Principal Kernel Analysis

A Tractable Methodology to Simulate Scaled GPU Workloads

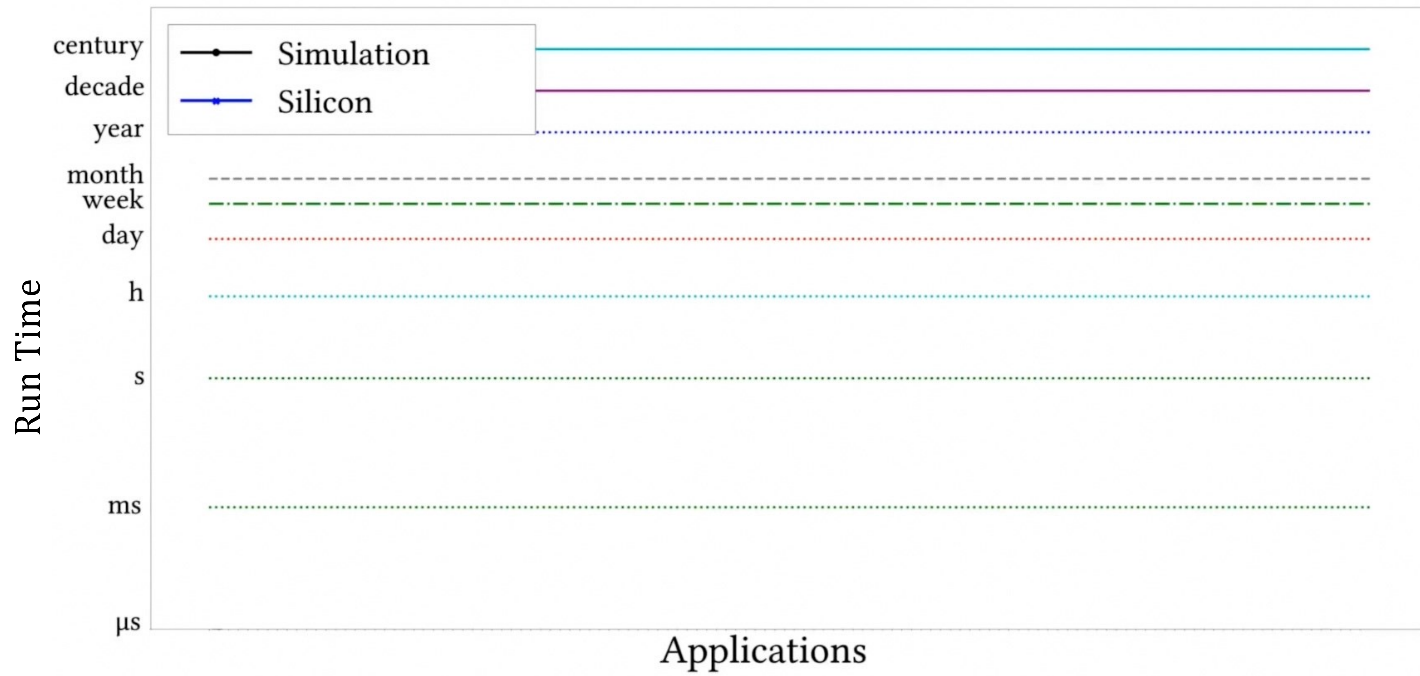
Authors: Cesar A. Baddouh[1], Mahmoud Khairy[1], Roland Green[1], Mathias Payer[2], Timothy G. Rogers[1]

[1] Purdue University

[2] EPFL

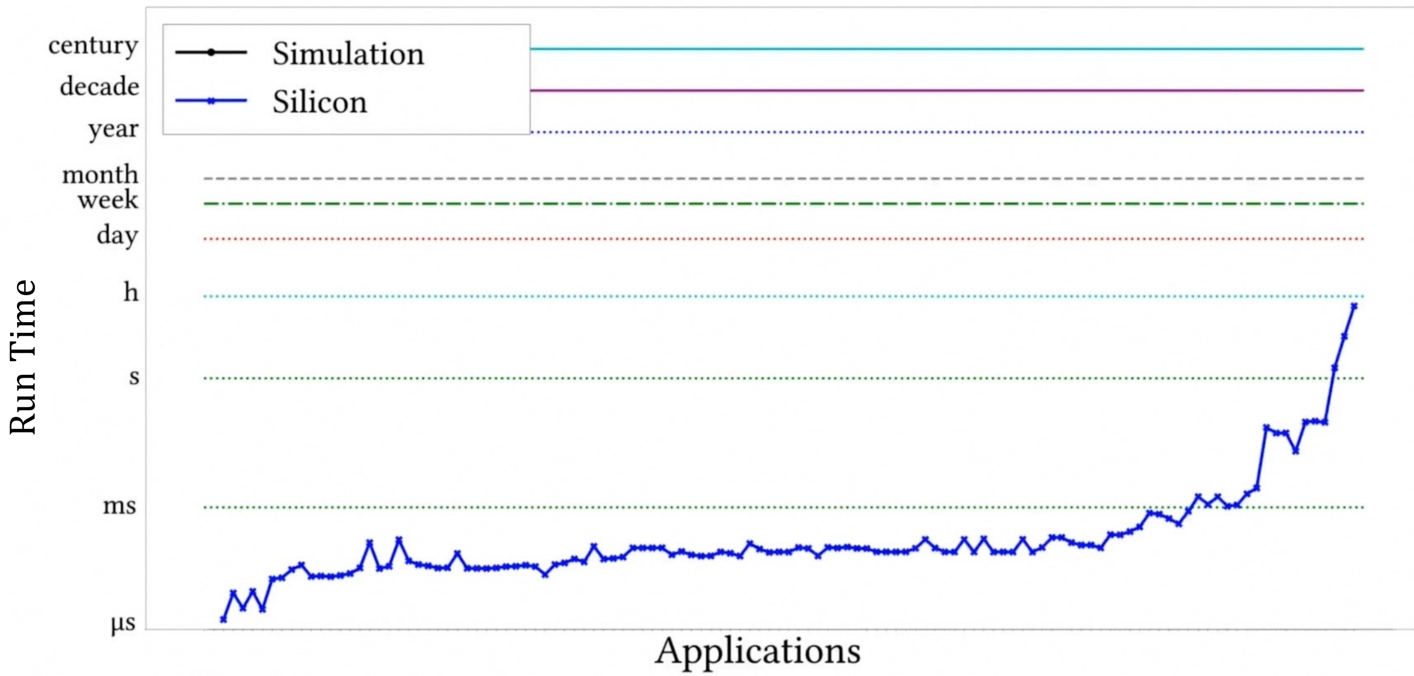
Motivation – Two Facts

1. Cycle-accurate GPU simulation is slow
2. Realistic benchmarks are impossible to fully simulate



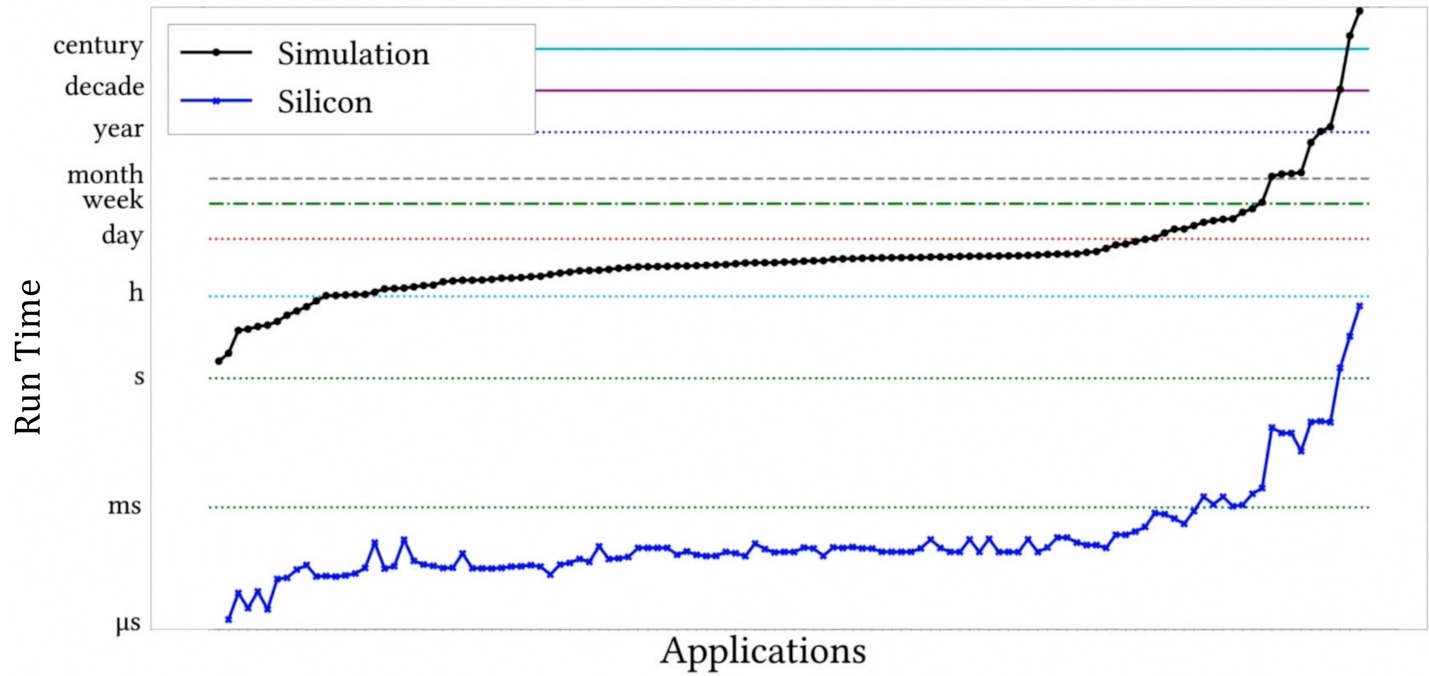
Motivation – Two Facts

1. Cycle-accurate GPU simulation is slow
2. Realistic benchmarks are impossible to fully simulate



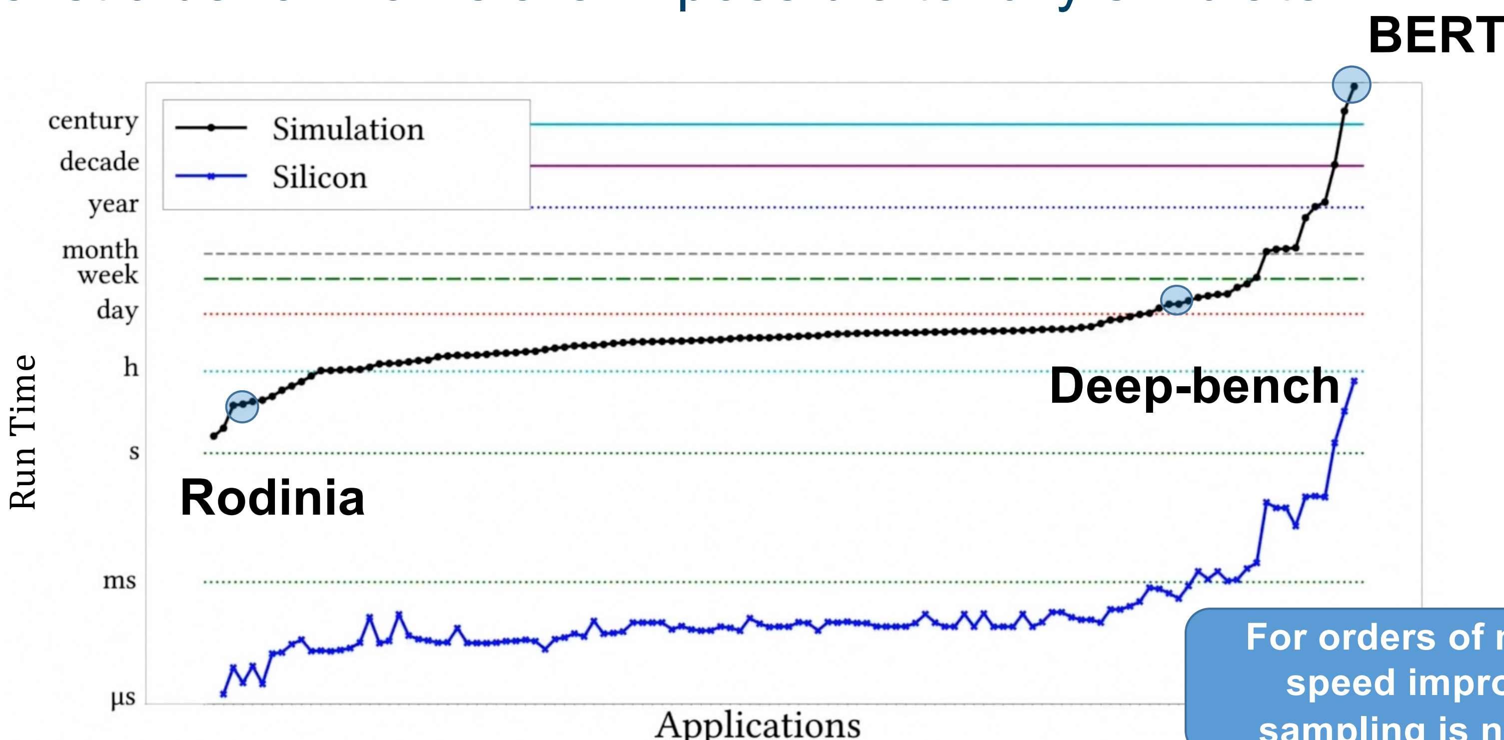
Motivation – Two Facts

1. Cycle-accurate GPU simulation is slow
2. Realistic benchmarks are impossible to fully simulate



Motivation – Two Facts

1. Cycle-accurate GPU simulation is slow
2. Realistic benchmarks are impossible to fully simulate



For orders of magnitude speed improvement sampling is necessary.

Related work

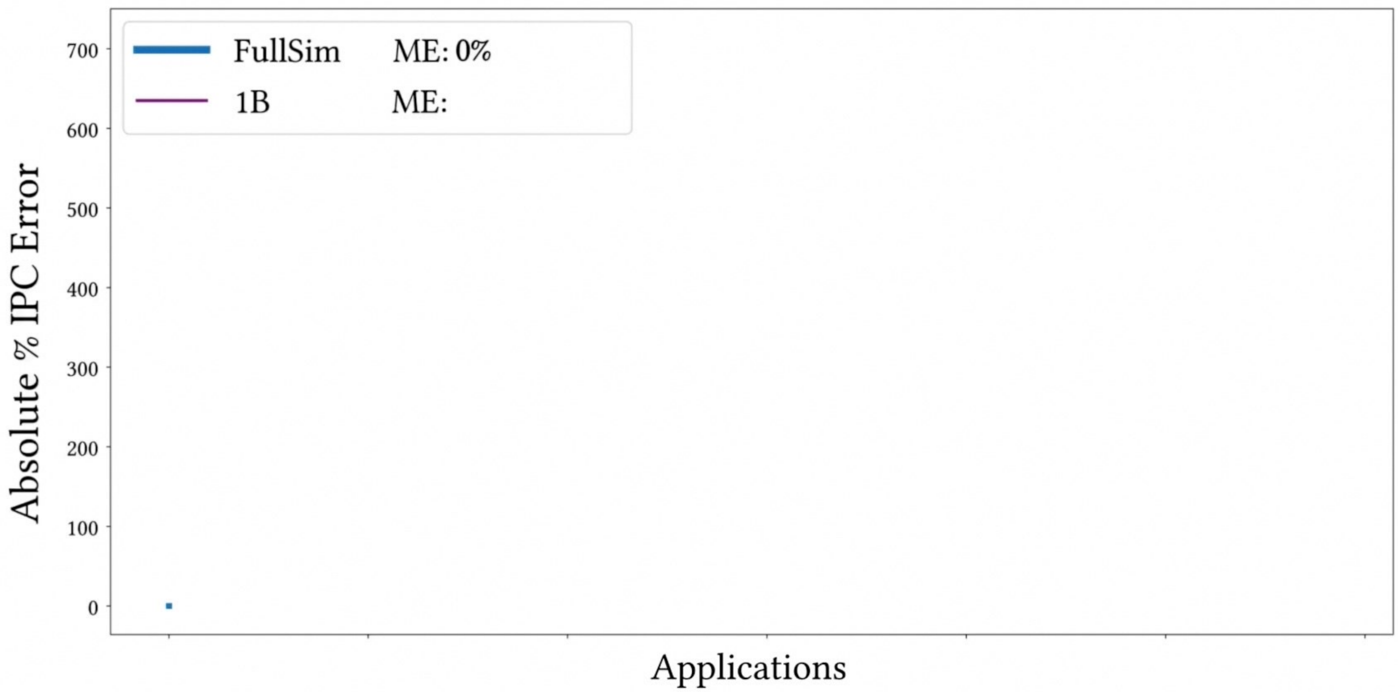
- A quick survey of the related work

Sampling Methodologies	Control-Flow Reduction [24, 45], [54, 64, 66]	Synchronization Regions [13, 22]	GPGPU -MiniBench [67, 68]	GT-Pin[30]	TBPoint [26], Clustering [21]	***Principal Kernel Analysis***
Threaded	Single	CPU Multi-Threaded	GPU Multi-Threaded	GPU Multi-Threaded	GPU Multi-Threaded	GPU Multi-Threaded
Mechanism	Identify common basic blocks	Inter-barrier regions	Intra-thread-block control flow analysis	Unique kernels & control flow analysis	Thread block reduction [26], kernel clustering	Thread block/kernel reduction
Inter-kernel	NA	NA	X	✓	✓	✓
Intra-kernel	NA	NA	✓	X	[26]* Requires full functional simulation	✓
Sampling Clustering	Automated	Automated	Automated	Automated	Hierarchical hand-tuned	Automated
# GPU Workloads	NA	NA	23	25	12	147
Silicon Validated vs Century-Long Full-Simulation	X	X	X	X	X	✓

None of these sampling techniques are the de-facto standard in GPU arch. exploration

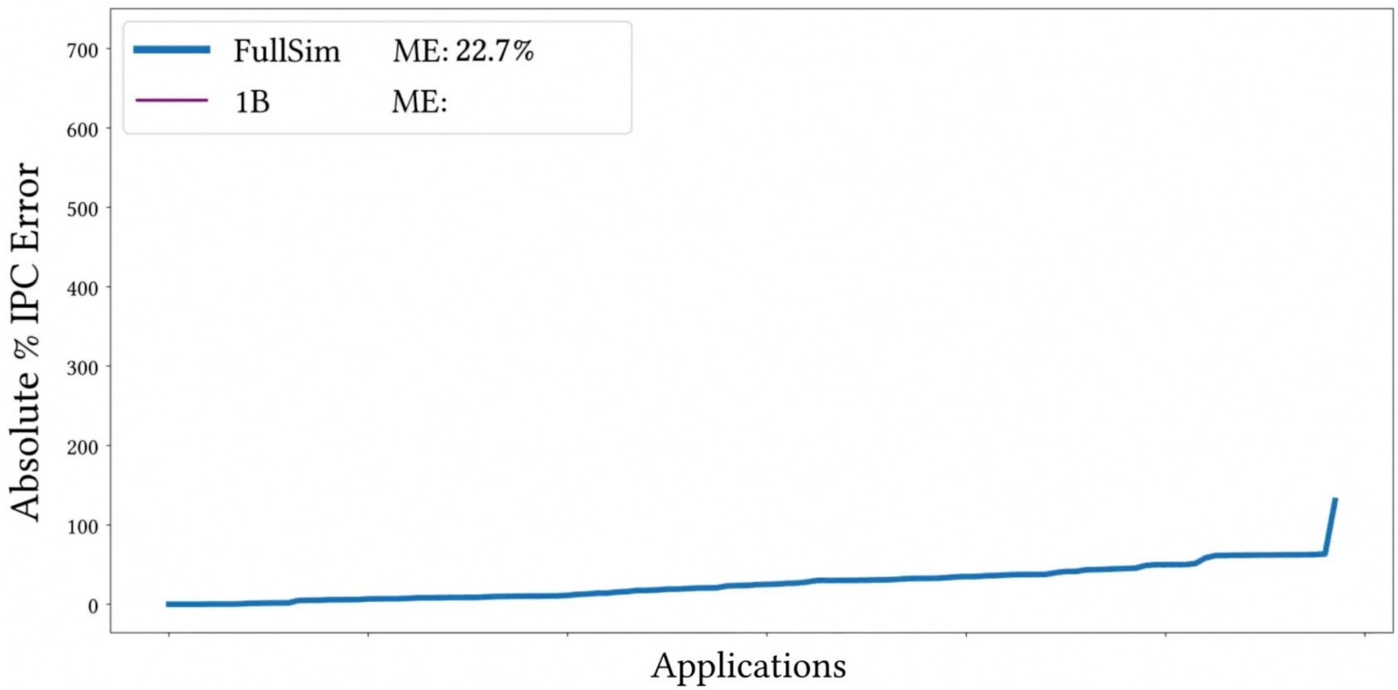
Current Solutions

- Two mainstream options:
  - Full simulation (if tractable)
  - Execute the first N Billion instructions



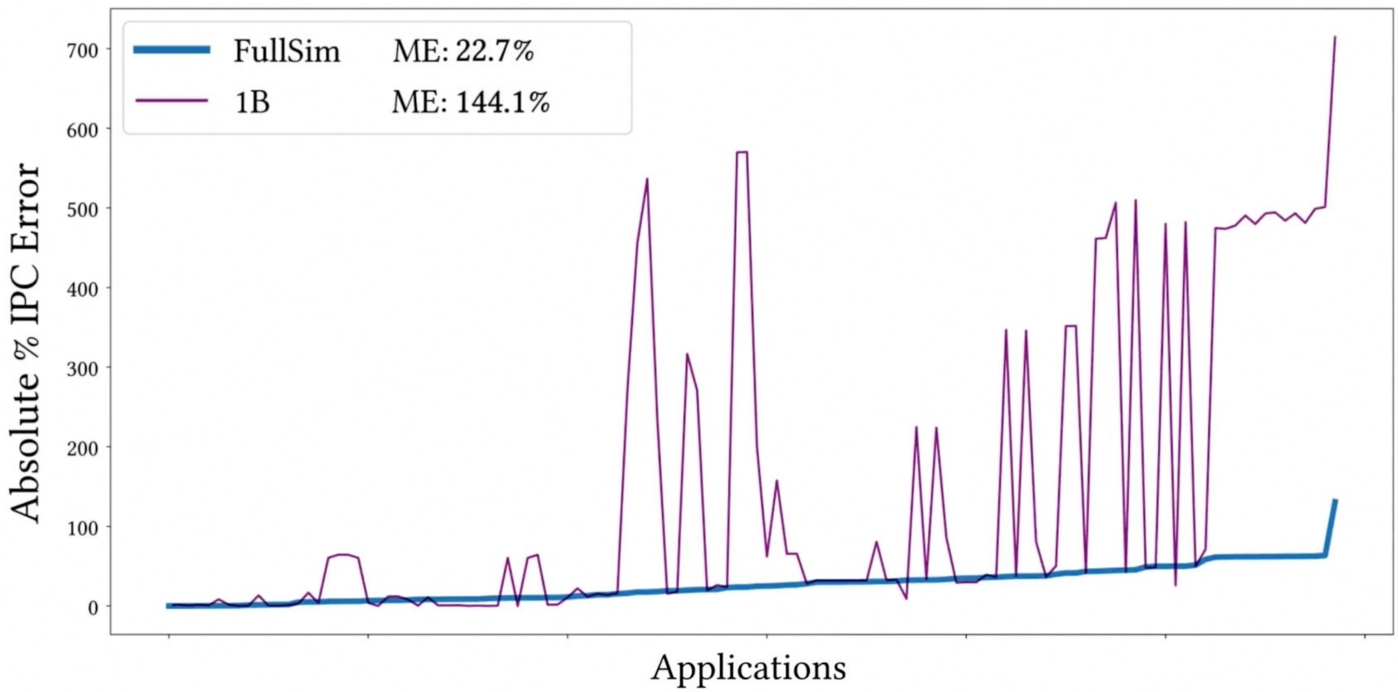
Current Solutions in practice

- Try to simulate the whole workload
  - Not often possible (i.e., no MLPerf)



Current Solutions in practice

- Simulate the first 1-5B instructions
  - Not great



Proposed method

- Take advantage of the GPU/Accelerator programming model
  - Natural synchronization points at kernel boundaries

- Within kernels, code performance is generally more uniform than CPU applications.

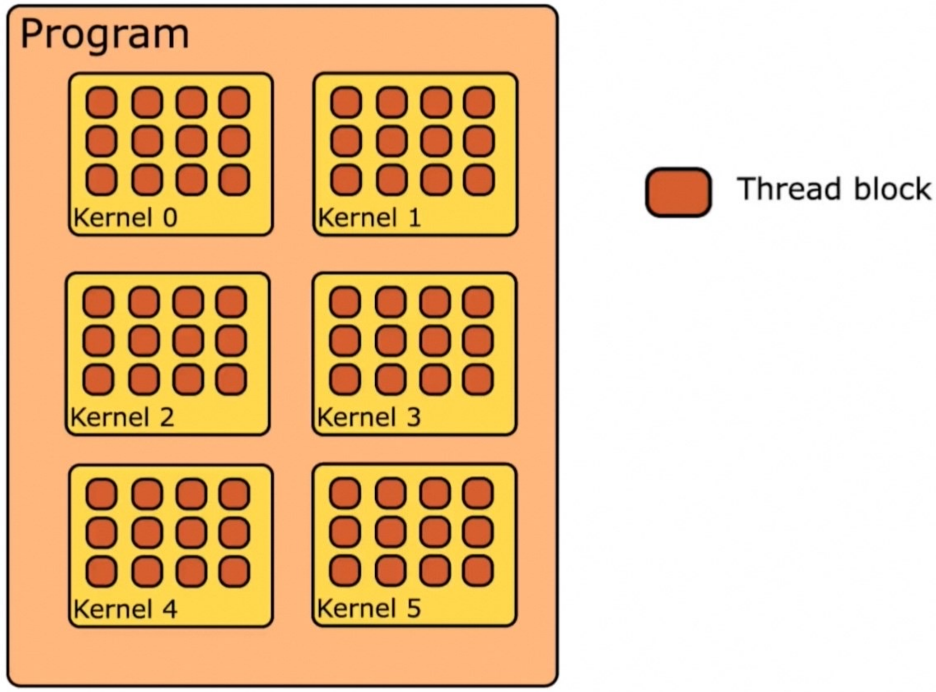
- Can we take advantage of these factors?

- **Key Objectives:**
  - **As hardware agnostic as possible:** want sampling to hold inter-generation
  - **Avoid application-specific characteristics:** want to run everything with zero tuning.

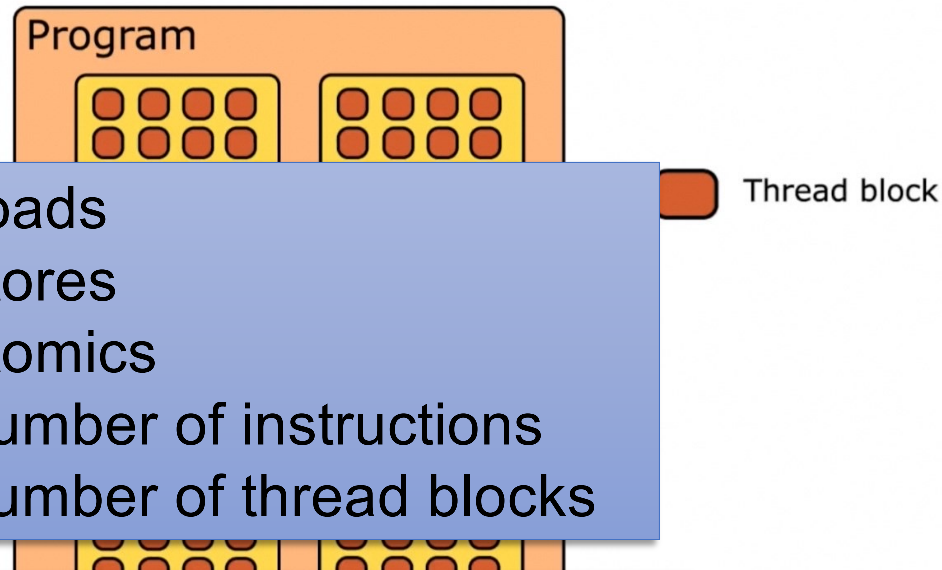
Two key elements

- Select a representative set of kernels
  - Contemporary workloads can run millions of kernels
  - Perform **Principal Kernel Selection (PKS)**
- Within kernels, code performance is generally more uniform than CPU applications
  - Leverage this fact: perform **Principal Kernel Projection (PKP)** of statistics
- Together these form our **Principal Kernel Analysis (PKA)** solution.

Proposed Method - Part 1 - PKS



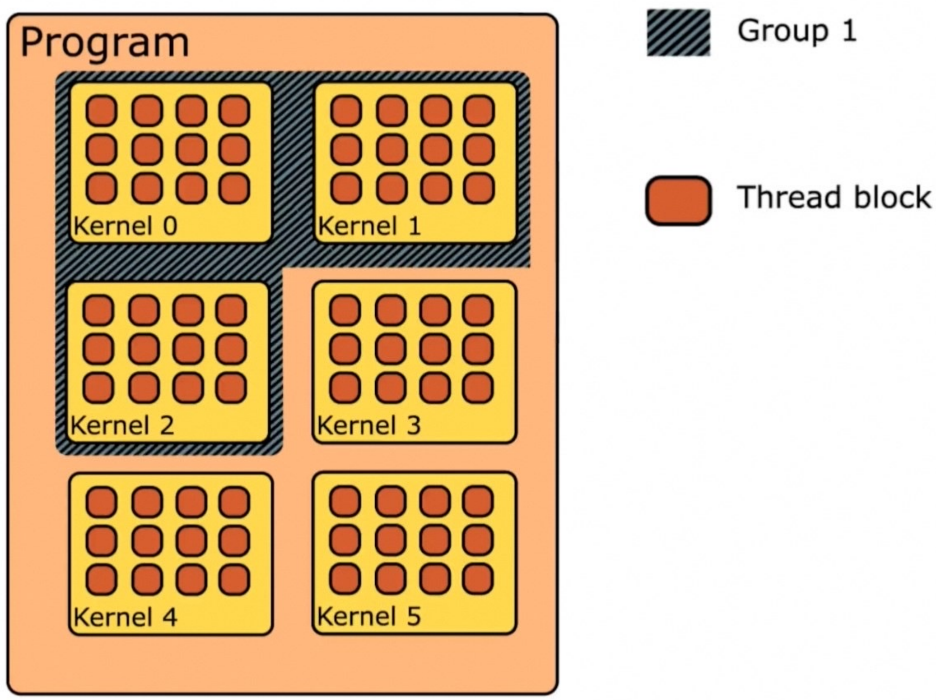
Proposed Method - Part 1 - PKS



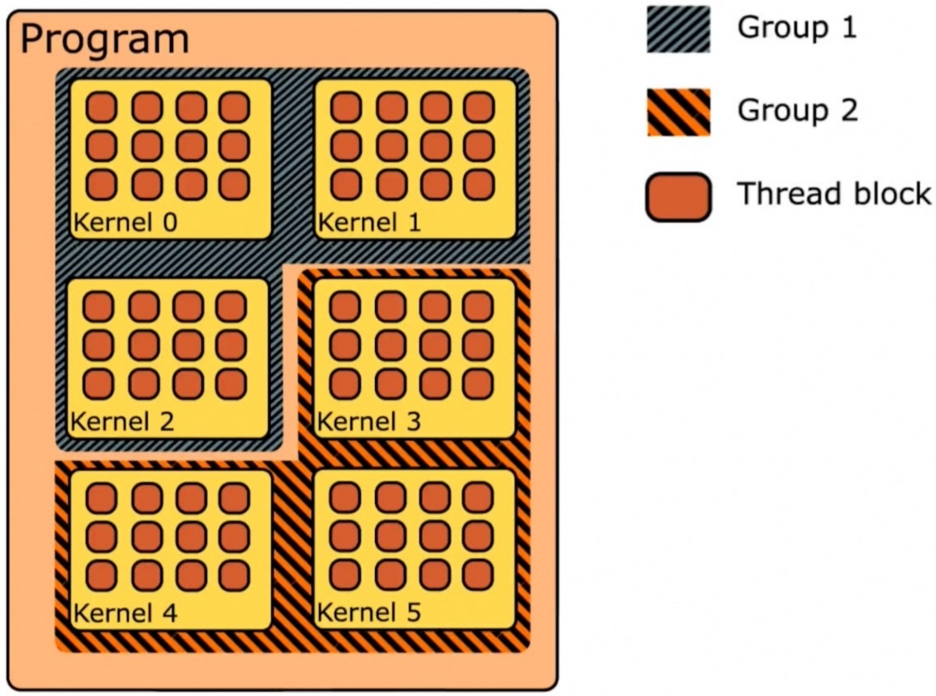
Loads
Stores
Atomics
Number of instructions
Number of thread blocks

Key:
Hardware-agnostic metrics

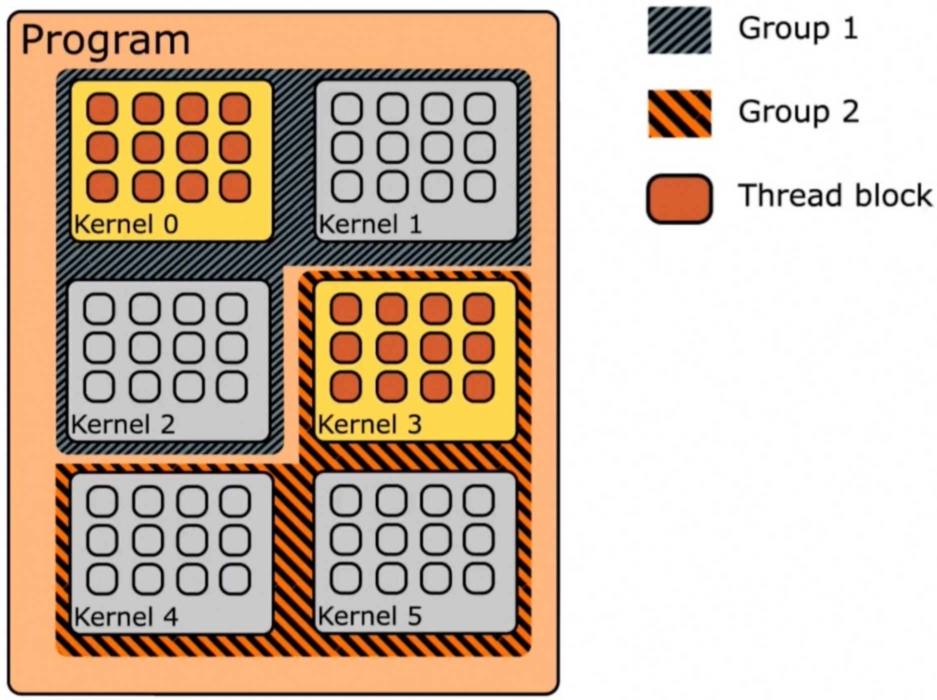
Proposed Method - Part 1 - PKS



Proposed Method - Part 1 - PKS

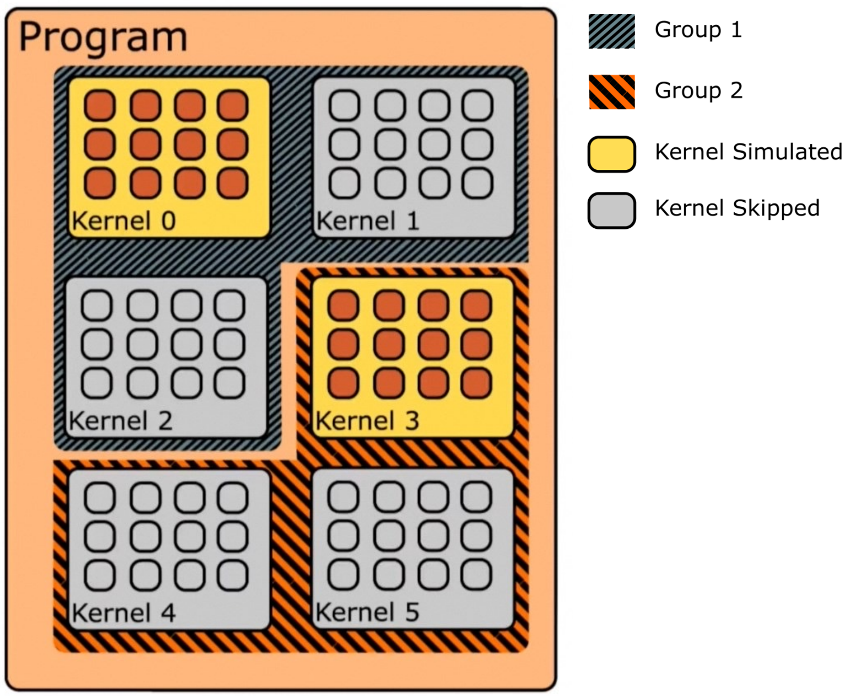


Proposed Method - Part 1 - PKS



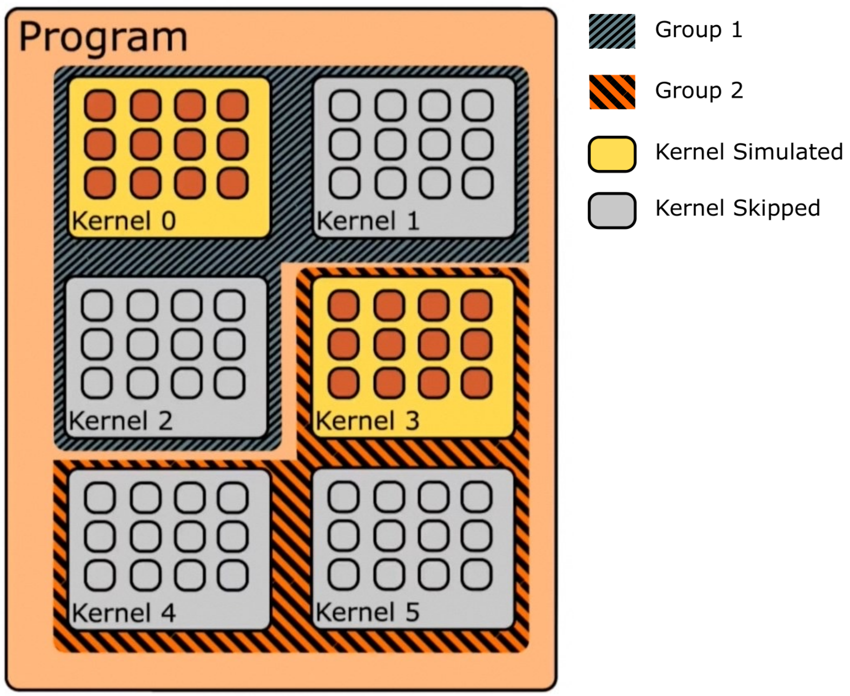
Proposed Method - Part 1 - PKS

- AIM: Reduce the number of simulated kernels. (i.e., inter-kernel reduction)
- We profile the program and obtain hardware-agnostic metrics
- Loads, stores, atomic instructions, etc., etc.



Proposed Method - Part 1 - PKS

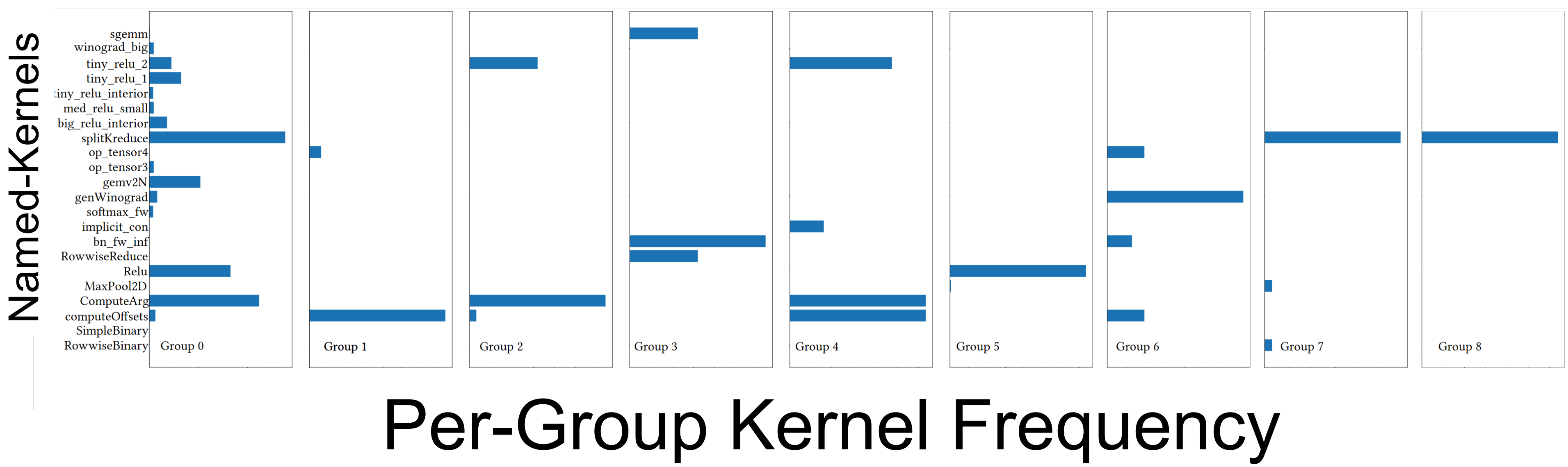
- We use PCA + K-Means to group similar kernels.
  - Technique scales to millions of discrete kernels
- Select one kernel from each group as the principal kernel, skipping all other kernels in a group.
- Project the performance of each group by scaling the performance of the principal kernel by the number of kernels in the group.



Group composition

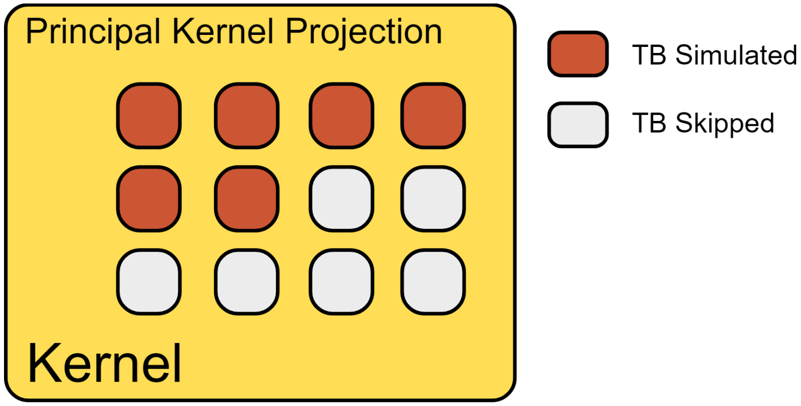
- Group composition not homogeneous
  - Different groups might contain the same named kernels as other groups

Example ResNet

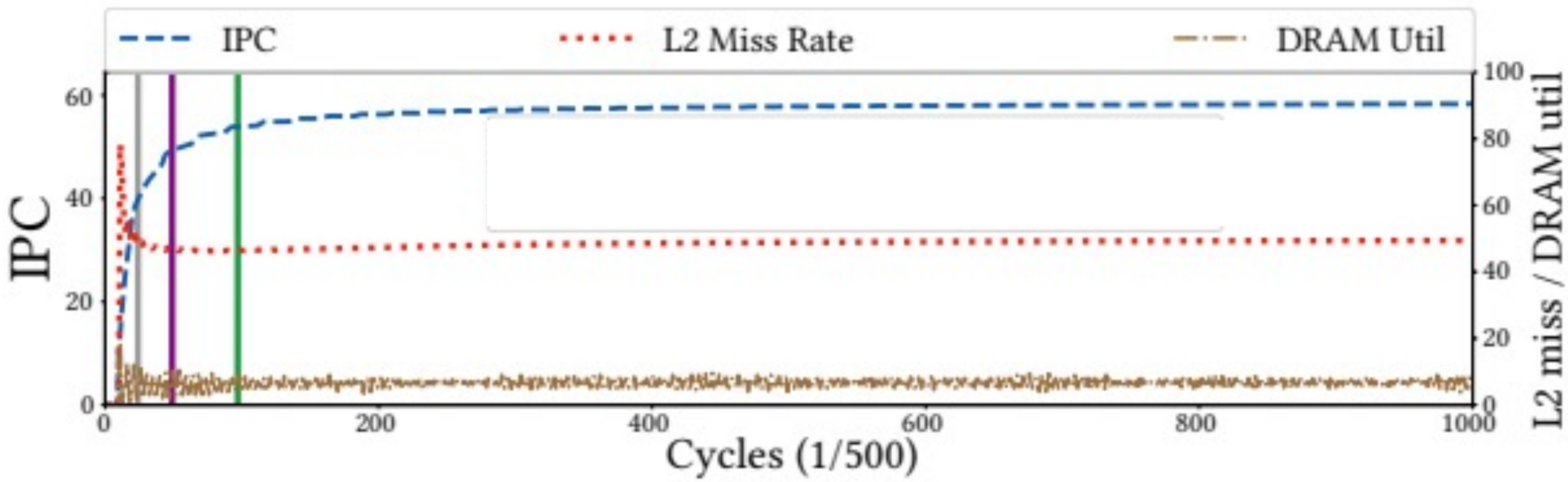


Proposed Method - Part 2 - PKP

- Individual kernels can still be too long.
- AIM: Reduce the execution time of long kernels. (i.e., intra-kernel)
- We observed that for some applications the IPC of a kernel stabilizes, even for workloads that would otherwise seem irregular.



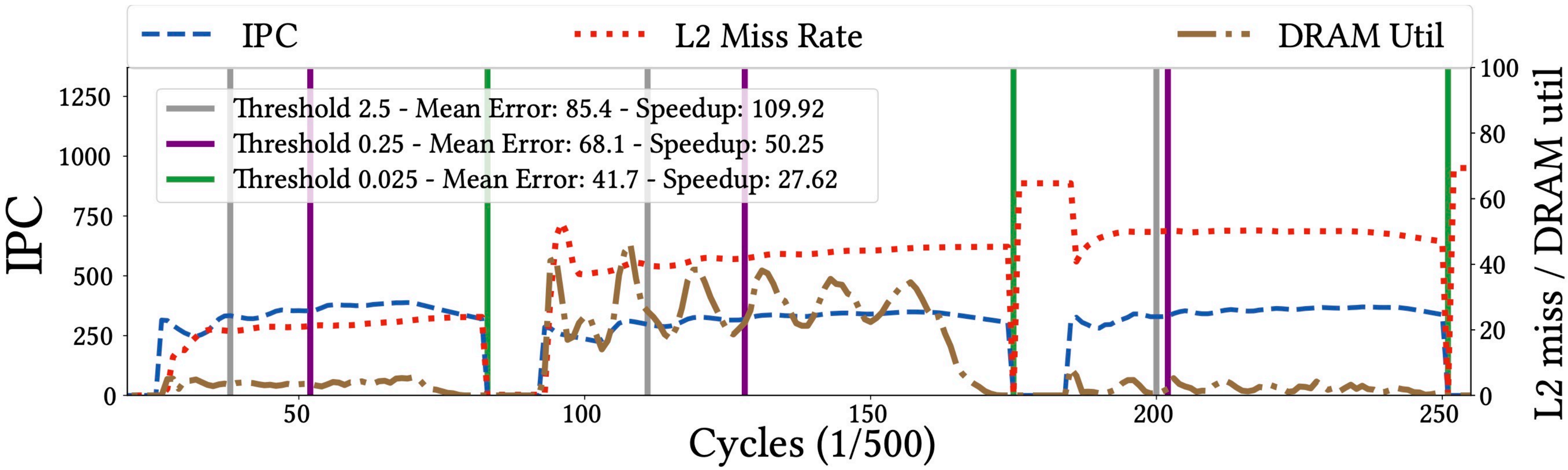
Kernel Stability in Regular Applications



(a) A single kernel from atax: A regular application.

Stability in Irregular Applications

BFS

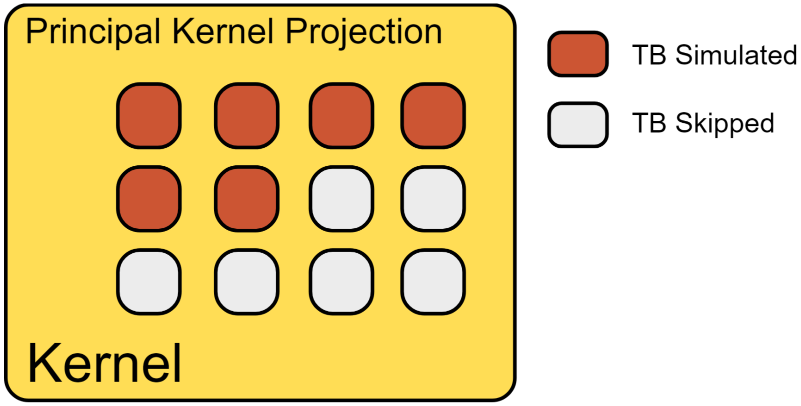


Stability Projection

- If the kernel is stable, we:
    1. Assume:
        - IPC is constant
    2. Know:
        - Number of Instructions remaining
        - Number of Thread blocks remaining

- We can project how long it would take to finish the kernel

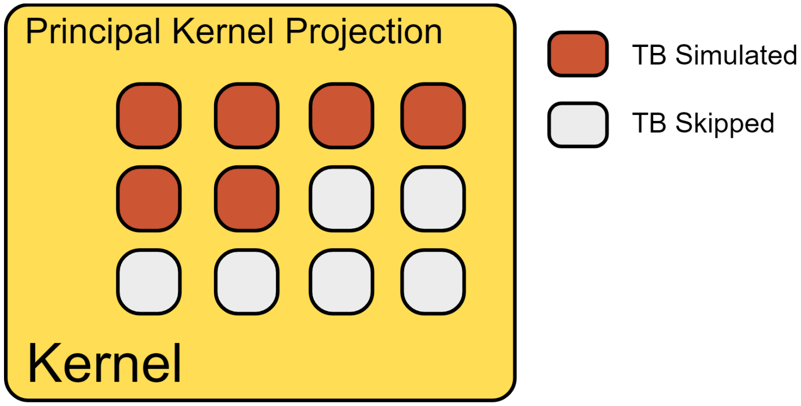
- We skip simulating all thread blocks after we project



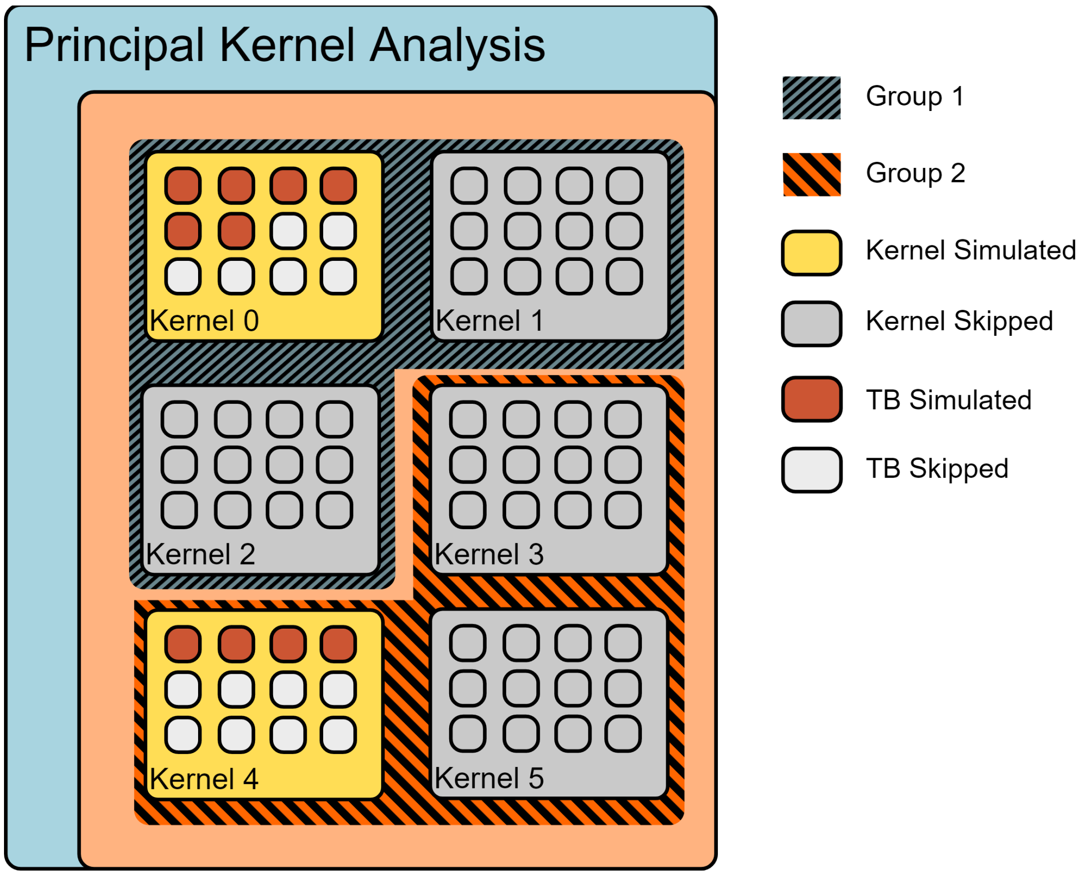
Stability Conditions

1. Stable IPC
   - Coefficient of variance less than some threshold *t*

2. Some thread-blocks have finished
   - Different rules depending on overall number of thread blocks and occupancy.

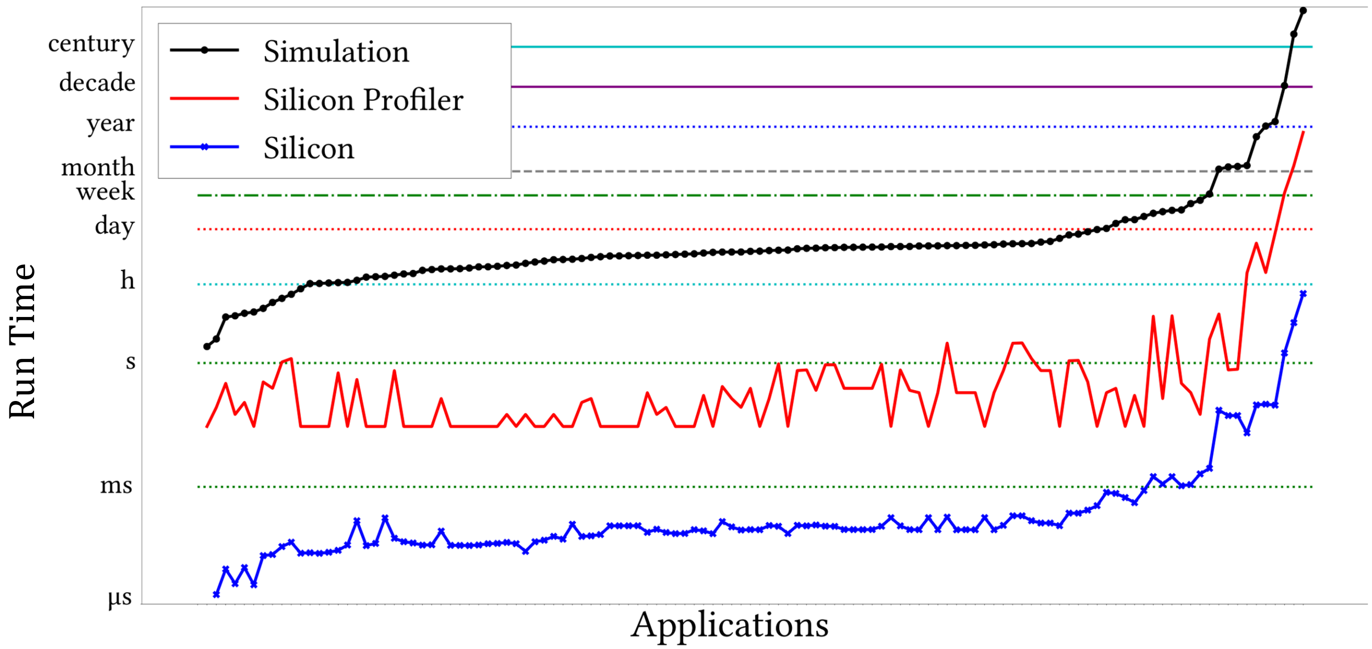


Proposed Joint Method - PKA

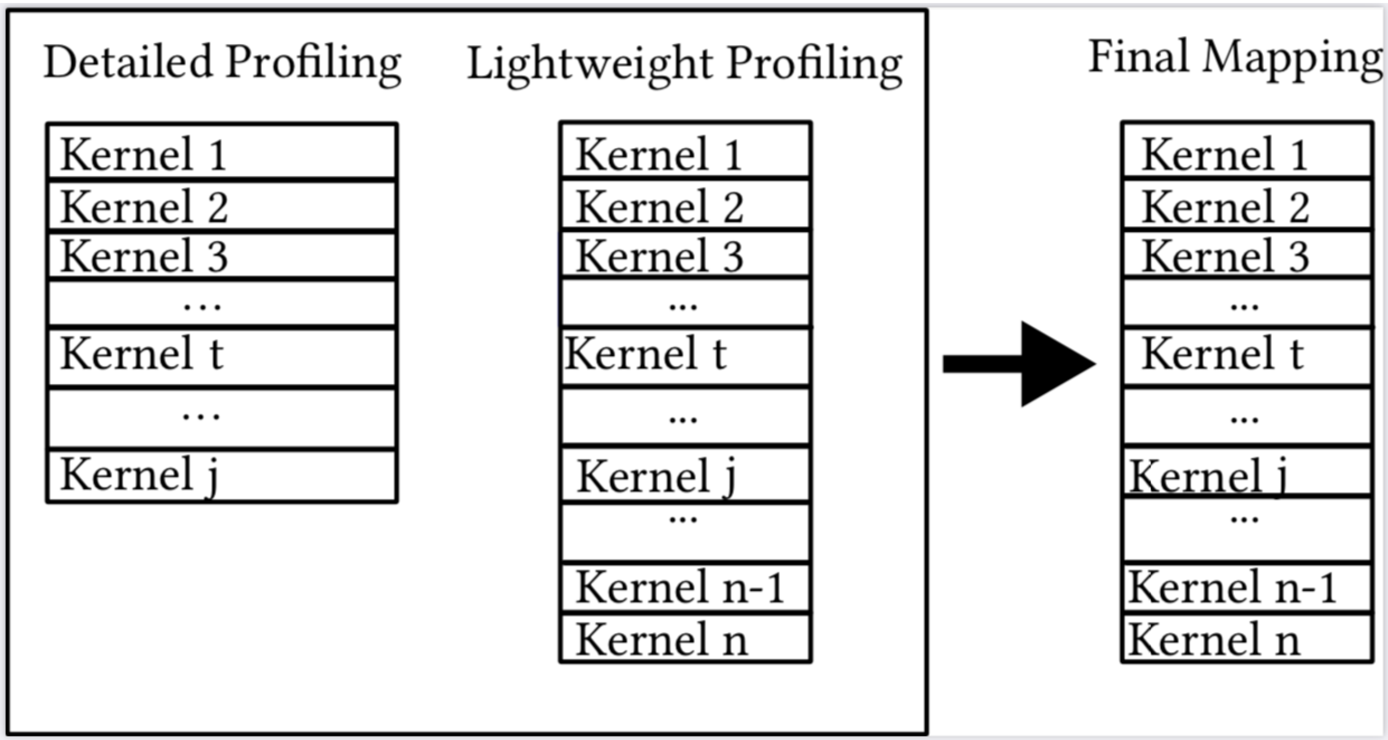


Proposed Method – One more fact

- Detailed profiling has an overhead
  - We calculate it to be $10^3 - 10^4$ slower than silicon.

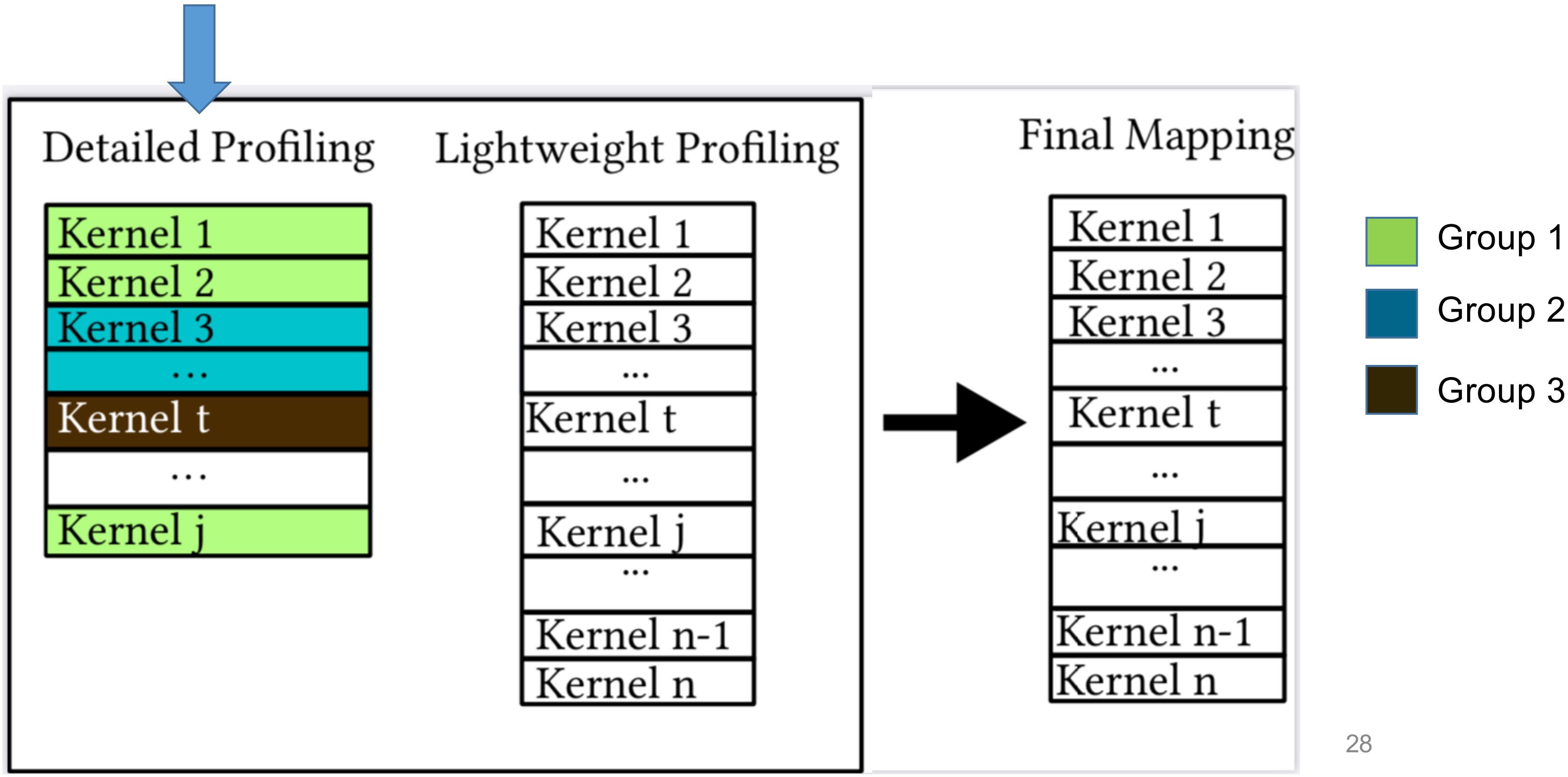


Hierarchical profiling



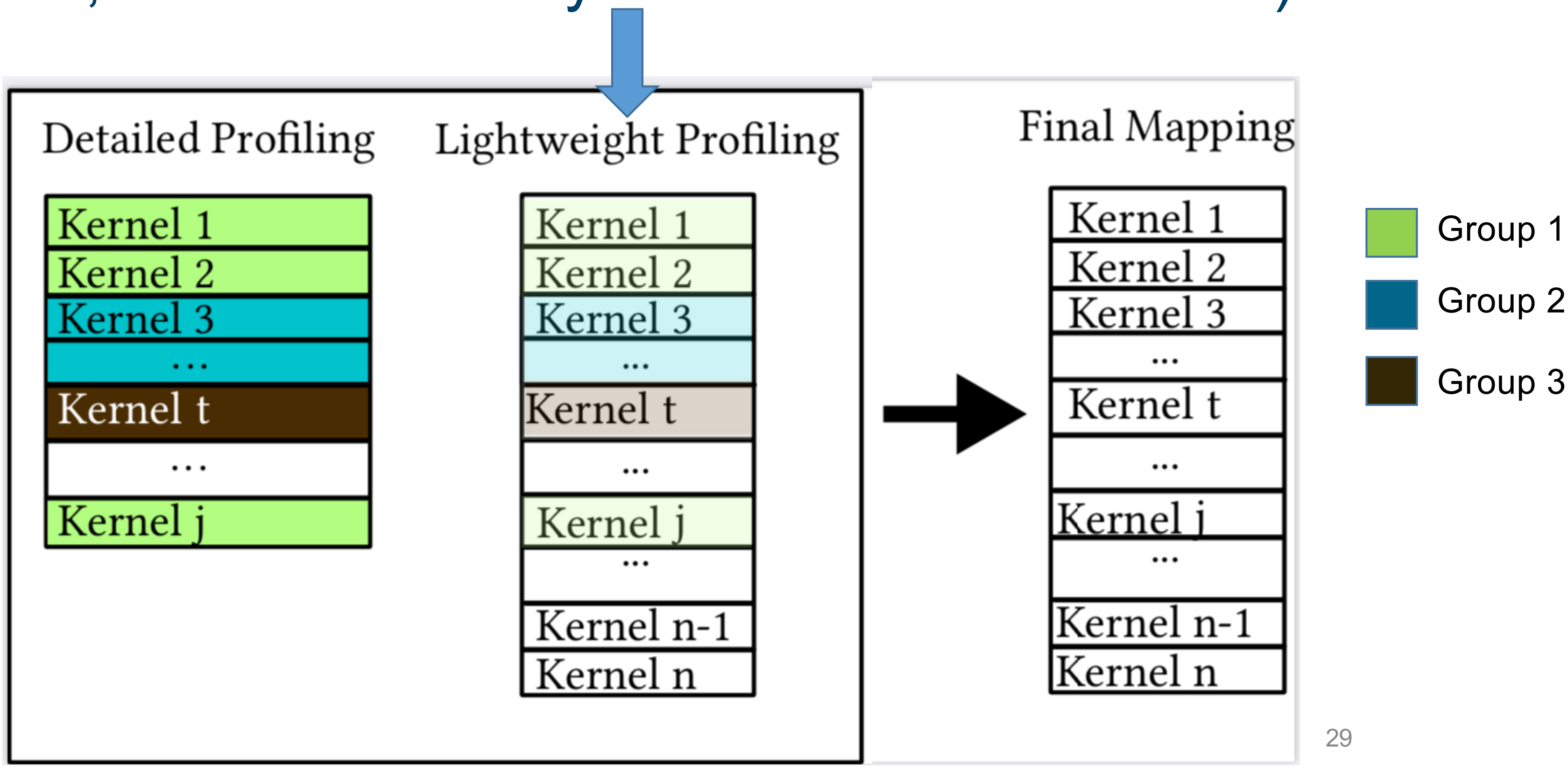
Hierarchical profiling

- Perform detailed profiling on the first **j** kernels where profiling time is practical. Perform PKS on these kernels.



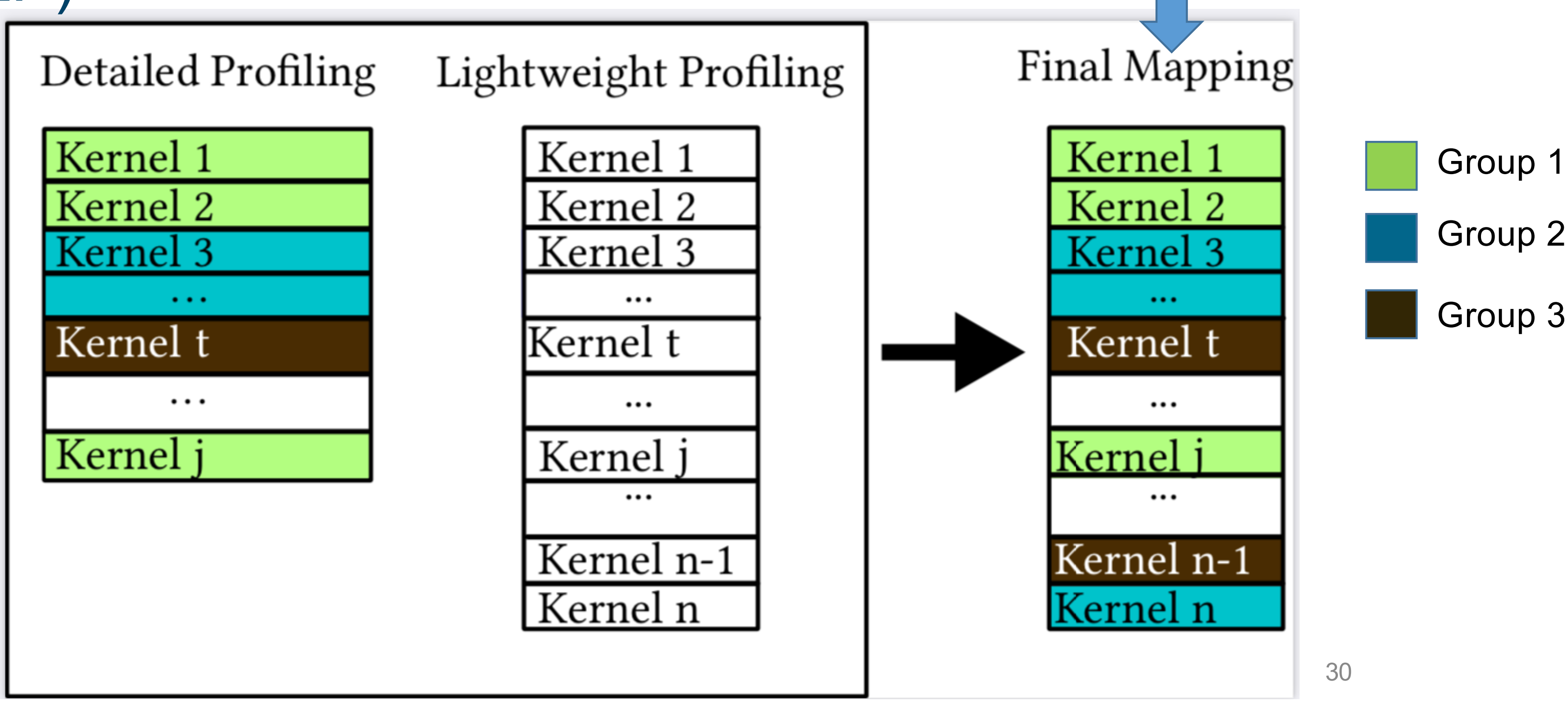
Hierarchical profiling

- Perform lightweight profiling on all the kernels (just get kernel name, dimensions + layer info for ML workloads)



Hierarchical profiling

- We map data from a partial detailed profiling and a complete lightweight profiling using classification algorithms (i.e., SGD, MLP)



Methodology

- Accel-Sim Simulation framework
  - Trace-based simulation framework integrated on GPGPU-sim.
  - Version 1.1
- Run 140+ benchmarks per architecture
  - Rodinia,Polybench,Parboil,Cutlass,Deepbench, MLPerf (SSD, ResNet, BERT, 3D Unet, GNMT)

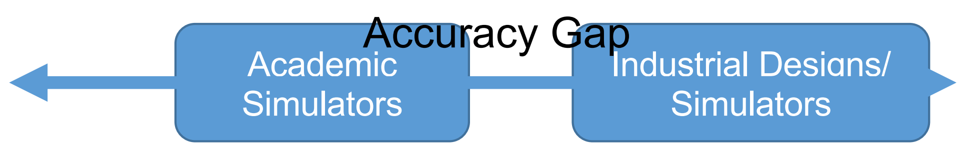
Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling – Khairy et al. - ISCA 2020

Architecture Simulators

- Simulation is commonly used to estimate the effectiveness of a new architectural design idea.

- The simula
  released for open use.

Research cannot look ahead, if its baseline assumptions are too far behind



Incorrect baseline assumptions
→ unrealistic issues or incorrect conclusions

GPU Accelerators are Evolving Rapidly

2009 — 20 — 19 — 2022

Fermi
- mISA sm20
- Caches/Atomics
- Dual warp scheduler

Kepler
- mISA sm30
- DP unit
- Dynamic Parallelism
- Dual issue

Maxwell
- mISA sm50
- Subcore model

Pascal
- mISA sm60
- Unified memory
- HBM
- FP16 support
- Streaming l1

Volta
- mISA sm70
- Scoped synchronization
- Tensor cores & INT unit
- Independent threads SIMT
- Cooperative Groups
- Unified adaptive cache

Turing
- mISA sm75
- New tensor cores
- RT-cores
- UDP cores

Ampere
- mISA sm80
- Sparse tensor cores
- Asynchronous copy and barriers
- HBM2

Hopper
??

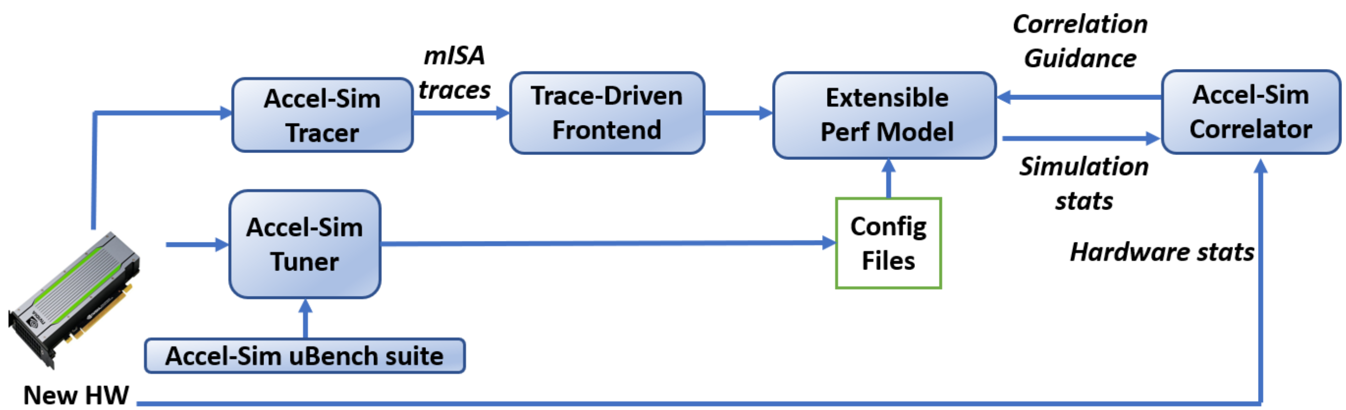
How can academic open-source simulators keep up with industrial designs quickly and accurately?

New machine ISA and architecture designs every 1-2 years!

We show here an example of Nvidia GPU. Similar trend was observed for other GPU vendors.

Accel-Sim *[ISCA'20]*

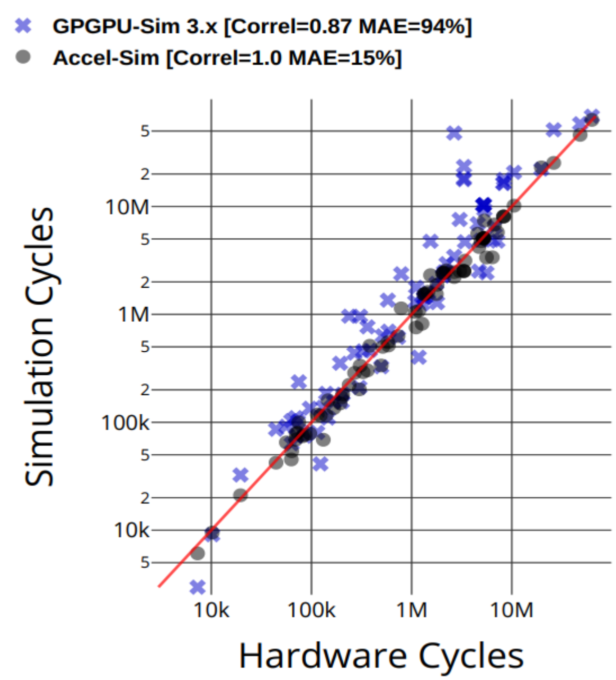
- Accel-Sim introduces a simulation framework to help solve the problem of keeping simulators up-to-date with contemporary designs.



- <u>Key Results:</u> Modeling and validating against five generations of NVIDIA GPUs ranging from Kepler to Ampere with correlation > 0.97 in all instances.

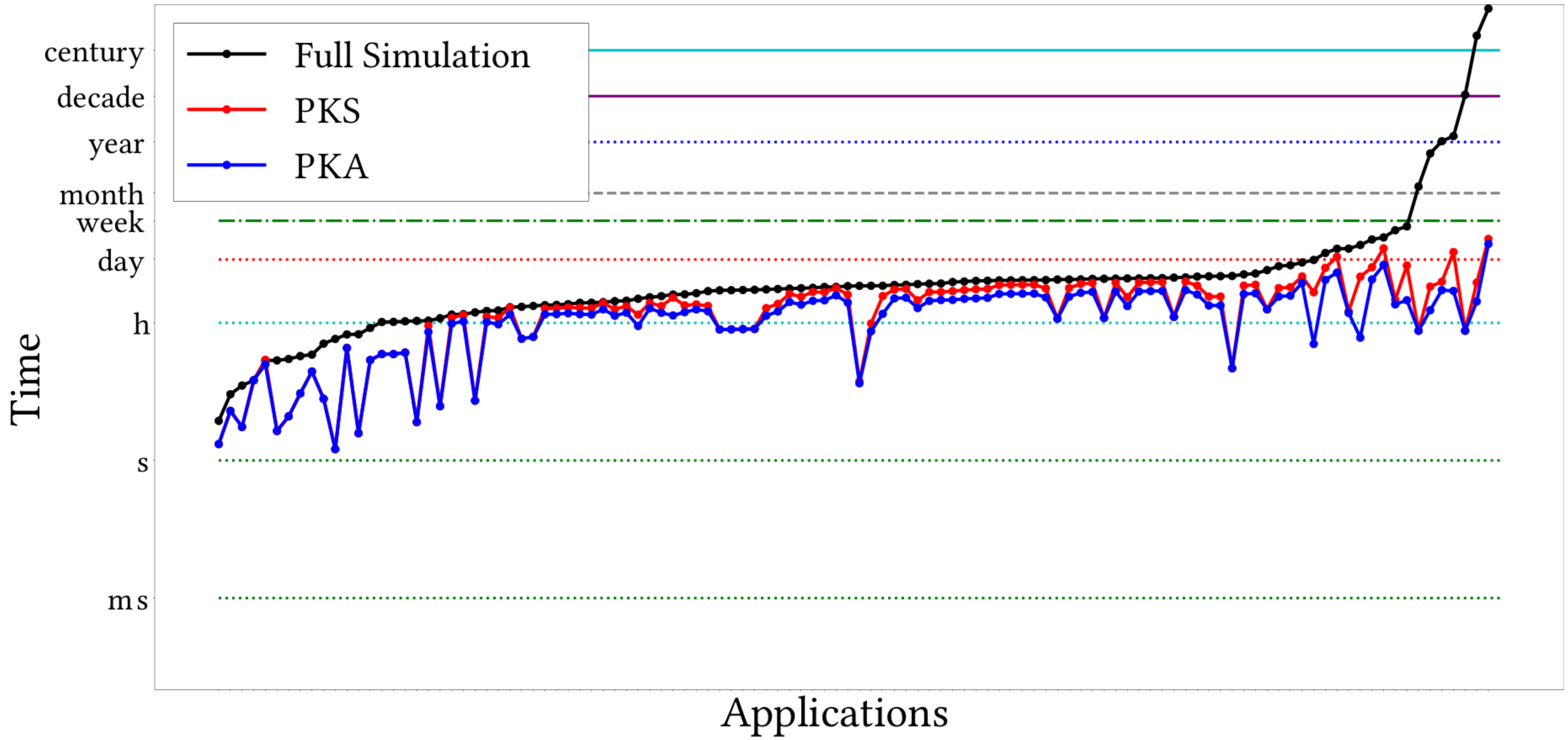
GPGPU-Sim 3.x vs Accel-Sim

- Accel-Sim decreases cycle error from 94% to 15%.

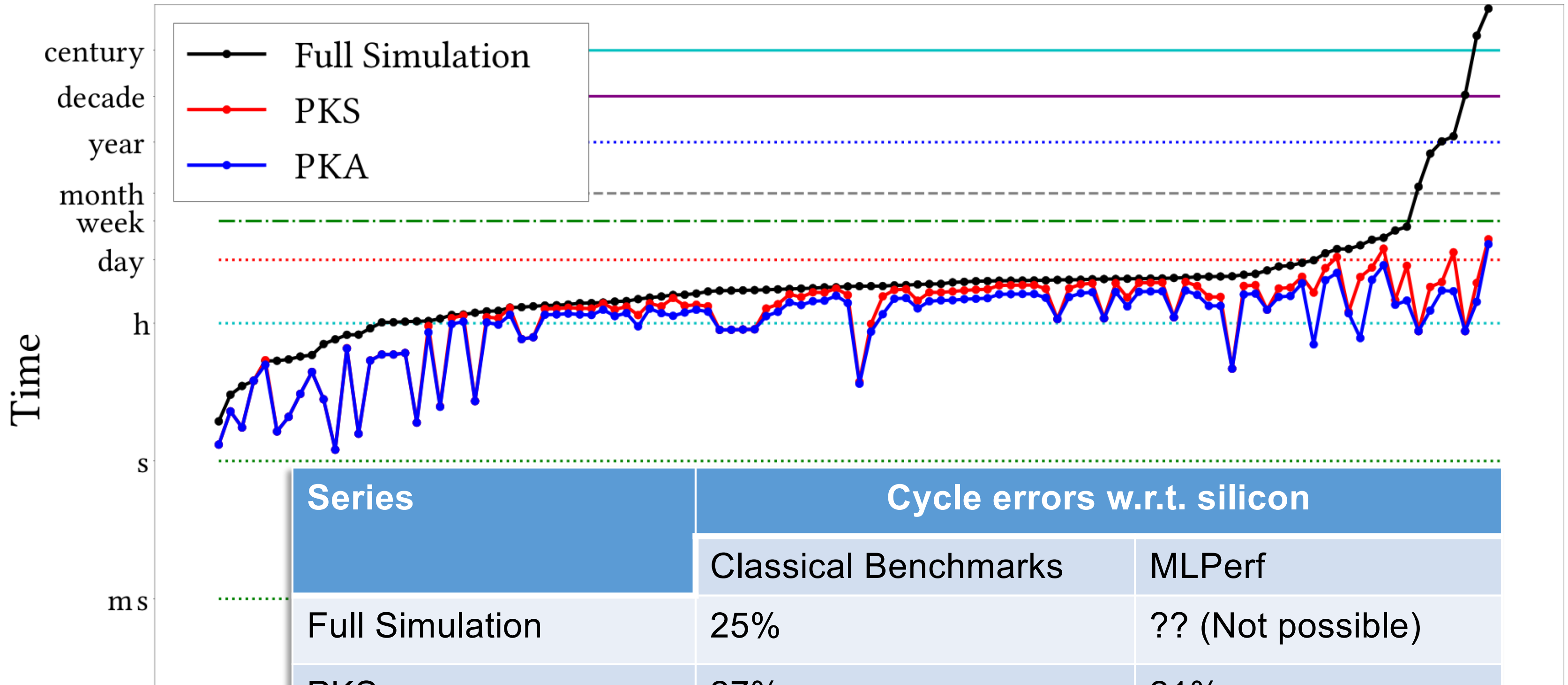


More detailed correlation results can be found in the paper.

PKA Results – Time reduction



Results – Time reduction



Series	Cycle errors w.r.t. silicon	
	Classical Benchmarks	MLPerf
Full Simulation	25%	?? (Not possible)
PKS	27%	21%
PKA	25%	28%

Results

Application	Silicon						Simulation					Metrics	
	Volta		Turing		Ampere		Volta					DRAM Util	
	Error [%]	SU	Error [%]	SU	Error [%]	SU	SimError	PKS Error	PKS SimTime [H] (SU)	PKA Error	PKA SimTime [H] (SU)	Full	PKA
Rodinia Suite													
b+tree	0	1	0	1	0	1	5.8	5.8	0.4 H (1.0)	3.5	0.2 H (1.7)	14.3	14.2
backprop	0	1	0	1	0	1	4.3	4.3	0.1 H (1.0)	4.3	0.1 H (1.0)	35.0	55.0
bfs1MW	5	1.5	0.4	1.2	0.7	1.3	36.7	34.5	1.4 H (1.5)	12.1	1.0 H (1.7)	24.0	30.4
bfs4096	1.6	1.2	2	1.2	1.8	1.2	15.5	23.0	0.1 H (1.2)	23.0	0.1 H (1.2)	0	0
bfs65536	1.9	19.6	35.6	31.1	2.8	19.4	14.2	12.1	0.0 H (21.4)	12.5	0.0 H (22.2)	0	0
dwt2d_192	1.2	3.5	1.4	3.2	6.3	3.3	45.2	48.3	0.0 H (3.5)	48.3	0.0 H (3.5)	0	0
dwt2d_rgb	0.3	2.3	1.2	2	0.1	2	1.6	0.1	0.1 H (2.4)	0.1	0.1 H (2.4)	25.4	41.36
gauss_208	5	435.6	7.8	449	7.2	446.1	56.7	63	0.0 H (429.6)	51	0.0 H (431.1)	0	0
gauss_mat4	1.8	5.9	0.9	5.9	1.1	6.1	77.8	86.8	0.0 H (6.0)	86.8	0.0 H (6.0)	0	0
gauss_s16	2.5	14.9	2.9	14.8	0.1	14.5	73.5	84.5	0.0 H (15.0)	73.5	0.0 H (20.1)	0	0
gauss_s64	0.7	60.1	1.6	61.3	2.4	62	69.8	79.0	0.0 H (63.7)	67.9	0.0 H (74.0)	0	0
gauss_s256	0.4	226.3	8.5	167.9	3.8	232.4	53.4	65.8	0.0 H (248.0)	50.8	0.0 H (258.4)	0	0
hots_1024	0	1	0	1	0	1	3.9	3.1	0.2 H (1.0)	9.1	0.1 H (1.3)	23.5	20.4
hots_512	0	1	0	1	0	1	16.1	16.1	0.0 H (1.0)	16.1	0.0 H (1.0)	0	0
hstort_500k	4.8	4.4	6	4.6	3.9	4.4	45.1	46.5	0.3 H (4.3)	46.5	0.3 H (4.3)	1.0	1.28
hstort_r	4.6	5.6	7.8	6.6	5.9	6	49.5	47.8	2.3 H (5.6)	45.4	2.2 H (5.7)	14.1	34.9
kmeans_28k	1.4	1.6	0	1.3	0	1.6	15.8	16.6	17 M (1.6)	16.6	17 M (1.6)	9.4	6.6
kmeans_819k	0	1.2	0	1.3	0.1	1.4	60.8	38.9	5.1 H (1.1)	3	1.5 H (3)	31.2	32.6
kmeans_oi	0.1	1.2	0	1.3	0.1	1.4	57.6	32.8	3.8 H (1.1)	0.2	1.8 H (2.0)	29.8	32.0
lavaMD	0	1	0	1	0	1	13.2	13.2	8.0 H (1.0)	0.1	6.7 H (1.2)	*	*
lud_i	2	19.5	6.7	13.2	4	16	10.6	15.8	0.0 H (18.2)	11.6	0.0 H (18.7)	0.4	0.0
lud_256	0.4	8.5	0.5	7.8	0.6	8	11.8	15.7	0.0 H (7.6)	11.8	0.0 H (7.2)	0.1	0.0
myocyte	*	*	*	*	*	*	*	*	*	*	*	*	*
nn	0	1	0	1	0	1	38	38	0.0 H (1.0)	38	0.0 H (1.0)	0	0
nw	3.6	88.2	7.7	92.1	2.9	87.5	0.1	1.3	0.0 H (87.1)	2.5	0.0 H (87.6)	0	0
scluster	0.9	128.9	1.9	127.5	1.2	128.5	25.9	30.4	0.0 H (125.5)	30.4	0.0 H (119.5)	*	*
srad_v1	2	98.2	0.9	99.2	0.6	99.5	2	2.3	0.1 H (101.8)	2.3	0.1 H (101.8)	0	0
Parboil Suite													
bfs	4.2	1.1	3.9	1.1	4	1.1	37.8	40.4	0.9 H (1.1)	40.4	0.9 H (1.1)	*	*
cutcp	3.3	4.1	2.9	4	3	4	17.5	19.5	0.9 H (4.0)	19.5	0.9 H (4.0)	*	*
histo	0.4	20.1	0.2	20	0.3	19.9	60.9	57.4	0.2 H (18.4)	57.4	0.2 H (18.4)	14.0	14.5
mri	0.4	3	0.2	3	0.3	3	8.2	8.2	0.2 H (2.9)	8.2	0.2 H (2.9)	0.3	2.1
sad	0	1	0	1	0	1	7.8	7.8	0.3 H (1.0)	7.8	0.3 H (1.0)	10.0	10.0
sgemm	0	1	0	1	0	1	153.9	153.9	2.9 H (1.0)	153.9	2.9 H (1.0)	5.1	5.1
spmv	2.2	48.9	0.8	50.4	0.5	50.3	14.2	12.4	0.1 H (50.9)	12.4	0.1 H (50.9)	*	*
stencil	0	100	1.3	101.3	0.3	99.7	30.1	30.1	0.0 H (1)	30.1	0.0 H (1)	0.1	5
Polybench Suite													
2Dcnn	0	1	0	1	0	1	12	17	1.3 H (1.0)	42	0.2 H (4.6)	53.5	36.0
2mm	0	2	0.1	2	0	2	6.8	1.7	99.7 H (2.0)	15	3.8 H (1.3)	*	*
3dconvolution	4.6	242.9	2.2	259.8	0.4	253	50.3	56.6	0.0 H (243.7)	56.6	0.0 H (249.7)	0	0
3mm	0.4	3	0.1	3	0.5	3	11.4	11.6	1.7 H (3.0)	7.9	1.3 H (4.0)	0.4	0.6
atax	0	1	0	1	0	1	22.4	22.4	2.3 H (1.0)	22.4	2.3 H (1.0)	6.5	6.5
bicg	0	1	0	1	0	1	23	23	2.2 H (1.0)	23	2.2 H (1.0)	6.5	6.5
correlation	0	1	0	1	0	1	42.8	42.8	494.4 H (1.0)	42.8	494.4 H (1.0)	*	*
covariance	0	1	0	1	0	1	43.4	43.4	502.6 H (1.0)	43.4	502.6 H (1.0)	*	*
fdtd2d	1.6	711.1	1.3	722.5	1.6	706.9	6.5	2.6	0.3 H (725.6)	2.6	0.1 H (2725.5)	*	*
gemm	0	1	0	1	0	1	12.8	12.8	1.9 H (1.0)	7.5	1.5 H (1.3)	0.5	0.7
gsummv	0	1	0	1	0	1	0.1	0.1	2.5 H (1.0)	0.1	2.5 H (1.0)	6.7	5.9
gramschmidt	4.9	498.2	6.8	507.1	4.3	494.5	27.8	26.3	1.1 H (500)	26.3	1.1 H (500)	*	*
mvt	0	1	0	1	0	1	22.9	22.9	2.3 H (1.0)	22.9	2.3 H (1.0)	6.5	6.5
syr2k	0	1	0	1	0	1	119	188	50 D (1.0)	11.0	24 H (50)	0.1	0.2
syrk	0	1	0	1	0	1	1.7	1.7	45.2 H (1.0)	17.6	8.2 H (5.5)	*	*
Cutlass Perf Suite SGEMM (10 inputs)													
Mean	0.3	6.0	0.0	6.0	0.0	6.0	1.9	1.9	4.9 H (6.1)	3.7	2.4 H (7.6)	6.1	5.3
Cutlass Perf Suite WGEMM (TensorCore) (10 inputs)													
Mean	0.3	7.0	0.7	7.0	0.1	7.0	44.9	45.0	1.8 H (7.0)	42.7	0.4 H (12.3)	11.0	10.3
Deepbench Suite - Convolution - Inference (5 inputs)													
Mean	0.8	1.5	0.9	1.5	0.6	1.6	13.4	13.5	2.3 H (1.4)	13.6	2.1 H (1.5)	1.2	0.6
Deepbench Suite - Convolution - Training (5 inputs)													
Mean	1.3	2.8	51.3	5.0	0.5	3.6	*	*	*	*	*	1.8	6.1
Deepbench Suite - Convolution - Inference (TensorCore) (5 inputs)													
Mean	0.9	1.5	0.2	1.5	0.2	1.5	11.1	11.9	2.9 H (1.4)	13.0	2.5 H (1.6)	1.8	0.8
Deepbench Suite - Convolution - Training (TensorCore) (5 inputs)													
Mean	2.1	1.9	*	*	*	*	21.6	25.8	14.8 H (1.7)	28.3	12.5 H (2.9)	0.6	2.0
Deepbench Suite - GEMM bench - Inference (5 inputs)													
Mean	2.4	1.1	4.1	1.2	4.2	1.2	10.3	12.4	2.2 H (1.2)	12.4	2.2 H (1.3)	21.1	38.0
Deepbench Suite - GEMM bench - Training (5 inputs)													
Mean	0.9	1.3	0.2	1.6	0.6	1.5	12.6	11.6	3.5 H (1.3)	11.6	3.4 H (1.4)	23.4	29.3
Deepbench Suite - GEMM bench - Inference (TensorCore) (5 inputs)													
Mean	2.4	1.1	4.0	1.2	4.0	1.2	10.4	12.5	3.1 H (1.2)	12.5	3.1 H (1.2)	21.1	38.1

Application	Silicon						Simulation					Metrics	
	Volta		Turing		Ampere		Volta					DRAM Util	
	Error [%]	SU	Error [%]	SU	Error [%]	SU	SimError	PKS Error	PKS SimTime [H] (SU)	PKA Error	PKA SimTime [H] (SU)	Full	PKA
Deepbench Suite - GEMM bench - Train (TensorCore) (5 inputs)													
Mean	0.8	1.3	0.1	1.5	0.8	1.5	12.7	11.8	4.2 H (1.3)	11.8	4.1 H (1.3)	25.2	27.0
Deepbench Suite - RNN bench - Inference (9 inputs)													
Mean	3.3	3.0	5.6	5.3	3.2	4.5	18.7	13.0	6.1 H (1.9)	13.0	6.1 H (1.9)	0.1	6.0
Deepbench Suite - RNN bench - Train (5 inputs)													
Mean	0.5	1.1	1.5	1.2	1.1	1.1	19.4	18.8	6.3 H (1.2)	18.8	6.3 H (1.2)	0.3	5.8
Deepbench Suite - RNN bench - Inference (TensorCore) (10 inputs)													
Mean	3.4	3.2	6.6	5.0	3.6	4.3	18.8	13.3	5.7 H (2.1)	13.3	5.7 H (2.1)	0.1	6.0
Deepbench Suite - RNN bench - Train (TensorCore) (5 inputs)													
Mean	0.6	1.1	1.6	1.2	0.7	1.1	19.6	19.0	6.0 H (1.2)	19.0	6.0 H (1.2)	0.3	5.0
MLPerf Suite													
BERT Offline Inference	12.5	21564	*	*	*	*	*	29.51	0.4 H	29.51	0.4 H (1)	*	*
SSD Training	32.5	13000	*	*	*	*	*	35.9	4.5 H	28	0.5 M (500)	*	*
ResNet 50 64b Inference	3.2	1144	*	*	*	*	*	6.4	10 H	18	1.3 H (17)	*	*
ResNet 50 128b Inference	3.8	851	*	*	*	*	*	3.5	8 H	12	1.5 H (5)	*	*
ResNet 50 256b Inference	0.7	330	*	*	*	*	*	2.2	18 H	24	1.6 H (11)	*	*
GNMT Training	16.2	9630	*	*	*	*	*	17.0	36 H	39	25 H (1.4)	*	*
3D-Unet Inference	2.8	141	*	*	*	*	*	49.3	0.1 H	49.3	0.1 H (1)	*	*

Results

MLPerf suite

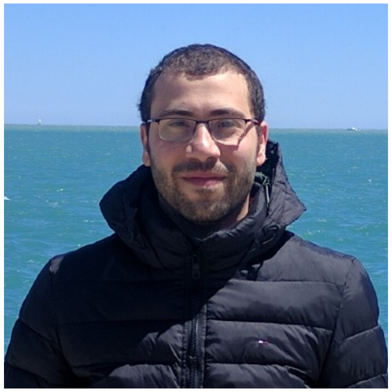
Application	Silicon						Simulation					Metrics	
	Volta		Turing		Ampere		Volta					DRAM Util	
	Error [%]	SU	Error [%]	SU	Error [%]	SU	SimError	PKS Error	PKS SimTime [H] (SU)	PKA Error	PKA SimTime [H] (SU)	Full	PKA
MLPerf Suite													
BERT Offline Inference	12.5	21564	*	*	*	*	*	29.51	0.4 H	29.51	0.4 H (1)	*	*
SSD Training	32.5	13000	*	*	*	*	*	35.9	4.5 H	28	0.5 M (500)	*	*
ResNet 50 64b Inference	3.2	1144	*	*	*	*	*	6.4	10 H	18	1.3 H (17)	*	*
ResNet 50 128b Inference	3.8	851	*	*	*	*	*	3.5	8 H	12	1.5 H (5)	*	*
ResNet 50 256b Inference	0.7	330	*	*	*	*	*	2.2	18 H	24	1.6 H (11)	*	*
GNMT Training	16.2	9630	*	*	*	*	*	17.0	36 H	39	25 H (1.4)	*	*
3D-Unet Inference	2.8	141	*	*	*	*	*	49.3	0.1 H	49.3	0.1 H (1)	*	*

PKA vs. Single iteration in ML Workloads

- Another increasingly popular option is to run a single iteration and scaling that by the number of iterations in the entire program
- Fast and accurate, but still orders of magnitude slower than PKA
- More involved process, must mark where an iteration starts, etc., etc. PKA is completely automatic, doesn't require context.
- Even more manual with sequence/input-dependent workloads like BERT.
  - Could use seqPoints for some SQNN's, point still stands, more involved, less automatic.

Thanks to the students

Cesar A. Baddouh



Mahmoud Khairy

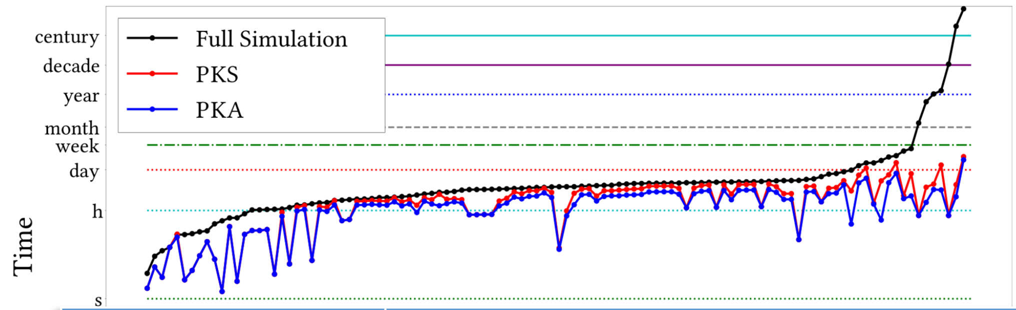
Roland Green

Principal Kernel Analysis

Questions?

- Key idea: Summarize a GPU program by **grouping** kernels together, simulating a **principal kernel** per group and **scaling** the performance. If the IPC is **stable**, **skip** remaining thread blocks and **project** the number of **cycles remaining**.

- Results:
  - Enable simulation of long running programs via an **automatic** process
  - **Centuries** long simulations now achievable in **hours**, at an acceptable error within **5%** of the full simulation
  - Validation of hardware-**invariance** across three GPU generations



Series	Cycle errors w.r.t. silicon	
	Classical Benchmarks	MLPerf
Full Simulation	25%	?? (Not possible)
PKS	27%	21%
PKA	25%	28%

- Artifact available