A Framework for Memory Oversubscription Management in Graphics Processing Units

Chen Li, Rachata Ausavarungnirun, Christopher J. Rossbach,
Youtao Zhang, Onur Mutlu, Yang Guo, Jun Yang

Executive Summary

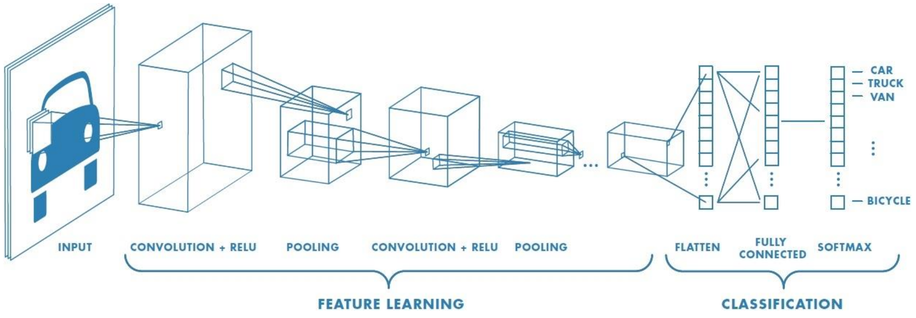
- **Problem:** Memory oversubscription causes GPU performance degradation or, in several cases, crash
- **Motivation:** Prior hand tuning techniques require heavy loads on programmers and have no visibility into other VMs in the cloud

  ➡ Application-transparent mechanisms in GPU are needed
- **Observations:** Different applications have different sources of memory oversubscription overhead
- **ETC**: an application-transparent framework that applies Eviction, Throttling and Compression selectively for different applications
- **Conclusion:** ETC outperforms the state-of-the-art baseline on all different applications

Outline

- Executive Summary
- Memory Oversubscription Problem
- Demand for Application-transparent Mechanisms
- Demand for Different Techniques
- ETC: An Application-transparent Framework
- Evaluation
- Conclusion

Memory Oversubscription Problem



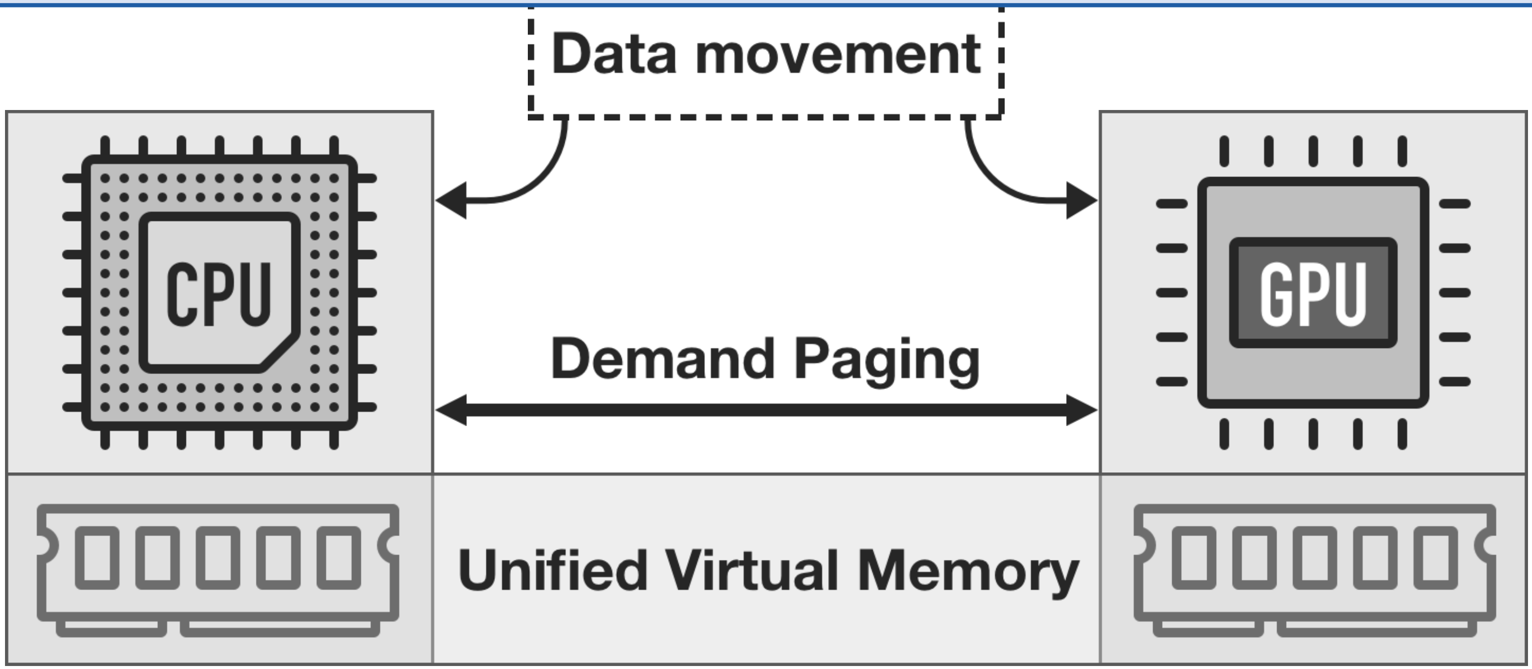
Cloud providers oversubscribe resource for better utilization

DNN training requires larger memory to train larger models

- Limited memory capacity becomes a first-order design and performance bottleneck

Memory Oversubscription Problem

Memory oversubscription causes GPU performance degradation or, in several cases, crash

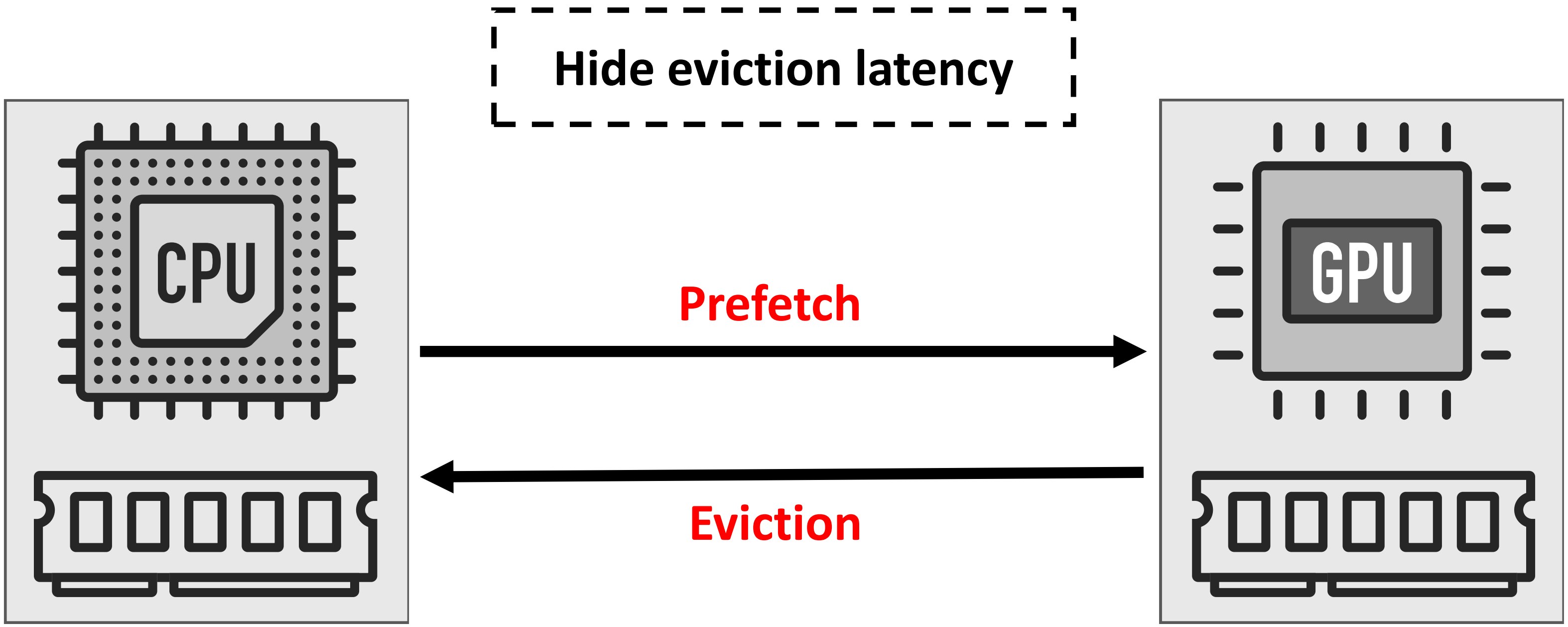


Outline

- Executive Summary
- Memory Oversubscription Problem
- Demand for Application-transparent Mechanisms
- Demand for Different Techniques
- ETC: An Application-transparent Framework
- Evaluation
- Conclusion

Demand for Application-transparent Framework

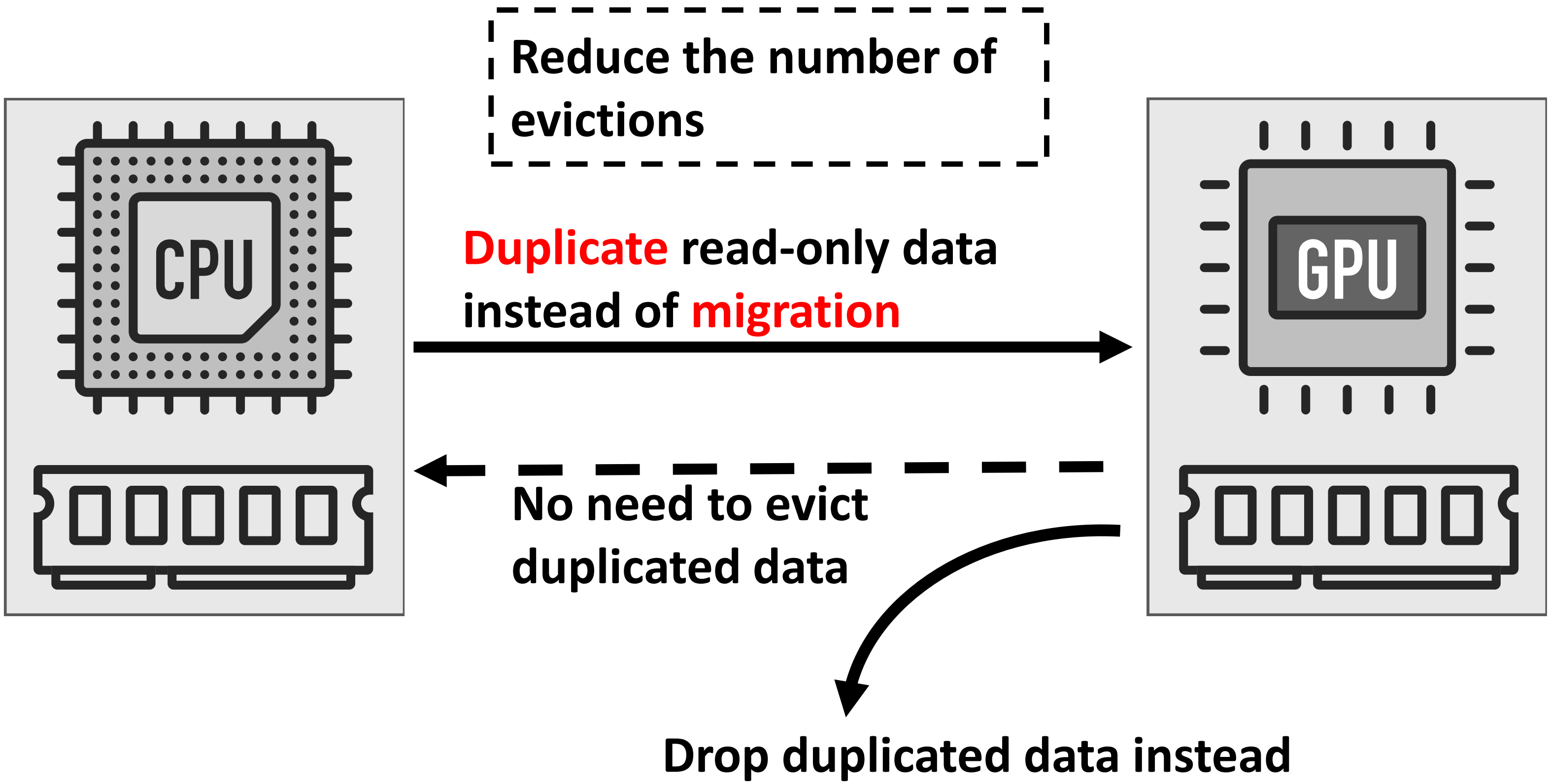
- Prior Hand-tuning Technique 1:
  - Overlap prefetch with eviction requests



Demand for Application-transparent Framework

- Prior Hand-tuning Technique 2:

  - Duplicate read-only data



Demand for Application-transparent Framework

- Prior Hand-tuning Techniques:
  - Overlap prefetch with eviction requests
  - Duplicate read-only data

✘ Requires programmers to manage data movement manually
✘ No visibility into other VMs in cloud environment

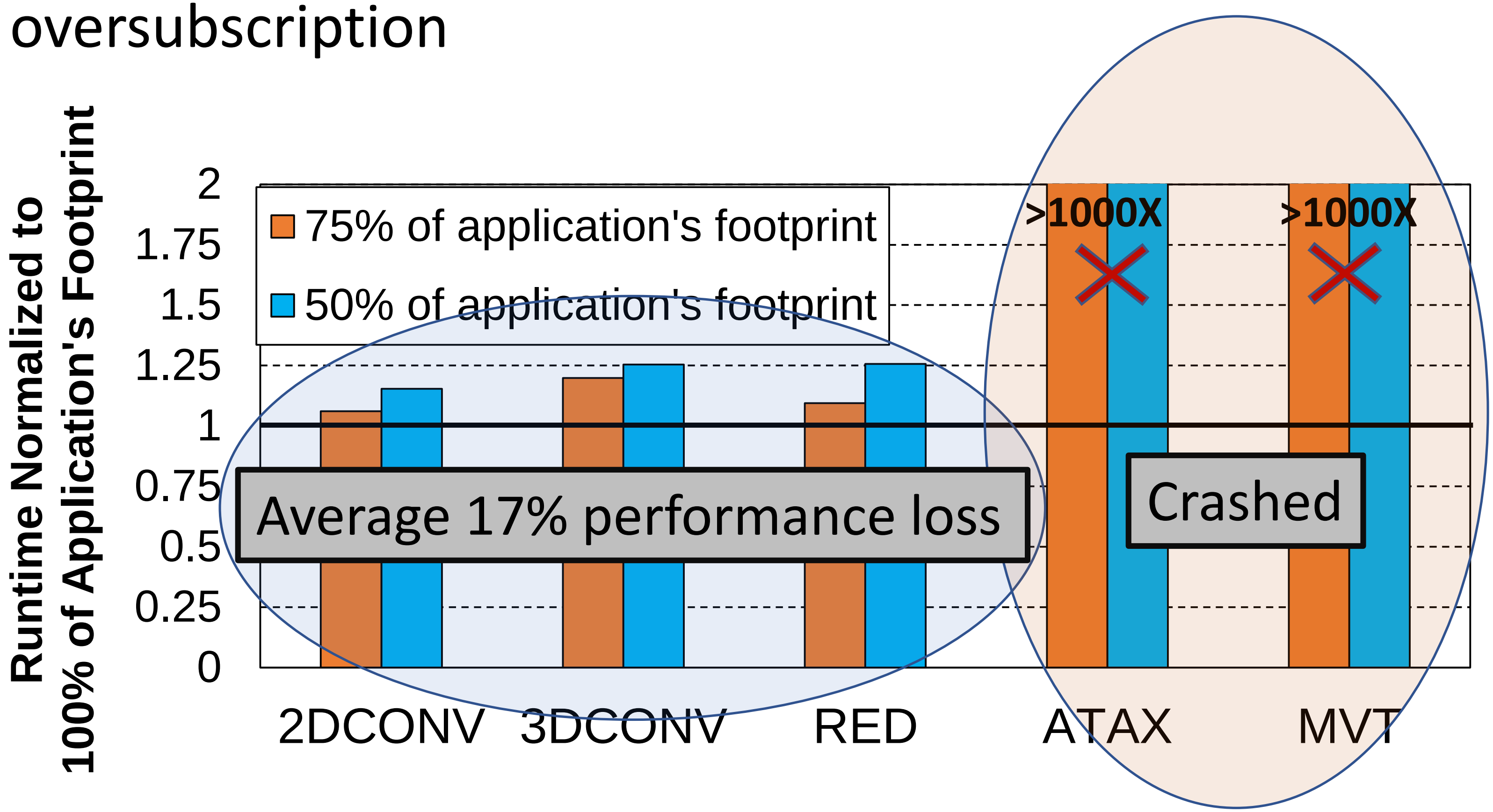
Application-transparent mechanisms are urgently needed

Outline

- Executive Summary
- Memory Oversubscription Problem
- Demand for Application-transparent Mechanisms
- Demand for Different Techniques
- ETC: An Application-transparent Framework
- Evaluation
- Conclusion

Demand for Different Techniques

- Different Applications behave differently under oversubscription

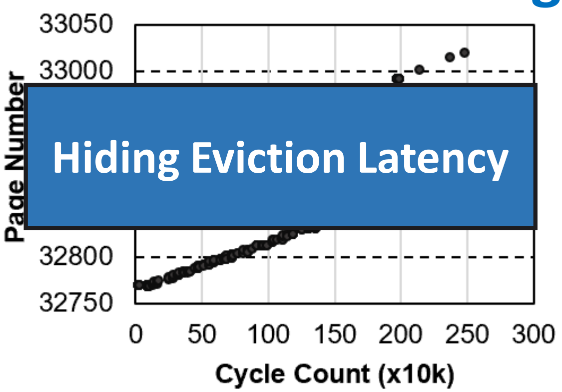


Collected from NVIDIA GTX1060 GPU

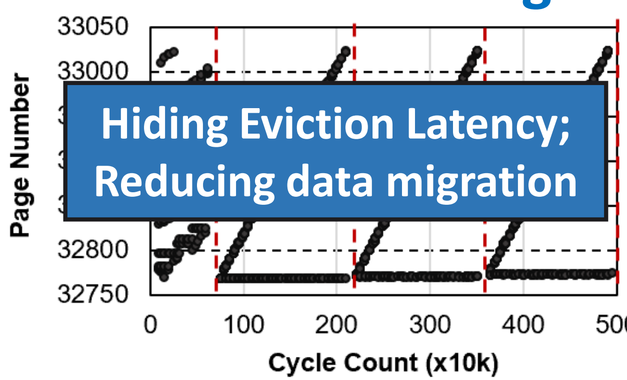
Demand for Different Techniques

- Representative traces of 3 applications

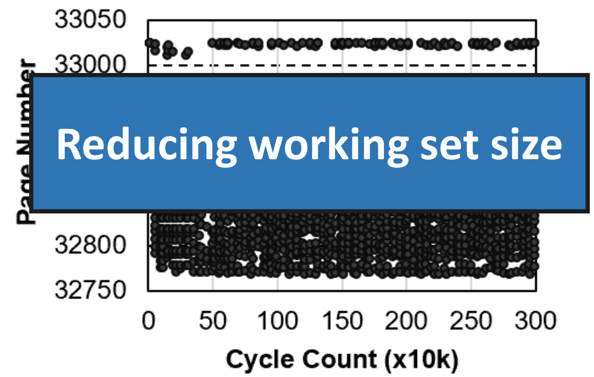
Regular applications with no data sharing



Regular applications with data sharing



Irregular applications



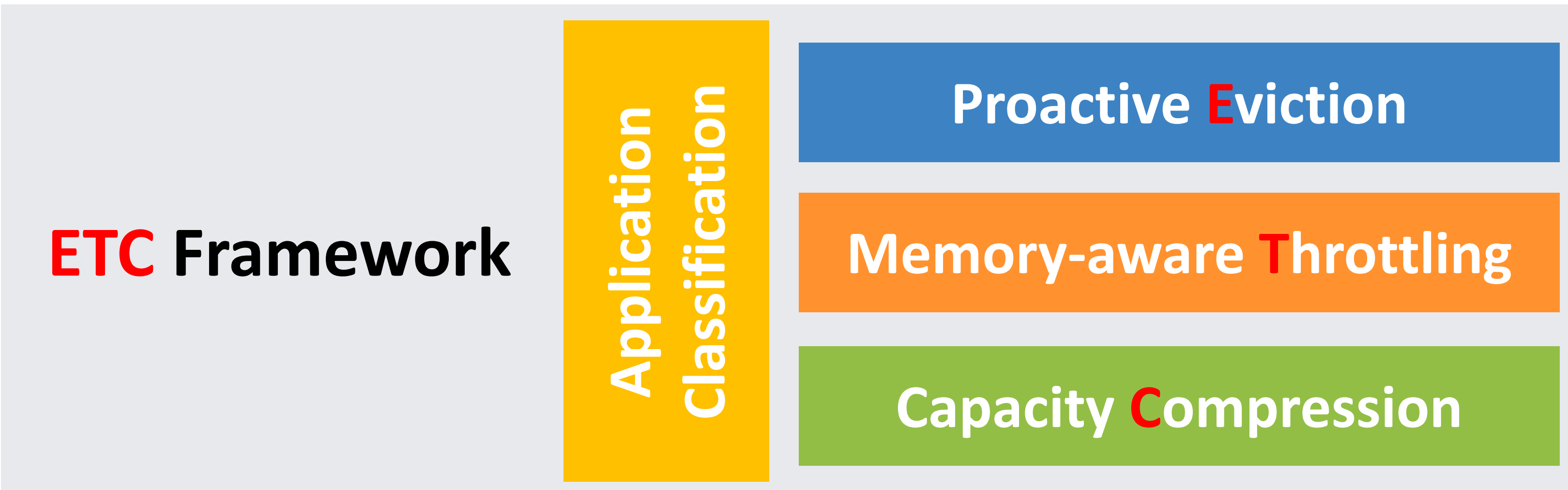
Different techniques are needed to mitigate different sources of overhead

Outline

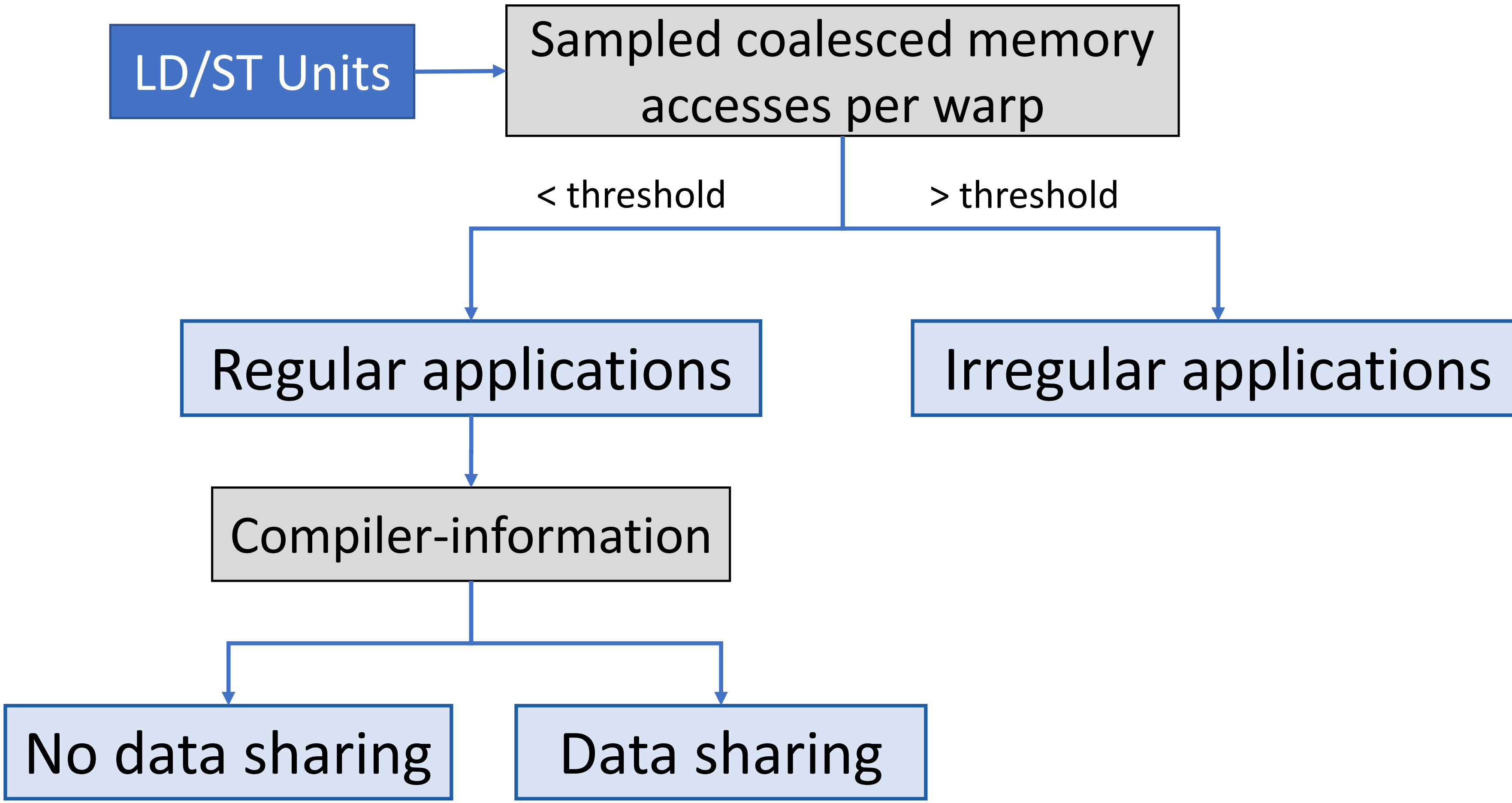
- Executive Summary
- Memory Oversubscription Problem
- Demand for Application-transparent Mechanisms
- Demand for Different Techniques
- ETC: An Application-transparent Framework
- Evaluation
- Conclusion

Our Proposal

- Application-transparent Framework

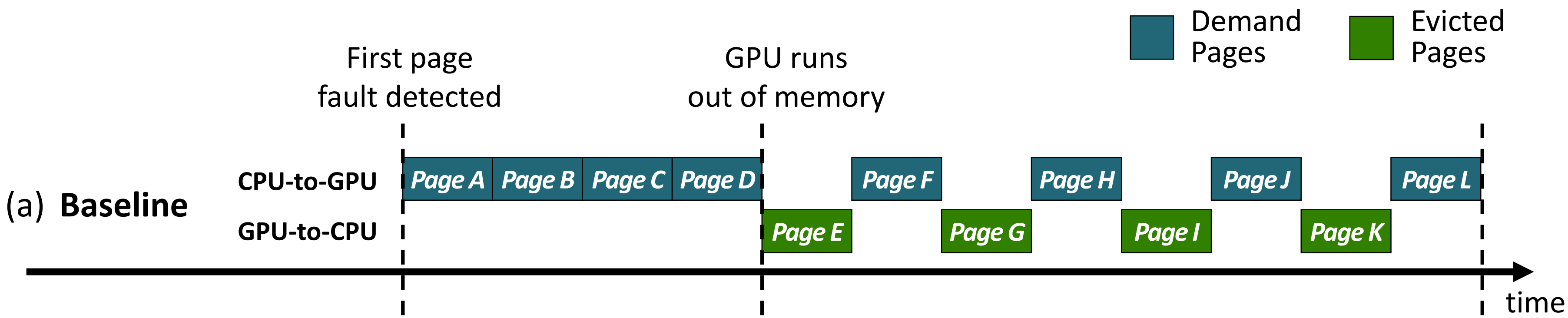


Application Classification

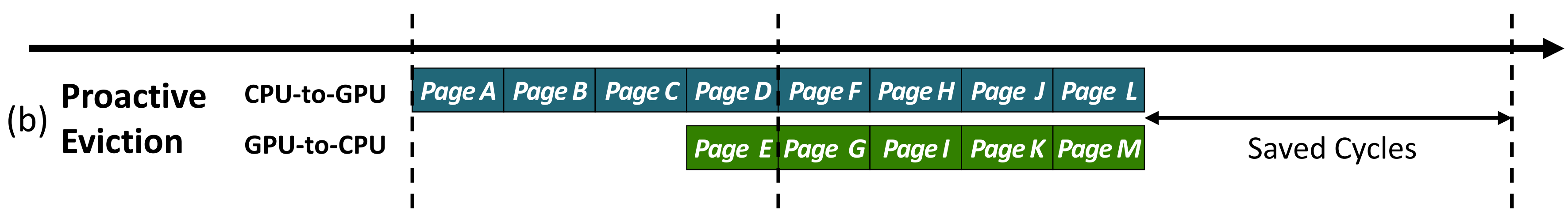


Regular Applications with no data sharing

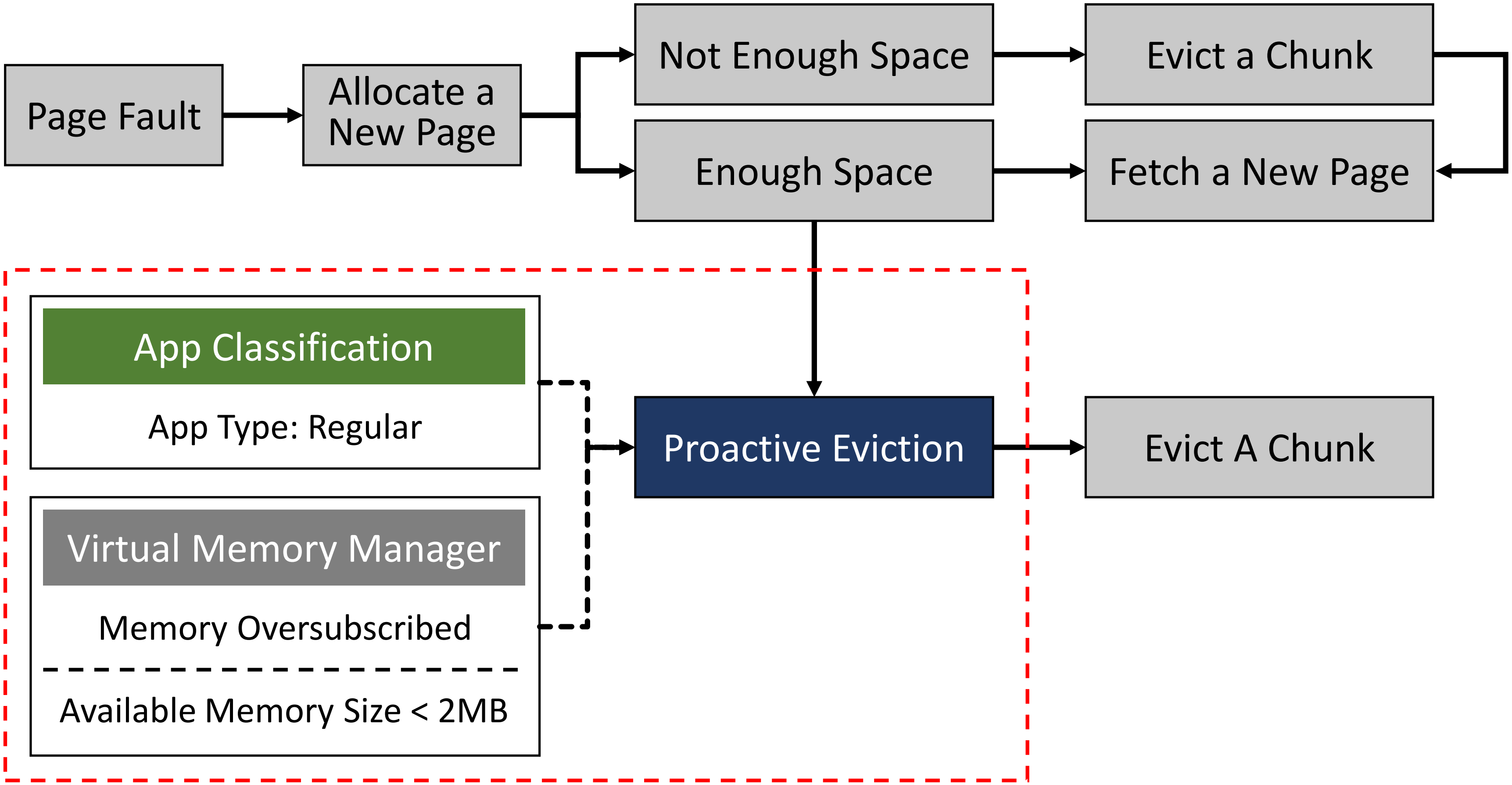
Proactive Eviction → **Waiting for Eviction**



- **Key idea of proactive eviction**: evict pages preemptively before GPU runs out of memory

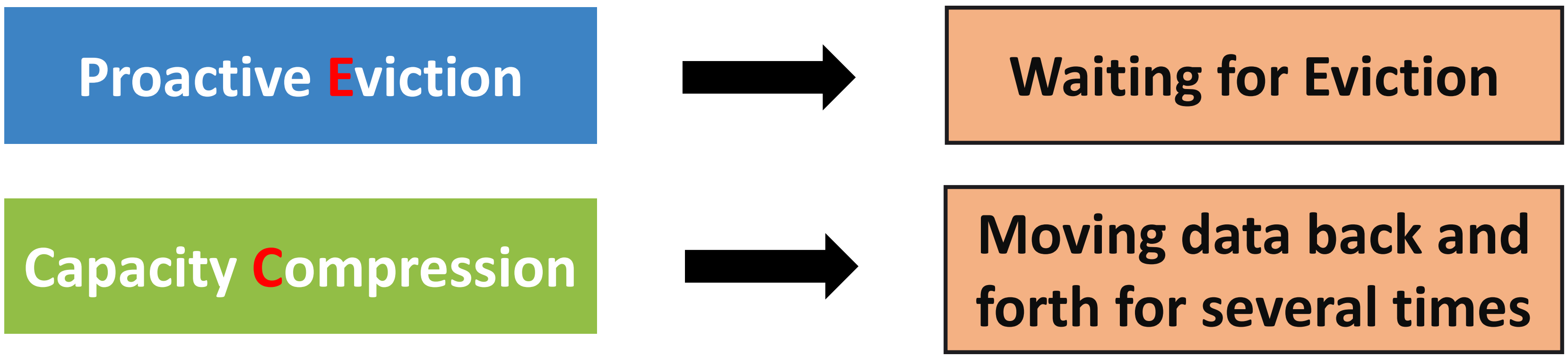


Proactive Eviction



ETC Implementation

Regular Applications with data sharing

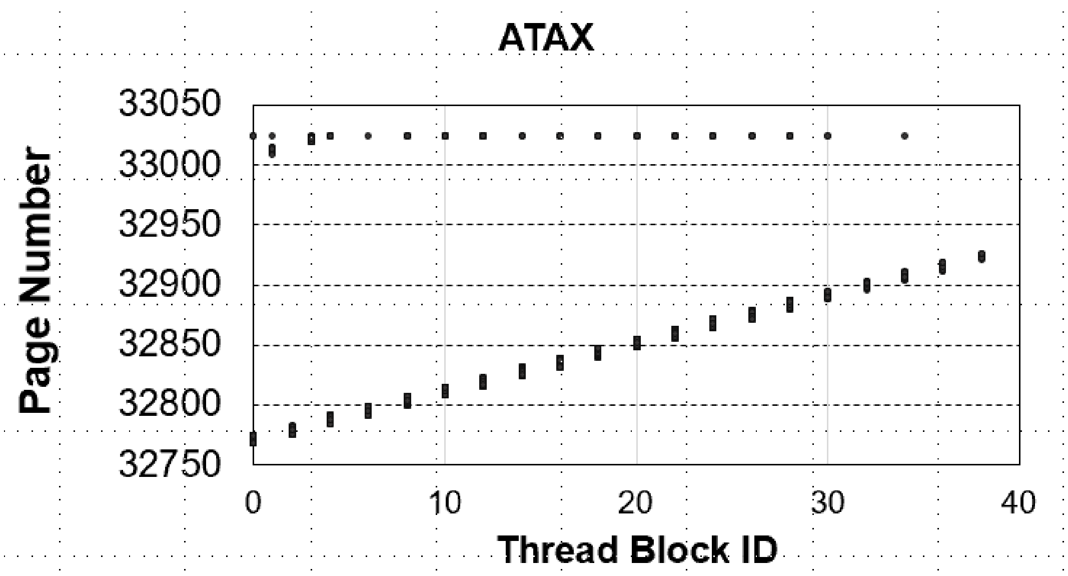


- **Key idea of capacity compression**: Increase the effective capacity to reduce the oversubscription ratio
- Implementation: transplants Linear Compressed Pages (LCP) framework [Pekhimenko et al., MCIRO'13] from a CPU system.

Irregular Applications

Memory-aware Throttling → **Thrashing**

- **Key idea of memory-aware throttling** : reduce the working set size to avoid thrashing

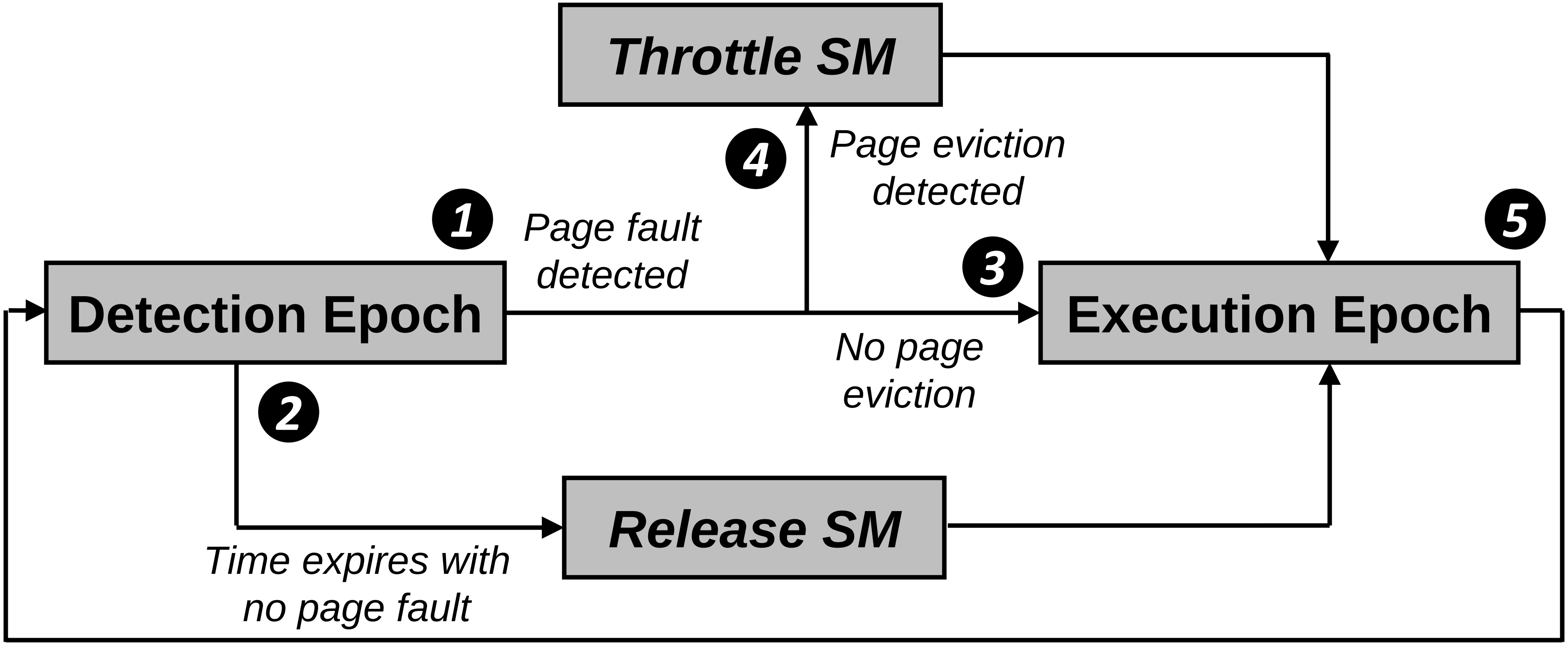


Reduce concurrent running thread blocks

↓

Fit the working set into the memory capacity

Memory-aware Throttling



ETC Implementation (SM Throttling)

Irregular Applications

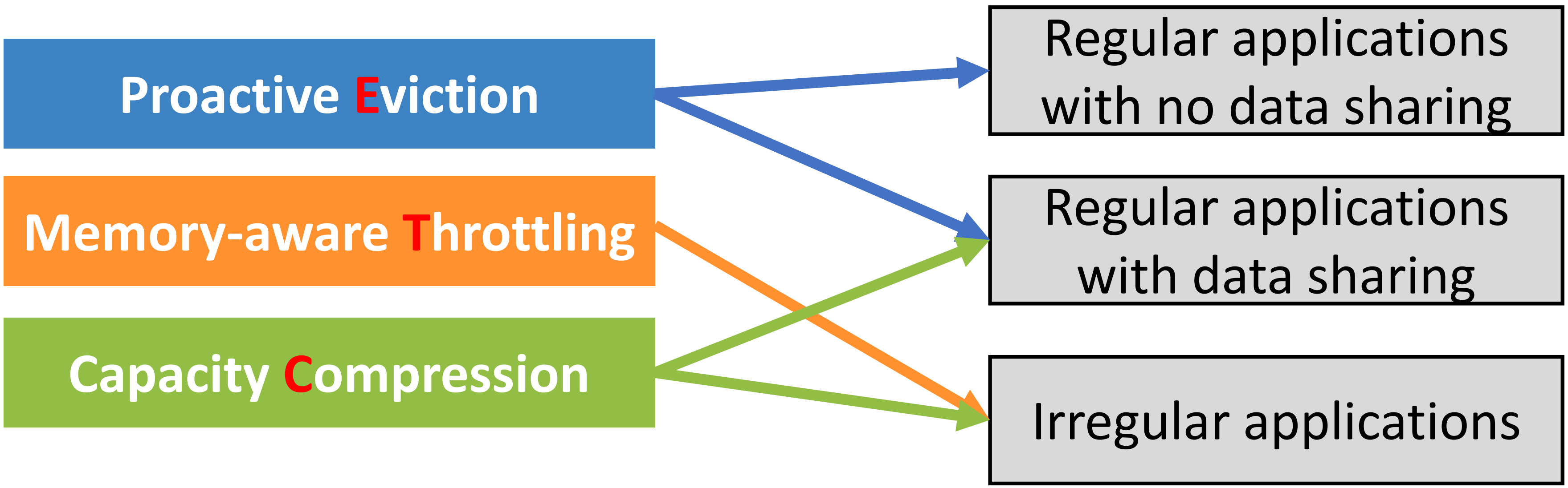
Memory-aware Throttling

Capacity Compression

→ **Thrashing**

Lower Thread Level Parallelism

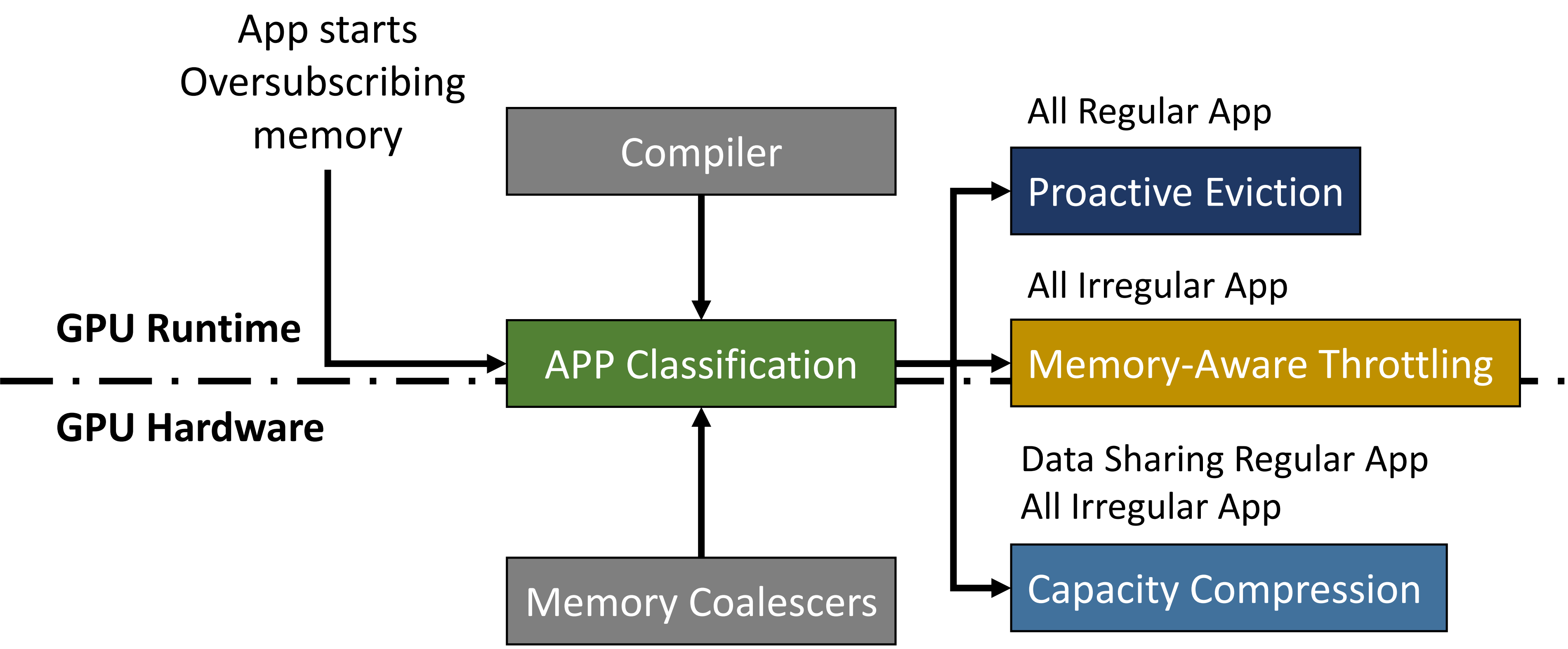
ETC Framework



No single technique can work for all applications

ETC Framework

- Application-transparent Framework



Outline

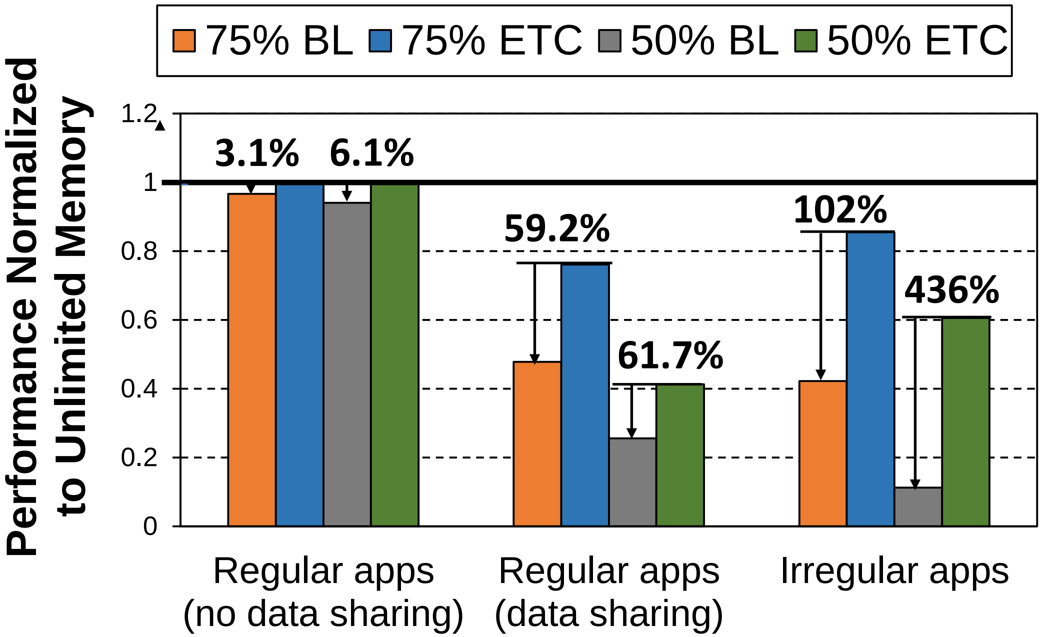
- Executive Summary
- Memory Oversubscription Problem
- Demand for Application-transparent Mechanisms
- Demand for Different Techniques
- ETC: An Application-transparent Framework
- Evaluation
- Conclusion

Methodology

- Mosaic simulation platform [Ausavarungnirun et al., MICRO’17]
  - Based on GPGPU-Sim and MAFIA [Jog et al., MEMSYS ’15]
  - Models demand paging and memory oversubscription support
- Real GPU evaluation
  - NVIDIA GTX 1060 GPU with 3GB memory
- Workloads
  - CUDA SDK, Rodinia, Parboil, and Polybench benchmarks
- Baseline
  - BL: the state-of-the-art baseline with prefetching [Zheng et al., HPCA’16]
  - An ideal baseline with unlimited memory

Performance

- ETC performance normalized to a GPU with unlimited memory



Compared with the state-of-the-art baseline,

Regular applications with no data sharing

Fully mitigates the overhead

Regular applications with data sharing

60.4% of performance improvement

Irregular applications

270% of performance improvement

Other results

- In-depth analysis of each technique
- Classification accuracy results
  - Cache-line level coalescing factors
  - Page level coalescing factors
- Hardware overhead
- Sensitivity analysis results
  - SM throttling aggressiveness
  - Fault latency
  - Compression ratio

Outline

- Executive Summary
- Memory Oversubscription Problem
- Demand for Application-transparent Mechanisms
- Demand for Different Techniques
- ETC: An Application-transparent Framework
- Evaluation
- Conclusion

Conclusion

- **Problem:** Memory oversubscription causes GPU performance degradation or, in several cases, crash
- **Motivation:** Prior hand tuning techniques require heavy loads on programmers and have no visibility into other VMs in the cloud

  ➡ Application-transparent mechanisms in GPU are needed

- **Observations:** Different applications have different sources of memory oversubscription overhead
- **ETC**: an application-transparent framework that
  - Proactive Eviction → Overlaps eviction latency of GPU pages
  - Memory-aware Throttling → Reduces thrashing cost
  - Capacity Compression → Increases effective memory capacity
- **Conclusion:** ETC outperforms the state-of-the-art baseline on all different applications

A Framework for Memory Oversubscription Management in Graphics Processing Units

Chen Li, Rachata Ausavarungnirun, Christopher J. Rossbach,
Youtao Zhang, Onur Mutlu, Yang Guo, Jun Yang