A Cross-Input Adaptive Framework for GPU Program Optimizations

Yixun Liu, **Eddy Z. Zhang**, Xipeng Shen

Computer Science Department
The College of William & Mary

Outline

- **GPU overview**
- G-Adapt Framework
- Evaluation
- Related & Future Work
- Conclusion

GPU (Graphics Processing Unit)

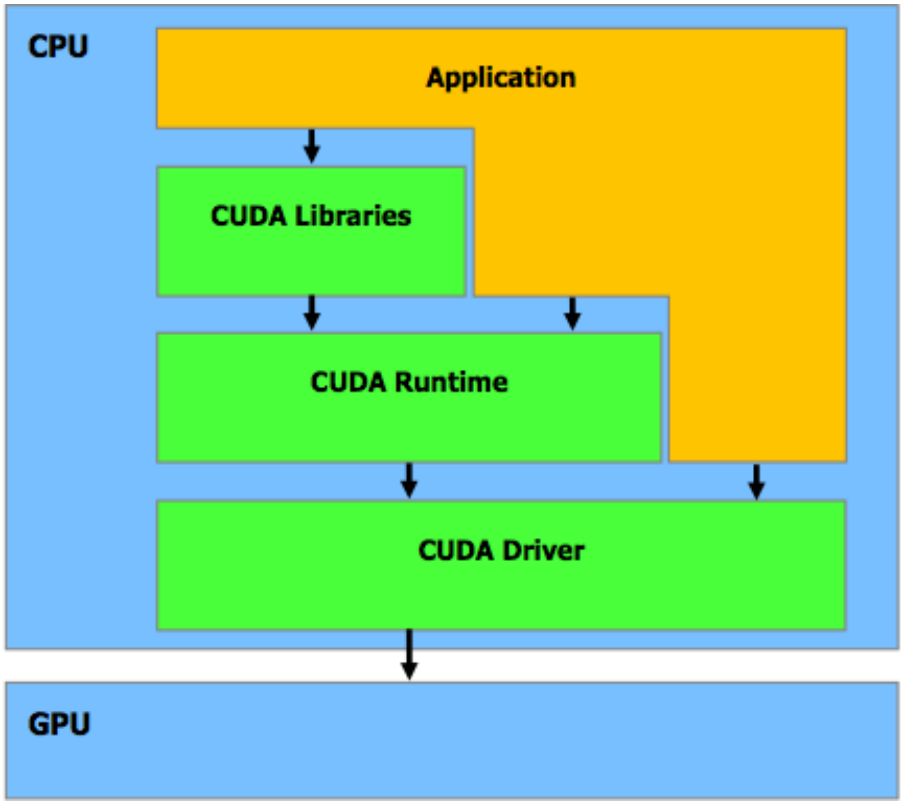
- Architecture
  - SIMD parallel
  - Multithreaded
  - Many core

- Feature
  - Tremendous computational horsepower
  - High mem badwidth

- Applications
  - Traditional graphic rendering
  - Emerging: general data parallel computing

Programming GPU

- High level model
  - Abstraction to multithread platform
  - C-like programming
  - No explicit mapping to graphics rendering
  - E.g, CUDA, Brook+, openCL
- NVIDIA CUDA
  - Kernel func. on GPU
  - Threads->blocks->grids



NVIDIA® CUDA

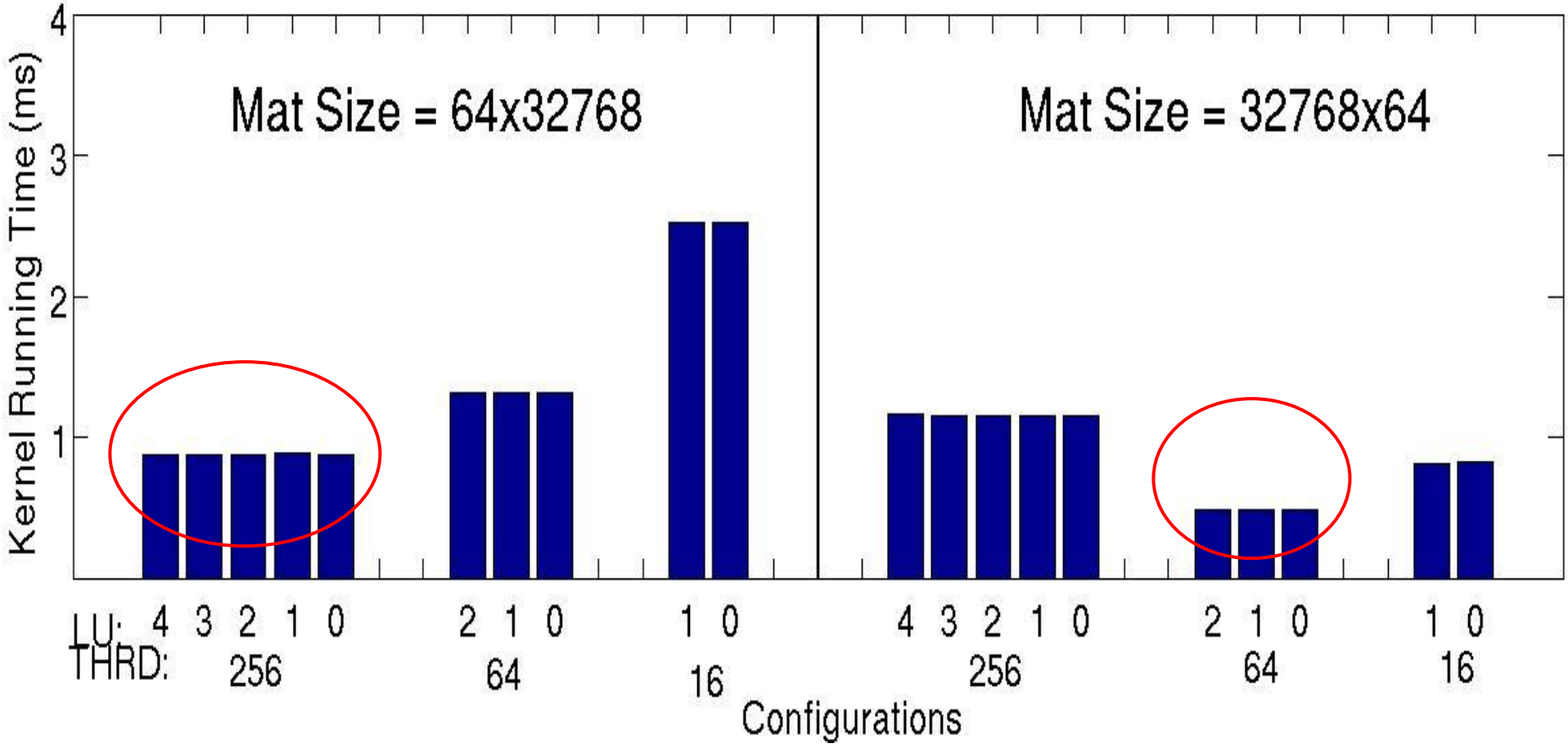
Graph From CUDA Manual

Optimization Challenges

- Goal
  - Maximize throughput
    - Increase occupancy, reduce latency, dynamic instr.

- Difficulties
  - Hard to predict optimization effects
    - Non-linearity, coupling, undisclosed CUDA details
  - GPU hardware complexities
    - Limits: 512 threads per block, 768 threads per SM, etc
    - Various types of memories: constant, texture, etc
  - Input sensitivity

Matrix-Vector Multiplication



Outline

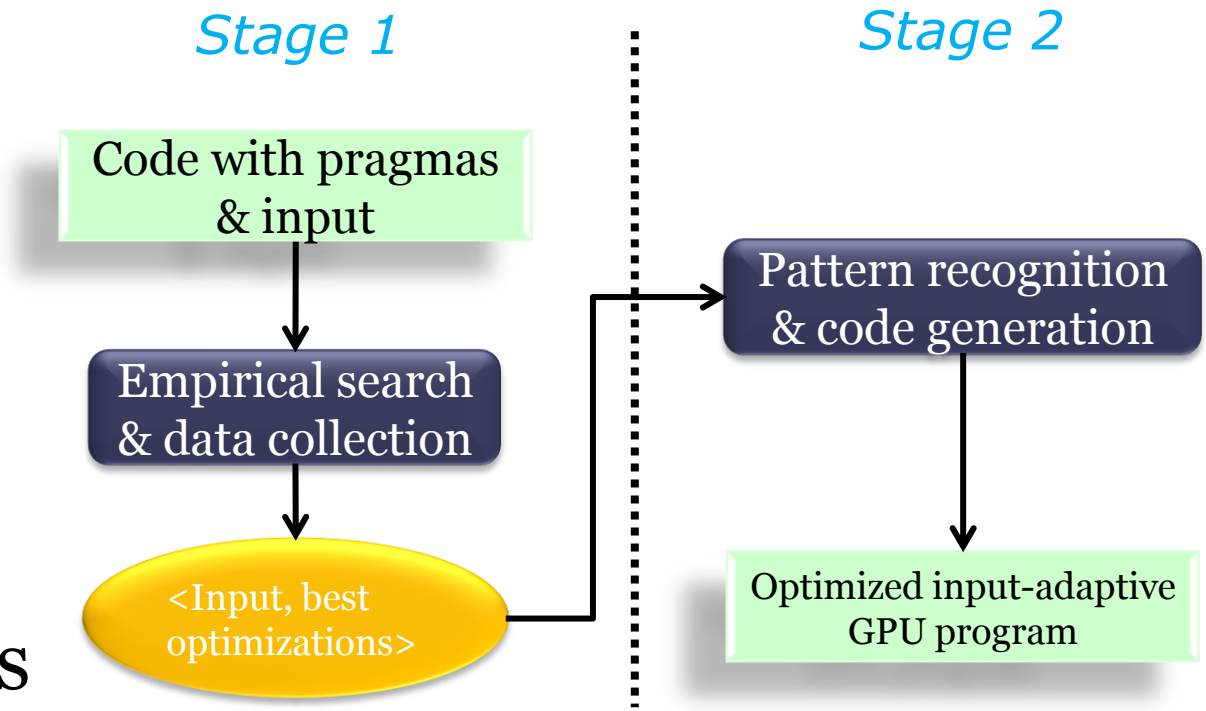
- GPU Overview
- **G-Adapt Framework**
- Evaluation
- Related & Future Work
- Conclusion

G-ADAPT

- Empirical search-based optimization
  - Three obstacles to address
    - Construction of the optimization space
    - Space pruning
    - Cross-input adaptation

G-ADAPT: Overview

- Source-to-source compiler
- Cross-input adaptation
- Automatic search & transformations



- Easy integration of user knowledge through pragmas

G-ADAPT Pragmas

- Supports a programmer-compiler synergy
- Covers 2 levels of optimizations
  - Execution configurations
    - E.g, thread block dimensions
  - Code transformations
    - E.g, loop tile size, unrolling levels

Pragma Examples

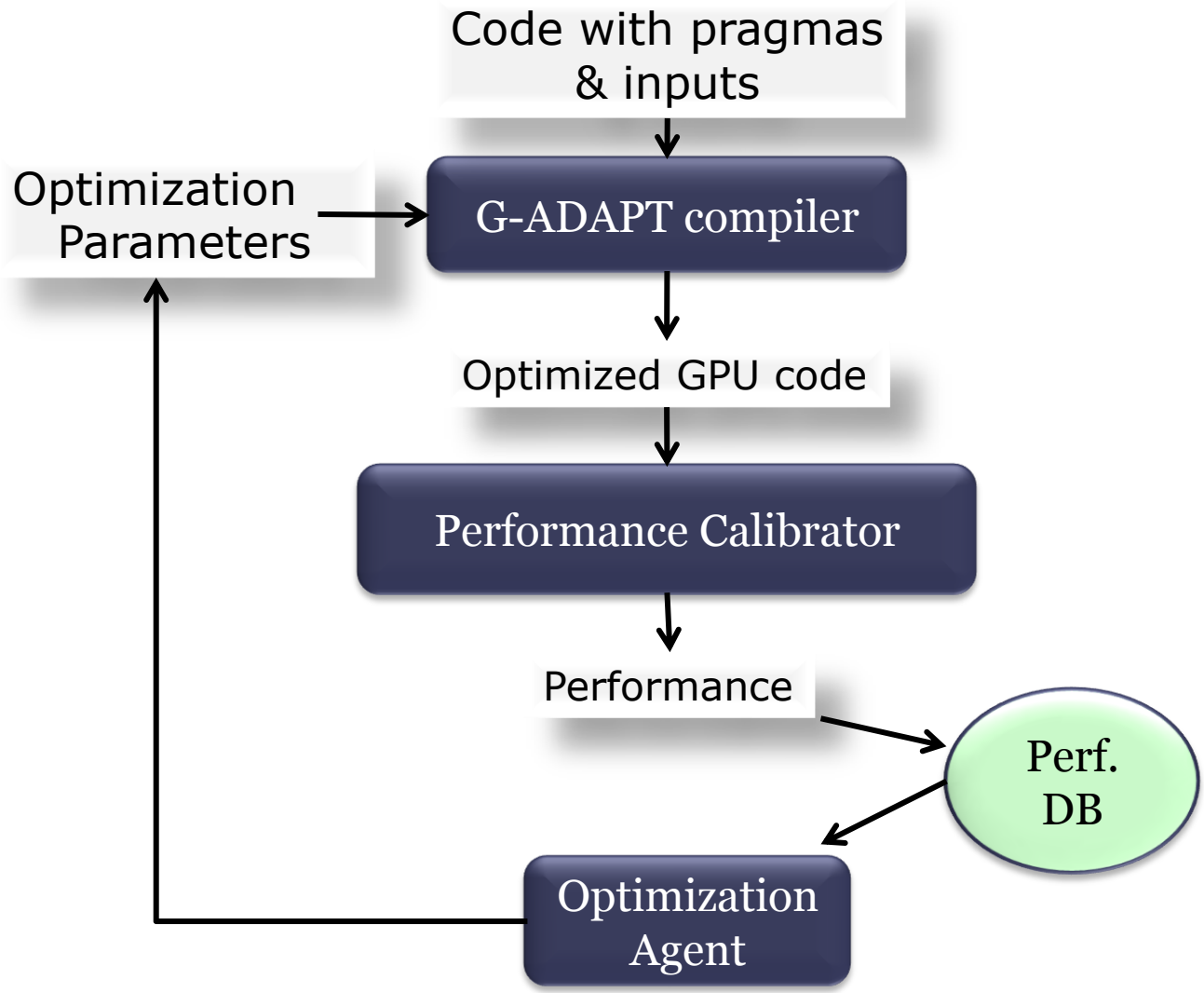
```
#pragma erange 64, 512, 2
#define BLKSZ 256
```

BLKSZ ranges from 64 to 512, increasing exponentially, doubling each time.

```
#pragma lpur_lrange 0, min(BLKSZ, 16), 2
For (i=1; i < BLKSZ; i++) {
   ……
}
```

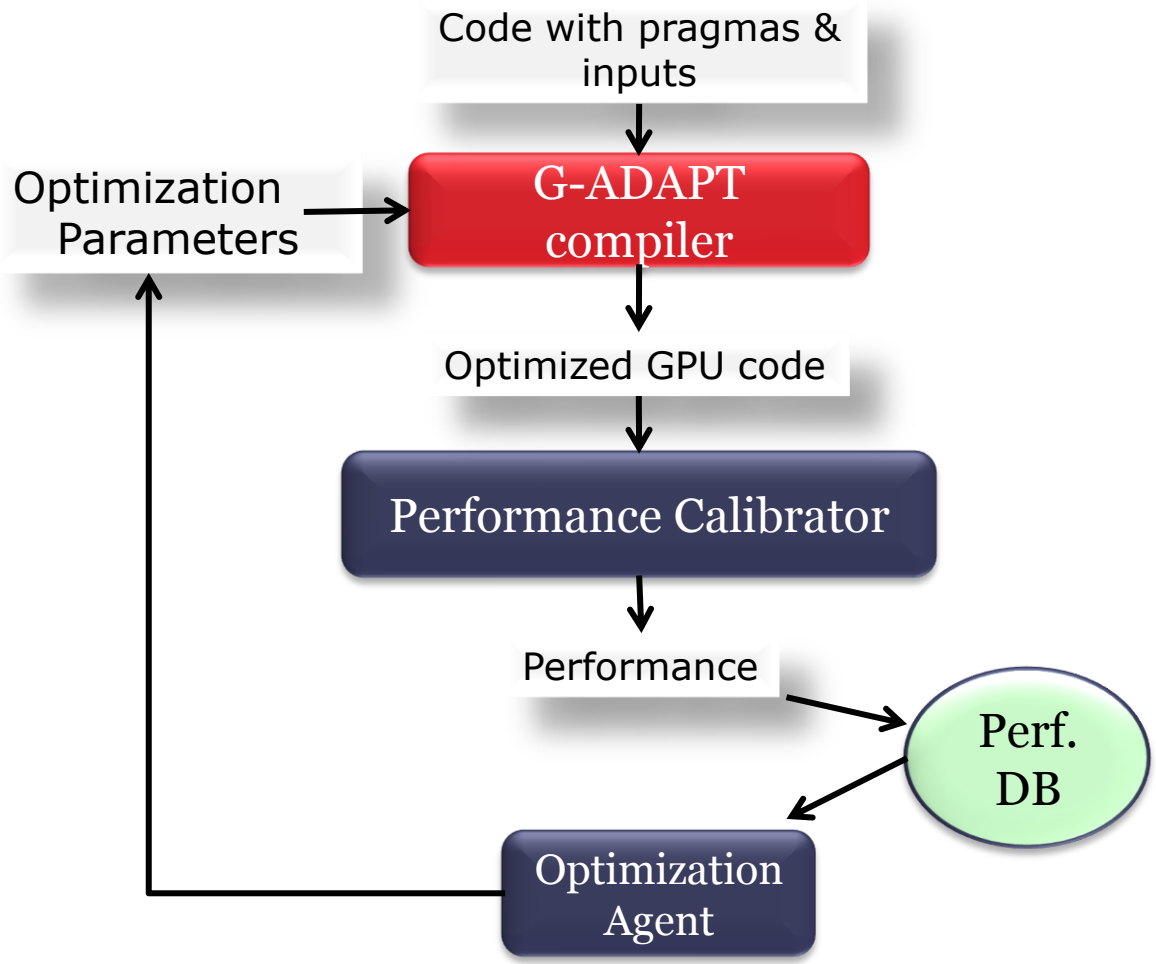
the loop unrolling level ranges from 0 to min(BLKSZ, 16), increasing linearly, each time by 2.

Stage I: Search & Collect



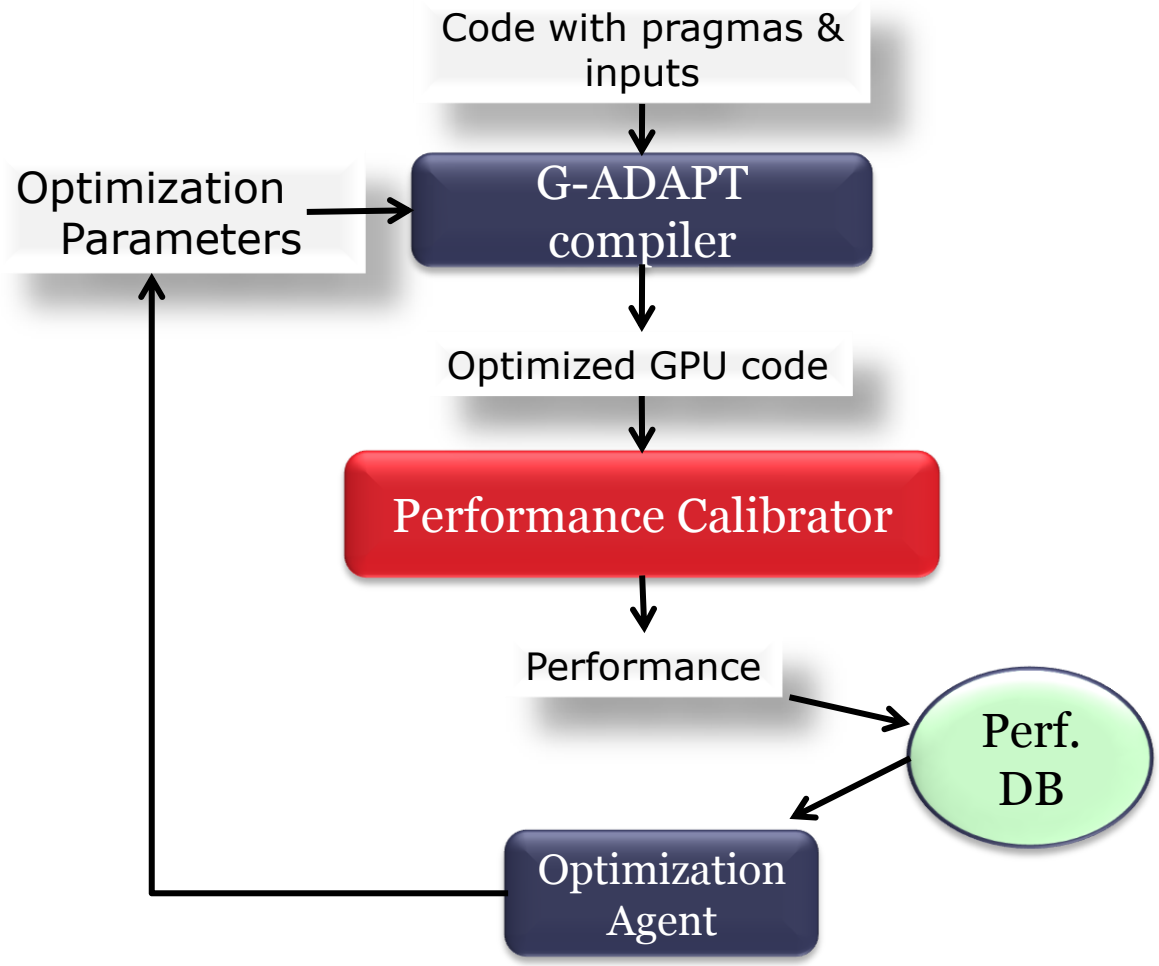
G-ADAPT Compiler

- Two functionalities
  - Recognize opt. space
  - Program transformations
- Based on Cetus [Purdue Univ]
- Source-to-source
- GPU extensions
- Support G-ADAPT pragmas



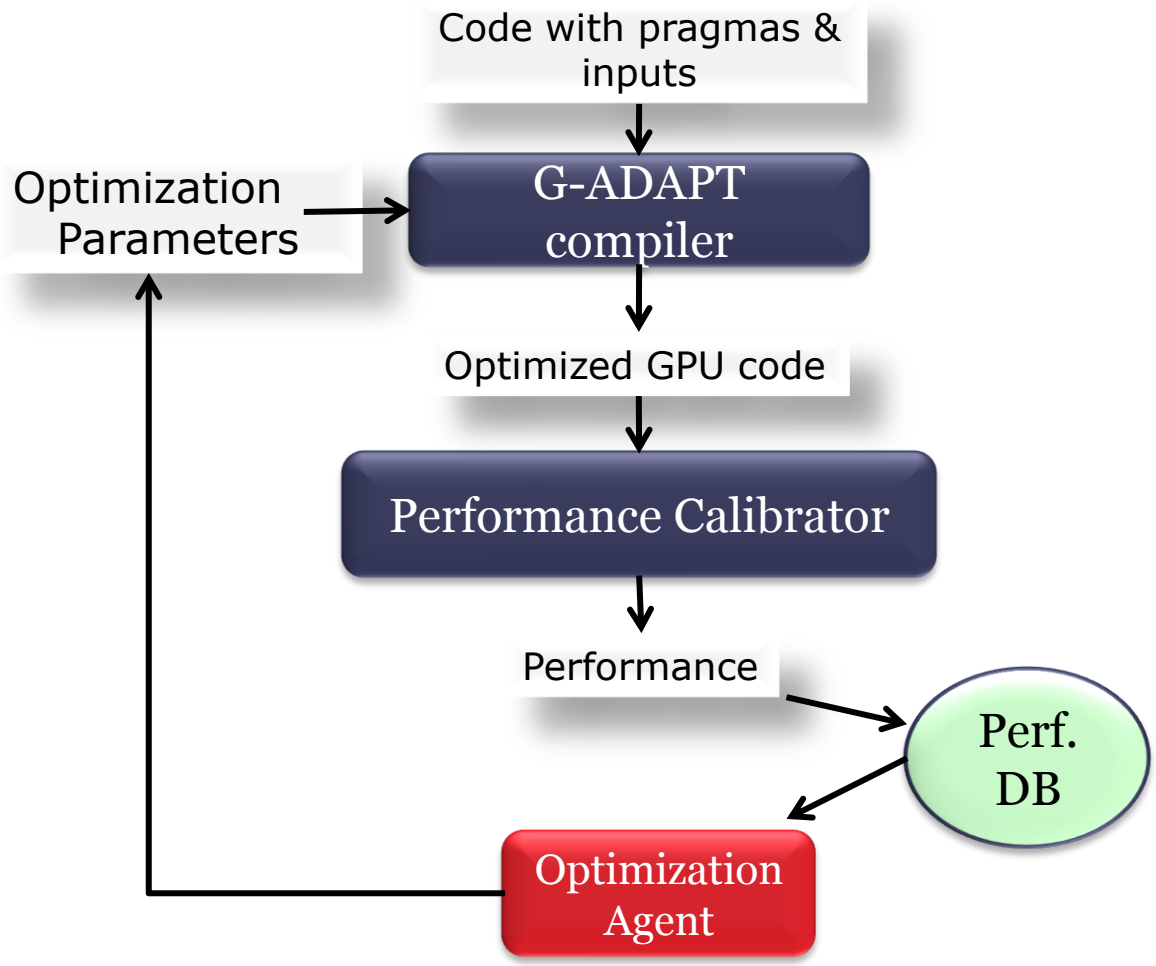
Performance Calibrator

- Invokes CUDA compiler and runs the executable
- Collect running time and GPU occupancy

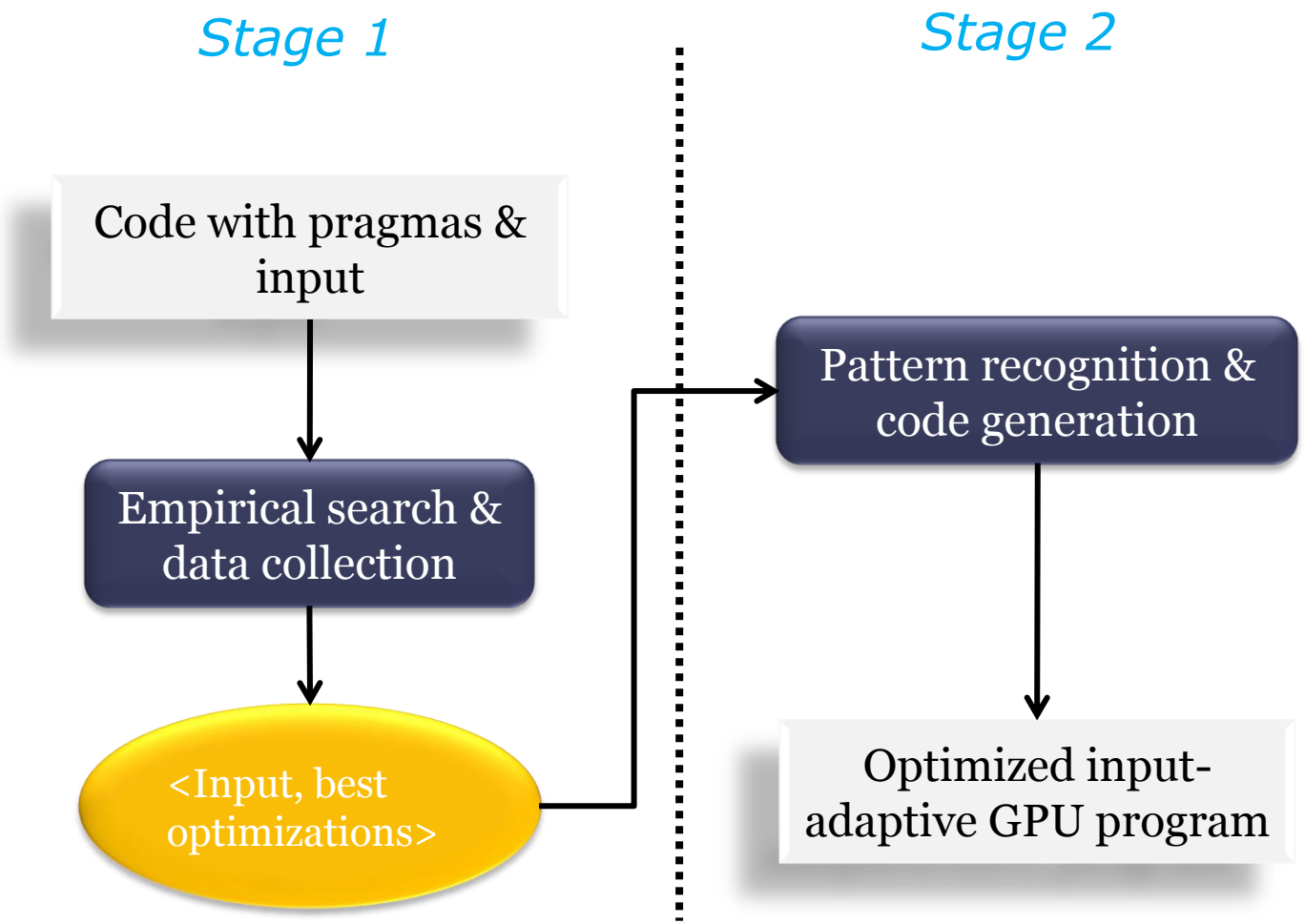


Optimization Agent

- Determines the optimization param. to try next
- Uses hill climbing to overcome space explosion problem



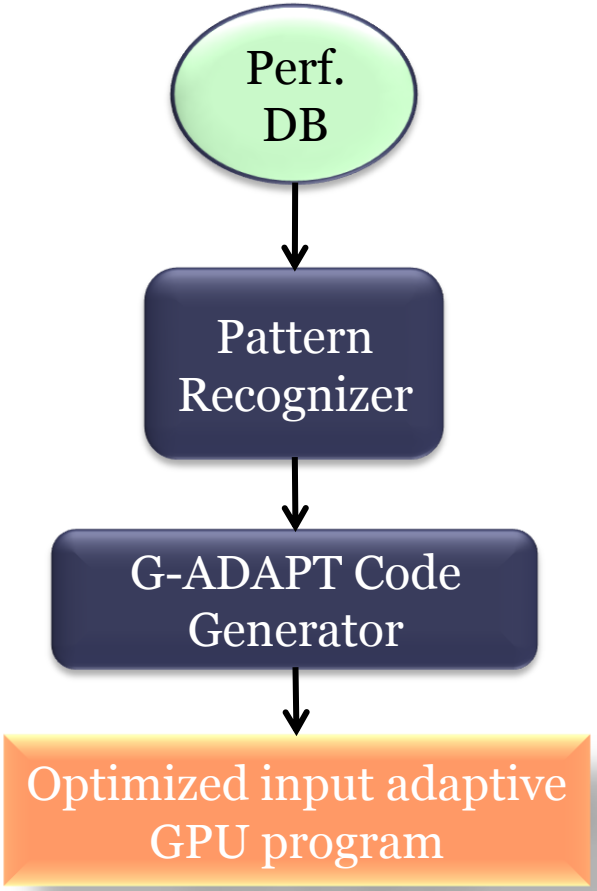
G-ADAPT: Overview



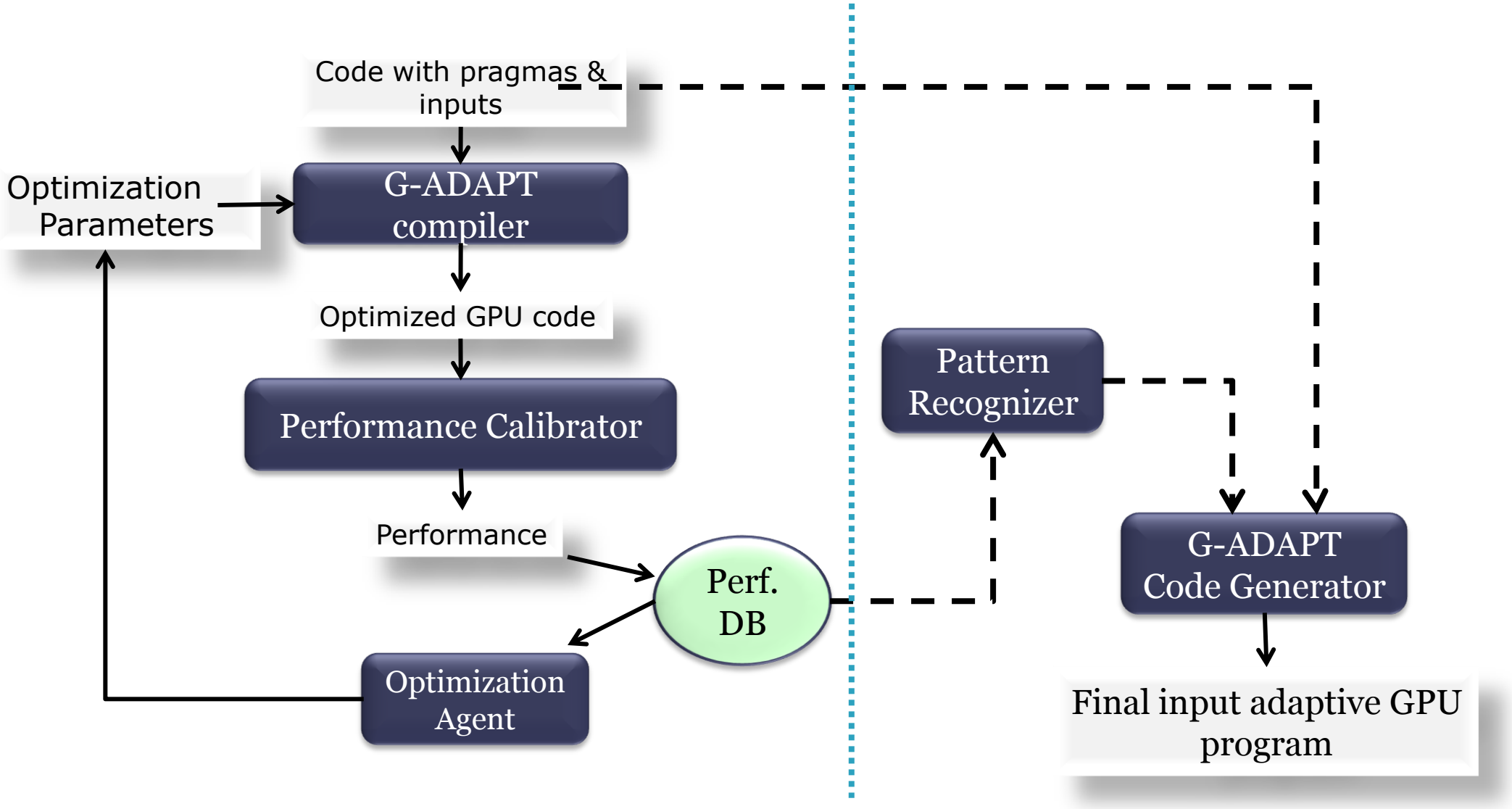
Stage II: PR & Code Gen

- Pattern recognizer
  - Recognize
    input ⟷ best parameters
  - Regression Trees with Least Mean Square

- Options for code generator
  - Multiple versions
  - JIT compilers
  - Linker



G-ADAPT


Code with pragmas & inputs
Optimization Parameters
G-ADAPT compiler
Optimized GPU code
Performance Calibrator
Performance
Perf. DB
Optimization Agent
Pattern Recognizer
G-ADAPT Code Generator
Final input adaptive GPU program

Outline

- GPU overview
- G-Adapt Framework
- **Evaluation**
- Related & Future Work
- Conclusion

Evaluation - Platform

- GPU: NVIDIA GeForce 8800 GT
  - 14 multiprocessors (MP), 112 cores
  - 512M global mem, 16KB shared mem/MP, 8192 registers/MP
  - CUDA 2.0
- Host: Intel Xeon 3.6 GHz, Suse Linux 2.6.22

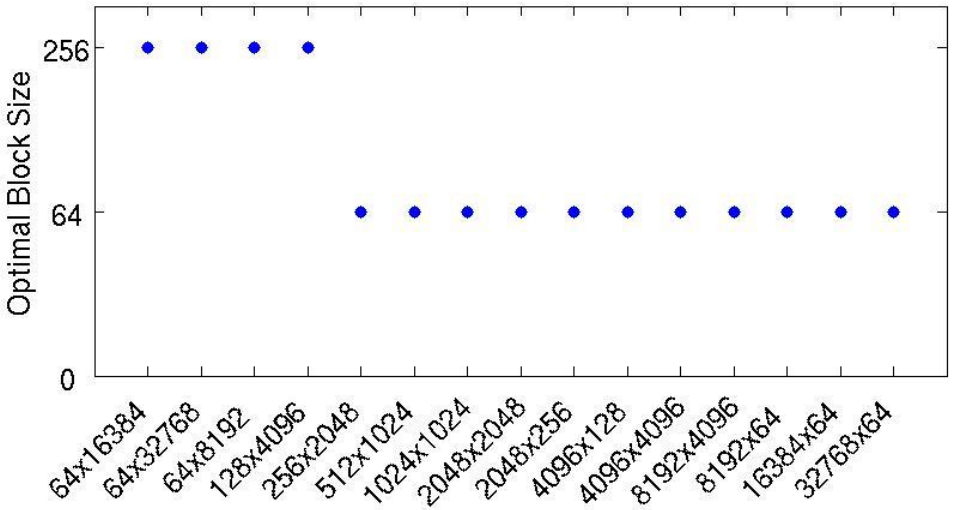
Benchmarks

Benchmark	Description	#of Inputs
Convolution	Convolution filter of a 2D signal	10
matrixMul	Dense matrix multiplication	9
mvMul	Dense matrix vector multiplication (by Fujimoto)	15
reduction	Sum of an array	15
scalarProd	Scalar products	7
transpose	Matrix transpose	18
Transpose-co	Coalescing matrix transpose	18

Training and Prediction

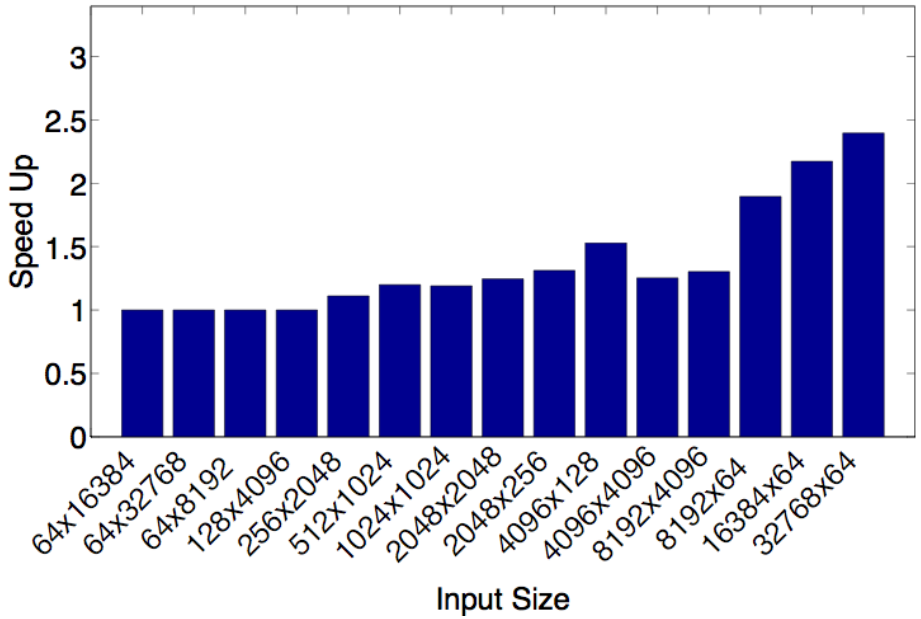
Benchmark	Training iterations	Training time (s)	Prediction accuracy
convolution	200	2825	100%
matrixMul	196	2539	100%
mvMul	124	124	93.3%
reduction	75	29	80%
scalarProd	93	237	100%
transpose	54	1639	100%
Transpose-co	54	631	100%

Matrix Vector Multiplication

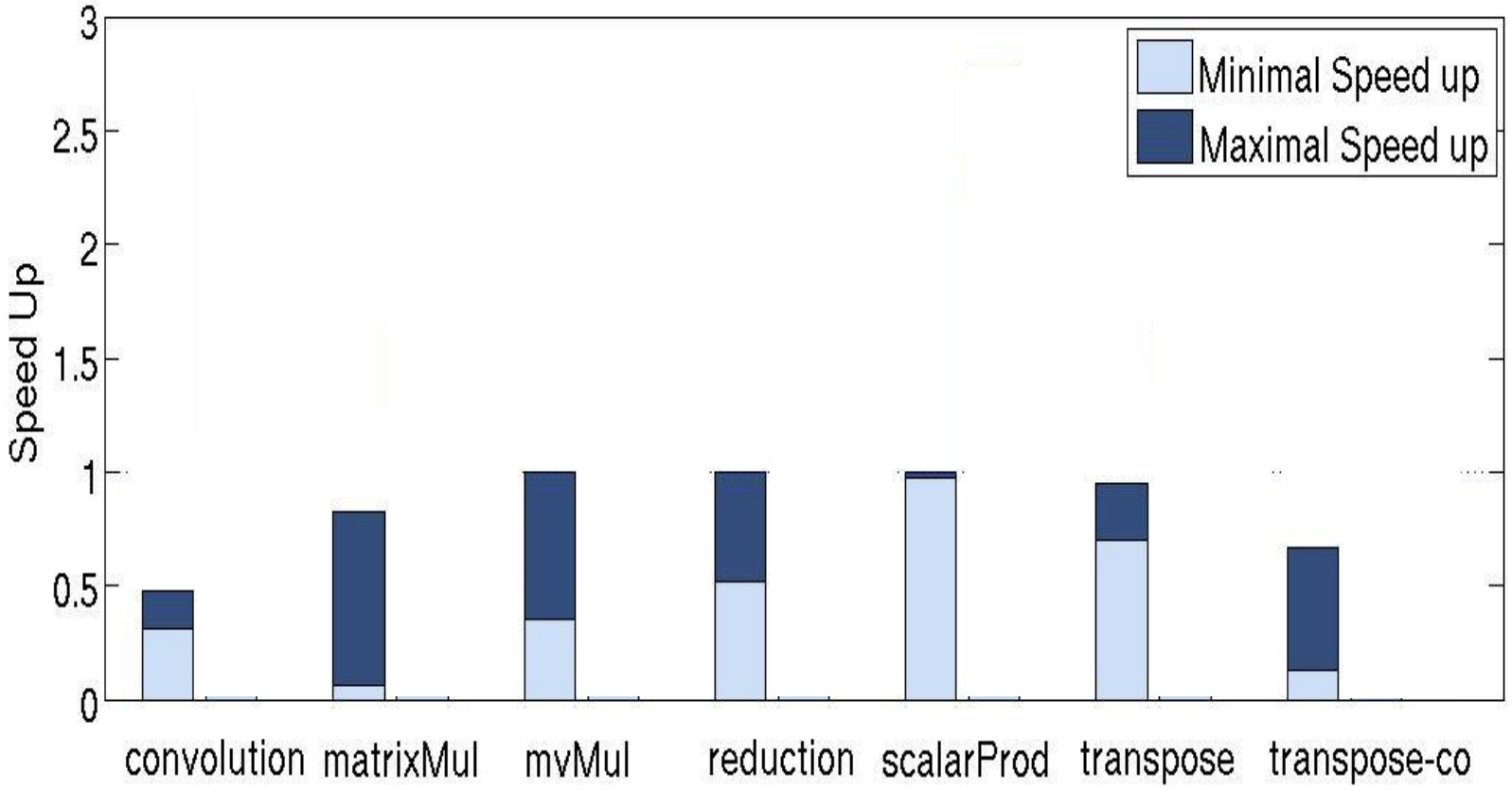


Best Parameter V.S. Input

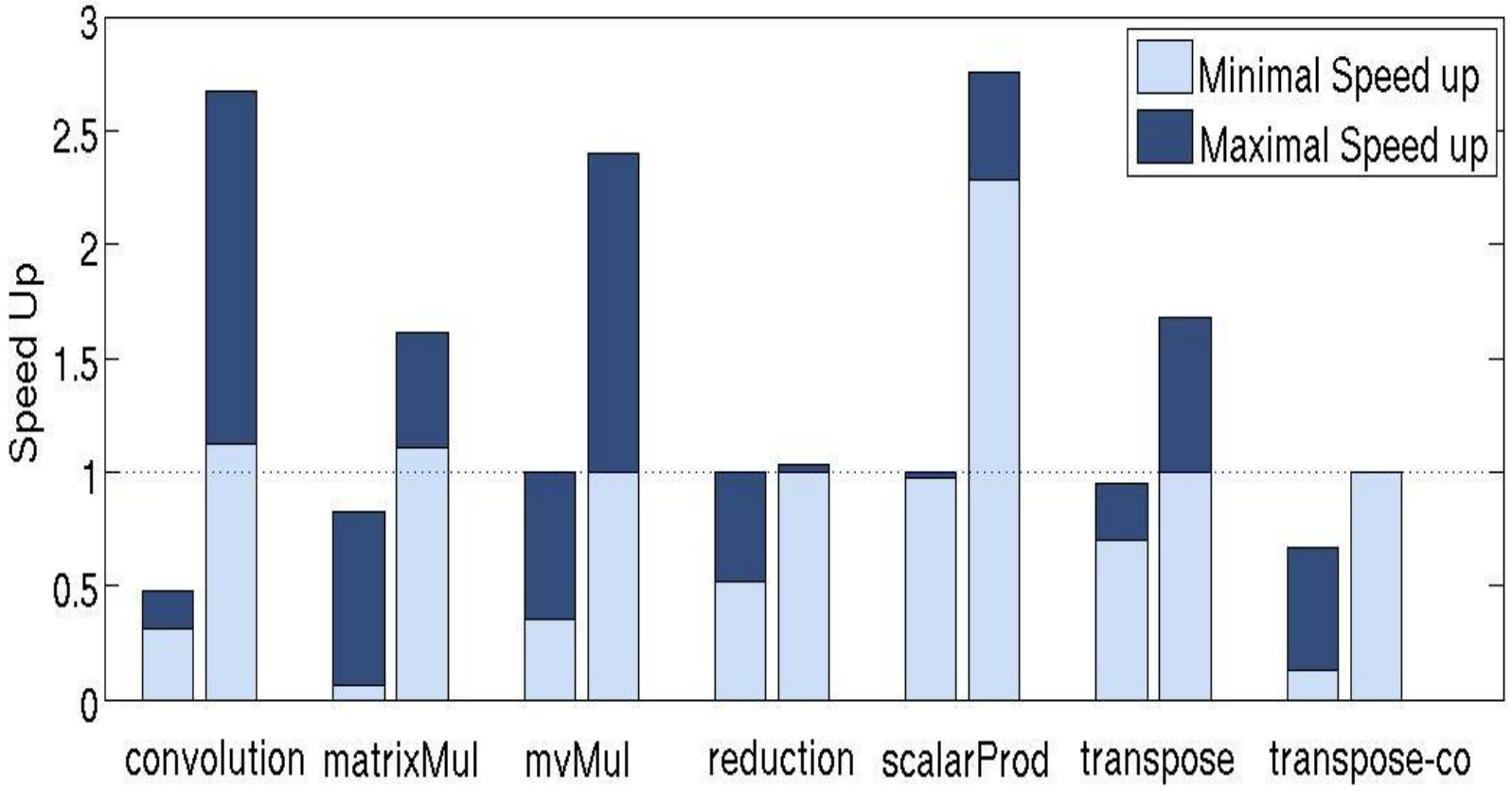
Speed up V.S. Input



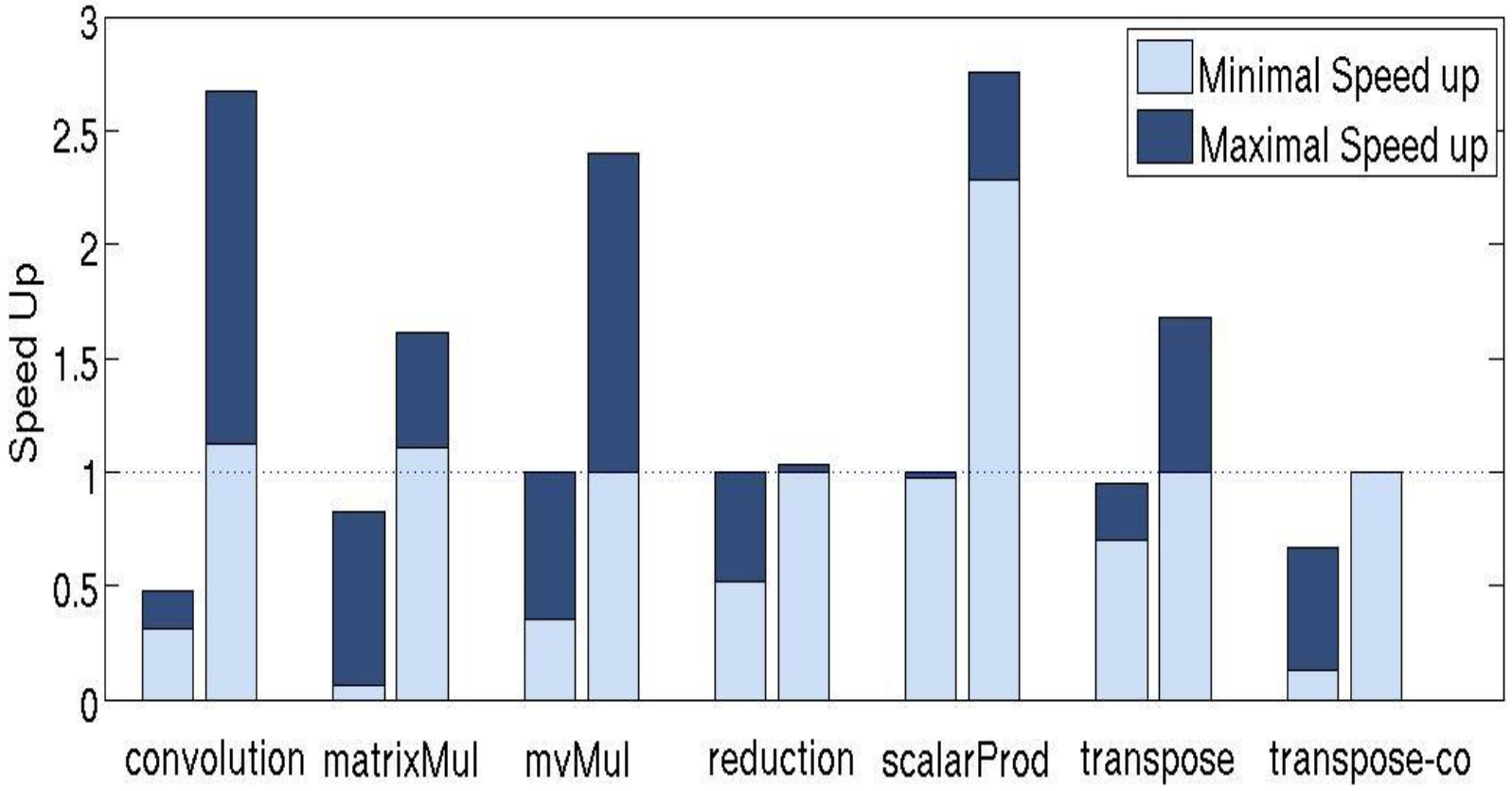
Speed up over default



Speed up over default



Speed up over default



http://www.cs.wm.edu/~xshen/Publications/ipdps09.pdf

Outline

- GPU overview
- G-Adapt Framework
- Evaluation
- **Related & Future Work**
- Conclusion

Related Work

- Ryoo+: CGO'08
  - Efficiency and utilization model for search
  - Manual transformation; assumptions on applications
- Baskaran+:ICS'08
  - Polyhedral model for optimizing memory access
  - Limited to affine loop nests

Features of G-Adapt

First generally applicable framework

Cross-input adaptation

Future Work

- More optimization options
  - Algorithm Selection
  - Memory optimization
  - Divergence Elimination
- General Support
  - Cetus – ANSI C compiler
    - Non ANSI C features, C++
  - CUDA built-in types
    - E.g. float4, texture and etc

Outline

- GPU overview
- G-Adapt Framework
- Evaluation
- Related & Future Work
- **Conclusion**

Conclusion

- A general tool for GPU optimization
- Cross-input adaptation
- Synergy between compilers and programmers
- Alternative of manual tuning, enabling easy adaptation across architectures

Acknowledgement

- Cetus authors at Purdue
  - Group led by Eigenmann and Midkiff
- John Owens
- NVIDIA
  - donation of device
- NSF grants

Thank you!