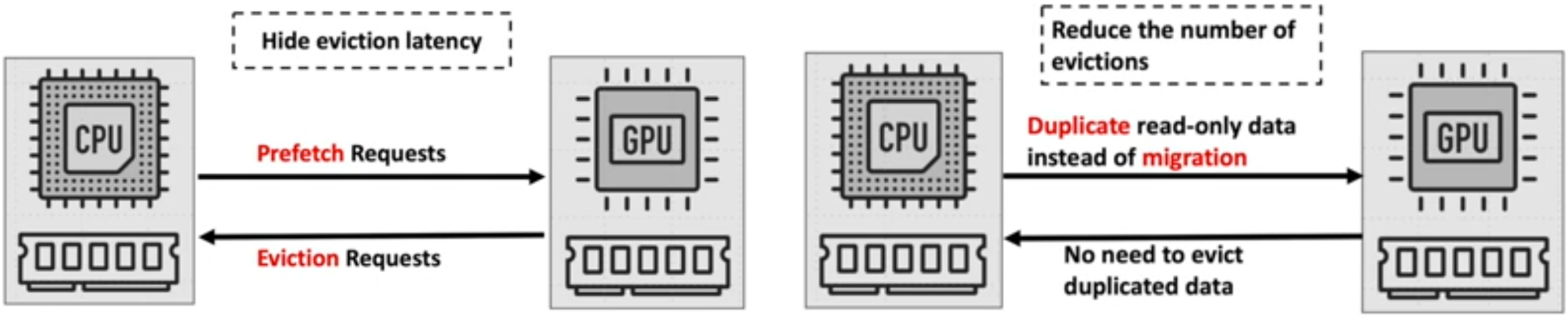
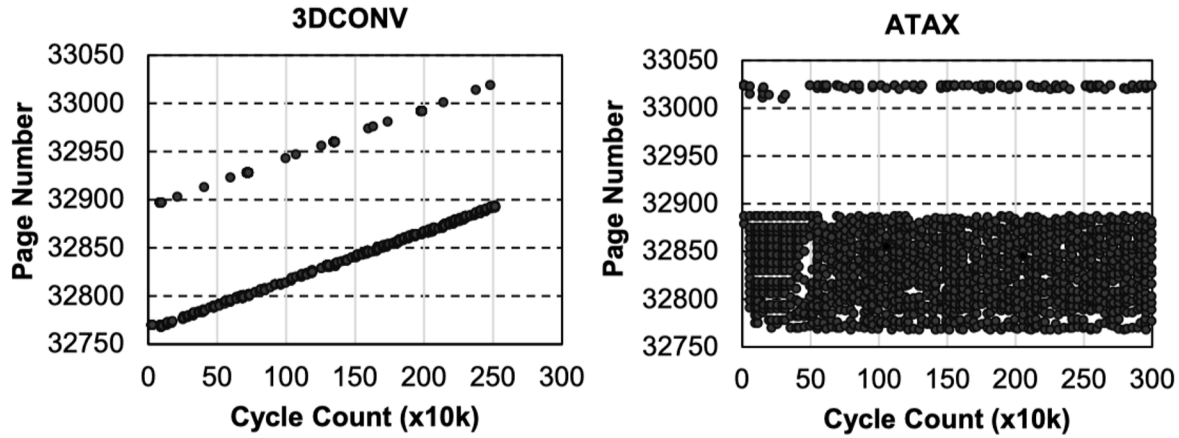
A Framework for Memory Oversubscription Management in Graphics Processing Units

Background

- Modern GPU’ s features:
  - Unified memory
  - Demand paging
- These features free developers from manually managing data movement between the CPU and GPU memory
- But in some kernels it seems not so perfect.
- i.e. GPU kernel working set exceeds the GPU physical memory capacity.

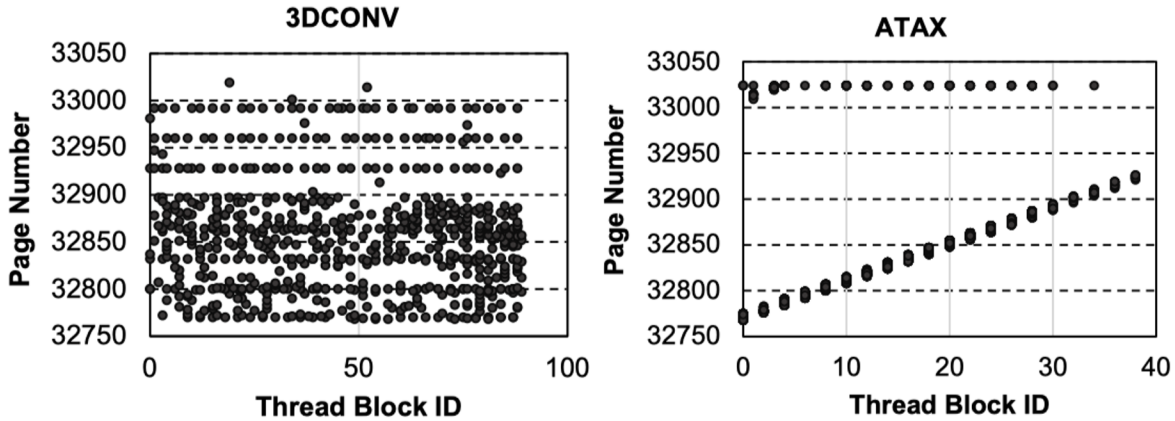


- Drawbacks
  - distinguish between read and write data explicitly.
  - understand and leverage data locality occurring
  - manually manage data migration
  - Even worse in a cloud environment



(a) Regular Application **(b)** Irregular Application

Figure 2. Example page access patterns of (a) a regular (streaming) application, and (b) an irregular (random access) application.



(a) Regular Application **(b)** Irregular Application

Figure 3. Pages accessed by each thread block in example GPGPU applications: (a) regular, (b) irregular.

Key properties

- Different memory access behavior => performance degradation
- Thrashing and long-latency evictions

Solution

- categorize applications into **regular** and **irregular** applications
- propose a memory oversubscription management framework named
- **Eviction-Throttling-Compression, ETC**
  - Application Classification (AC),
  - Proactive Eviction (PE),
  - Memory-aware Throttling (MT),
  - Memory Capacity Compression (CC).

Proactive Eviction

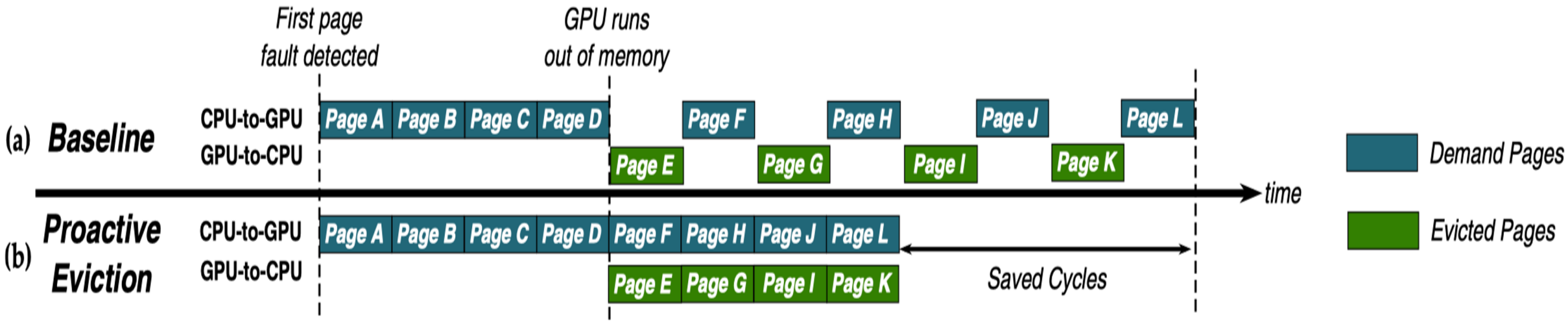


Figure 5. Proactive eviction technique.

Memory-aware Throttling

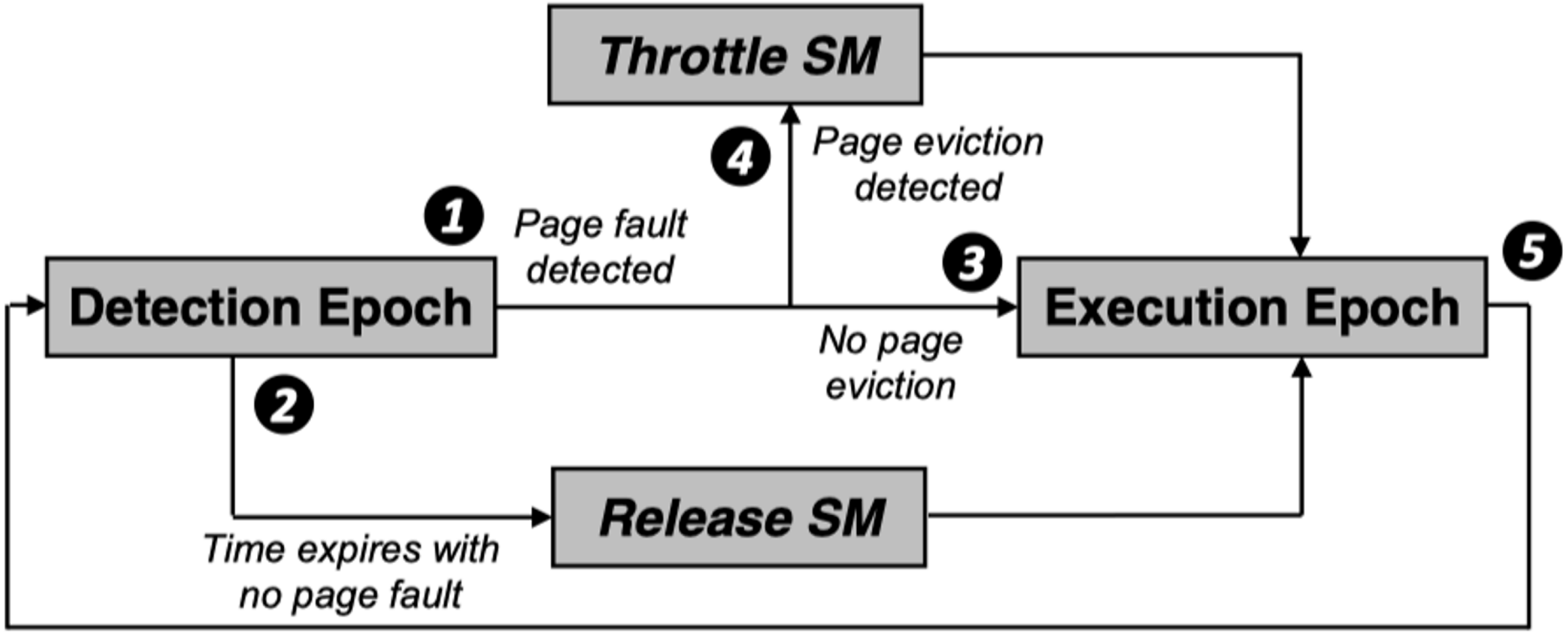


Figure 7. ETC's memory-aware throttling scheme.

Overview

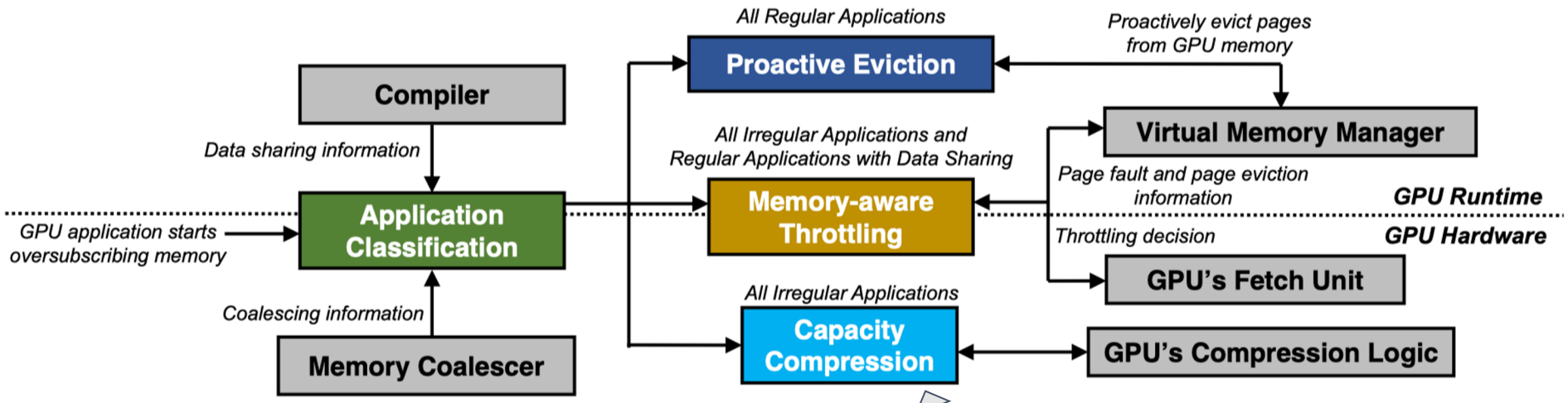


Figure 9. High level overview of ETC showing its four components: Application Classification (**AC**), Proactive Eviction (**PE**), Memory-aware Throttle (**MT**) and Capacity Compression (**CC**).

Linearly Compressed Pages (LCP)