Zorua: A Holistic Approach to Resource Virtualization in GPUs

Nandita Vijaykumar

Kevin Hsieh, Gennady Pekhimenko, Samira Khan,
Ashish Shrestha, Saugata Ghose, Adwait Jog, Phillip B. Gibbons, Onur Mutlu

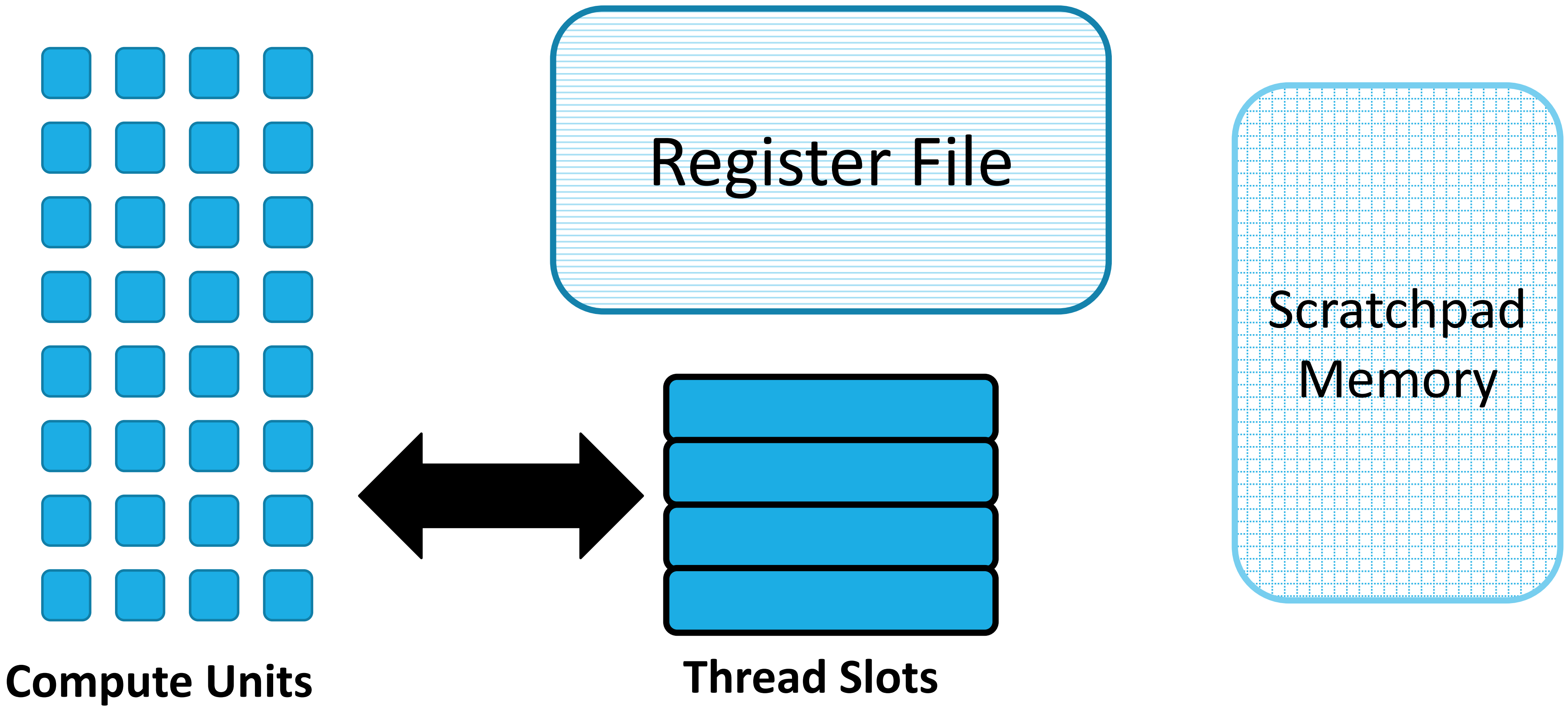
Carnegie Mellon University
Microsoft Research
University of Virginia
William & Mary
ETH Zürich

Overview

- **Problem:** Major on-chip resources in GPUs are managed by the ***programmer/software***
- **Key Issues:** Leads to several challenges in obtaining high performance:
  - **Programming Ease:** Requires programmer effort to optimize resource usage
  - **Performance Portability:** Optimizations do not port well across different GPUs
  - **Resource Inefficiency:** Underutilized resources even in optimized code
- **Our Goal:**
  - Reduce dependence of performance on programmer-specified resource usage
  - Enhance resource efficiency for optimized code
- **Our Approach:** ***Decouple*** the programmer-specified resource usage from the allocation in the hardware
- **Zorua:** A Holistic Resource Virtualization Framework for GPUs
- **Key Results:** Zorua enhances programming ease, performance portability and performance for optimized code

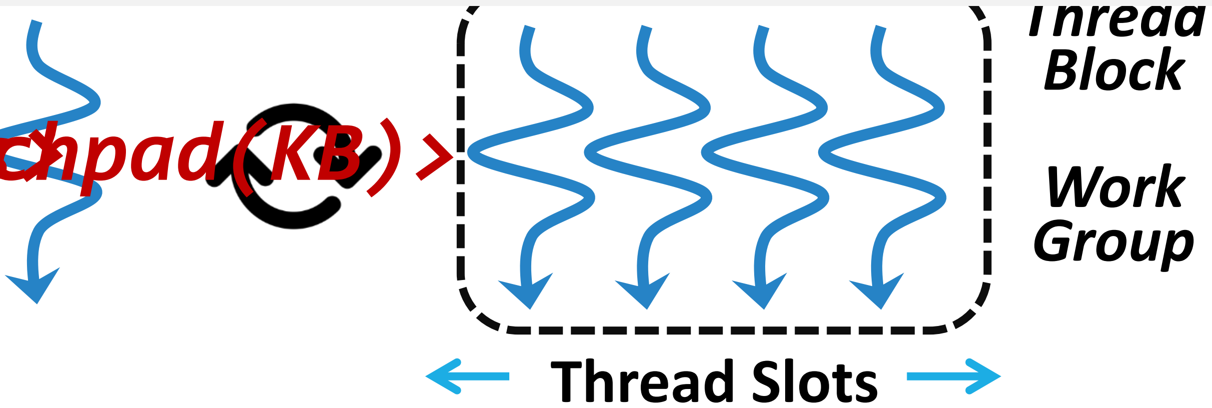
GPUs today are used across many classes of applications …

On-Chip Resources in GPUs



Every thread in a thread block needs to be allocated enough (worst-case) resources to execute and complete

<#Threads, #Registers, Scratchpad (KB)>



Abstraction of On-Chip Resources

Programmer/Software

Tight coupling **between resource specification and allocation**

Key Issues

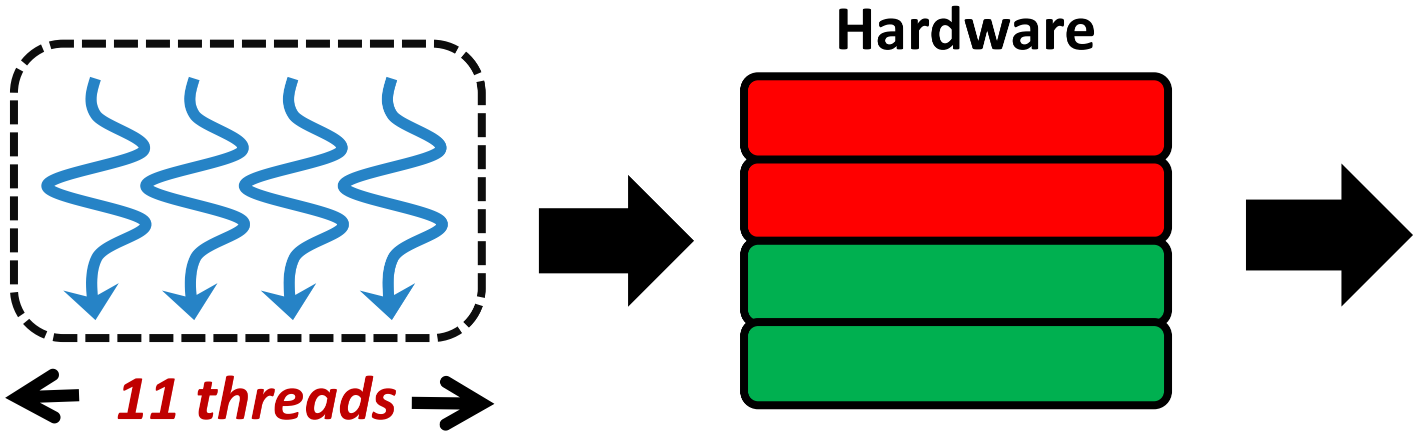
1. ***Static Underutilization***

2. ***Dynamic Underutilization***

1. Static Underutilization

<#Threads,#Registers,Scratchpad(KB)>

Static underutilization may lead to loss in parallelism



Parallelism:

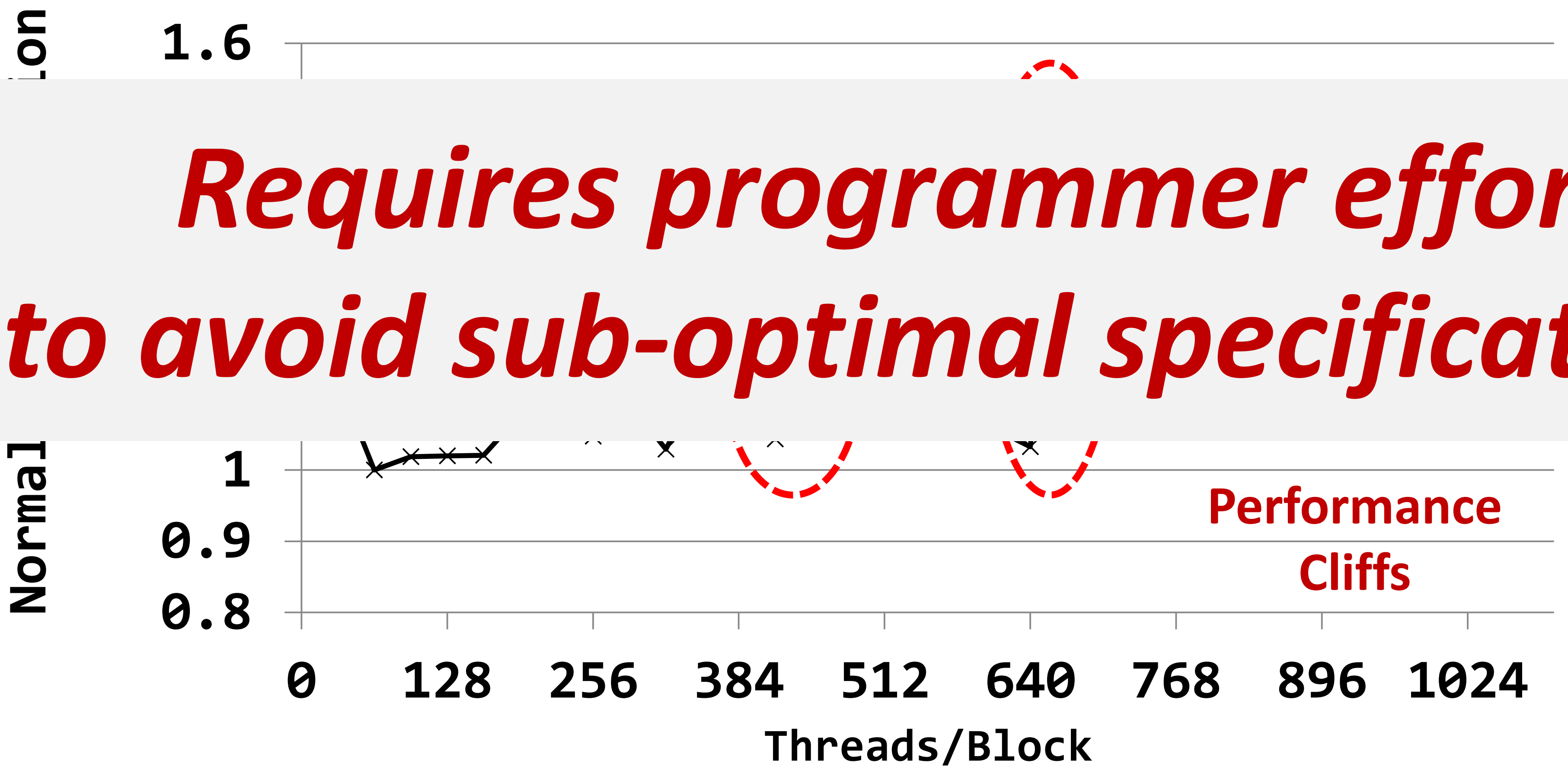
1 thread block

11 threads

To make things worse…

- Same problem exists for other on-chip resources
  - registers, scratchpad memory, thread blocks
- The programmer needs to get it right for ***all*** of them at the ***same time***

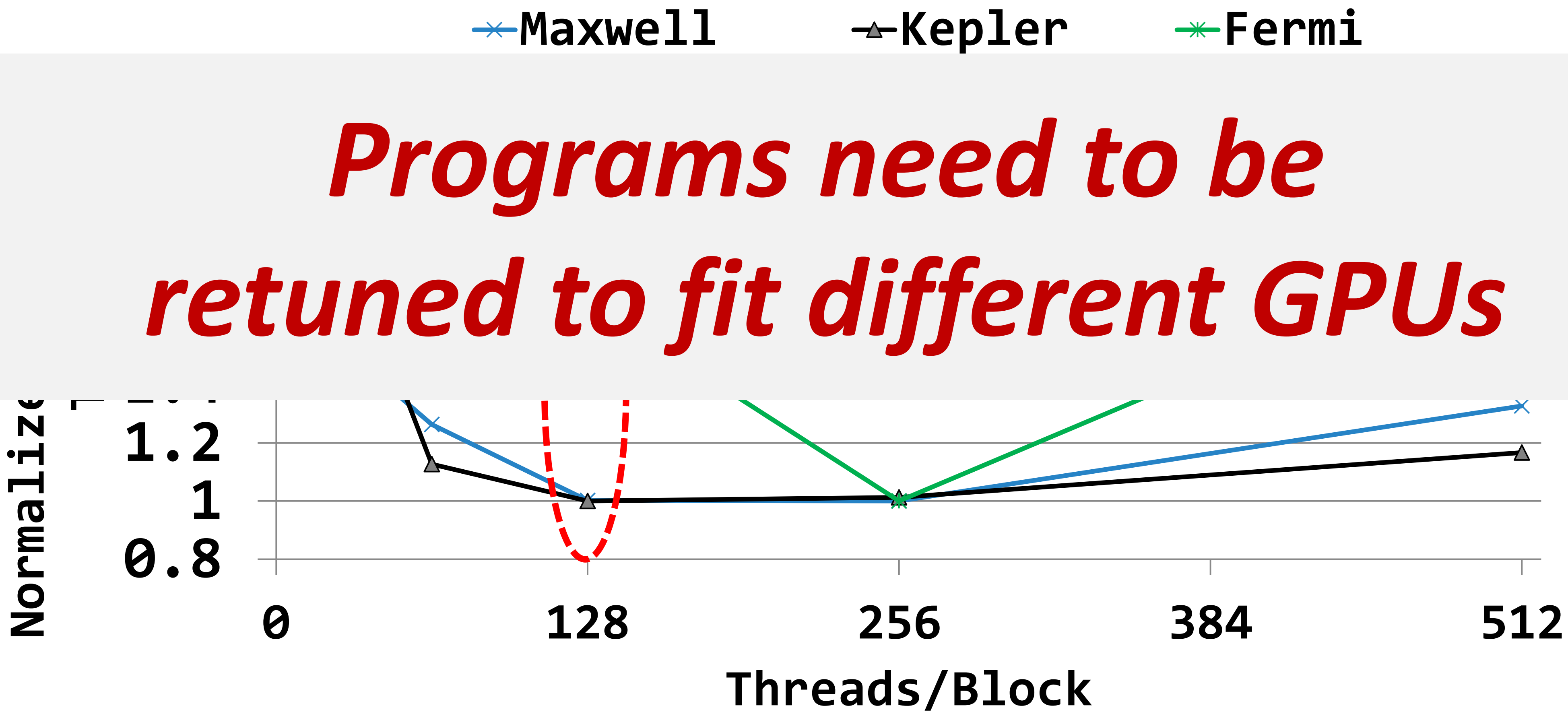
Implication 1: Programming Ease



Requires programmer effort
to avoid sub-optimal specifications

MST (*Minimum Spanning Tree*)

Implication 2: Performance Portability



Programs need to be retuned to fit different GPUs

DCT (*Discrete Cosine Transform*)

Key Issues

1. ***Static Underutilization***
2. ***Dynamic Underutilization***

2. Dynamic Underutilization

Resource requirements of a thread vary throughout execution

Implication:
Resource inefficiency due to worst-case static allocation

```
    for(unsigned int i = 0; i < B; i++)
              dst[i *I] = bl_ptr[i * X];
}
```

16 regs

Our Goal

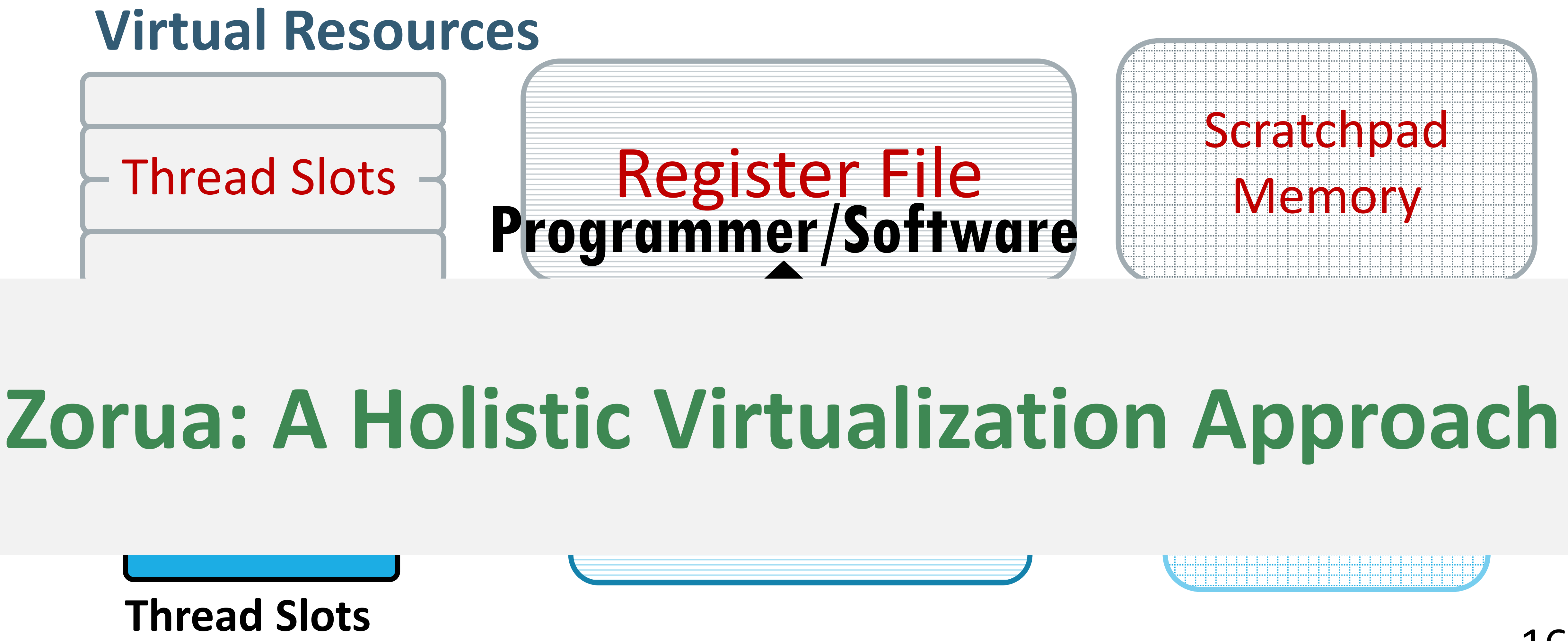
- Reduce the dependence of performance on resource specification
  - *Programming Ease*
  - *Performance Portability*

- Improve efficiency of resource utilization
  - *Higher performance for optimized code*

Outline

- Problem: Tight Coupling
- Key Implications
- Our Goal
- Our Approach: Zorua
  - *Virtualization Strategy*
  - *Design Challenges*
  - *Design Ideas*
- Evaluation

Our Approach



How do we design a virtualization strategy to effectively address the key issues?

1. Static Underutilization

<#Threads,#Registers,Scratchpad(KB)>

Flexibility in available resources helps restore parallelism

11 threads

Thread Slots in Hardware

Parallelism:
1 thread block
11 threads

Addressing Key Issues

1. ***Static Underutilization***

   Provide an illusion of a flexible amount of resources

2. ***Dynamic Underutilization***

   Enable dynamic allocation/deallocation of resources

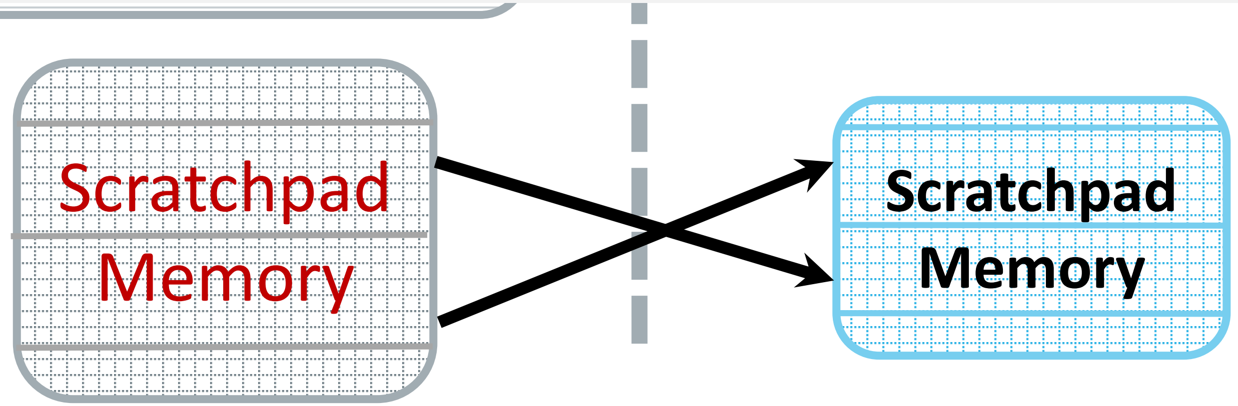
Outline

- Problem: Tight Coupling
- Key Implications
- Our Goal
- Our Approach: Zorua
  - *Virtualization Strategy*
  - *Design Challenges*
  - *Design Ideas*
- Evaluation

Zorua: Virtualization Strategy

Virtual Resources | **Physical Resources**

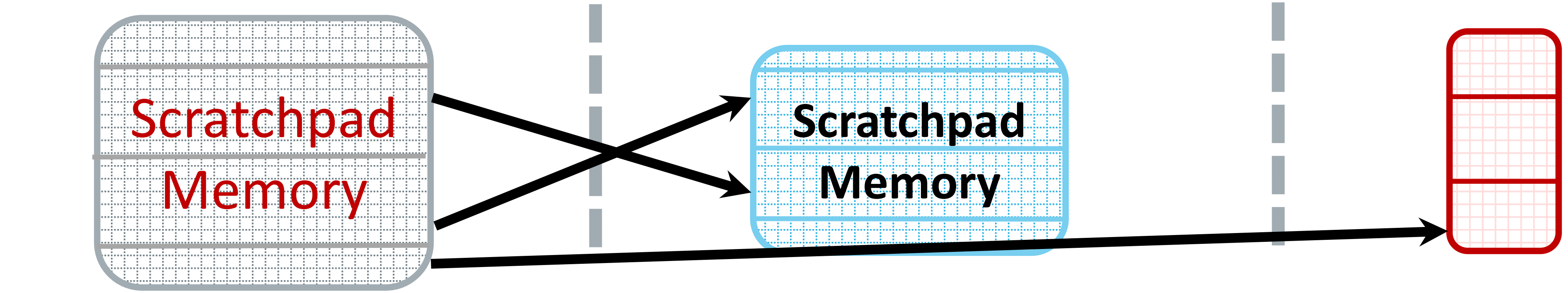
Fine-grained dynamic allocation provides resource efficiency



Zorua: Virtualization Strategy

Virtual Resources | **Physical Resources** | **Swap Space** ***(in the mem. hierarchy)***

<u>Careful</u> oversubscription using a swap space provides flexibility in the amount of resources



Outline

- Problem: Tight Coupling
- Key Implications
- Our Goal
- Our Approach: Zorua
  - *Virtualization Strategy*
  - *Design Challenges*
  - *Design Ideas*
- Evaluation

Zorua: Design Challenges

- **Challenge 1:** **Controlling** the ***extent*** of oversubscription
  - Spills are expensive
- **Challenge 2:** **Coordinating** virtualization of ***multiple*** on-chip resources
  - Resources are independently virtualized

Resource requirements vary during execution

Zorua Design: Key Questions

- How do we determine the variation in resource requirements?

- How do we use this knowledge to:
  - **control** how much we oversubscribe at run time?
  - **coordinate** allocation of *multiple* resources to maximize parallelism within the oversubscription budget?

Outline

- Problem: Tight Coupling
- Key Implications
- Our Goal
- Our Approach: Zorua
  - *Virtualization Strategy*
  - *Design Challenges*
  - *Design Ideas*
- Evaluation

Component 1: *The Compiler*

- Leverage software to determine variation in resource requirements

Use the compiler to:

- ***Statically partition the program into phases***
- ***Add annotations with per-phase resource requirements***

```
    ...
    CUDAsubroutineInplaceDCTvector(…);

    for(unsigned int i = 0; i < B; i++)
                dst[i *I] = bl_ptr[i * X];
}
```

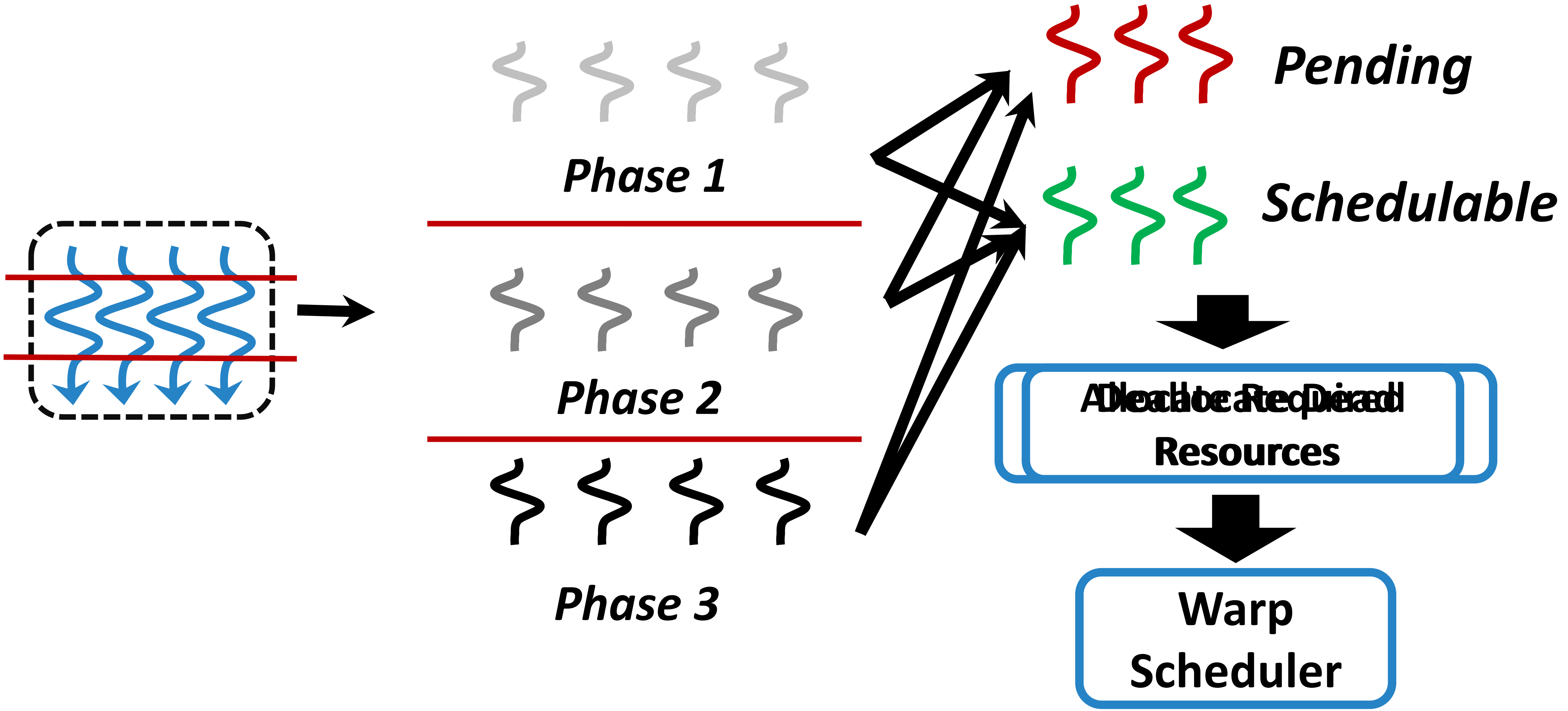
32 regs

16 regs

Zorua Design: Key Questions

- How do we determine the variation in resource requirements?
- How do we use this knowledge to:
  - **control** how much we oversubscribe at run time?
  - **coordinate** allocation of *multiple* resources to maximize parallelism within the oversubscription budget?

Component 2: *Hardware Runtime System*



Putting It All Together

Zorua: A hardware-software cooperative framework

- **The compiler:** annotates the program to partition it into ***phases*** and specify the resource needs of each phase
- **The coordinator:** a hardware runtime system that makes oversubscription decisions and allocates/deallocates resources
- **Hardware virtualization support:**
  - Mapping tables for each resource (1.85kB ≈ 0.134% of the die area)
  - Machinery to swap data between on-chip hardware & swap space

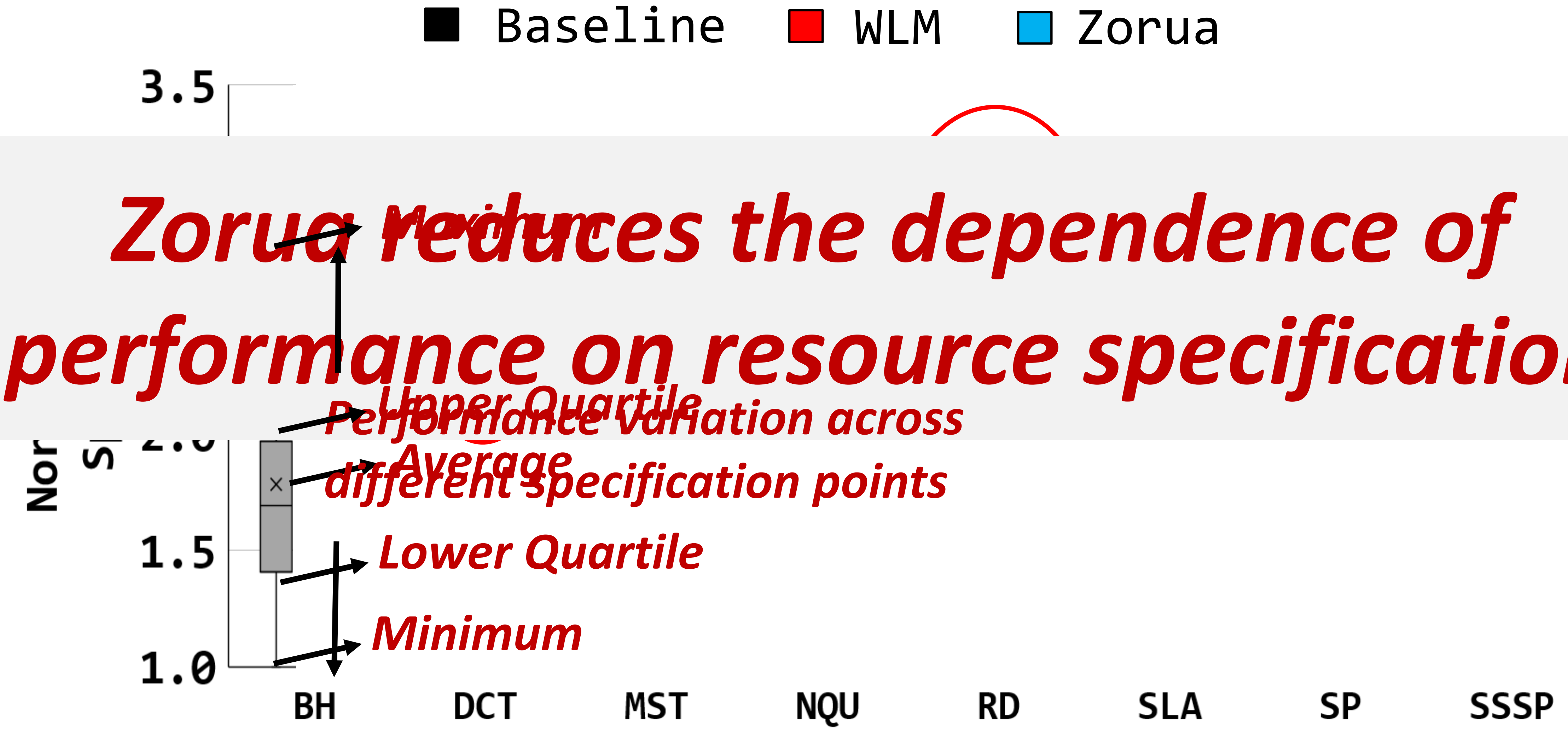
Outline

- Problem: Tight Coupling
- Key Implications
- Our Goal
- Our Approach: Zorua
  - *Virtualization Strategy*
  - *Design Challenges*
  - *Design Ideas*
- Evaluation

Methodology

- **Evaluation Infrastructure:** Real GPUs (Fermi/Kepler/Maxwell), GPGPUSim, GPUWattch
- **Workloads**
  - Lonestar, CUDA SDK
- **System Parameters**
  - 15 SMs, 32 threads/warp
  - **Warps/SM:** Fermi: 48, Kepler/Maxwell: 64
  - **Registers:** Fermi: 32768, Kepler/Maxwell: 65536
  - **Scratchpad:** Fermi/Kepler: 48KB, Maxwell: 64KB
  - **Core:** 1.4GHz, GTO scheduler , 2 schedulers/SM
  - **Memory:** 177.4GB/s BW, 6 GDDR5 Memory controllers
- **Overheads of Zorua**
  - 2-cycle latency for mapping table lookup for each resource
  - Memory requests for swap space accesses

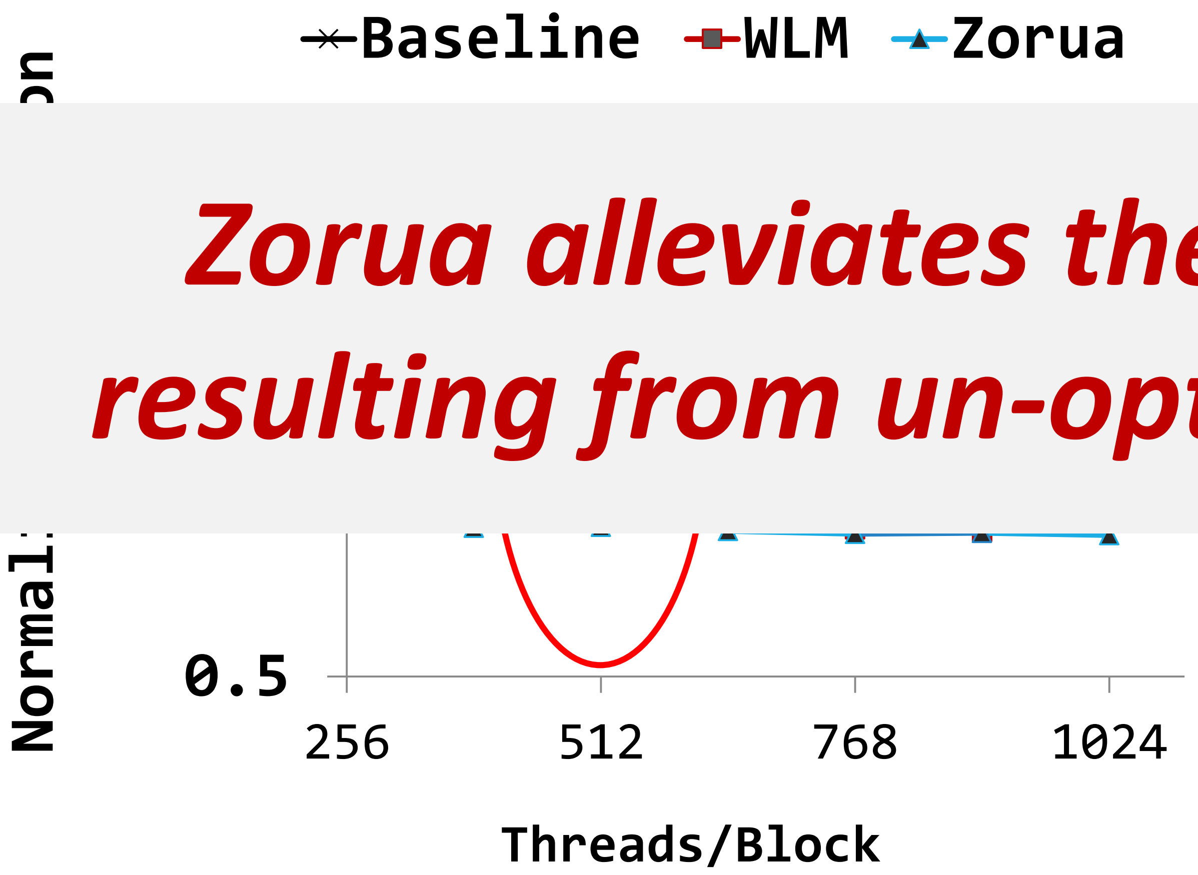
Effect on Performance Variation



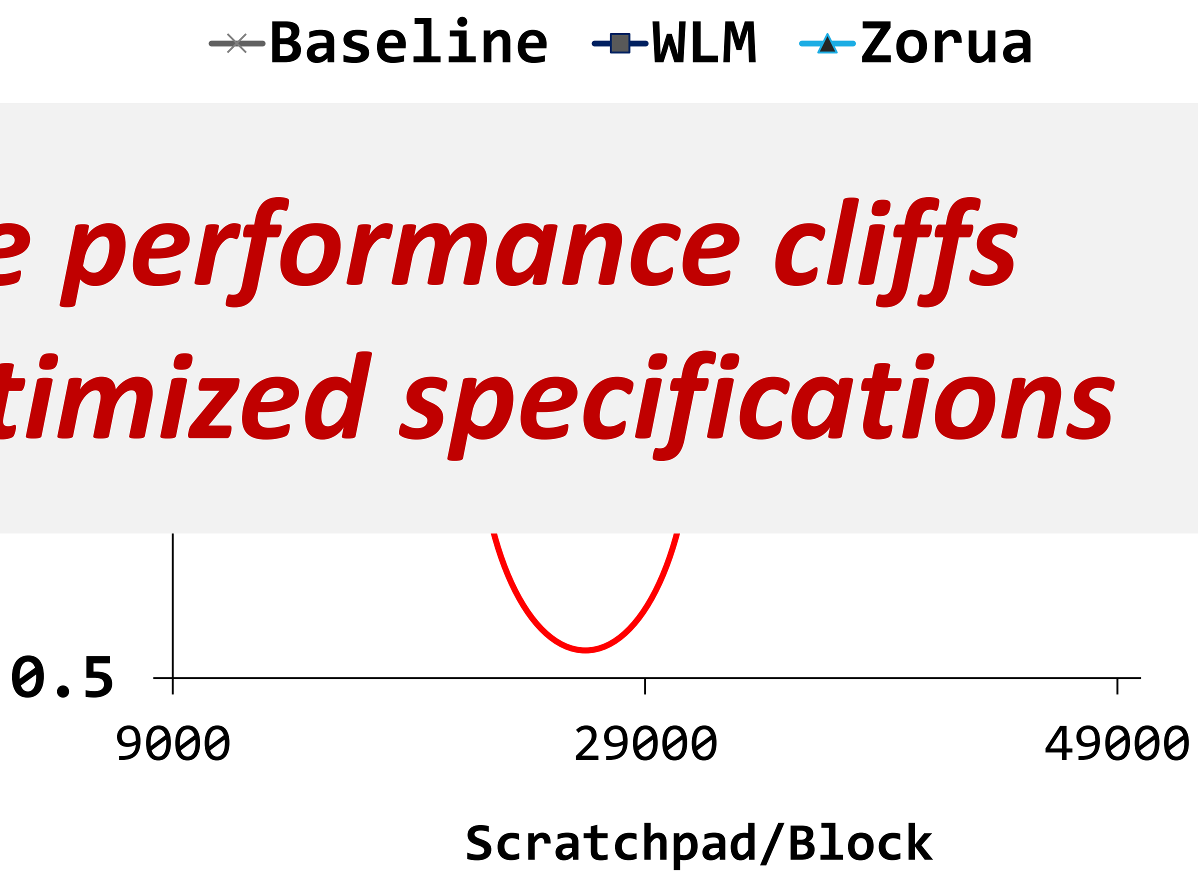
Zorua reduces the dependence of performance on resource specification

* Xiang et al., HPCA '14

Effect on Performance Cliffs



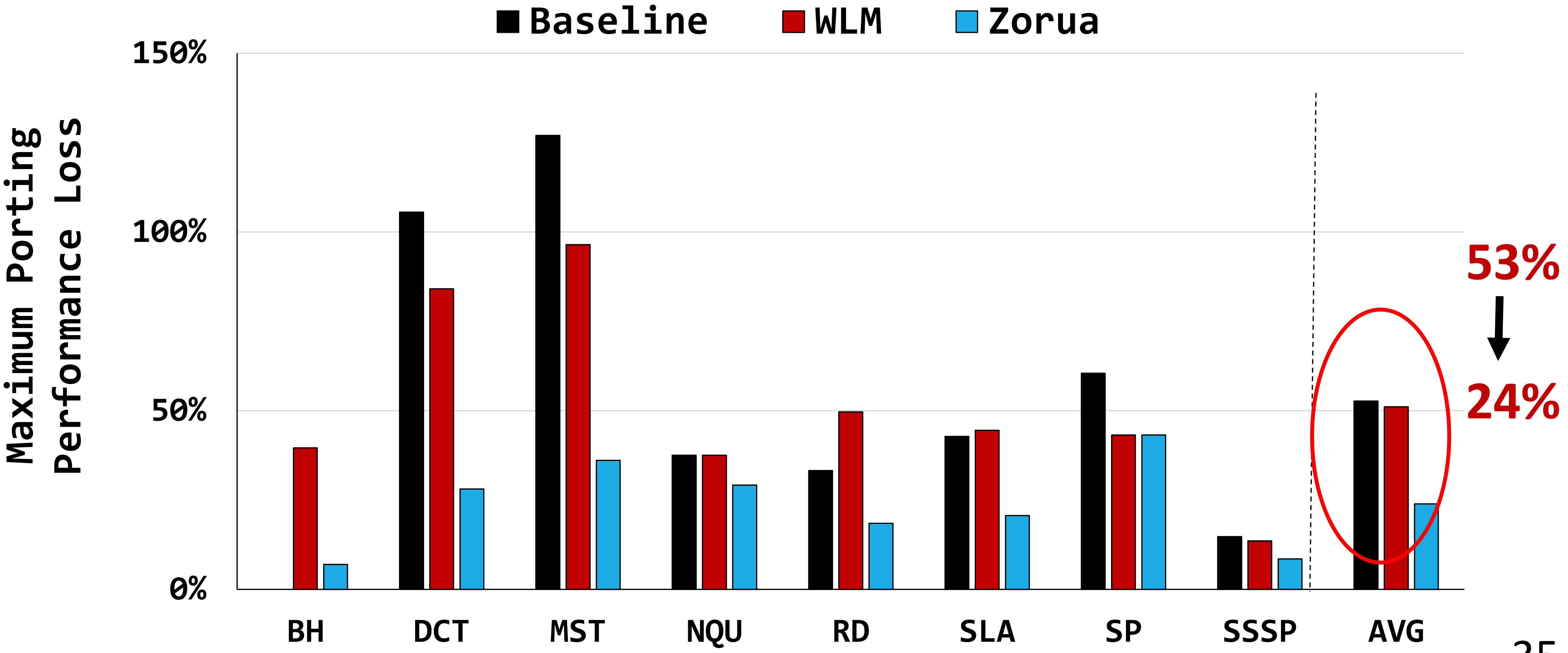
MST



NQU

Zorua alleviates the performance cliffs resulting from un-optimized specifications

Effect on Performance Portability



Other Uses

- Resource sharing in multi-programmed environments
- Low latency preemption of kernels
- Dynamic parallelism
- …

Conclusion

- **Problem:** The ***tight coupling*** between programmer-specified resource usage and allocation of on-chip resources leads to challenges in:
  - ***programming ease, performance portability, resource efficiency***
- **Our Approach:** ***Decouple*** specification and management of on-chip resources
- **Our Solution: Zorua:** A holistic approach to virtualizing multiple on-chip resources in GPUs
- **Key Results:**
  - Zorua reduces dependence of performance on programmer-specified resource usage
    - *Zorua enhances **programming ease** and **performance portability***
  - Zorua improves performance with more ***efficient resource utilization***
- **Future Work:** Zorua enables several other use cases

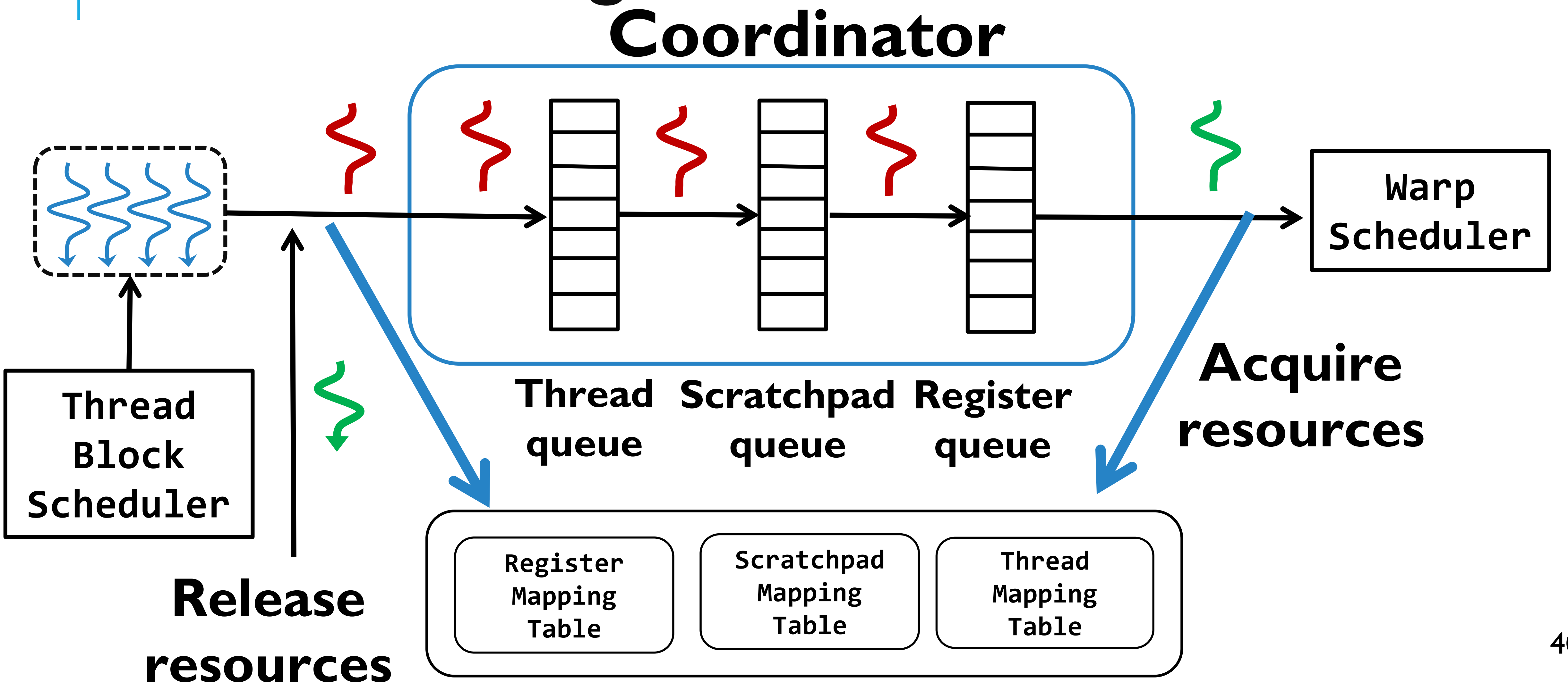
Zorua: A Holistic Approach to Resource Virtualization in GPUs

Nandita Vijaykumar

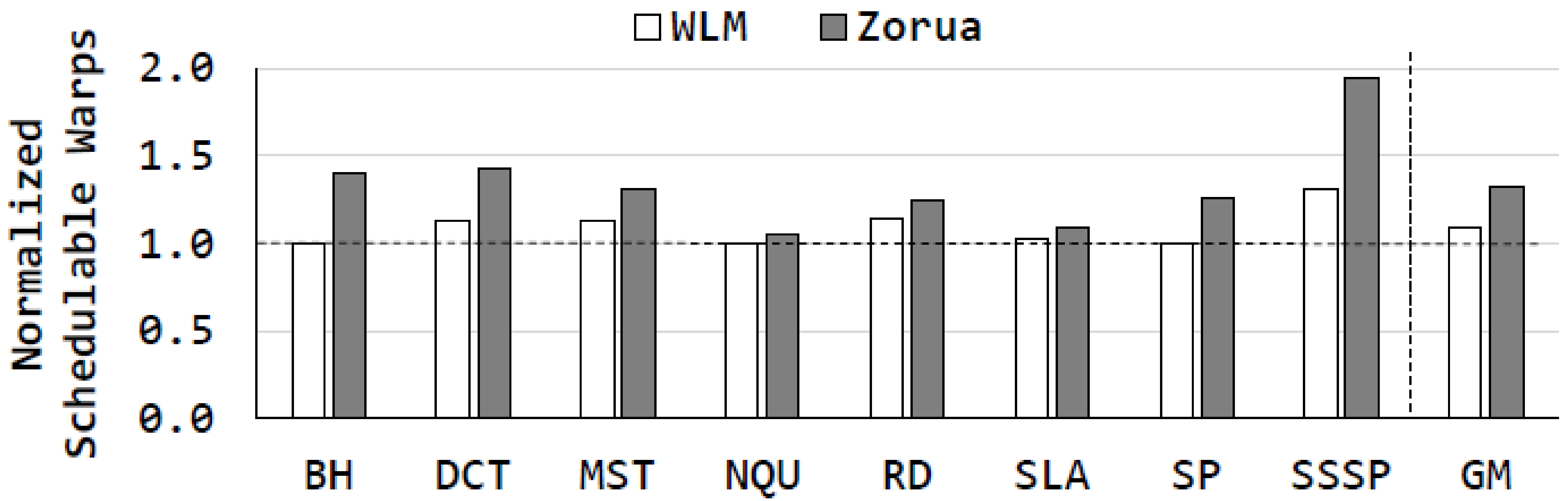
Kevin Hsieh, Gennady Pekhimenko, Samira Khan,
Ashish Shrestha, Saugata Ghose, Adwait Jog, Phillip B. Gibbons, Onur Mutlu

Carnegie Mellon University
Microsoft Research
University of Virginia
William & Mary
ETH Zürich

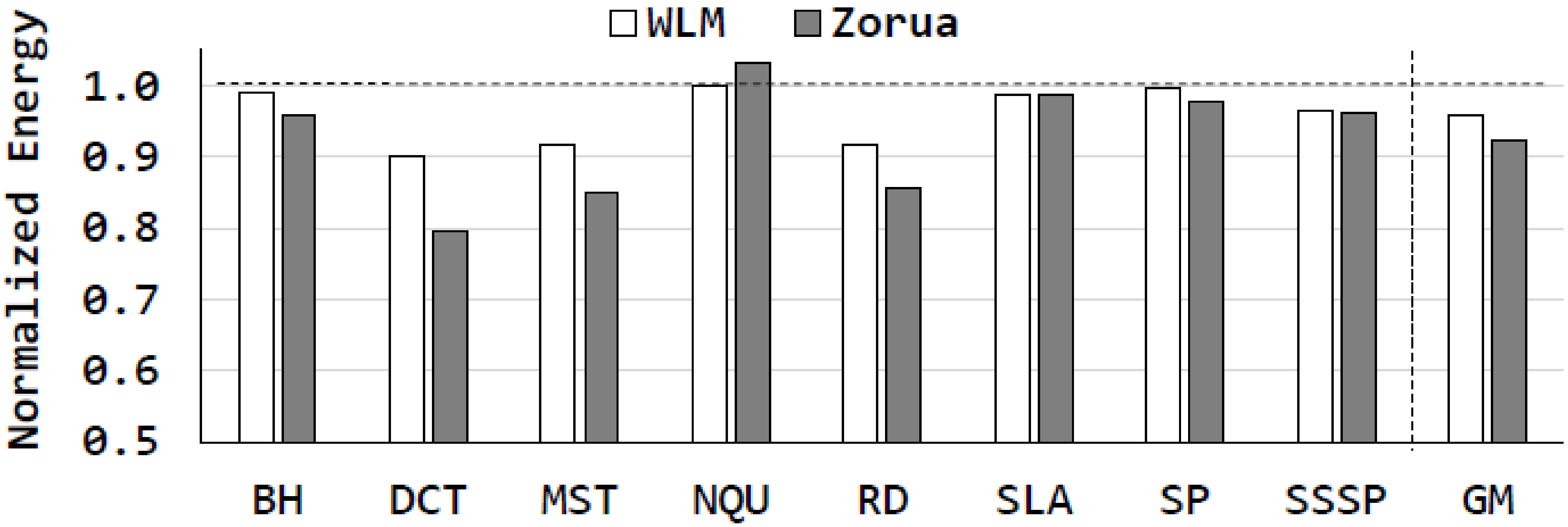
A Walkthrough



Effect on schedulable warps



Effect on energy consumption



Summary of applications

Name (Abbreviation)	(R: Register, S: Scratchpad, T: Thread block) Range
Barnes-Hut (BH) [8]	R:28-44 × T:128-1024
Discrete Cosine Transform (DCT) [52]	R:20-40 × T: 64-512
Minimum Spanning Tree (MST) [8]	R:28-44 × T: 256-1024
Reduction (RD) [52]	R:16-24 × T:64-1024
N-Queens Solver (NQU) [11] [5]	S:10496-47232 (T:64-288)
Scan Large Array (SLA) [52]	R:24-36 × T:128-1024
Scalar Product (SP) [52]	S:2048-8192 × T:128-512
Single-Source Shortest Path (SSSP) [8]	R:16-36 × T:256-1024