based cache fault handling**
- Optimization#2: **Pre-writeback** for non-blocking replacement



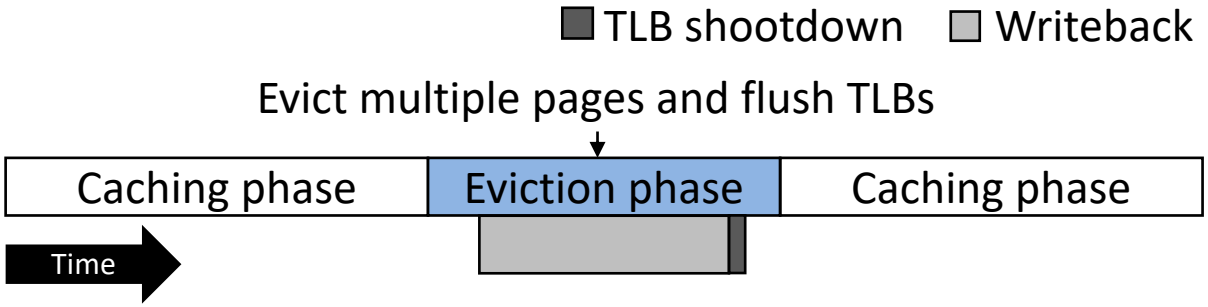
Optimization#1: HW-based Cache Fault Handling



- HW-based fault handling resolves the **trap and synchronization overhead.**
- Therefore, Genie Cache is tolerant of excessive and **concurrent cache misses.**

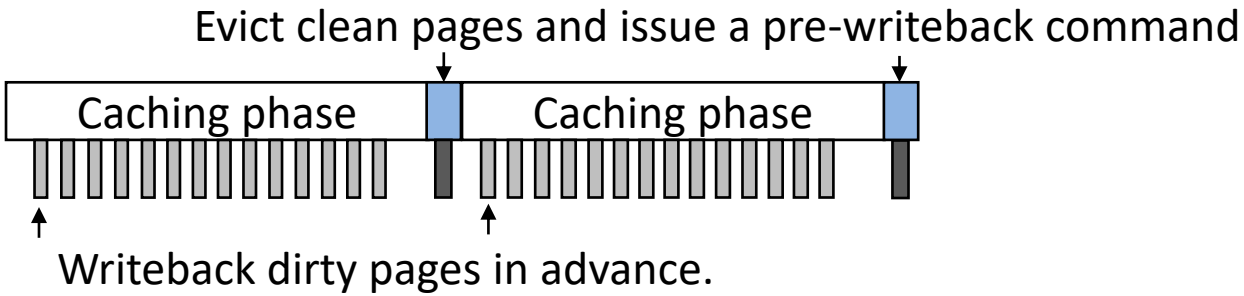
Optimization#2: Pre-Writeback Mechanism

(1) Naïve batch eviction



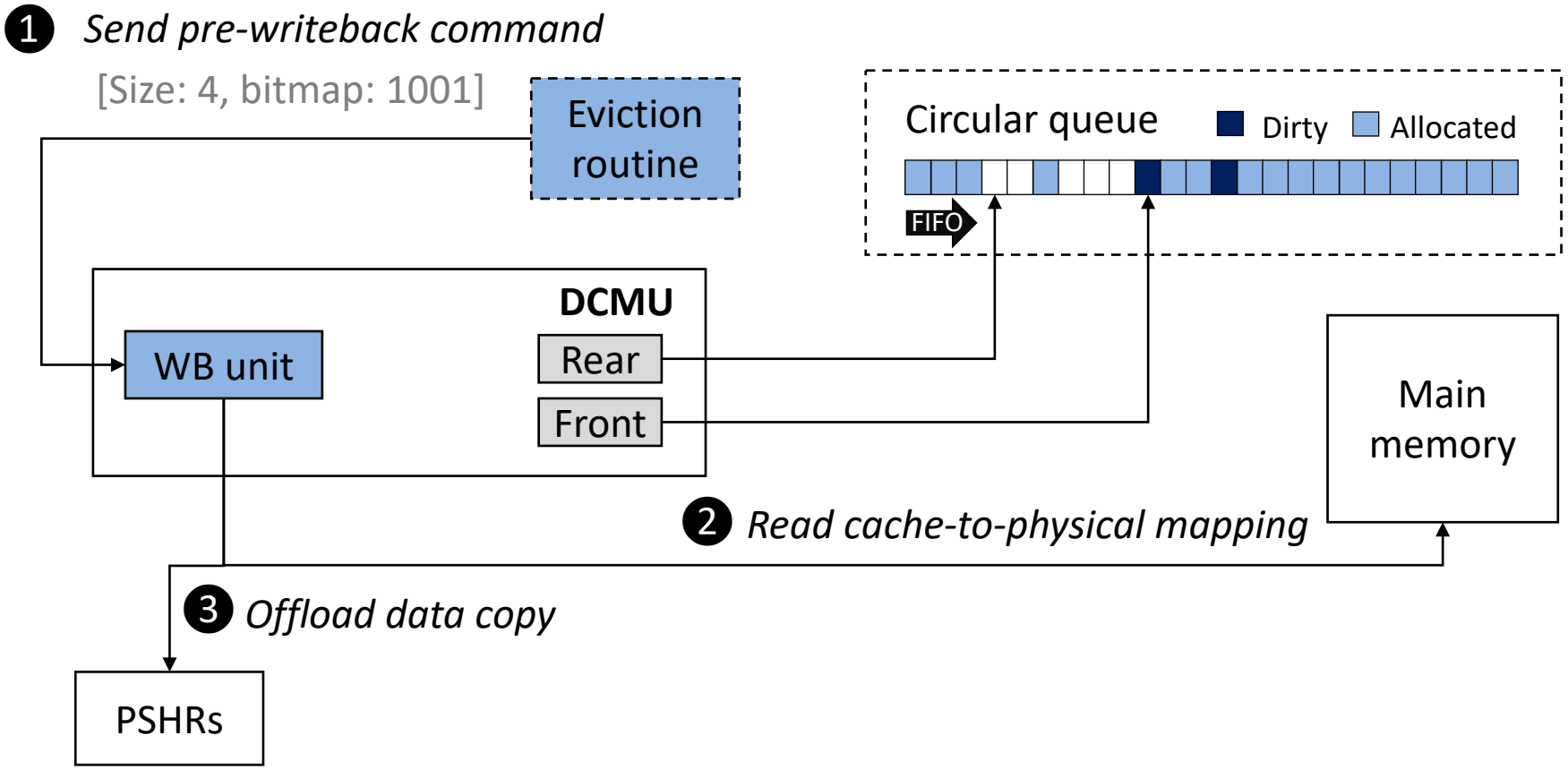
- Naïve batch eviction **effectively hides TLB shootdown overhead**.
- However, if there are too many dirty pages, applications are likely to be stalled by accessing victims.

(2) Pre-writeback



- Since the eviction routine does not copy data while deferring TLB shootdown, it **minimizes application stall time** due to the batch eviction.

Optimization#2: Pre-Writeback Mechanism

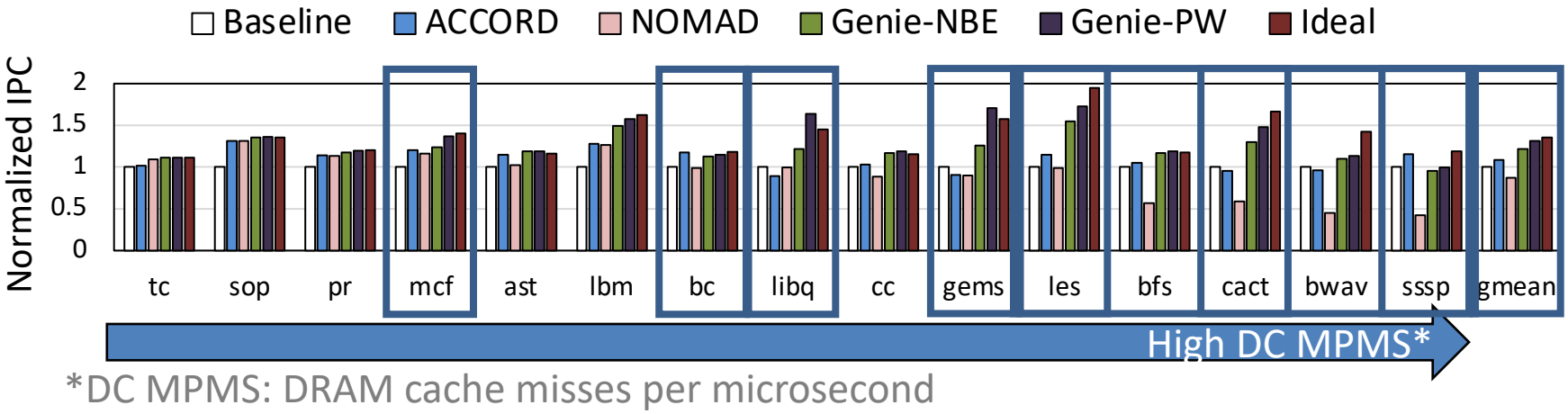


- The eviction routine sends dirty bitmap to the DCMU, and the DCMU writes each dirty pages back.

Experiment environment

- Simulator - gem5 SE mode + DRAMsim3
- Benchmark suites – SPEC CPU@ 2006, GAPBS
- Memory schemes
  - Baseline (a system without HBM)
  - ACCORD (HW-based DRAM cache with 64B line size)
  - NOMAD (state-of-the-art PT-based DRAM cache)
  - Genie-NBE (Genie Cache w/ naïve batch eviction)
  - Genie-PW (Genie Cache w/ pre-writeback)
  - Ideal (PT-based DRAM cache w/ zero fault handling and data copy overhead)

Evaluation results



*DC MPMS: DRAM cache misses per microsecond

- Genie-NBE outperforms NOMAD by **40.2% on average (Opt.#1)**.
- Genie-PW achieves **up to 35% speedup** over Genie-NBE for workloads with high dirty page ratios and improves **average performance by 7.9% (Opt.#2)**.
- For workloads w/ **good temporal** and **bad spatial locality**, HW-based cache achieves near-ideal performance, and DRAM caches with large line sizes like **Genie Cache may not be suitable for such cases.**
- Genie-PW shows better performance than Ideal for workloads with **high fill buffer hit rates (35% of reads are served by fill buffers)**.