Poise:

Balancing Thread-Level Parallelism and Memory System Performance in GPUs using Machine Learning

Saumay Dublish*   Vijay Nagarajan‡   Nigel Topham‡

* Synopsys Inc.

‡ The University of Edinburgh

HPCA 2019

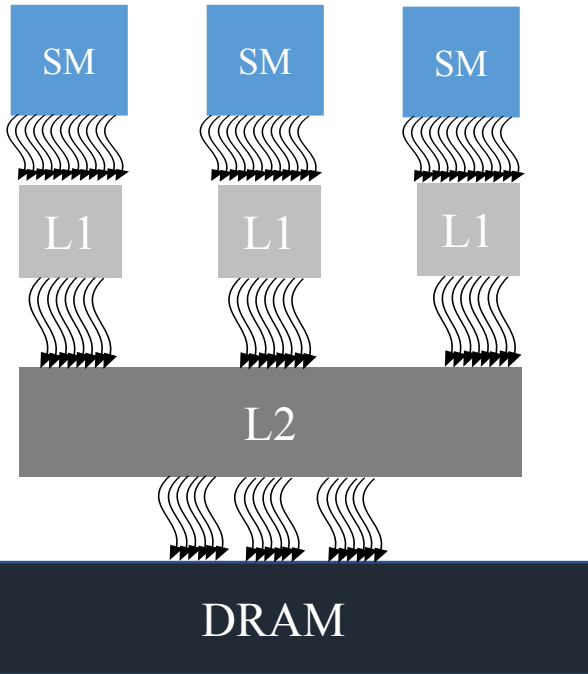Washington D.C., USA

19th February, 2019

SYNOPSYS®

THE UNIVERSITY *of* EDINBURGH

GPU Architecture

Overview

- GPUs are throughput-oriented systems

- Focus on overall system throughput

- Rely on high levels of multithreading

- Implemented by switching across warps
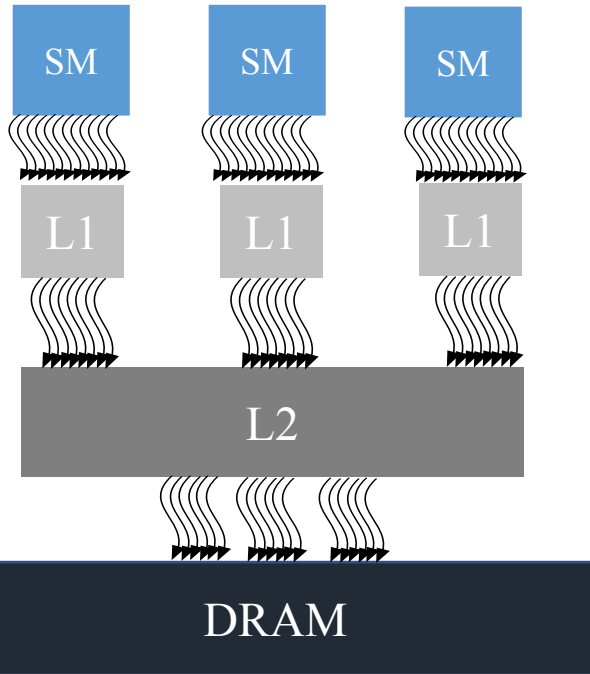
- Overlap latency with useful execution

GPU Architecture

Consequence of increasing TLP

- Increasing TLP not always useful

- Leads to cache thrashing

- Leads to bandwidth bottlenecks

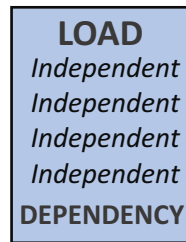- Results in high levels of congestion

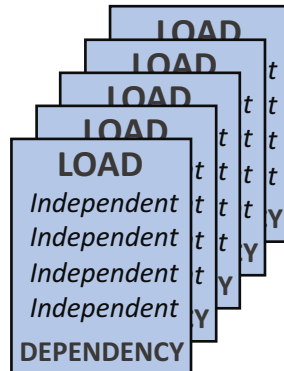- Latencies tend to be very high!

Can such high latencies be hidden?

Hiding Latencies in GPUs

Harnessing concurrency

Instruction concurrency

(Intra-warp concurrency)

| LOAD |
| Independent |
| Independent |
| Independent |
| Independent |
| DEPENDENCY |

Load latency

Execution

time

Warp concurrency

(Inter-warp concurrency)

| LOAD |
| Independent |
| Independent |
| Independent |
| Independent |
| DEPENDENCY |

Load latency

Execution

time

Hiding Latencies in GPUs

Harnessing concurrency

Instruction concurrency

(Intra-warp concurrency)

LOAD
Independent
Independent
Independent
Independent
DEPENDENCY

Load latency

Execution

stall

time

Warp concurrency

(Inter-warp concurrency)

LOAD
LOAD
LOAD
LOAD
LOAD
Independent
Independent
Independent
Independent
DEPENDENCY

Load latency

Execution

time

Hiding Latencies in GPUs

Harnessing concurrency

Instruction concurrency

(Intra-warp concurrency)

| LOAD |
| Independent |
| Independent |
| Independent |
| Independent |
| DEPENDENCY |

Load latency

Execution

stall

time

Warp concurrency

(Inter-warp concurrency)

| LOAD |
| LOAD |
| LOAD |
| LOAD |
| LOAD |
| Independent |
| Independent |
| Independent |
| Independent |
| DEPENDENCY |

Load latency

Execution

time

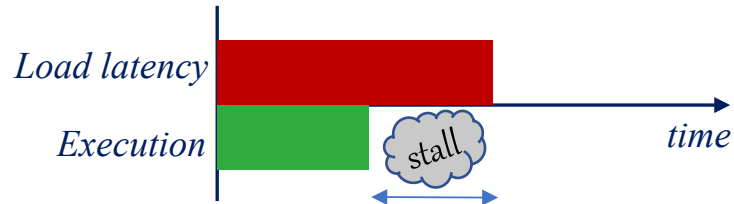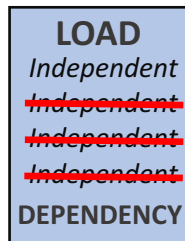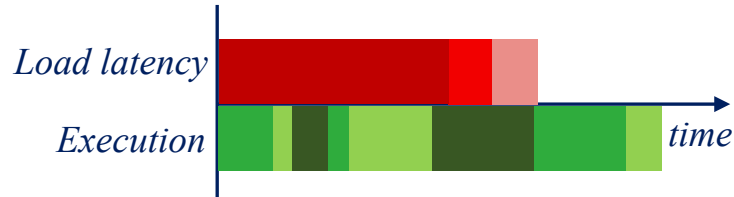**Works well in compute-intensive applications**

The Case of Limited Parallelism

Fewer independent operations

Instruction concurrency

(Intra-warp concurrency)

Warp concurrency

(Inter-warp concurrency)

The Case of Limited Parallelism

Fewer independent operations

Instruction concurrency

(Intra-warp concurrency)

LOAD
Independent
~~Independent~~
~~Independent~~
~~Independent~~
DEPENDENCY

Load latency

Execution

stall

time

Warp concurrency

(Inter-warp concurrency)

LOAD
LOAD
LOAD
LOAD
LOAD
Independent
~~Independent~~
~~Independent~~
~~Independent~~
DEPENDENCY

Load latency

Execution

stall

time

The Case of Limited Parallelism

Fewer independent operations

Instruction concurrency

(Intra-warp concurrency)

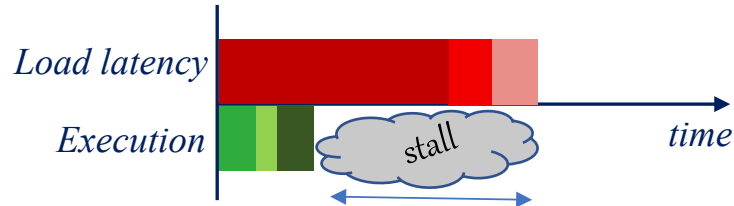| LOAD |
| --- |
| *Independent* |
| ~~*Independent*~~ |
| ~~*Independent*~~ |
| ~~*Independent*~~ |
| **DEPENDENCY** |

Load latency

Execution

stall

time

Warp concurrency

(Inter-warp concurrency)

LOAD
LOAD
LOAD
LOAD
LOAD
Independent
~~*Independent*~~
~~*Independent*~~
~~*Independent*~~
DEPENDENCY

Higher load latency due to congestion

Load latency

Execution

time

Impractically large number of warps required to completely hide latency

Need For Balance

Tension between TLP and memory system performance

- Increase TLP to improve concurrency – latency worsens
- Reduce TLP to reduce latency – concurrency worsens

Concurrency   **Memory Performance**
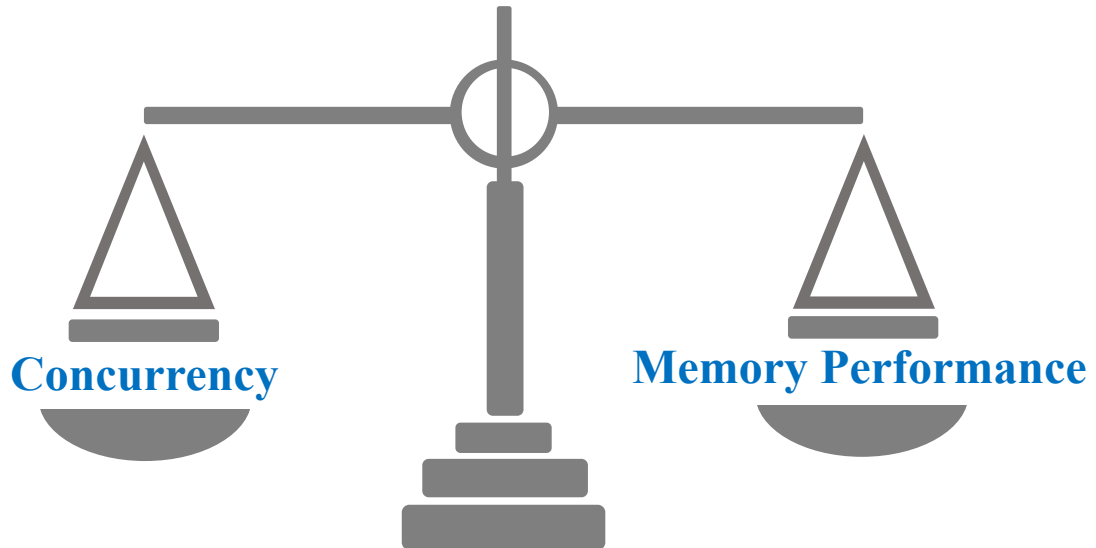
Need For Balance

Tension between TLP and memory system performance

- Increase TLP to improve concurrency – latency worsens
- Reduce TLP to reduce latency – concurrency worsens

Need For Balance

Tension between TLP and memory system performance

- Increase TLP to improve concurrency – latency worsens
- Reduce TLP to reduce latency – concurrency worsens

Concurrency

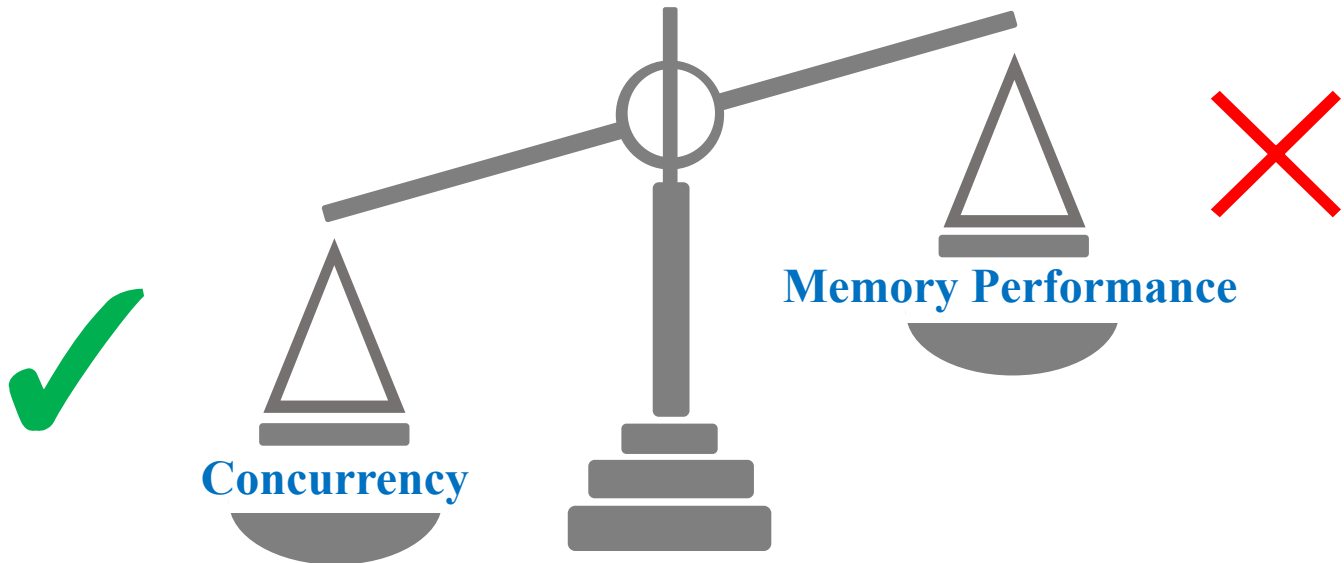**Memory Performance**

Need For Balance
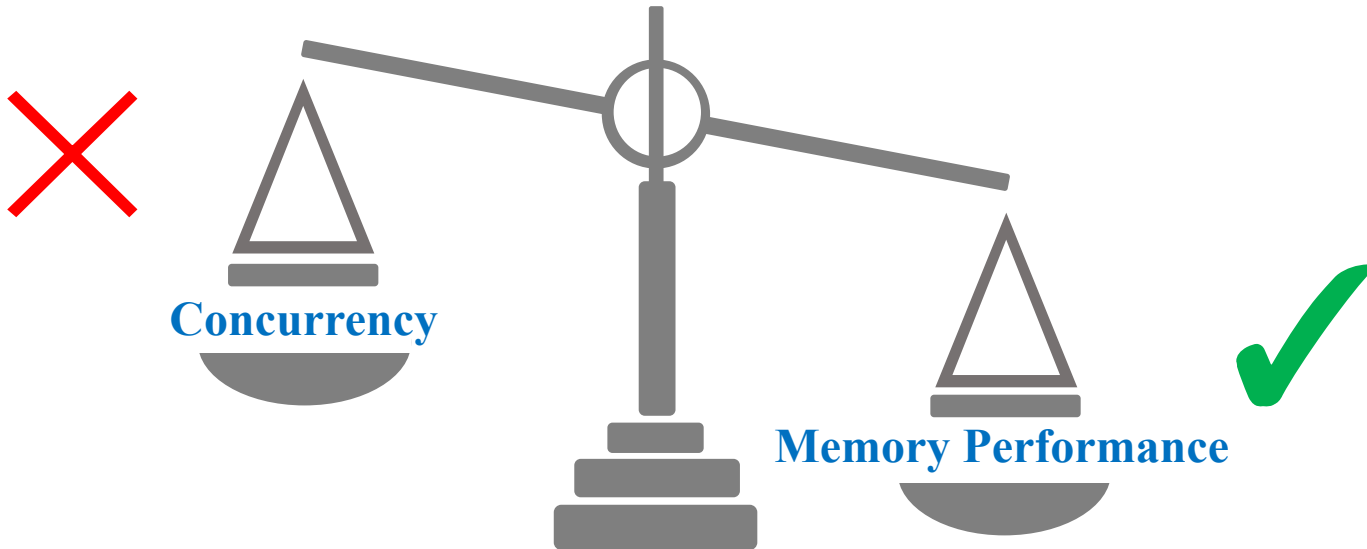
Tension between TLP and memory system performance

- Increase TLP to improve concurrency – latency worsens
- Reduce TLP to reduce latency – concurrency worsens

Concurrency ✔   ✔ **Memory Performance**
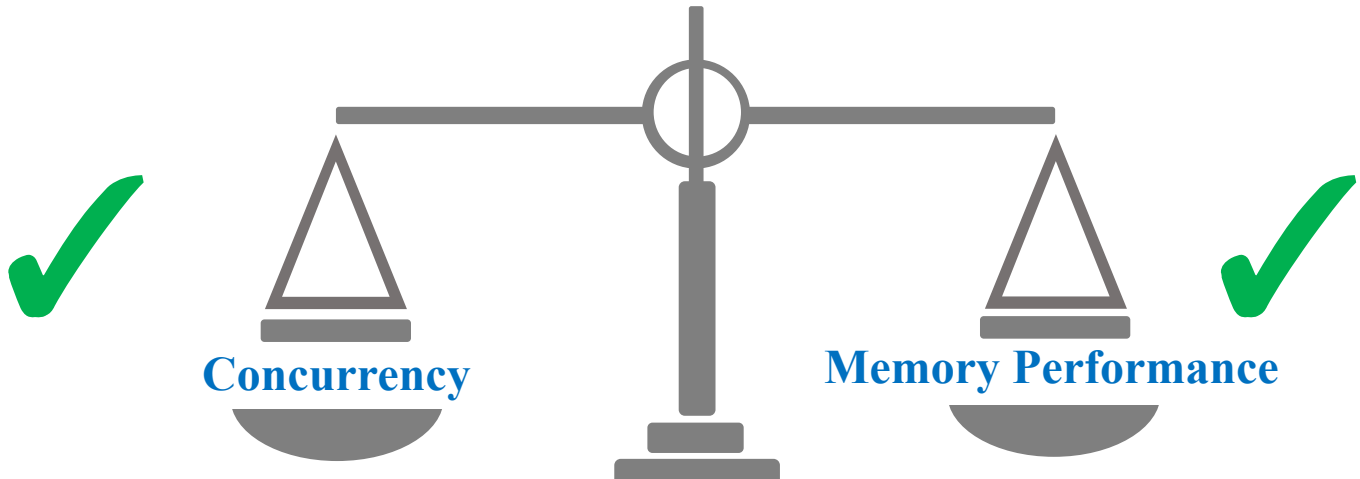
Optimal system throughput with balanced TLP and memory performance

Outline

- **Problem Statement**   Balancing TLP and memory performance

- **Prior state-of-the-art**   *CCWS and PCAL warp schedulers*

- **Pitfalls in prior techniques**   *Iterative search and prone to local optima*

- **Goals**   *Computing the best warp scheduling decisions*

- **Proposal**   *Poise*

- **Results**   *Experimental results*

- **Conclusion**   *Key takeaways*

Prior state-of-the-art

Warps

W1 W2 W3 W4

L1 cache

W1 W2 W3 W4
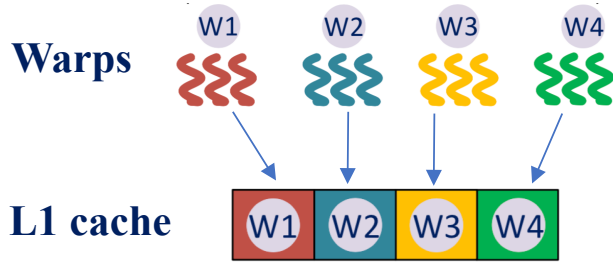
Cache Thrashing

Memory Congestion

Prior state-of-the-art

Cache-conscious wavefront scheduling (CCWS)

Limits the degree of multithreading
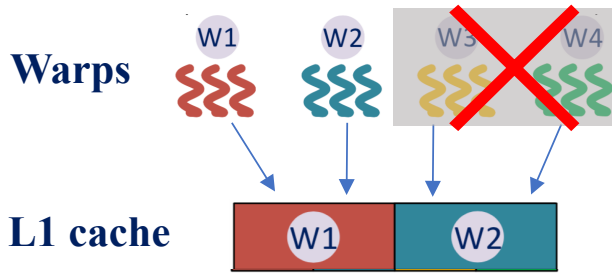
Reduces cache thrashing

Relieves congestion

Shortcomings
- Restricted coupling of warps with cache performance
- Underutilization of shared memory resources
- Dynamic policy has significant performance and cost overheads
- Static policy burdens the user with the task of profiling every workload

Prior state-of-the-art

Priority-based cache allocation (PCAL)

Alter parallelism independent of memory system performance
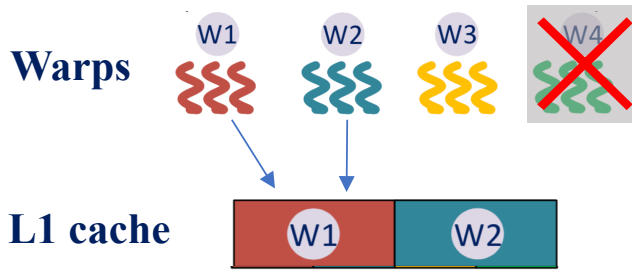
Warps

W1 W2 W3 W4

L1 cache

W1   W2

Prior state-of-the-art

Priority-based cache allocation (PCAL)

Vital warps (W1, W2, W3)

Warps

W1 W2 W3 W4

L1 cache

W1 W2

Cache-polluting warps
(W1, W2)

Cache-polluting warps

CCWS search space

PCAL search space

Vital warps

Prior state-of-the-art

Priority-based cache allocation (PCAL)

Vital warps *(N)*

Determine degree of multithreading

Cache-polluting warps *(p)*

Subset of vital warps
Ability to allocate and evict the L1 cache
Reduce cache contention

Warp-tuple *{ N, p }*
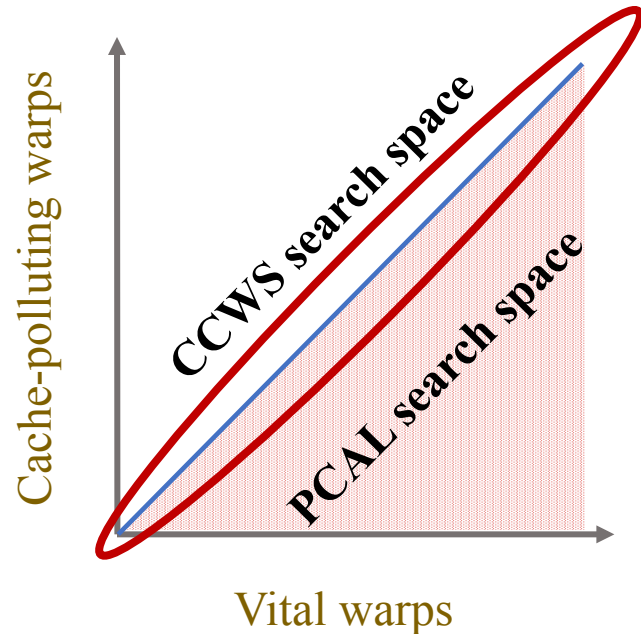
Limitations of PCAL

- Heuristic-based iterative search are slow in hardware

- Prone to local optima in presence of multiple performance peaks

- These two limitations lead to sub-optimal solutions

Goals
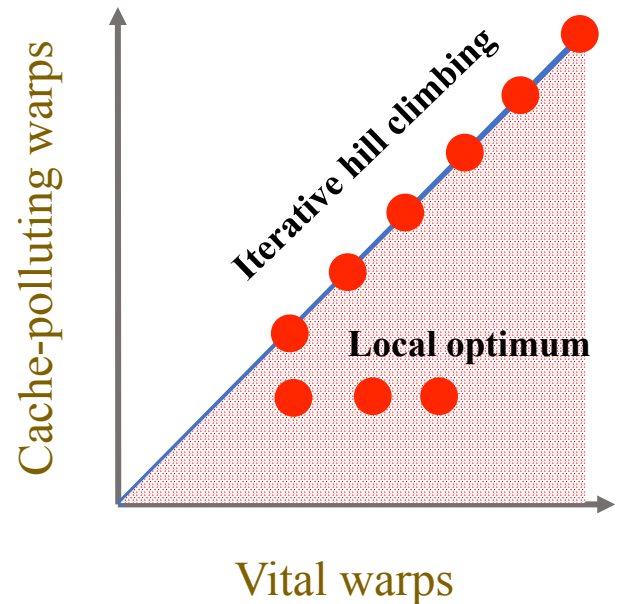
How to find the best warp-tuple?

- Balance TLP and memory performance

- Avoid local optima

- Converge expeditiously

- Low sampling and hardware overhead

- Avoid burdening the user

Cache-polluting warps

Best warp-tuple?

Vital warps
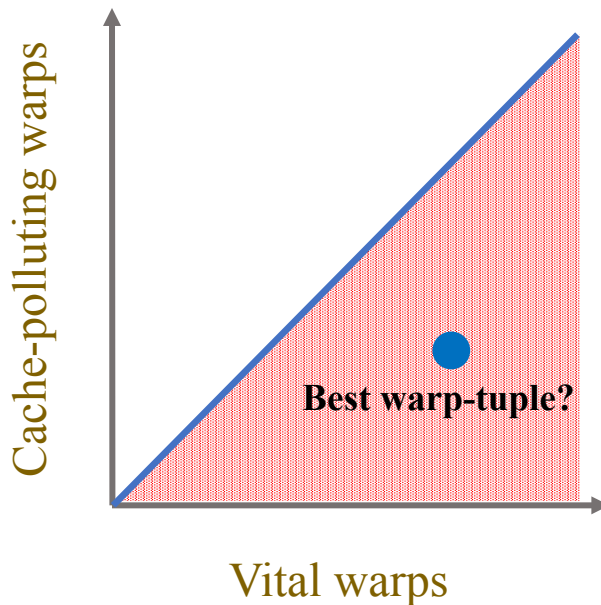
Proposal

Poise

A technique to dynamically balance TLP and memory system performance

Machine Learning Framework

Supervised learning

Hardware Inference Engine

Runtime prediction

Unseen user application

Runtime Input

Feature Set

Sample Input

Training Dataset

Regression Model

Feature weights

via compiler

Prediction Stage & Local Search

Poise Prediction

Best warp-tuple

Sample Output

Best warp-tuple

Profiled Kernels

Machine Learning Framework

Analytical Model

- Analytical model uses domain knowledge to identify reliable features

- Allows us to reason about the effectiveness of different features

- Proposed feature vector consists of only seven features

More details about the analytical model in the paper

Machine Learning Framework

Regression Model

- We use Negative Binomial regression to perform supervised learning

- Inputs are mapped to the output using a log-linear link function

- Reasons for selecting Negative Binomial regression:

  - Predicts discrete non-negative warp-tuple values

  - Lightweight in training time and dataset

  - Low computational demand for training and inference

Hardware Inference Engine

- Computes runtime predictions about good warp-tuples for new workloads

- Constitutes a *prediction stage* and *local search*
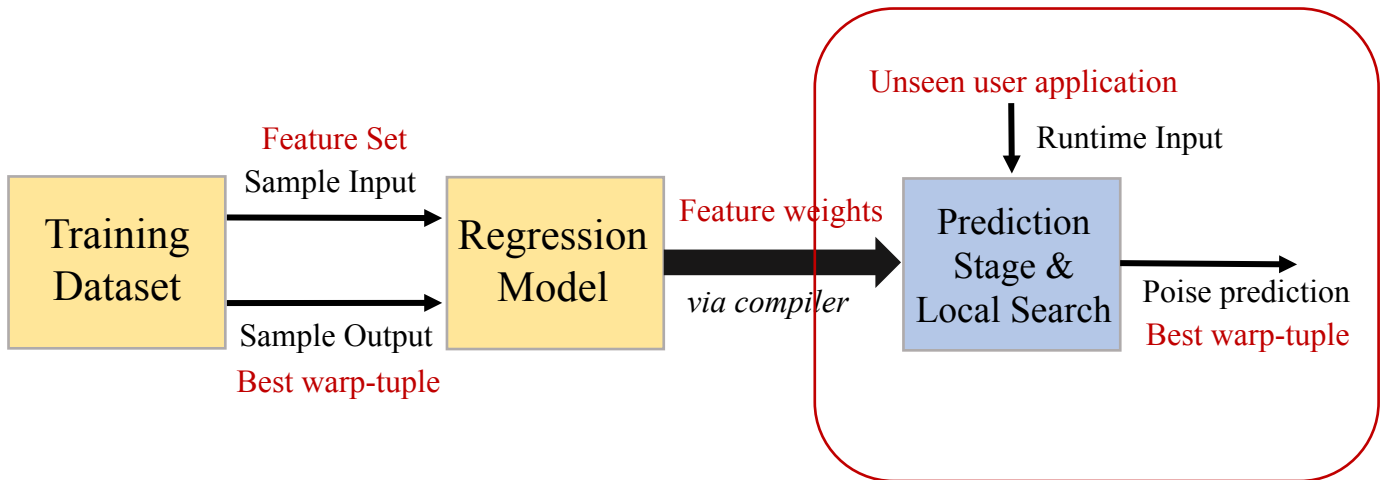
Hardware Inference Engine

Prediction Stage

Perform predictions at runtime using new features and learned mapping

Unseen user application

Runtime Input

Feature weights

via compiler

Prediction Stage

Predicted Output

Good warp-tuple

Runtime Feature Collection
Performance Counters

Dot product
Weights ● Features

Inference
Log-linear link function
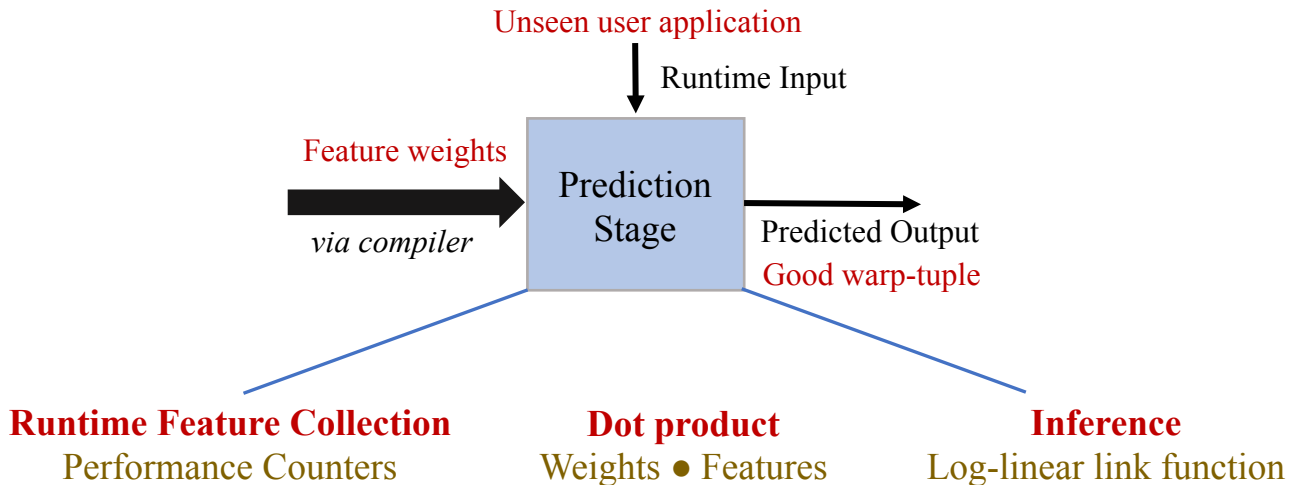
Hardware Inference Engine

Local Search

Mitigate statistical errors in prediction with a near-neighborhood search via gradient ascent
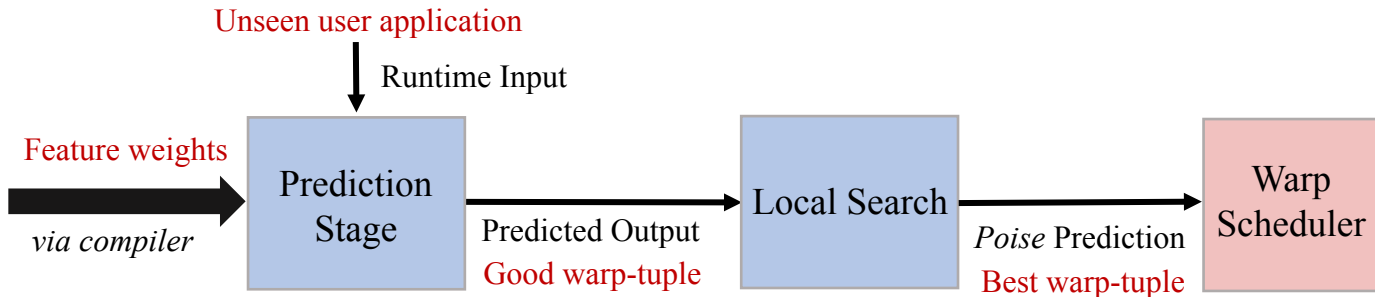
Local search is less prone to getting trapped at local optima due to proximity to performance peaks

Working Summary

PCAL

Iterative hill climbing

Local optimum

Cache-polluting warps

Vital warps

Poise

Feature collection

Feature collection

Local Search

Prediction

Cache-polluting warps

Vital warps
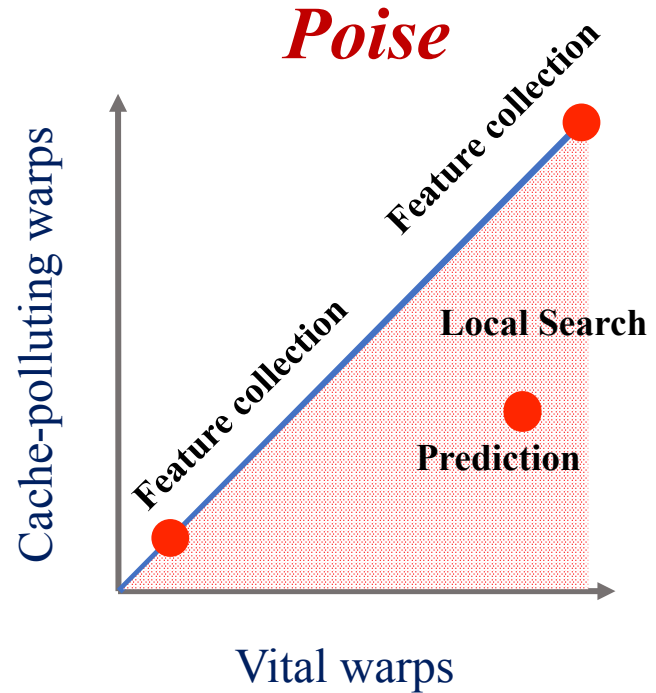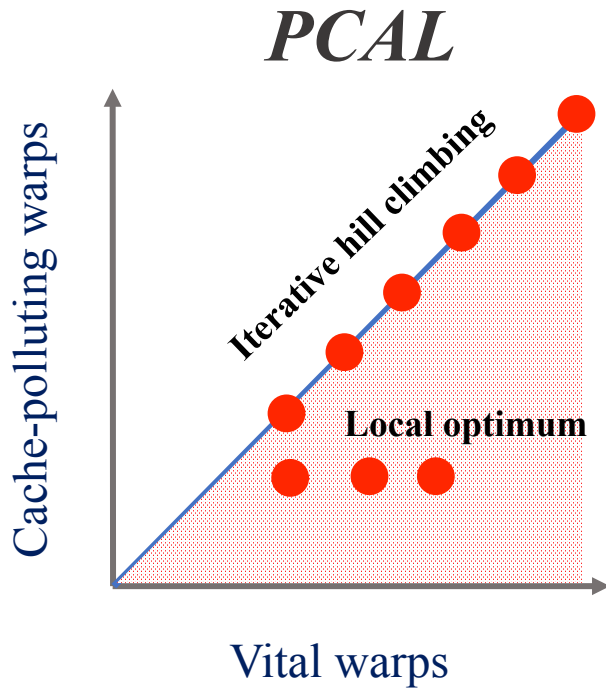
Warp Scheduler Architecture

GTO warp scheduler

Warp Scheduler Queue

Latest	W_{MAX-1}
	$\cdots$
	$\cdots$
	$\cdots$
	$\cdots$
	W_2
	W_1
Oldest	W_0

Warp-ID
bits

Warp Scheduler Architecture

Warp Scheduler Queue

Warp-ID bits	Vital bit	Pollute bit
W_{MAX-1}	1	1
. . .	1	1
. . .	1	1
. . .	1	1
. . .	1	1
W_2	1	1
W_1	1	1
W_0	1	1

Latest — top, *Oldest* — bottom

Poise   Warp Scheduler Architecture

Warp Scheduler Architecture

Compiler

Feature weights

Constant Memory

Hardware Inference Engine

Vital warps **(N)**
Cache-polluting warps **(p)**

Warp Scheduler Queue

Latest

Warp-ID bits	Vital bit	Pollute bit
W_{MAX-1}	0	1
...	0	1
...	1	1
...	1	1
...	1	1
W_2	1	1
W_1	1	1
W_0	1	1

Oldest

N
Vital Warps

Warp Scheduler Architecture

Compiler

Feature weights

Constant Memory

↓

Hardware Inference Engine

Vital warps **(N)**
Cache-polluting warps **(p)**

Warp Scheduler Queue

Latest

Warp-ID bits	Vital bit	Pollute bit
W_{MAX-1}	0	0
...	0	0
...	1	0
...	1	0
...	1	0
W_2	1	1
W_1	1	1
W_0	1	1

Oldest

Warp-ID bits   *Vital* bit   *Pollute* bit

p
Cache-polluting warps

Warp Scheduler Architecture

Compiler

Feature weights

Constant Memory

↓

Hardware Inference Engine

Vital warps (N)
Cache-polluting warps (p)

Warp Scheduler Queue

Latest

Warp-ID bits	Vital bit	Pollute bit
W_{MAX-1}	0	0
...	0	0
...	1	0
...	1	0
...	1	0
W_2	1	1
W_1	1	1
W_0	1	1

Oldest

→ Do not participate in TLP

→ **LOAD [a]**
Do not pollute cache
(bypass on read miss)

→ **LOAD [b]**
Allocate and replace cache lines

L1 Cache

Evaluation

- **Platform**

  - Statsmodels   – *regression analysis*
  - GPGPU-Sim (v3.2.2)   – *cycle-accurate simulator*
  - GPUWattch (McPAT) – *energy and area estimation*

- **Benchmark Suites ***

  - Rodinia
  - MapReduce
  - Graph Suite
  - Polybench

  *Training and evaluation are done on disjoint set of benchmarks

Evaluation   Methodology

Evaluation

- **Baseline GPU configuration**

  - 32 Streaming Multiprocessors (SM)

  - 16 KB Private L1 Cache

  - 2.25 MB Shared L2 Cache

 - GTO warp scheduler

  - 48 warps per SM

Evaluation

- **Warp Scheduling Schemes**

  - GTO
    - Baseline greedy-then-oldest warp scheduler
    - Maximum warps enabled per SM for multithreading

  - SWL
    - Static Warp Limiting from the CCWS scheduler
    - No runtime overheads in a static policy

  - PCAL-SWL
    - Dynamic PCAL policy with SWL for initial start

  - Static-Best
    - Each kernel run at best performing warp-tuple
    - Determined by offline profiling of each kernel

Results

Performance
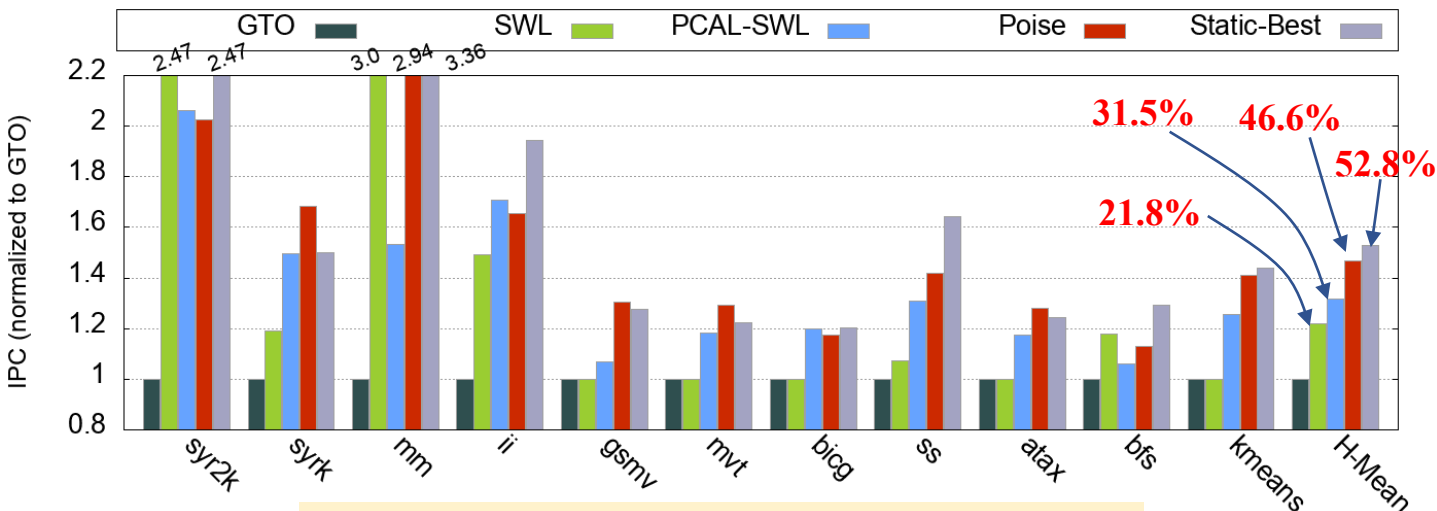
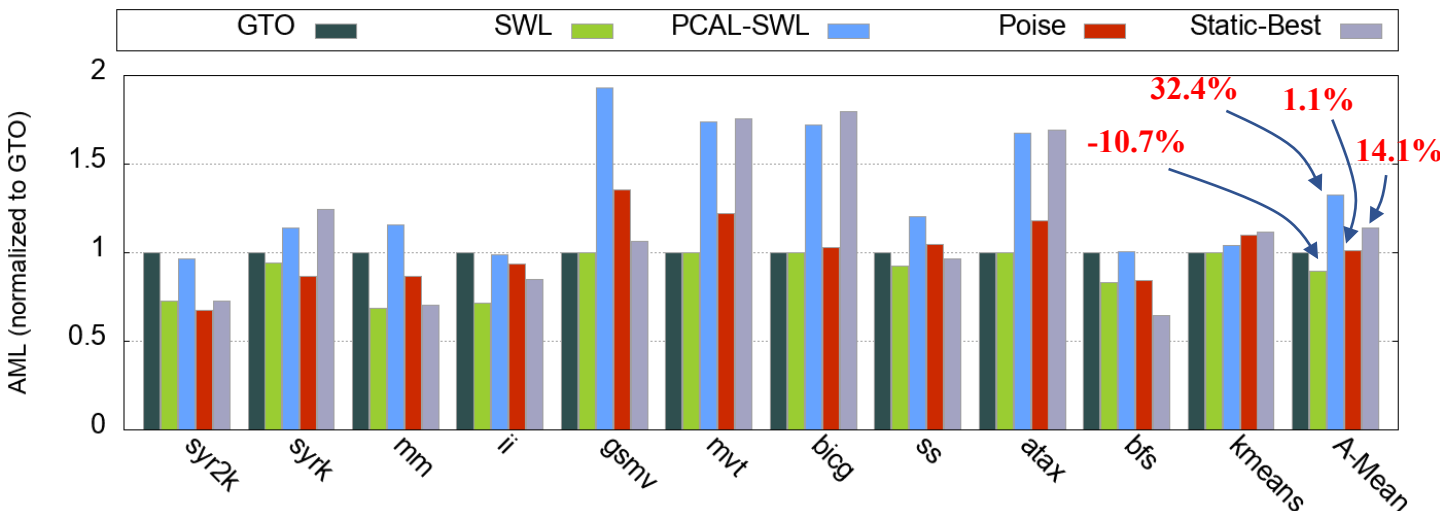*Poise* outperforms PCAL-SWL by 15.1% on average

Results

L1 Hit Rate
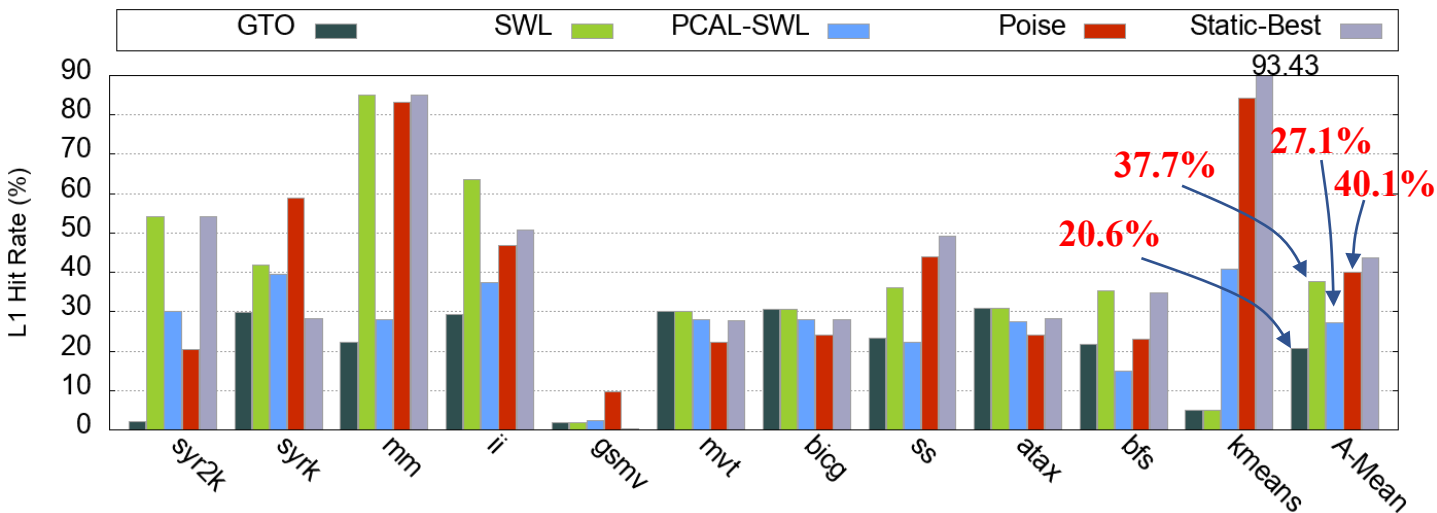
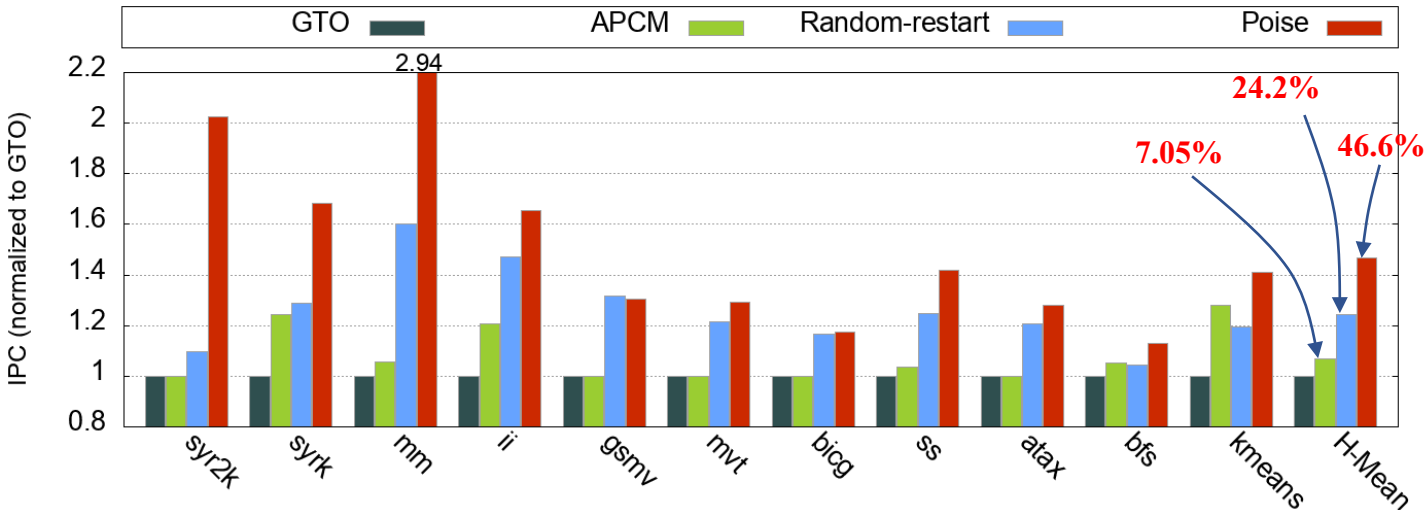*Poise* reduces cache thrashing and reduces pressure on memory system

Results

Average Memory Latency

Poise increases the AML by only 1.1% over GTO

Evaluation Results

Results

Cache Bypassing & Stochastic Search

Results

Energy consumption

79%

51.6%

GTO   Poise

Energy consumption (normalized)

1
0.8
0.6
0.4
0.2
0

syr2k   syrk   mm   ii   gsmv   mvt   bicg   ss   atax   bfs   kmeans   H-Mean
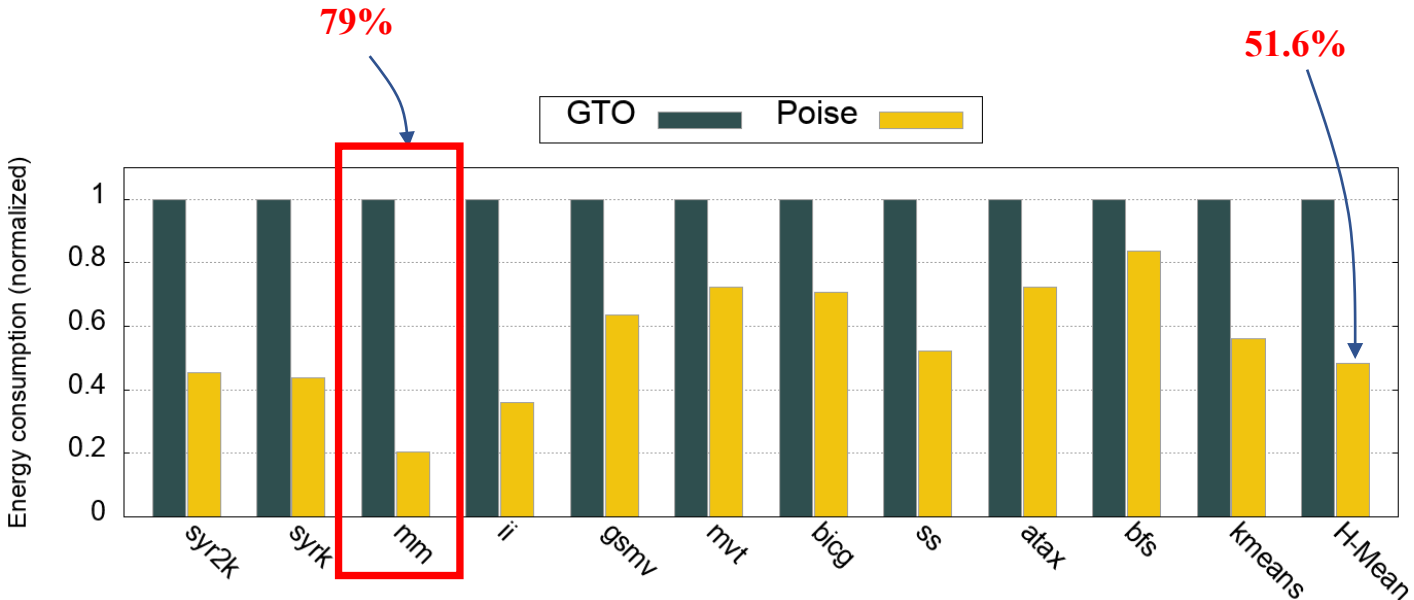
Poise reduces the energy consumption by 51.6% over GTO

Evaluation   Results

Hardware Overhead

- **Arithmetic Units for link function computation**
  - Enough spare cycles in existing FP units
  - Time-multiplexing existing FP units on SM
  - *No extra hardware needed*

- **Feature collection**
  - Seven 32-bit hardware performance counters per SM

- **Finite State Machine**
  - Two 3-bit registers per SM

- **Modified Warp Scheduler**
  - 2-bits per entry in warp scheduler queue

Net storage overhead of **40.75 bytes** per SM

Discussion

- **Why not larger models such as DNNs?**

    - Bulky nature of complex models

    - Generate prohibitively large feature weight matrices with high storage needs

    - High computational demands for training and inference

    - Black box nature of complex models and feature sets

    - Lack of mathematical insights prevents reasoning

Discussion

- ***Poise* – a machine learning based architecture technique**

  - Harness domain knowledge to reduce model size and feature vector

  - Small, yet effective regression model

  - Inference has low computational and storage needs

  - Viable architectural mechanism

  - Demonstrate an effective use of ML to solve an architectural problem

Conclusion

- **Problem**
  - Conflict between TLP and memory system performance
  - Traditional techniques to balance are slow and sub-optimal
  - Goal is to find good warp-tuples expeditiously in hardware

- **Proposal**
  - *Poise* – a machine learning based architectural technique
  - Offline training to learn about good warp scheduling decisions
  - Use prior knowledge to make good runtime predictions

- **Results**
  - Harmonic mean speedup of 46.6% over baseline GTO scheduler
  - Extremely lightweight in terms of hardware overheads
  - Demonstrate an effective use of ML to solve an architectural problem

Poise:
Balancing Thread-Level Parallelism and Memory System Performance in GPUs using Machine Learning

Questions?

Saumay Dublish
saumay.dublish@synopsys.com
http://homepages.inf.ed.ac.uk/s1433370/

SYNOPSYS®

THE UNIVERSITY *of* EDINBURGH