

A Case for Core-Assisted Bottleneck Acceleration in GPUs

*Enabling Flexible Data Compression
with Assist Warps*

Nandita Vijaykumar

Gennady Pekhimenko, Adwait Jog, Abhishek Bhowmick,
Rachata Ausavarangnirun, Chita Das, Mahmut Kandemir,
Todd C. Mowry, Onur Mutlu

SAFARI

Carnegie Mellon

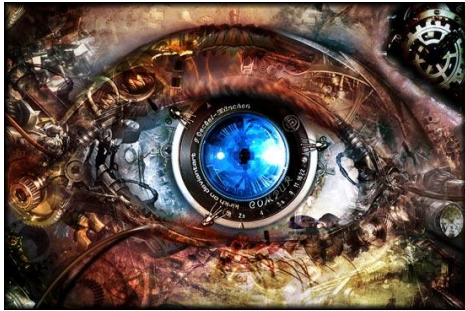
PENNSSTATE



Executive Summary

- **Observation:** Imbalances in execution leave GPU resources underutilized
- **Our Goal:** Employ underutilized GPU resources to do something useful – **accelerate bottlenecks using helper threads**
- **Challenge:** How do you efficiently **manage and use** helper threads in a **throughput-oriented** architecture?
- **Our Solution: CABA (Core-Assisted Bottleneck Acceleration)**
 - ▣ A new framework to enable helper threading in GPUs
 - ▣ Enables flexible data compression to alleviate the memory bandwidth bottleneck
 - ▣ A wide set of use cases (e.g., prefetching, memoization)
- **Key Results:** Using CABA to implement data compression in memory improves performance by 41.7%

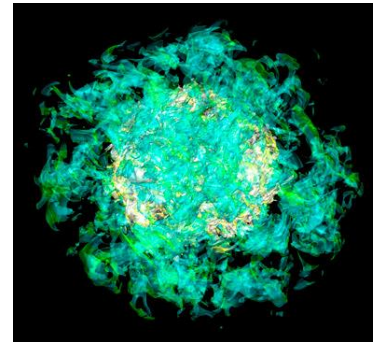
GPUs today are used for a wide range of applications ...



Computer Vision



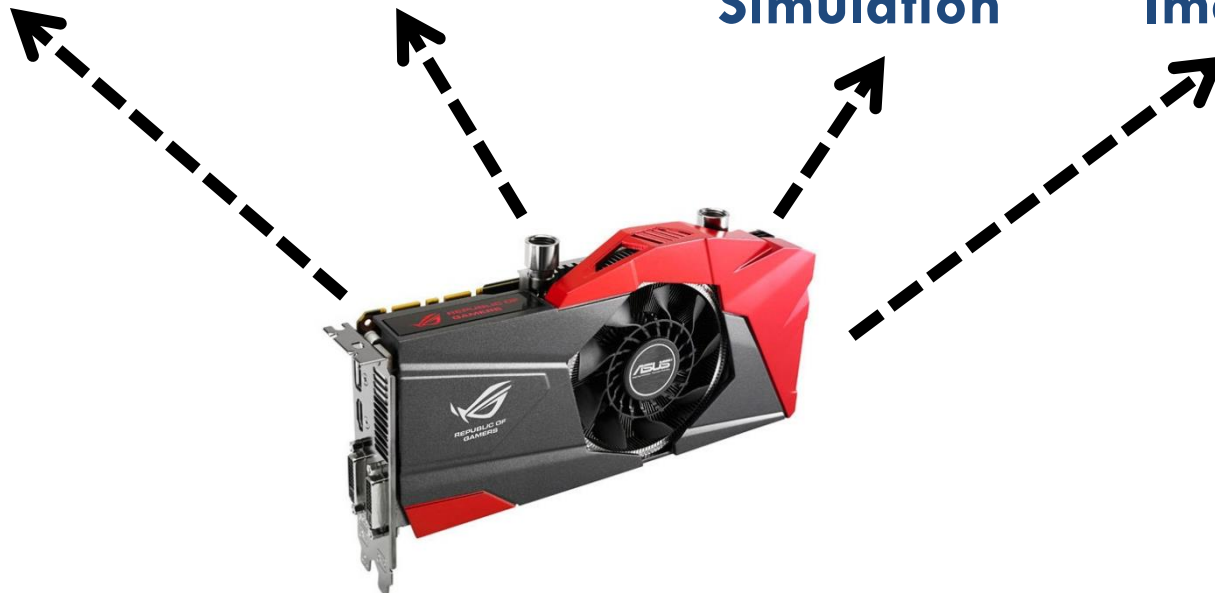
Data Analytics



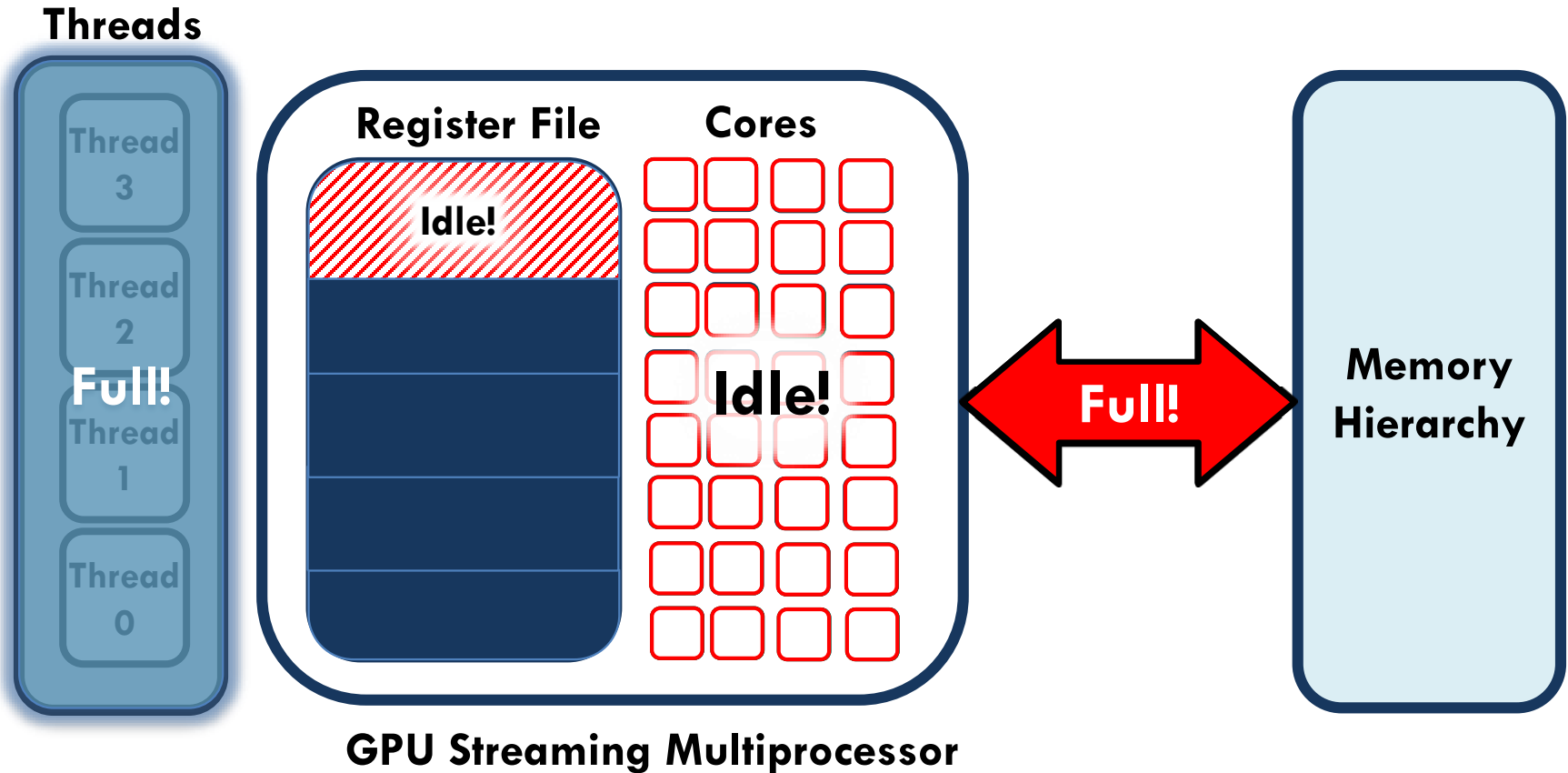
**Scientific
Simulation**



**Medical
Imaging**

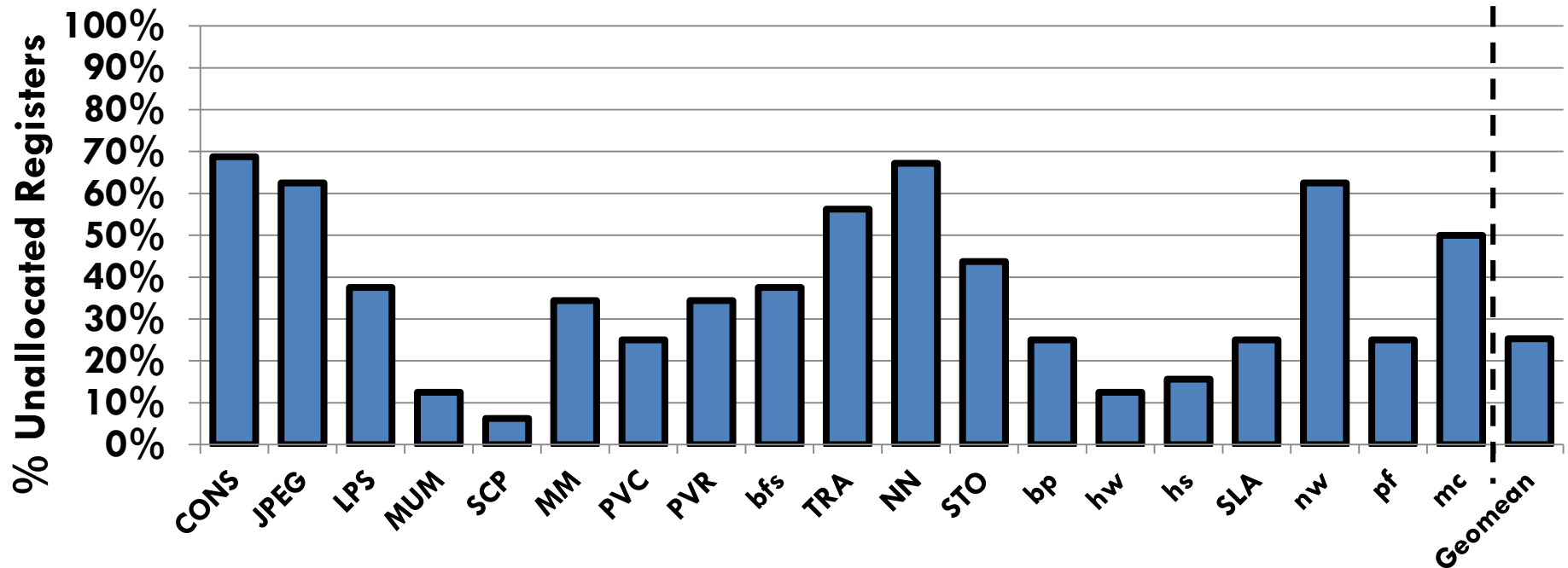


Challenges in GPU Efficiency



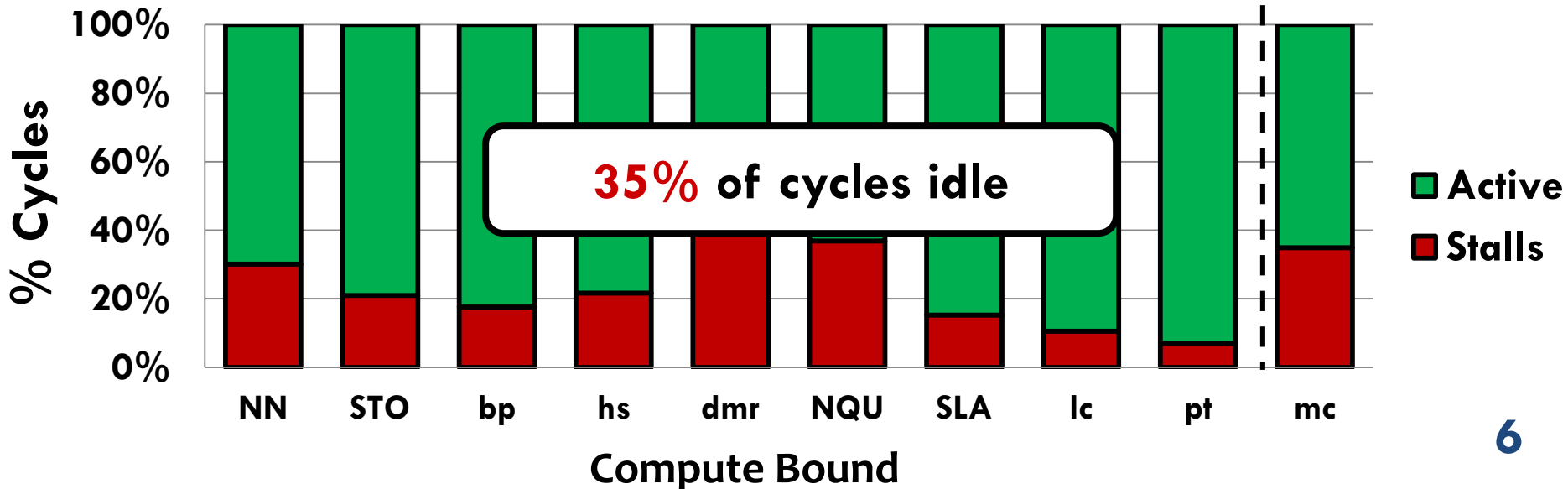
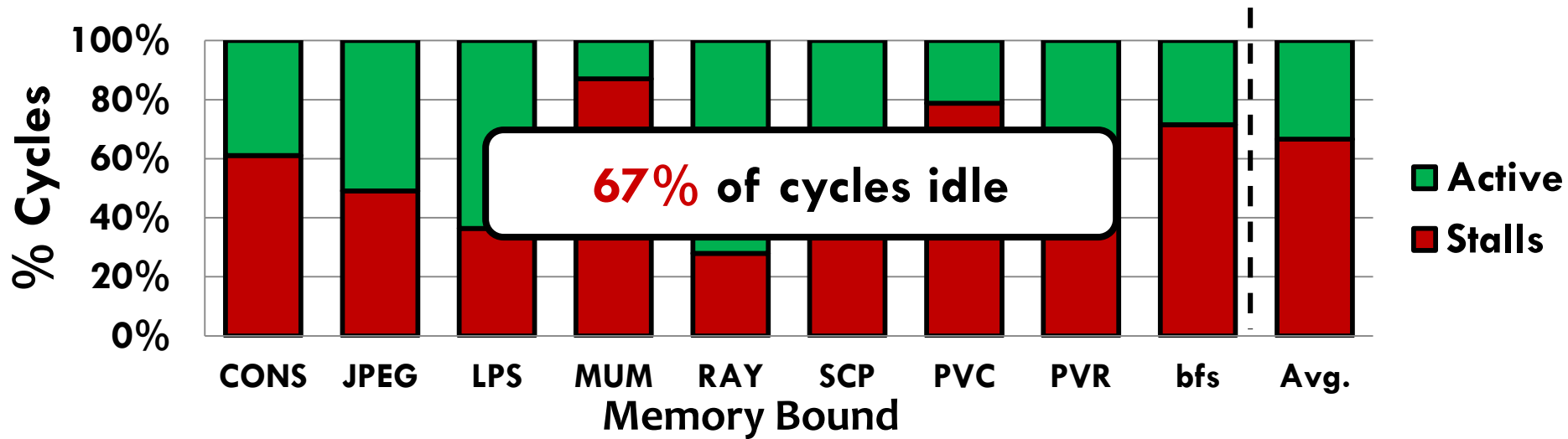
The memory hierarchy has a bottleneck with a full register file

Motivation: Unutilized On-chip Memory



- 24% of the register file is unallocated on average
- Similar trends for on-chip scratchpad memory

Motivation: Idle Pipelines



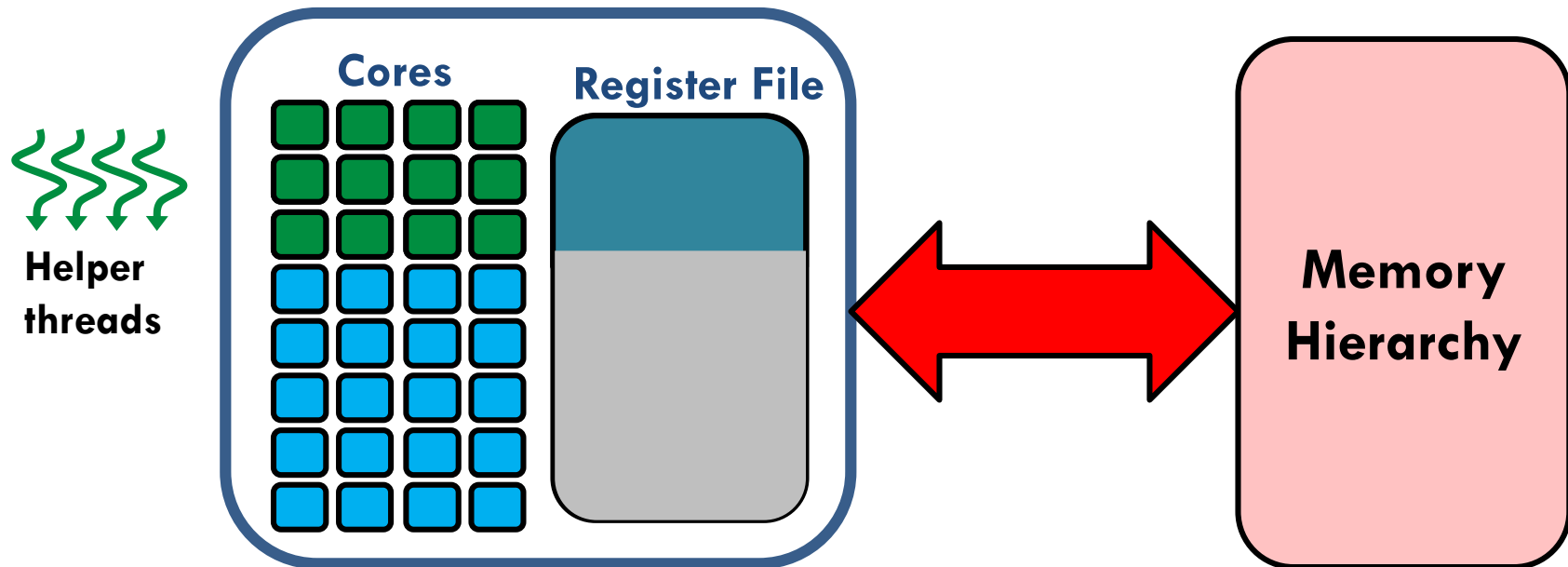
Motivation: Summary

Heterogeneous application requirements lead to:

- **Bottlenecks** in execution
- **Idle** resources

Our Goal

- Use idle resources to do something useful:
accelerate bottlenecks using helper threads



- A flexible framework to enable helper threading in GPUs:
Core-Assisted Bottleneck Acceleration (CABA)

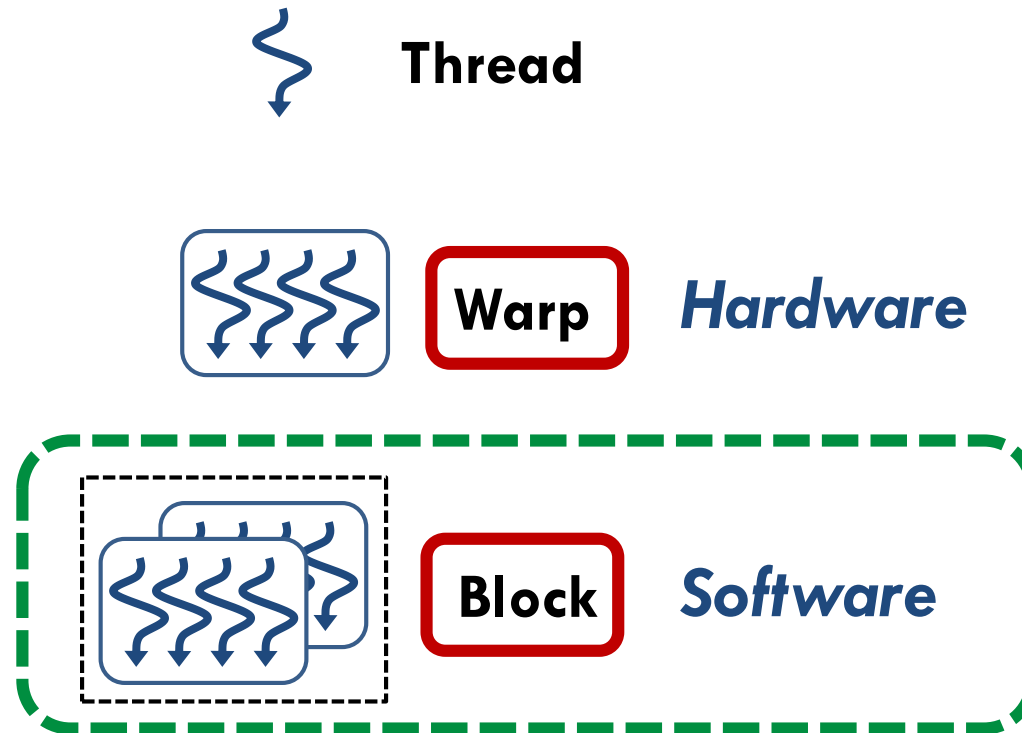
Helper threads in GPUs

- Large body of work in CPUs ...
 - ▣ [Chappell+ ISCA '99, MICRO '02], [Yang+ USC TR '98], [Dubois+ CF '04], [Zilles+ ISCA '01], [Collins+ ISCA '01, MICRO '01], [Aamodt+ HPCA '04], [Lu+ MICRO '05], [Luk+ ISCA '01], [Moshovos+ ICS '01], [Kamruzzaman+ ASPLOS '11], etc.
- **However, there are new challenges with GPUs...**

Challenge

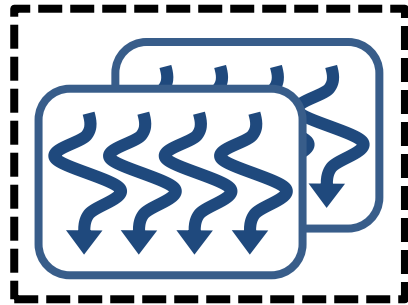
How do you efficiently
manage and use helper threads
in a throughput-oriented architecture?

Managing Helper Threads in GPUs

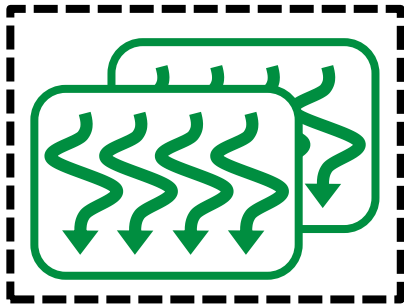


Where do we add helper threads?

Approach #1: Software-only



Regular threads

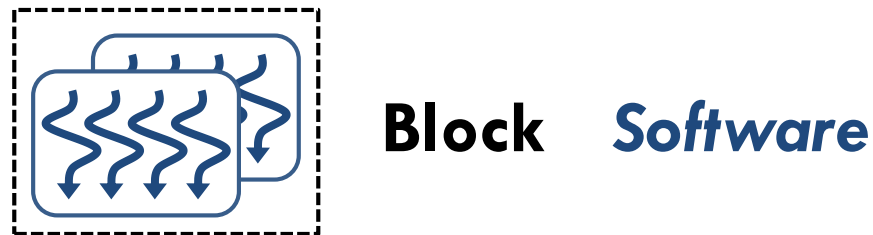


Helper threads

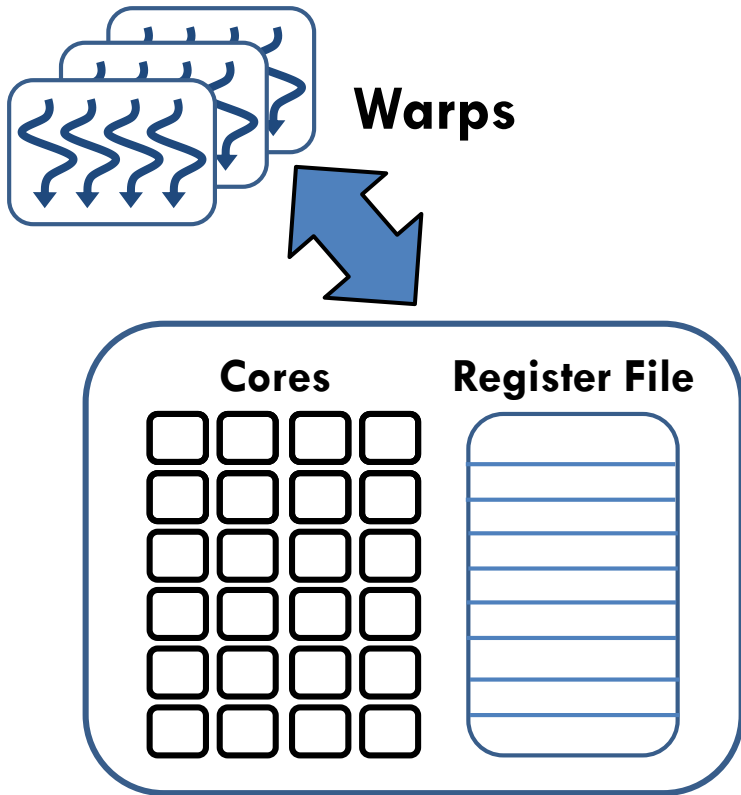
- ✓ No hardware changes
- ✗ Coarse grained
- ✗ Synchronization is difficult
- ✗ Not aware of runtime program behavior

Where Do We Add Helper Threads?

 **Thread**



Approach #2: Hardware-only

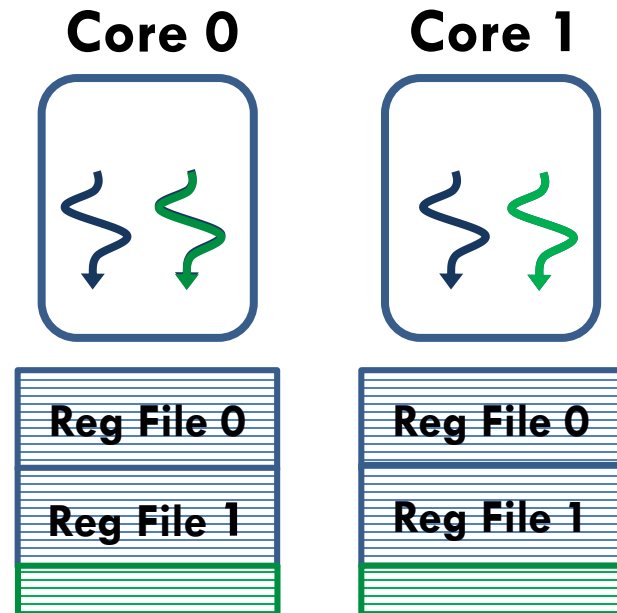


GPU

✗ Providing contexts efficiently is difficult

- ✓ **Fine-grained control**
- **Synchronization**
- **Enforcing Priorities**

CPU



CABA: An Overview

- **“Tight coupling”** of helper threads and regular threads
 - ✓ Efficient context management
 - ✓ Simpler data communication

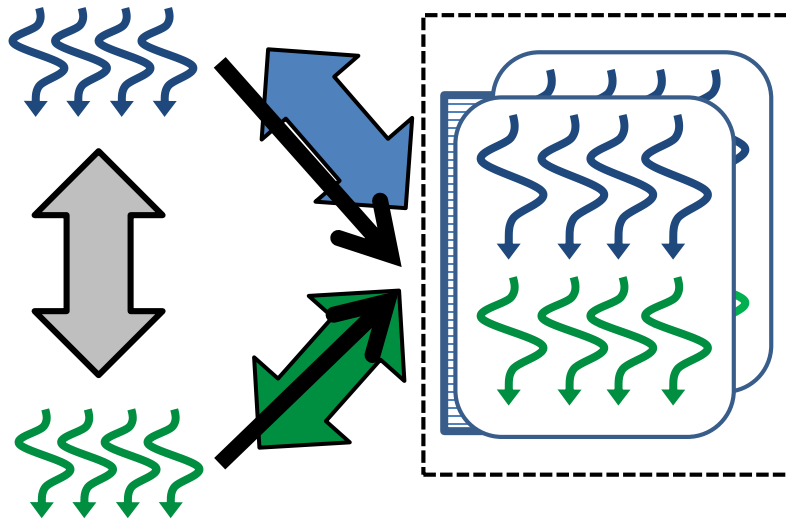
SW

HW

- **“Decoupled management”** of helper threads and regular threads
 - ✓ Dynamic management of threads
 - ✓ Fine-grained synchronization

CABA: 1. In Software

Regular threads



Helper threads

Block

Helper threads:

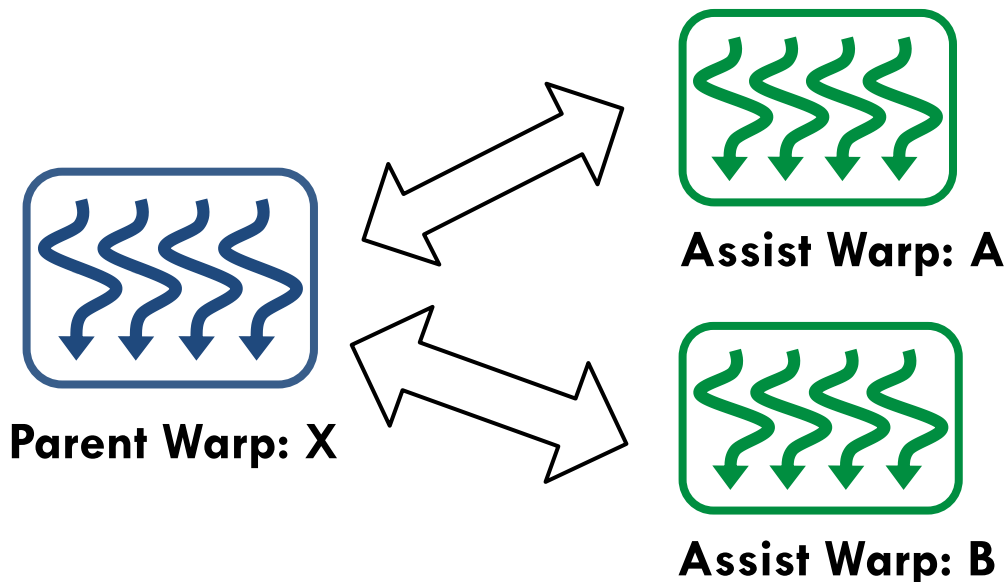
- ❑ **Tightly coupled** to regular threads
- ❑ Simply instructions injected into the GPU pipelines
- ❑ Share the same context as the regular threads

- ✓ **Efficient context management**
- ✓ **Simpler data communication**

CABA: 2. In Hardware

Helper threads:

- **Decoupled** from regular threads
- Tracked at the granularity of a **warp** – **Assist Warp**
 - Each regular (**parent**) warp can have different **assist warps**



✓ **Dynamic management of threads**

✓ **Fine-grained synchronization**

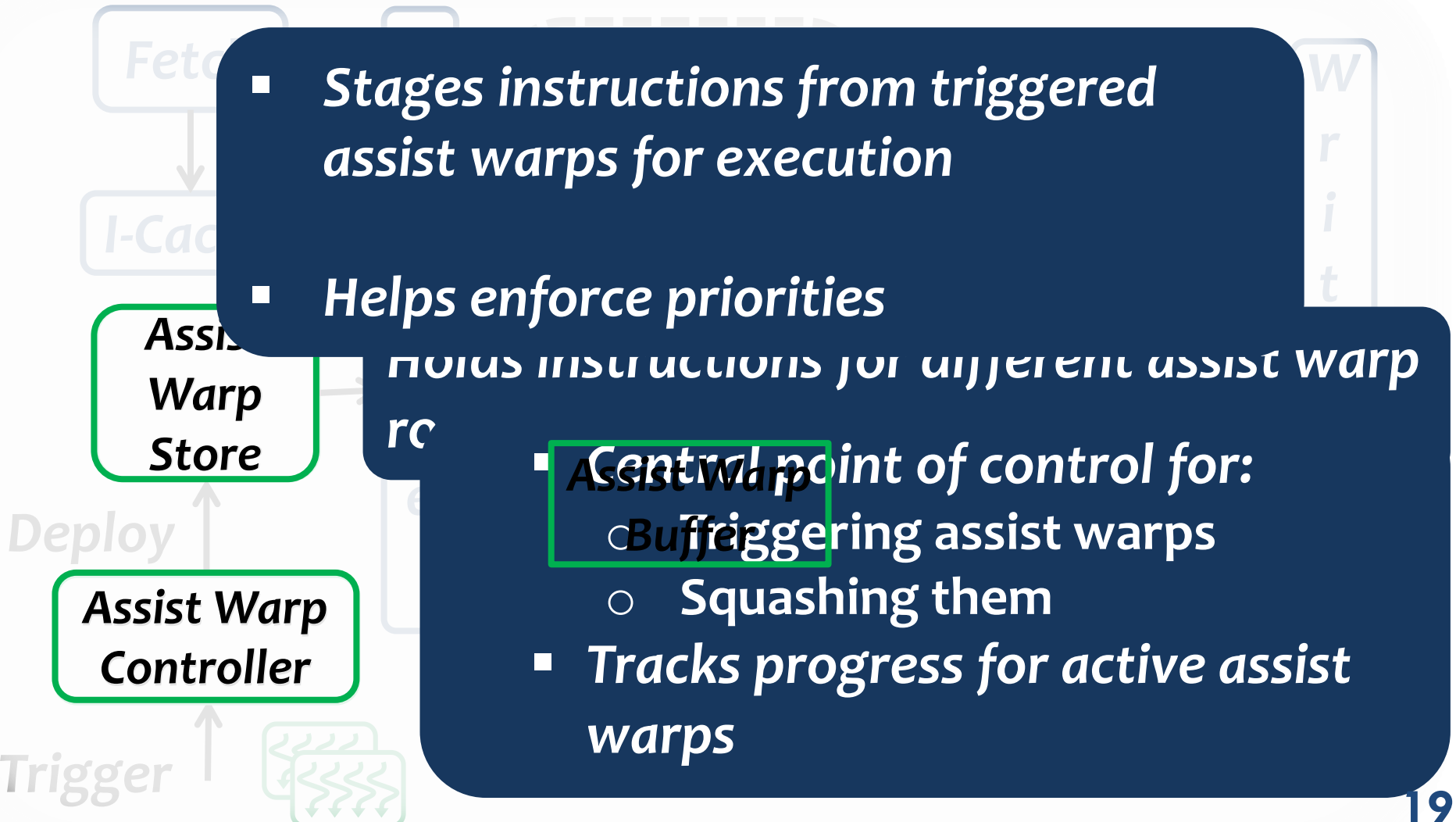
Key Functionalities

- **Triggering and squashing assist warps**
 - Associating events with assist warps
- **Deploying active assist warps**
 - Scheduling instructions for execution
- **Enforcing priorities**
 - Between assist warps and parent warps
 - Between different assist warps

CABA: Mechanism

- Stages instructions from triggered assist warps for execution
- Helps enforce priorities

- Central point of control for:
 - Buffer
 - Squashing them
- Tracks progress for active assist warps



Other functionality

In the paper:

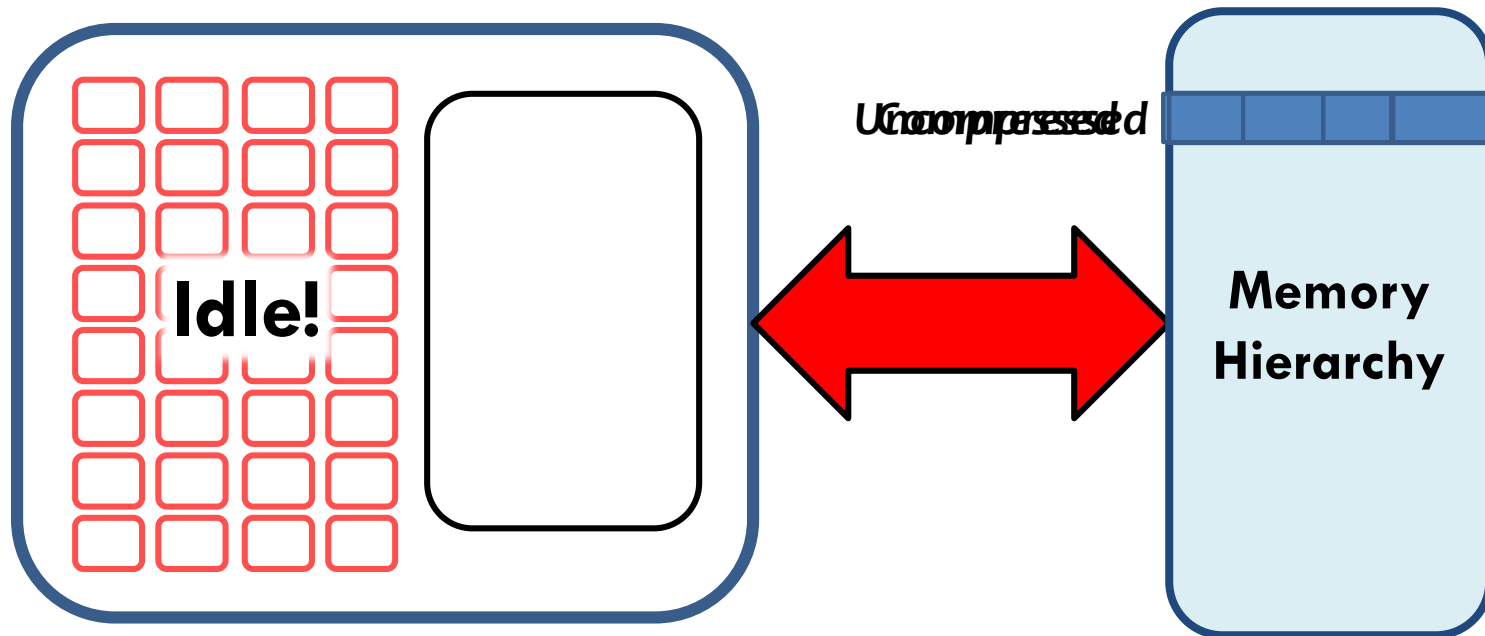
- More details on the hardware structures
- Data communication and synchronization
- Enforcing priorities

CABA: Applications

- Data compression
- Memoization
- Prefetching
- ...

A Case for CABA: Data Compression

- **Data compression** can help alleviate the **memory bandwidth bottleneck** - transmits data in a more condensed form

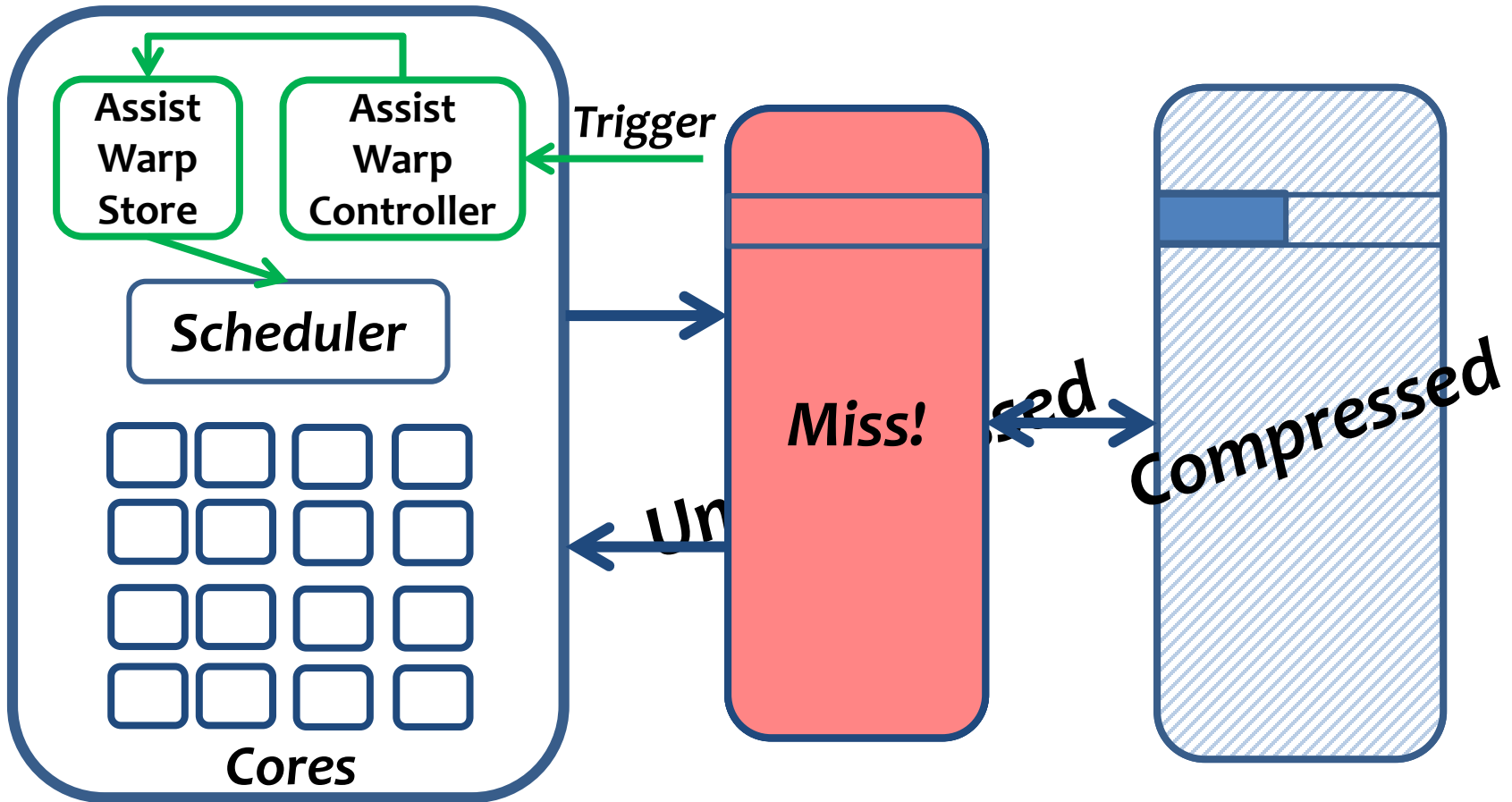


- CABA employs idle compute pipelines to perform compression

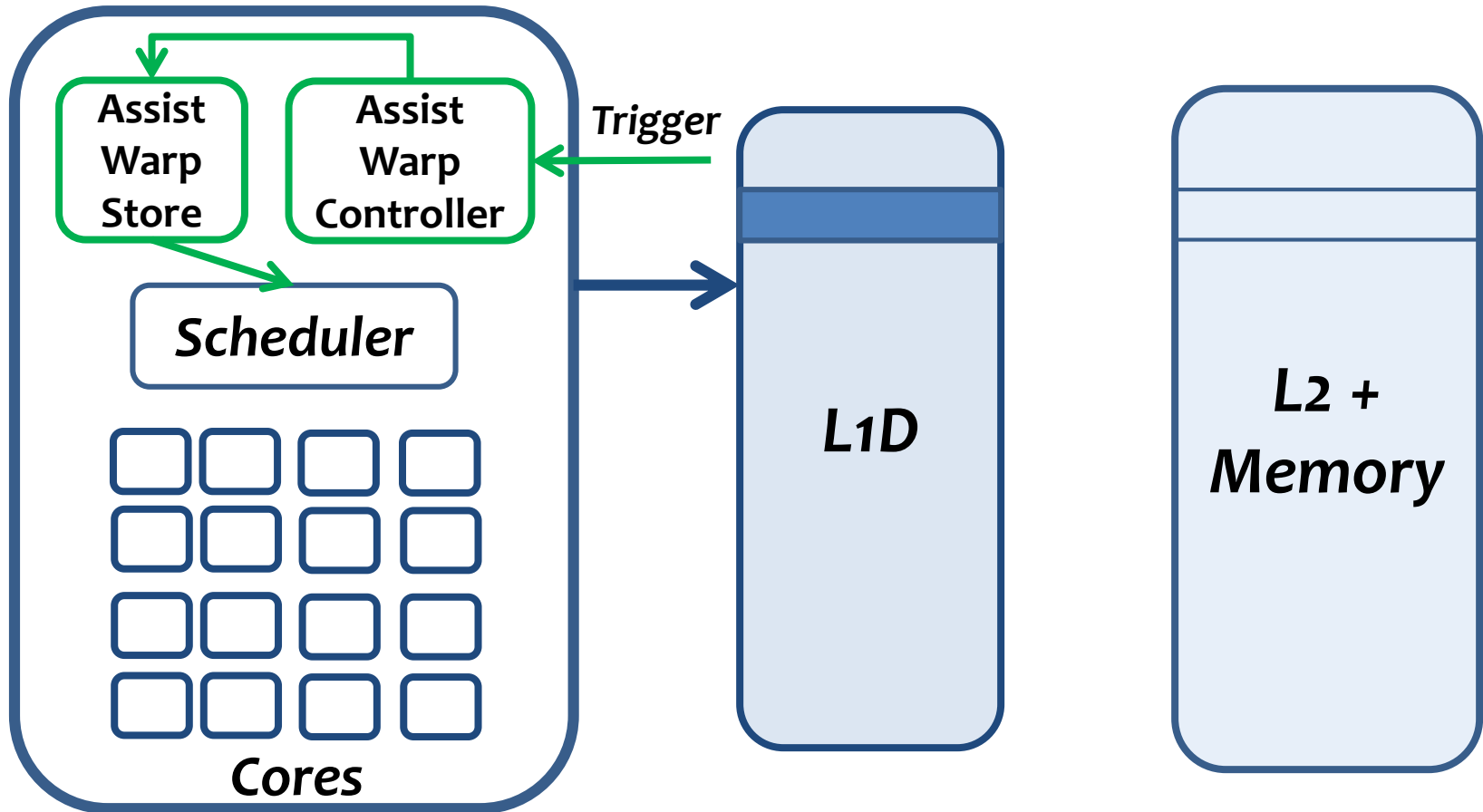
Data Compression with CABA

- Use assist warps to:
 - ▣ Compress cache blocks before writing to memory
 - ▣ Decompress cache blocks before placing into the cache
- **CABA flexibly enables various compression algorithms**
- Example: **BDI Compression** [Pekhimenko+ PACT '12]
 - ▣ Parallelizable across SIMT width
 - ▣ Low latency
- Others: **FPC** [Alameldeen+ TR '04], **C-Pack** [Chen+ VLSI '10]

Walkthrough of Decompression



Walkthrough of Compression



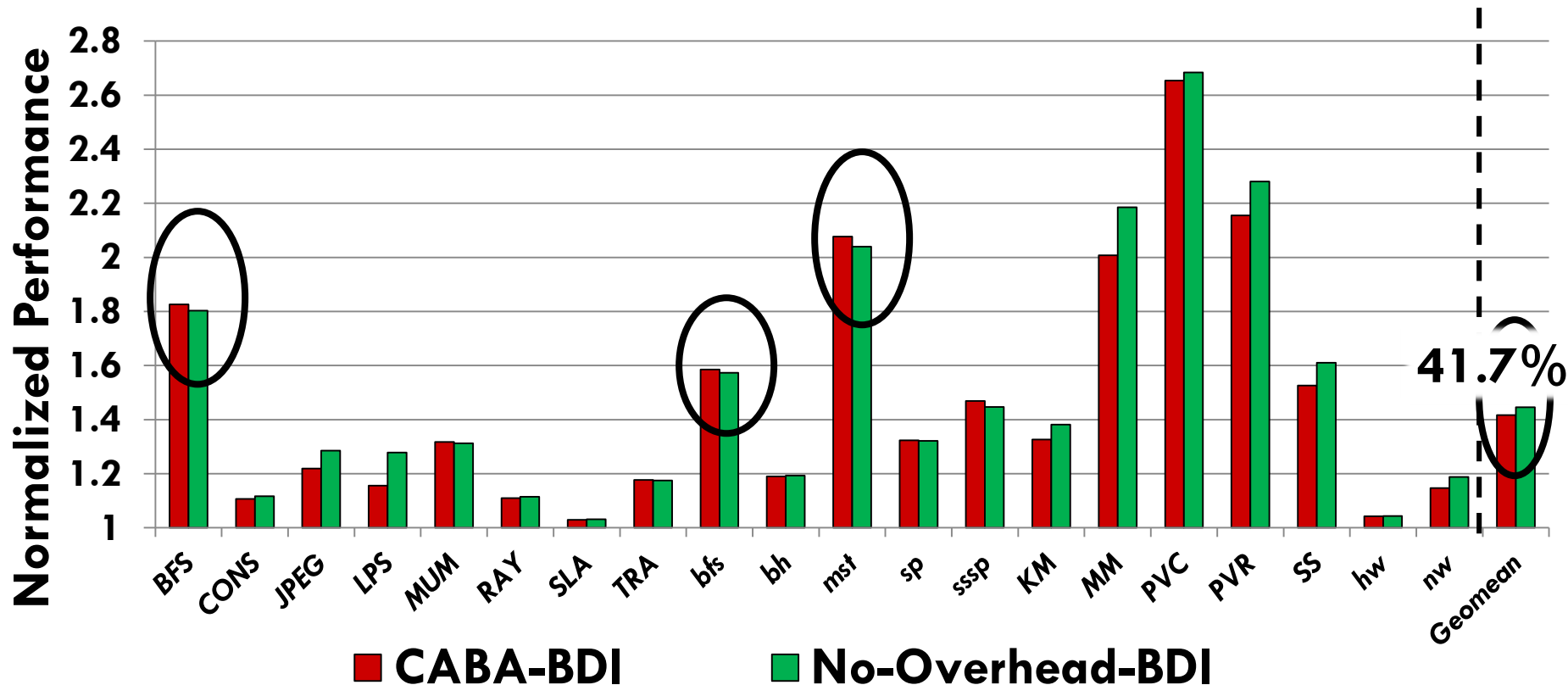
A horizontal bar at the top of the slide, divided into a red section on the left and a blue section on the right.

Evaluation

Methodology

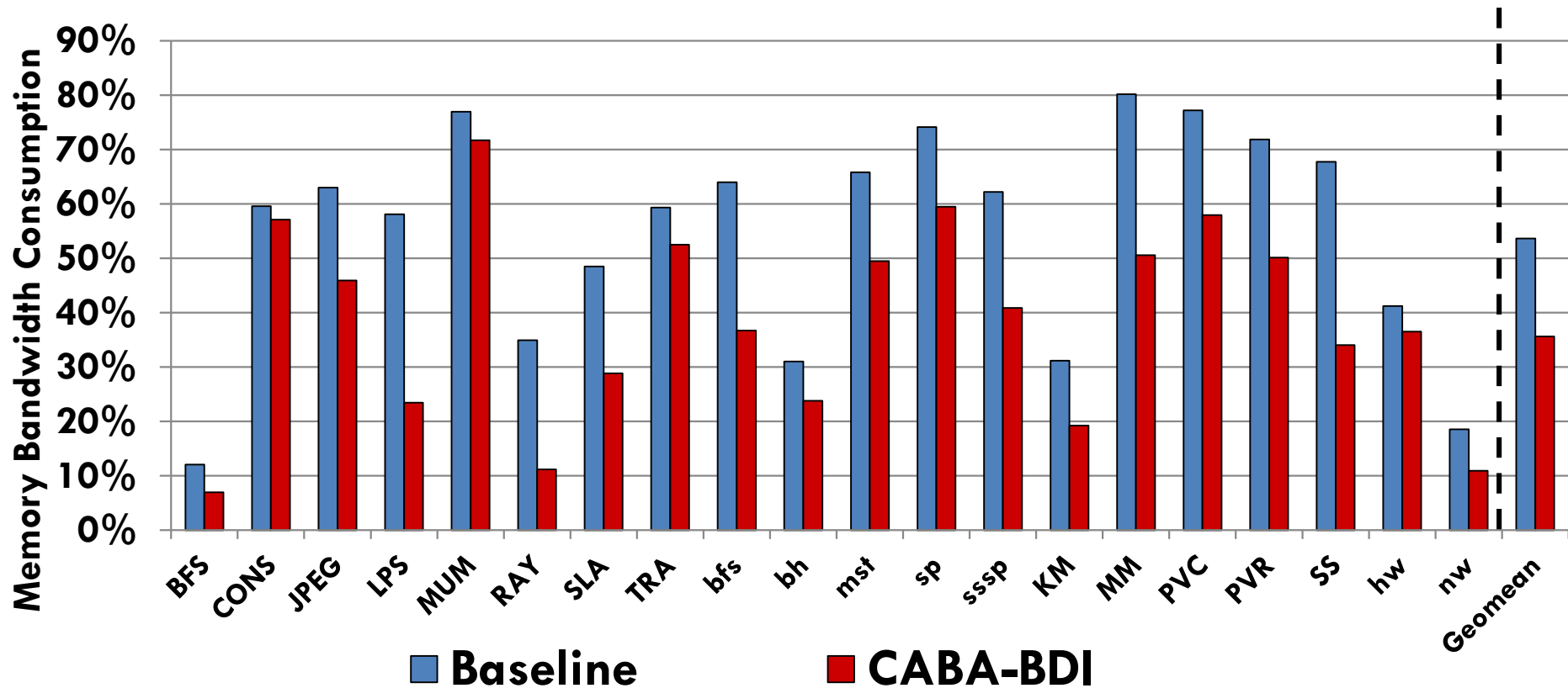
- **Simulator:** GPGPUSim, GPUWattch
- **Workloads**
 - ▣ Lonestar, Rodinia, MapReduce, CUDA SDK
- **System Parameters**
 - ▣ 15 SMs, 32 threads/warp
 - ▣ 48 warps/SM, 32768 registers, 32KB Shared Memory
 - ▣ Core: 1.4GHz, GTO scheduler , 2 schedulers/SM
 - ▣ Memory: 177.4GB/s BW, 6 GDDR5 Memory Controllers, FR-FCFS scheduling
 - ▣ Cache: L1 - 16KB, 4-way associative; L2 - 768KB, 16-way associative
- **Metrics**
 - ▣ Performance: Instructions per Cycle (IPC)
 - ▣ Bandwidth Consumption: Fraction of cycles the DRAM data bus is busy

Effect on Performance



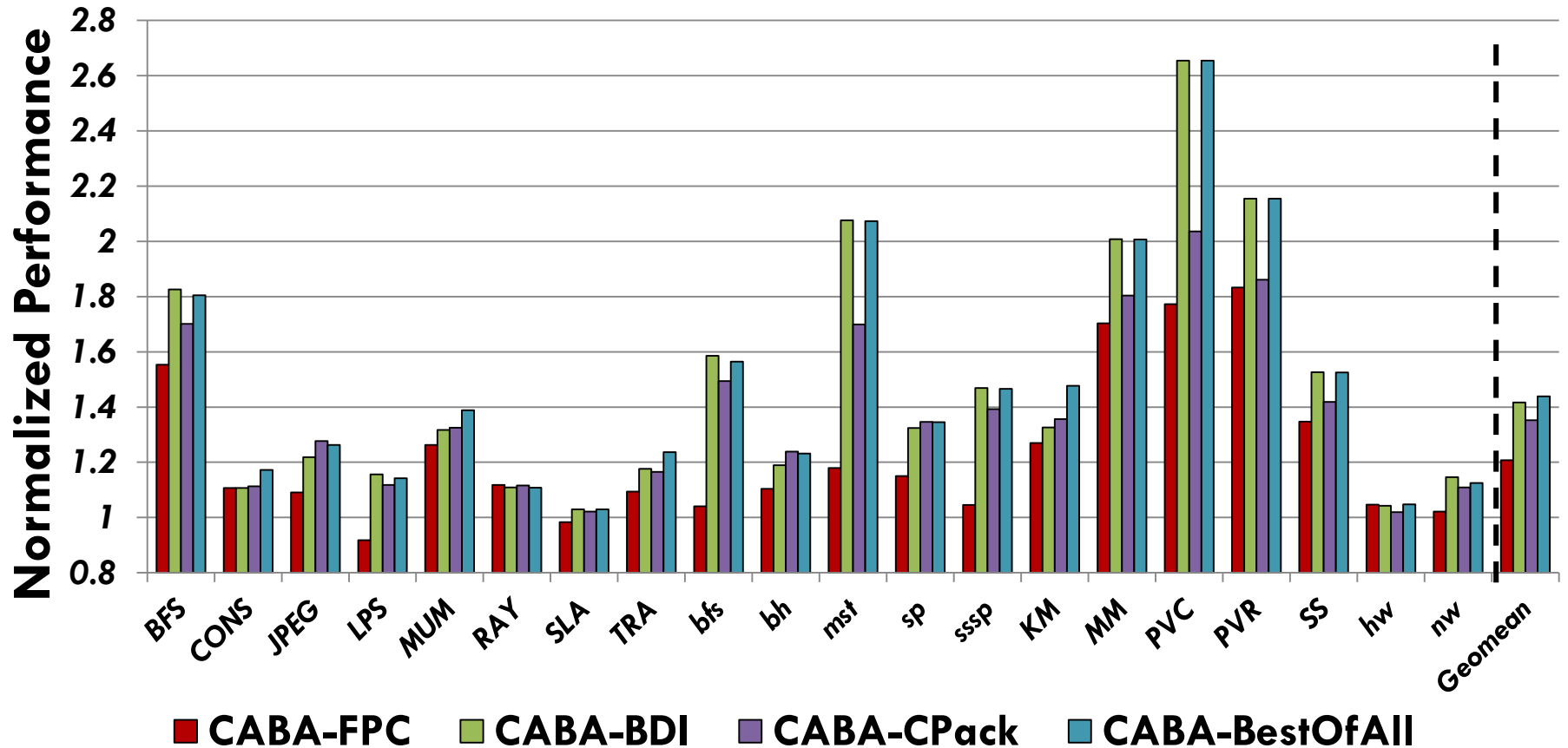
- CABA provides a 41.7% performance improvement
- CABA achieves performance close to that of designs with no overhead for compression

Effect on Bandwidth Consumption



Data compression with CABA alleviates the memory bandwidth bottleneck

Different Compression Algorithms



CABA is flexible: Improves performance with different compression algorithms

Other Results

- CABA's performance is similar to pure-hardware based BDI compression
- CABA reduces the overall system energy (22%) by decreasing the off-chip memory traffic
- Other evaluations:
 - ▣ Compression ratios
 - ▣ Sensitivity to memory bandwidth
 - ▣ Capacity compression
 - ▣ Compression at different levels of the hierarchy

Conclusion

- **Observation:** Imbalances in execution leave GPU resources underutilized
- **Our Goal:** Employ underutilized GPU resources to do something useful – **accelerate bottlenecks using helper threads**
- **Challenge:** How do you efficiently **manage and use** helper threads in a **throughput-oriented** architecture?
- **Our Solution: CABA (Core-Assisted Bottleneck Acceleration)**
 - ▣ A new framework to enable helper threading in GPUs
 - ▣ Enables flexible data compression to alleviate the memory bandwidth bottleneck
 - ▣ A wide set of use cases (e.g., prefetching, memoization)
- **Key Results:** Using CABA to implement data compression in memory improves performance by 41.7%

A Case for Core-Assisted Bottleneck Acceleration in GPUs

*Enabling Flexible Data Compression
with Assist Warps*

Nandita Vijaykumar

Gennady Pekhimenko, Adwait Jog, Abhishek Bhowmick,
Rachata Ausavarangnirun, Chita Das, Mahmut Kandemir,
Todd C. Mowry, Onur Mutlu

SAFARI

Carnegie Mellon

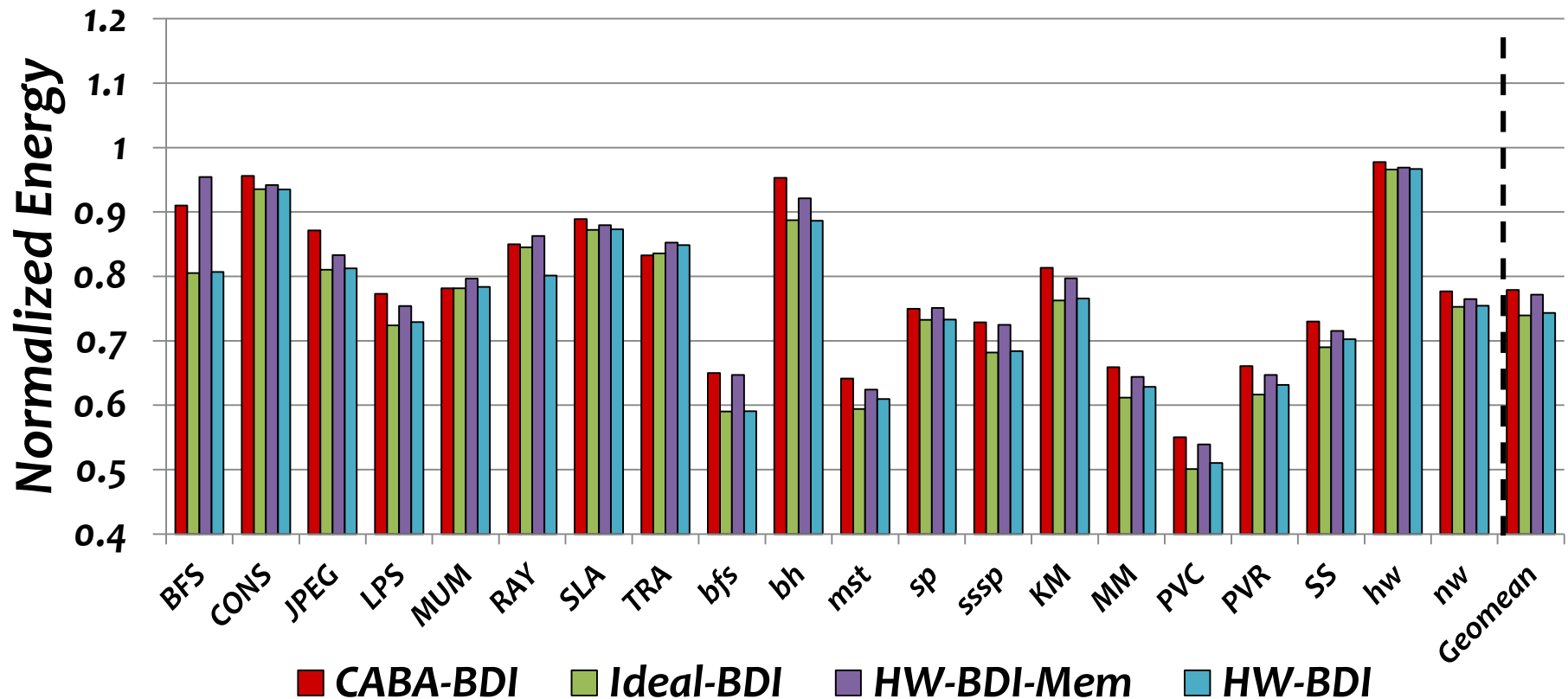
PENNSSTATE



34

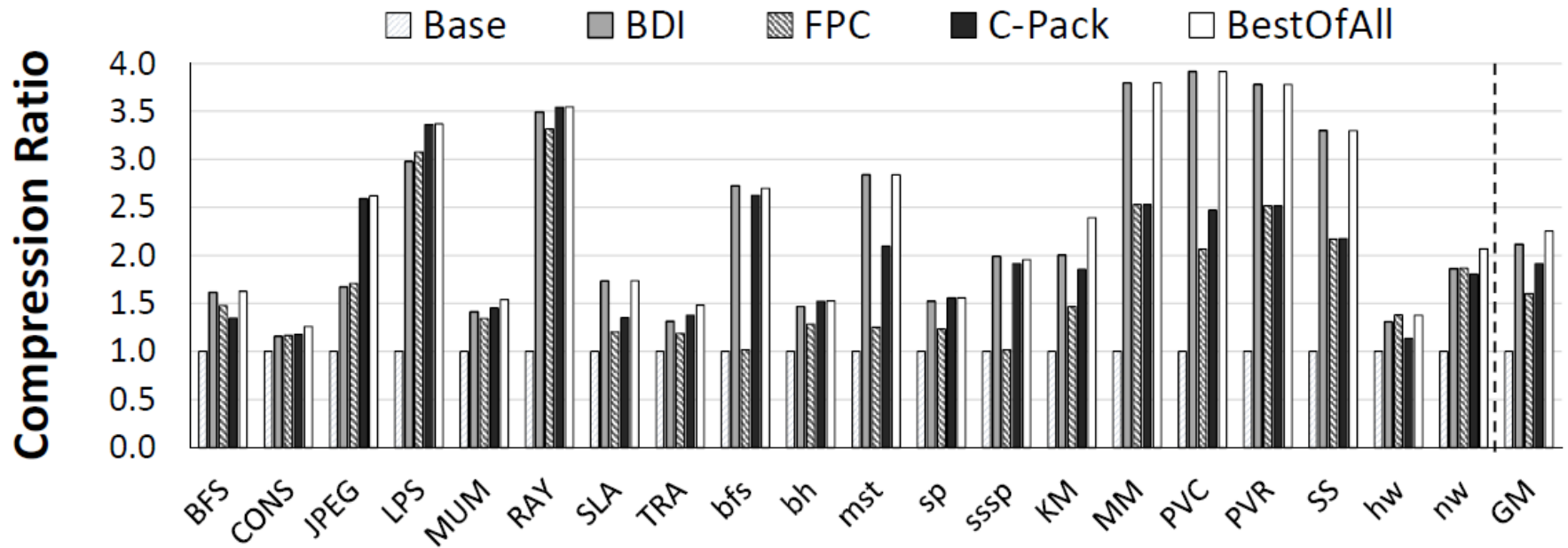
Backup Slides

Effect on Energy



CABA reduces the overall system energy by decreasing the off-chip memory traffic

Effect on Compression Ratio



Other Uses of CABA

- Hardware Memoization
 - ▣ Goal: avoid redundant computation by reusing previous results over the same/similar inputs
 - ▣ Idea:
 - hash the inputs at predefined points
 - use load/store pipelines to save inputs in shared memory
 - eliminate redundant computation by loading stored results
- Prefetching
 - ▣ Similar to CPU