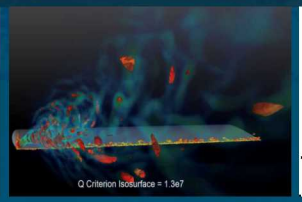
SAND2019-13453C

GPGPU-Sim Overview



PRESENTED BY

Roland Green, Purdue University

11/03/2019

SST Development Team, Sandia National Laboratories

Architecture Accelerator Lab, Purdue University

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Welcome!

Part 1: Introduction to SST *8:45 - 10:15*

- SST Overview
  - *Demos: Running a simple simulation; Enabling statistics; Running in parallel*
- SST Element Libraries: A tour

Break *10:15 - 10:45*

- SST Element Libraries: A tour
  - *Demos: L2 Cache; Backing store; Adding processors; Adding Memory; Node modeling*

Lunch *12:00 - 1:30*

Part 2: GPGPU-Sim ***1:30 – 2:30***

- **GPGPU-Sim Overview & New Features**

Part 3: The SST/GPGPU-Sim Integration *2:30 - 3:00*

Break *3:00 - 3:30*

GPGPU-Sim Exercises *3:30 - 5:00*

Outline

GPGPU-Sim Introduction
- GPU and programming model
- Functional model
- Performance model
- GPUWattch: power model

New Features in GPGPU-Sim
- Volta model
- Run closed source libraries
- Tensor Core
- Run CUTLASS library

Acknowledgements

Some slides are credited to GPGPU-Sim Micro Tutorial 2012

Some data and figures are credited to papers below:

- Akshay Jain, Mahmoud Khairy, Timothy G. Rogers, A Quantitative Evaluation of Contemporary GPU Simulation Methodology. SIGMETRICS 2018
- Mahmoud Khairy, Jain Akshay, Tor Aamodt, Timothy G Rogers, Exploring Modern GPU Memory System Design Challenges through Accurate Modeling, arXiv:1810.07269, (and ISPASS 2019 poster)
- Jonathan Lew, Deval Shah, Suchita Pati, Shaylin Cattell, Mengchi Zhang, Amruth Sandhupatla, Christopher Ng, Negar Goli, Matthew D. Sinclair, Timothy G. Rogers, Tor M. Aamodt, Analyzing Machine Learning Workloads Using a Detailed GPU Simulator, arXiv:1811.08933, (and ISPASS 2019 poster)
- Md Aamir Raihan, Negar Goli, Tor Aamodt, Modeling Deep Learning Accelerator Enabled GPUs, ISPASS 2019

We thank all contributors to GPGPU-Sim

All new features are based on the dev branch of GPGPU-Sim: https://github.com/gpgpu-sim/gpgpu-sim_distribution/tree/dev

Outline

GPGPU-Sim Introduction

- GPU and programming model
- Functional model
- Performance model
- GPUWattch: power model

New Features in GPGPU-Sim

- Volta model
- Run closed source libraries
- Tensor Core
- Run CUTLASS library

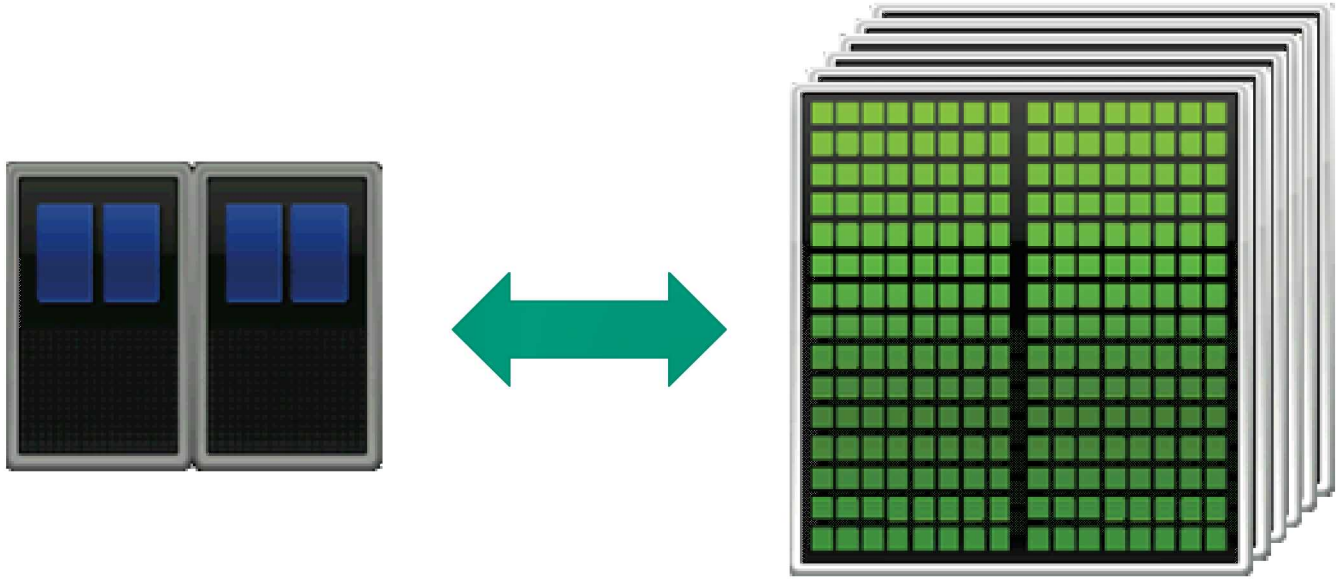
GPU Introduction

GPU = Graphics Processing Unit

- Optimized for Highly Parallel Workloads
- Highly Programmable
- Heterogeneous computing

NVidia Tesla GV100: 80 Stream Multiprocessors(SMs)

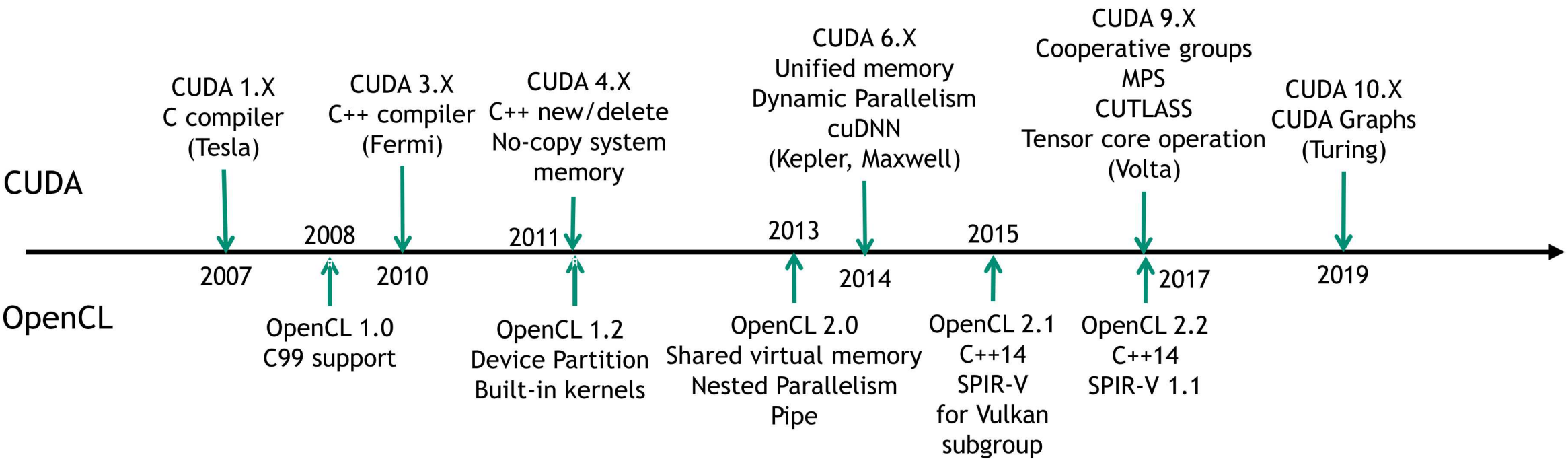
- Each SM has 64 INT32, 64 FP32, 32 FP64, 8 Tensor core



GPGPU Programming model evolution

GPGPU programming model:

- CUDA and OpenCL
- Support more features with newer architectures



GPU Microarchitecture[1]

NVidia GV100:

- Hierarchical compute unit: SM=>TPC=>GPC=>GPU
- Multi-level memory: L1/shmem=>L2=>HBM
- GPGPU-Sim simulator models components above

[1]Volta Whitepaper. https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

GPGPU-Sim Introduction

GPGPU-Sim

- Widely used GPU simulator in the research community (1300+ citations).
- The third most cited simulator in computer architecture field (after GEM5(+GEMS) and SimpleScalar)

Functional model

- Virtual ISA (vISA)
- Machine ISA (mISA)

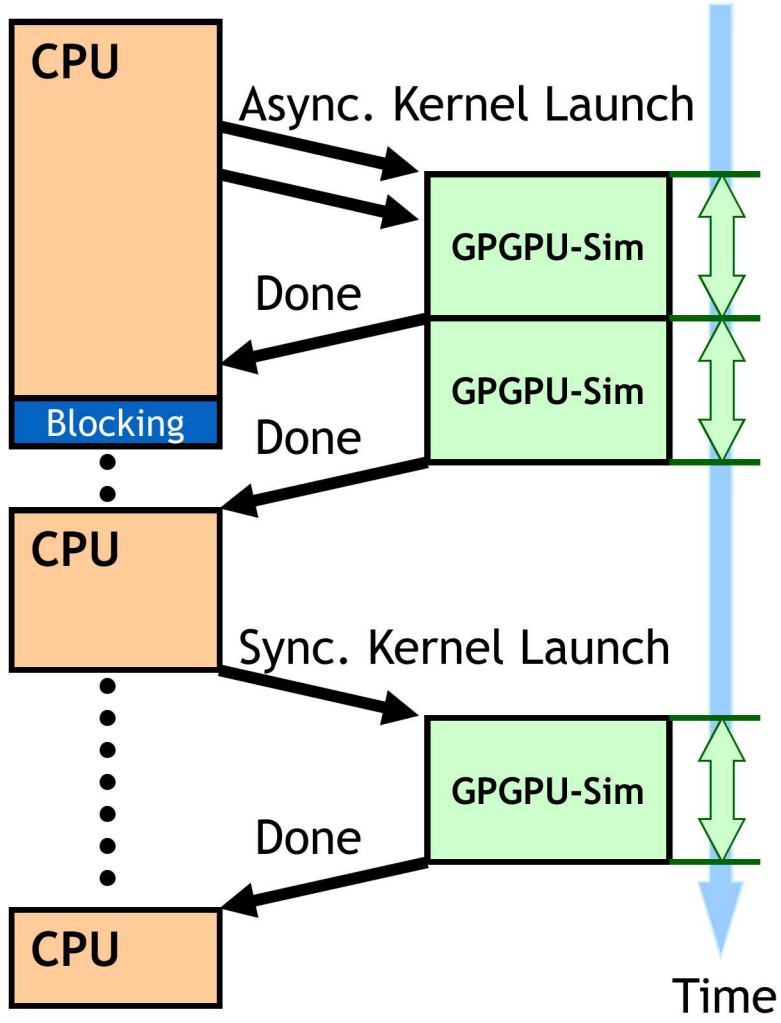
Performance model

- Model microarchitecture timing relevant to GPU compute

GPGPU-Sim Introduction

GPGPU-Sim simulates kernel

- Transfer data to GPU memory
- GPU kernels runs on GPGPU-Sim:
  - Reports statistics for the kernels
- Transfer data back to CPU memory



Functional model

Single Instruction Multiple Thread(SIMT):

- SIMD + multithreading
- Grid, Block, Warp, Thread

Virtual ISA vs. Machine ISA

- vISA: PTX = Parallel Thread eXecution: virtual ISA defined by Nvidia
- mISA: SASS = Native ISA for Nvidia GPUs
- GPGPU-Sim use PTXPlus to represent SASS
  - 1:1 mapping from SASS to PTXPlus

GPGPU-Sim supports:

- PTX for new architectures(CUDA 10): new, inaccurate, well documented
- SASS for architecture before Fermi(SM_1.X): old, accurate, less documented

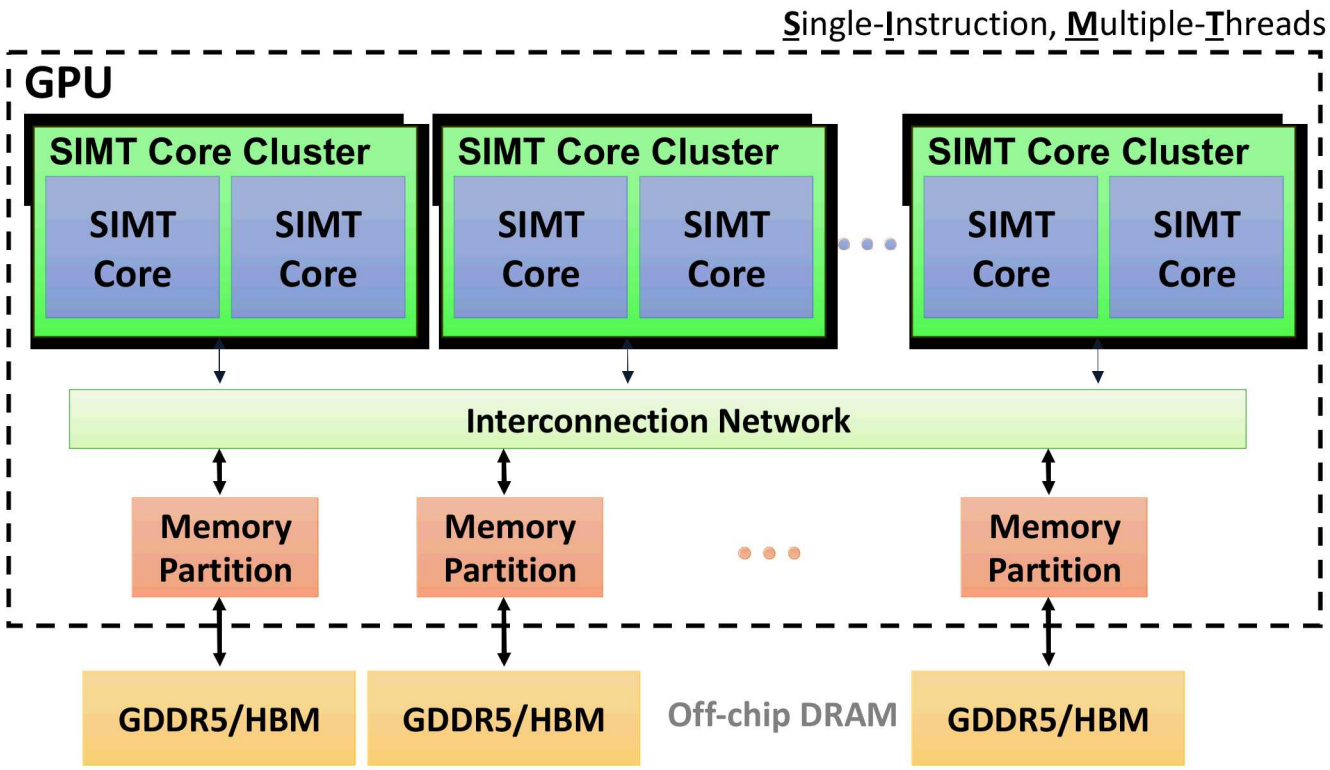
Performance model

GPGPU-Sim models timing

- SIMT Core
- Caches and texture/constant/shared memory
- Interconnection network(Booksim)
- DRAM(GDDR5/HBM)

DO NOT model

- Graphic Specific Hardware

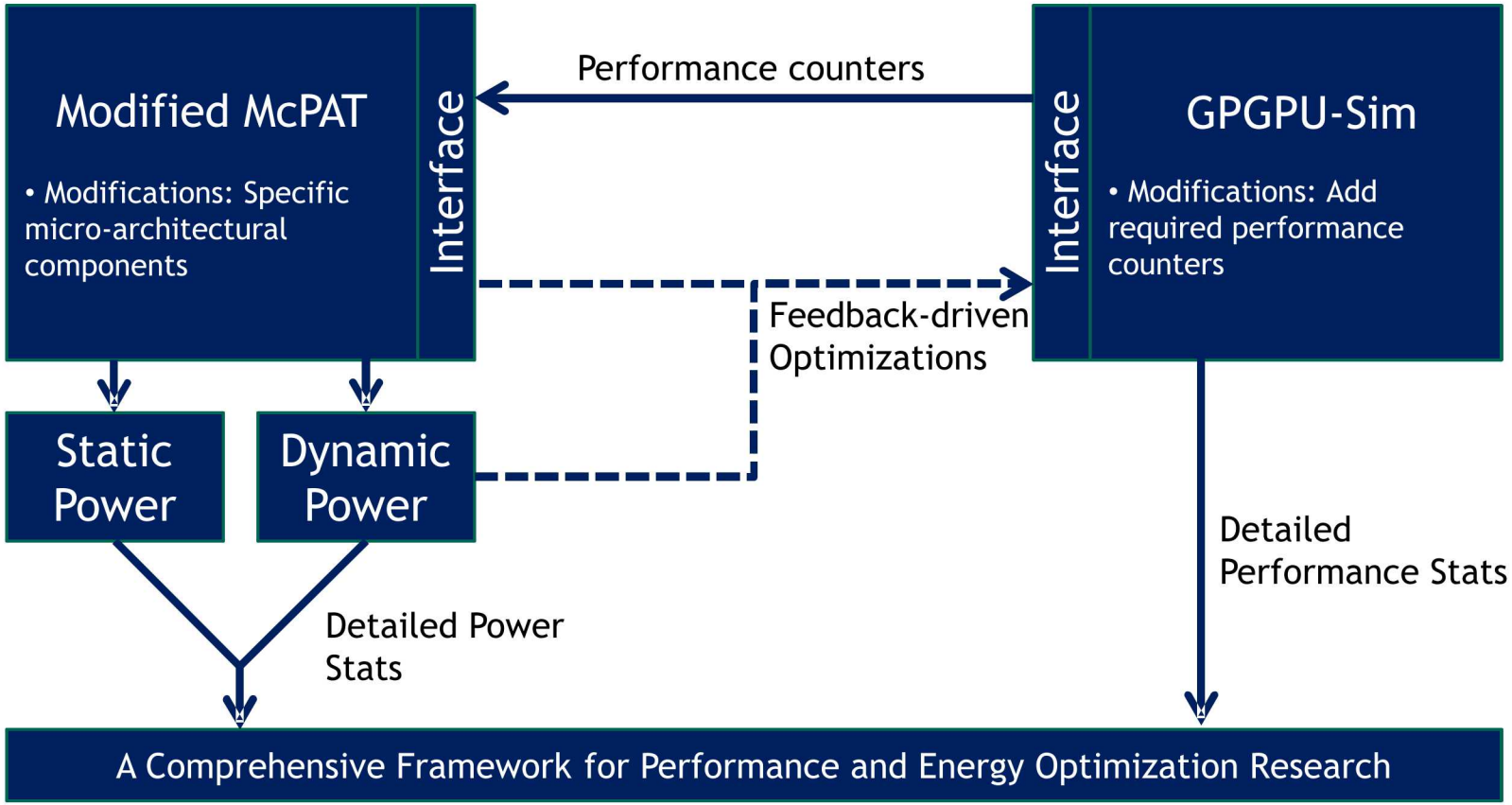


GPUWattch: Power model

Estimate power consumed by the GPU according to the timing behavior

Ideal for evaluating fine-grained power management mechanisms

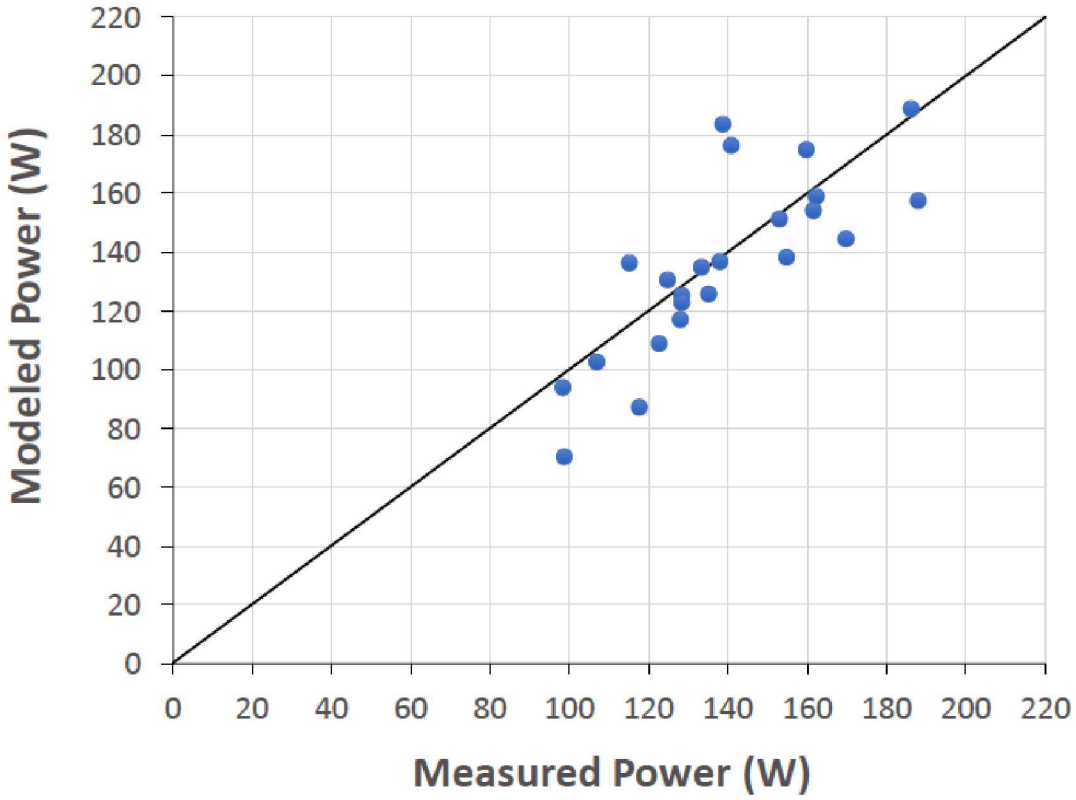
Validated with power measurements from GV100

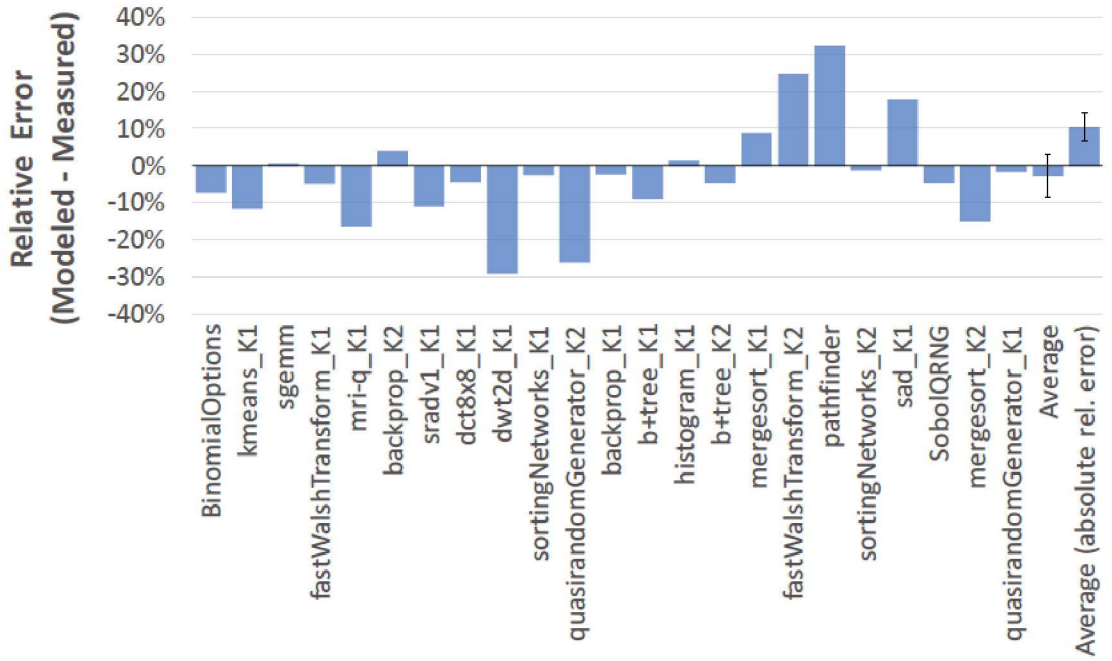


GPUWattch: Power model

Correlation for GV100

- Coefficient of 0.8
- Average relative error of $-2.67\% \pm 5.66\%$ (95% confidence interval)





Outline

GPGPU-Sim Introduction
- GPU and programming model
- Functional model
- Performance model
- GPUWattch: power model

New Features in GPGPU-Sim
- Volta model
- Run closed source libraries
- Tensor Core
- Run CUTLASS library

New Feature: Volta Model Motivation

ISA cycle correlation[1] before new Volta model[2]

- For Pascal Titan X

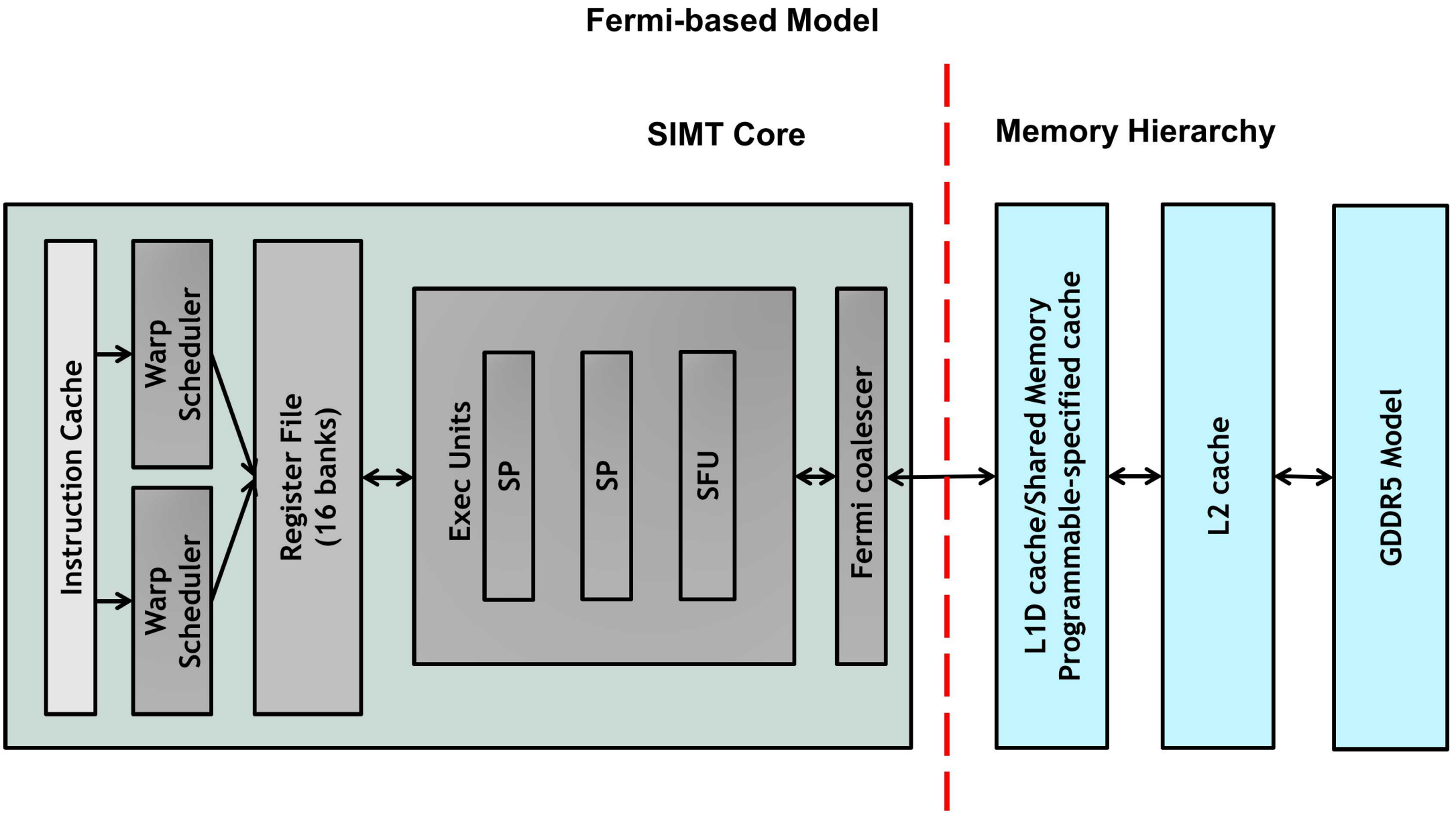
Benchmark	Means Abs Error		Correlation	
	vISA	mISA	vISA	mISA
Compute Intensive	43.3%	21.9%	91.1%	99.0%
Cache Sensitive	104.8%	100.4%	81.2%	82.0%
Memory Sensitive	31.6%	29.8%	96.0%	95.1%
Compute Balanced	58.5%	70.7%	96.1%	93.5%

Result

- Compute intensive: mISA > vISA
- Cache sensitive: both show inaccurate cache model
- Memory sensitive(streaming): not related to cache model
- Compute balanced: vISA > mISA

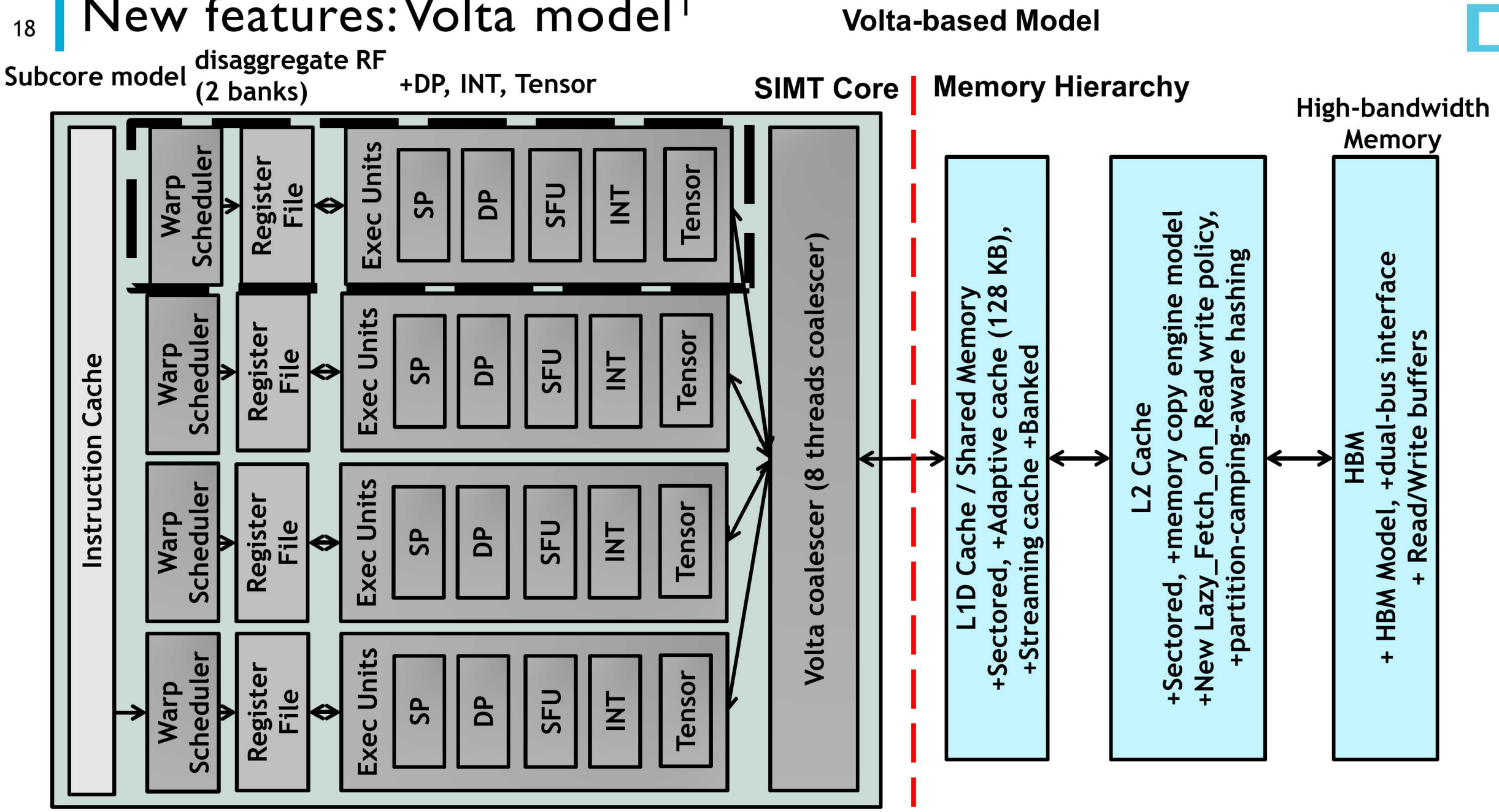
[1] Akshay Jain, Mahmoud Khairy, Timothy G. Rogers, A Quantitative Evaluation of Contemporary GPU Simulation Methodology. SIGMETRICS 2018
[2] Mahmoud Khairy, Jain Akshay, Tor Aamodt, Timothy G Rogers, Exploring Modern GPU Memory System Design Challenges through Accurate Modeling, arXiv:1810.07269

New features: Volta model[1]



[1] Mahmoud Khairy, Jain Akshay, Tor Aamodt, Timothy G Rogers, Exploring Modern GPU Memory System Design Challenges through Accurate Modeling, arXiv:1810.07269

New features: Volta model[1]



[1] Mahmoud Khairy, Jain Akshay, Tor Aamodt, Timothy G Rogers, Exploring Modern GPU Memory System Design Challenges through Accurate Modeling, arXiv:1810.07269

Hardware correlation for Volta model[1]

Statistic	Means Abs Error		Correlation	
	Old Model	New Model	Old Model	New Model
Execution Cycles	68%	27%	71%	96%
L1 Reqs	48%	0.5%	92%	100%
L1 Hit Ratio	41%	18%	89%	93%
L2 Reads	66%	1%	49%	94%
L2 Writes	56%	1%	99%	100%
L2 Read Hits	80%	15%	68%	81%
DRAM Reads	89%	11%	60%	95%

2.5X error reduction in exec time

0.5% error in L1 reqs (96x reqs error reduction)

SIMT Core

Memory Hierarchy

1% error in L2 behavior (66x read error reduction)

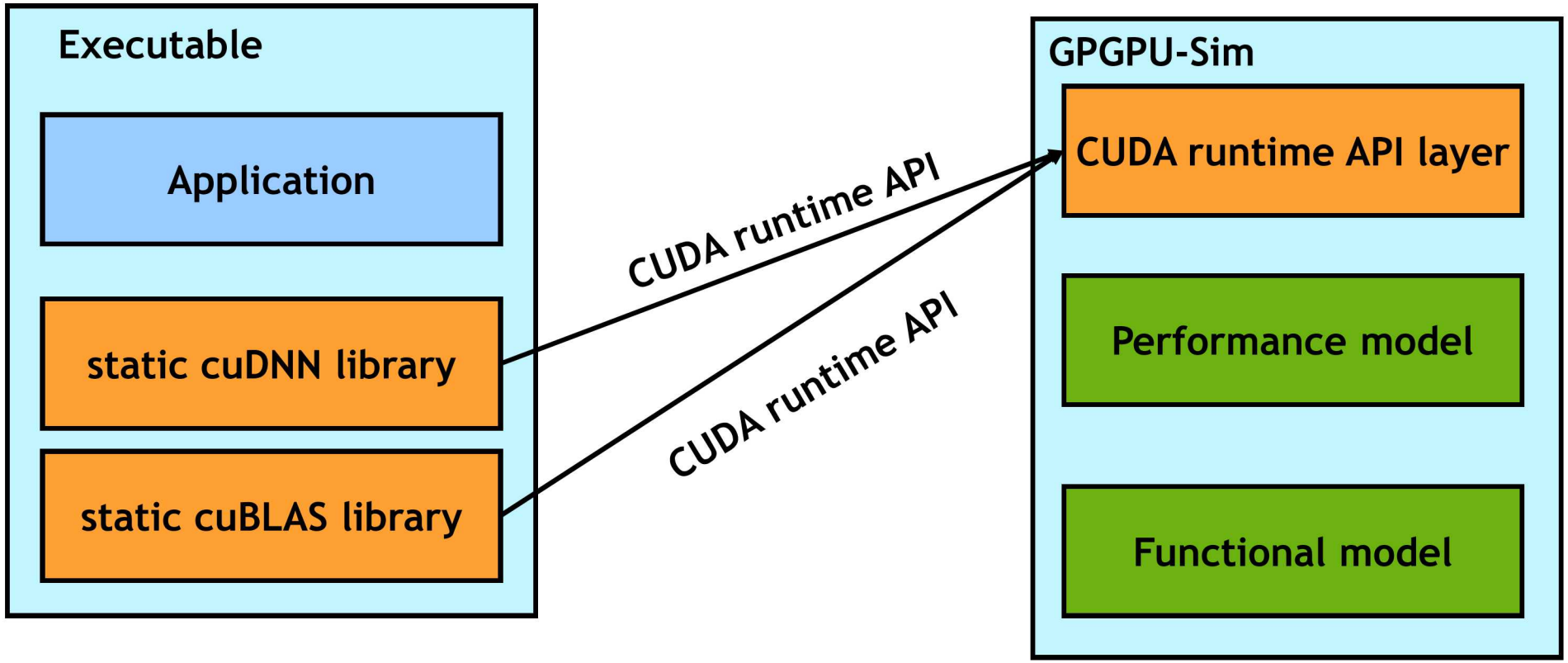
7X error reduction in DRAM reads

[1] Mahmoud Khairy, Jain Akshay, Tor Aamodt, Timothy G Rogers, Exploring Modern GPU Memory System Design Challenges through Accurate Modeling, arXiv:1810.07269

Run closed source model[1]

Run applications with cuDNN/cuBLAS

- Static linking closed source libraries
- LeNet for MNIST using cuDNN/cuBLAS



[1] Jonathan Lew, Deval Shah, Suchita Pati, Shaylin Cattell, Mengchi Zhang, Amruth Sandhupatla, Christopher Ng, Negar Goli, Matthew D. Sinclair, Timothy G. Rogers, Tor M. Aamodt Analyzing Machine Learning Workloads Using a Detailed GPU Simulator, arXiv:1811.08933

Tensor Core in GV100

Accelerate FP operations

8 Tensor Core/SM

Each perform 64 FP FMA/clock

512 FMA/clock/SM

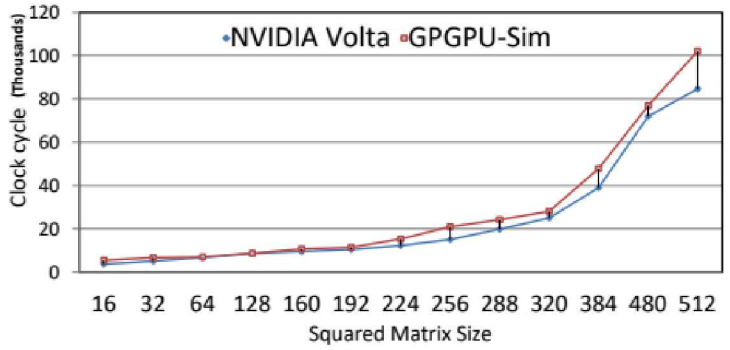
Or 1024 FP ops/clock/SM

Tensor Core in Tesla Titan V[1]

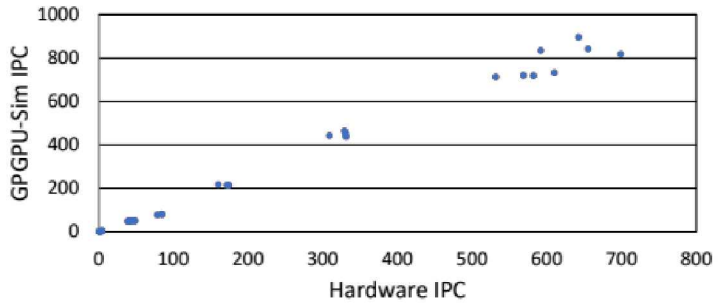
Standard deviation of less than 5%

99.60% IPC correlation

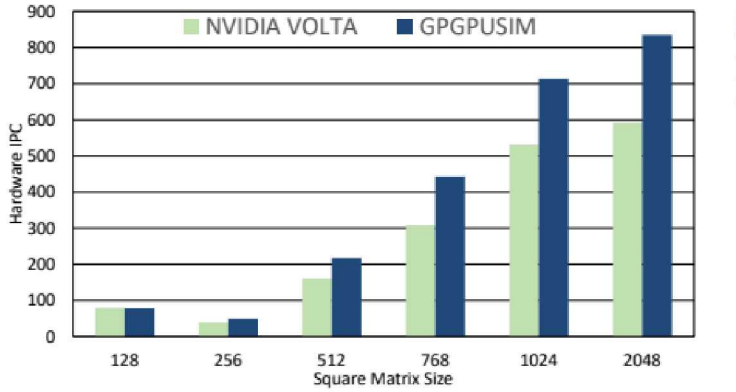
GPGPU-Sim shows higher perf than HW as matrix size increase



(a) WMMA-based GEMM kernel cycle count as matrix size varies.



(b) Instructions per cycle (IPC) correlation of CUTLASS GEMM kernel on GPGPU-Sim vs Titan V.



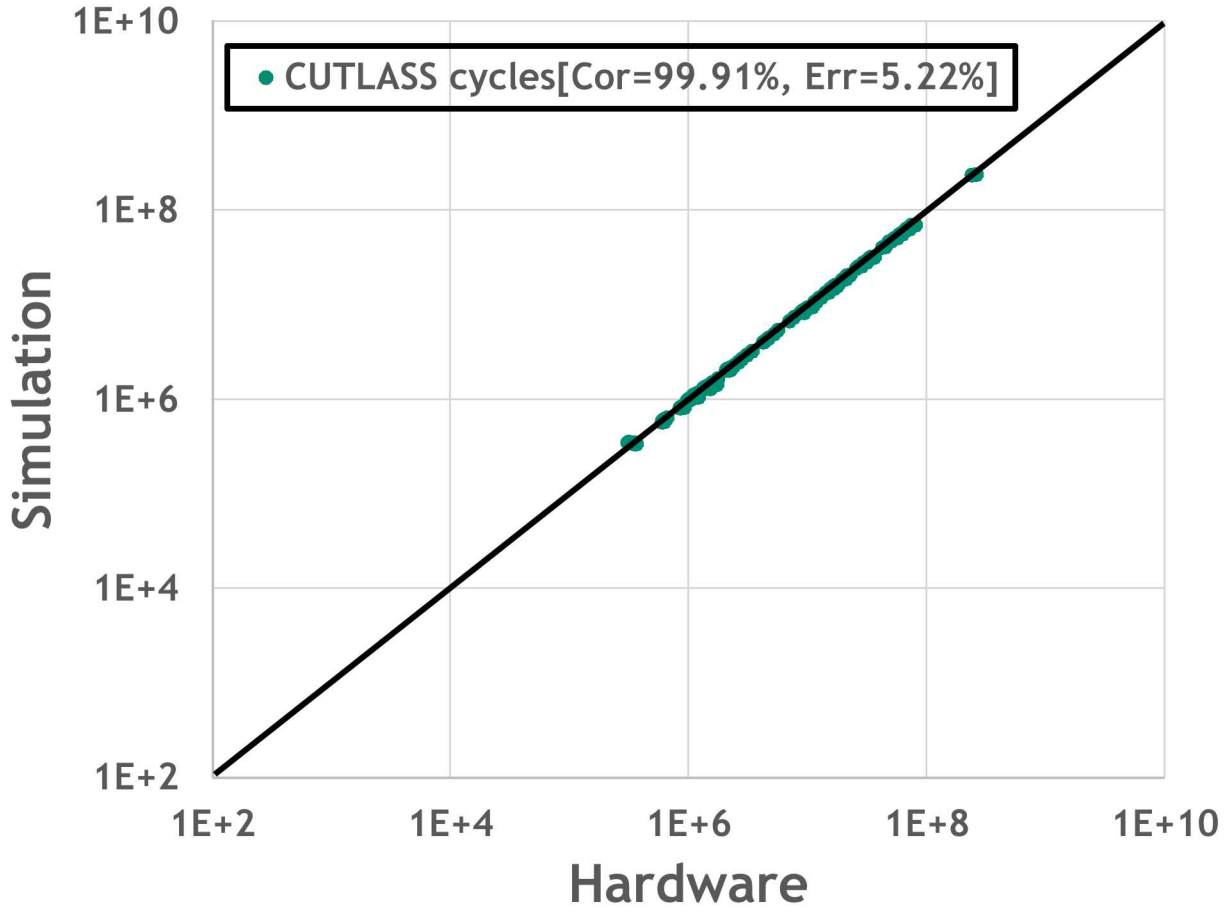
(c) CUTLASS-based GEMM kernel cycle count as matrix size varies.

[1] Md Aamir Raihan, Negar Goli, Tor Aamodt, Modeling Deep Learning Accelerator Enabled GPUs, ISPASS 2019

Run CUTLASS library

Combine CUTLASS, tensor core[1], and volta model[2]

Correlation: using Deepbench training/inference test from real scenario



[1] Md Aamir Raihan, Negar Goli, Tor Aamodt, Modeling Deep Learning Accelerator Enabled GPUs, ISPASS 2019
[2] Mahmoud Khairy, Jain Akshay, Tor Aamodt, Timothy G Rogers, Exploring Modern GPU Memory System Design Challenges through Accurate Modeling, arXiv:1810.07269

Summary

GPGPU-Sim:

- Simulates GPU compute unit and memory hierarchies.
- an evolutional simulator for new features in CUDA and GPUs.
  - actively including new features

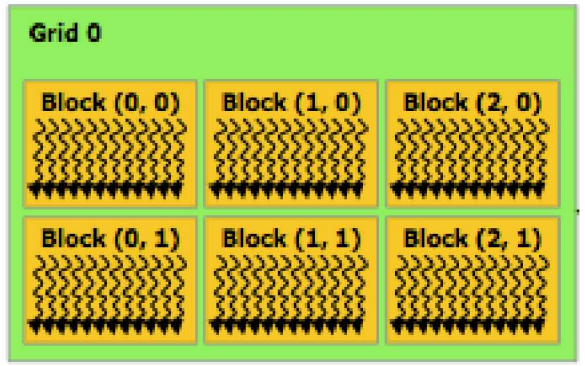
GPGPU-Sim with SST:

- GPGPU-Sim brings promising GPU model to SST
- GPGPU-Sim benefits for SST parallel simulation architecture

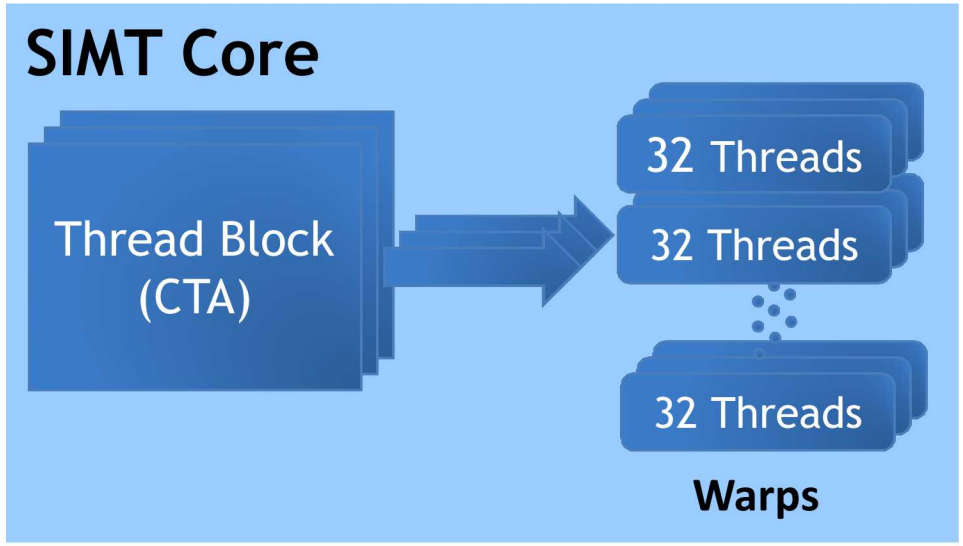

QUESTIONS

Thread Hierarchy Revisited

- Recall, kernel = grid of blocks of warps of threads
- Threads are grouped into *warps* in *hardware*

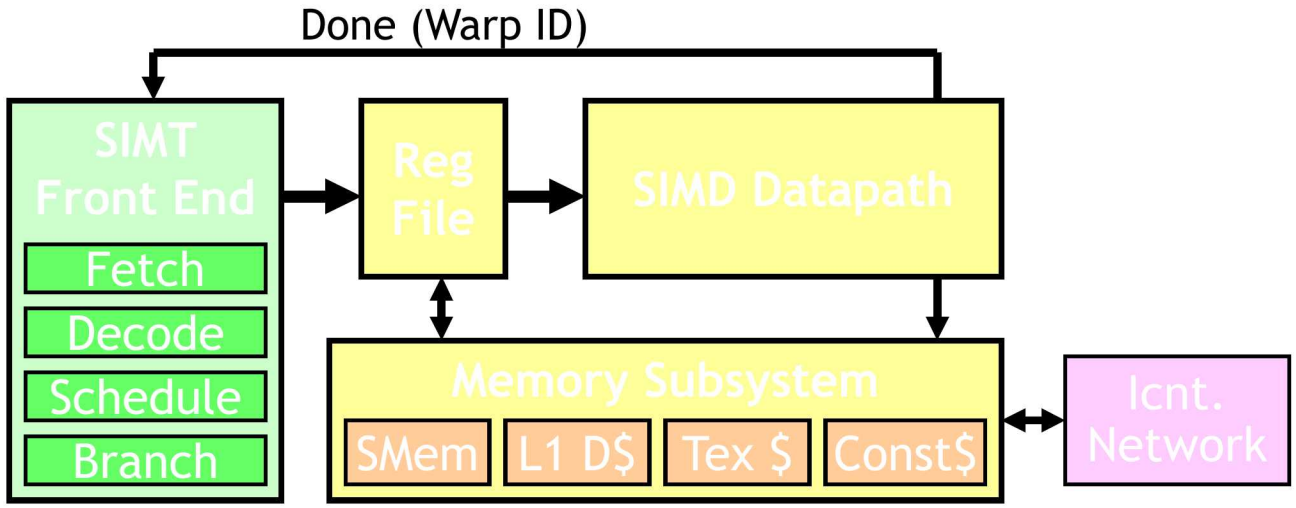


Source: NVIDIA



Each block is dispatched to a SIMT core as a unit of work: All of its warps run in the core's pipeline until they are all done.

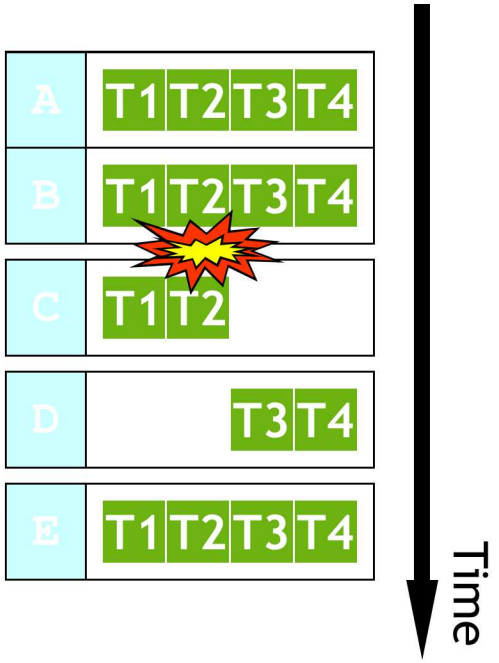
Inside a SIMT Core



- Fine-grained multithreading
  - Interleave warp execution to hide latency
  - Register values of all threads stays in core

SIMT Stack

```
foo[] = {4,8,12,16};
 A: v = foo[tid.x];
 B: if (v < 10)
 C:     v = 0;
    else
 D:     v = 10;
 E: w = bar[tid.x]+v;
```



One stack per warp
SIMT Stack

PC	RPC	Active Mask
E	-	1111
D	E	0011
C	E	1100

Handles *Branch Divergence*

Constant Cache

A Read-only cache for constant memory

GPGPU-Sim simulates 1 read ports

- A warp can access 1 constant cache locations in a single memory unit cycle
- If more than 1 locations accessed
  - reads are serialized causing pipeline stalls
- # of ports is not configurable

Coalescing

Combining memory accesses made by threads in a warp into fewer transactions

- E.g. if threads in a warp are accessing consecutive 4-byte sized locations in memory
  - Send one 128–byte request to DRAM (coalescing)
  - Instead of 32 4-byte requests

This reduces the number of transactions between SIMT cores and DRAM

- Less work for Interconnect, Memory Partition and DRAM

Interconnection Network Model

Intersim (Booksim) a flit level simulator
- Topologies (Mesh, Torus, Butterfly, …)
- Routing (Dimension Order, Adaptive, etc.)
- Flow Control (Virtual Channels, Credits)

We simulate two separate networks
- From SIMT cores to memory partitions
  - Read Requests, Write Requests
- From memory partitions to SIMT cores
  - Read Replies, Write Acks

PTX

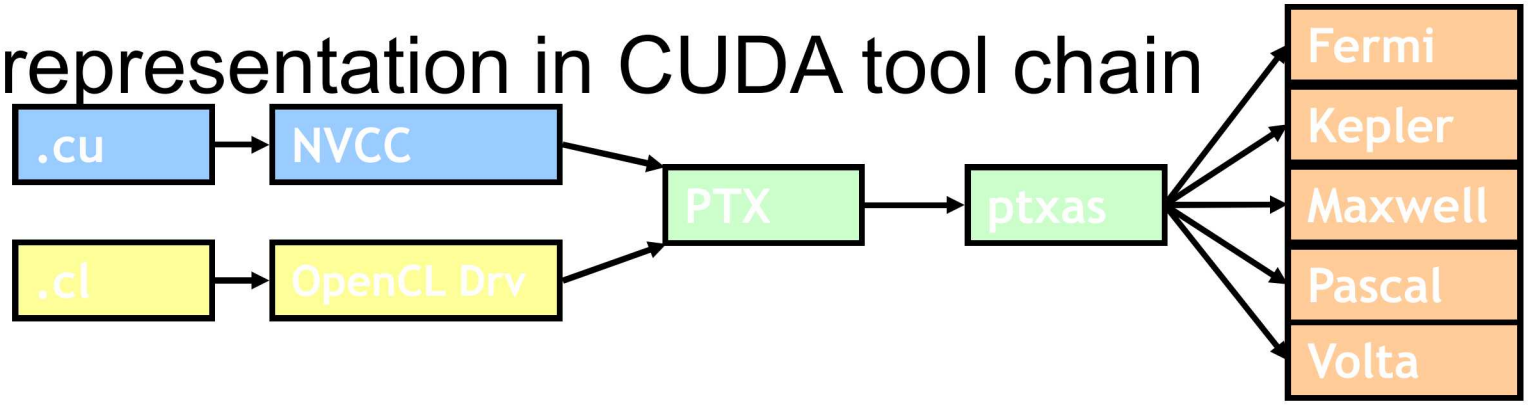
Low-level data-parallel virtual ISA

- Instruction level
- Unlimited registers
- Parallel threads running in blocks; barrier synchronization instruction

Scalar ISA

- SIMT execution model

Intermediate representation in CUDA tool chain



SASS

Native ISA for Nvidia GPUs

- Better correlation with HW but less documented
- Related with Stream Multiprocessor(SM) version.

Scalar ISA

GPGPU-Sim use PTXPlus to represent both SASS and PTX

SASS mapped 1:1 into PTXPlus instruction

PTX vs. SASS

PTX

```
$Lt_25_13570:
  ld.global.s32       %r9, [%rd5+0];
  add.s32             %r10, %r9, %r8;
  ld.global.s32       %r11, [%rd5+1024];
  add.s32             %r8, %r11, %r10;
  add.u32             %r5, %r7, %r5;
  add.u64             %rd5, %rd5, %rd6;
  ld.param.u32        %r6, [size];
  setp.lt.u32         %p2, %r5, %r6;
  @%p2 bra            $Lt_25_13570;
...
  mov.u32  %r12, 127;
  setp.gt.u32         %p3, %r3, %r12;
  @%p3 bra            $Lt_25_14082;
  ld.shared.s32       %r13, [%rd10+512];
  add.s32             %r8, %r13, %r8;
  st.shared.s32       [%rd10+0], %r8;
$Lt_25_14082:
  bar.sync  0;
```

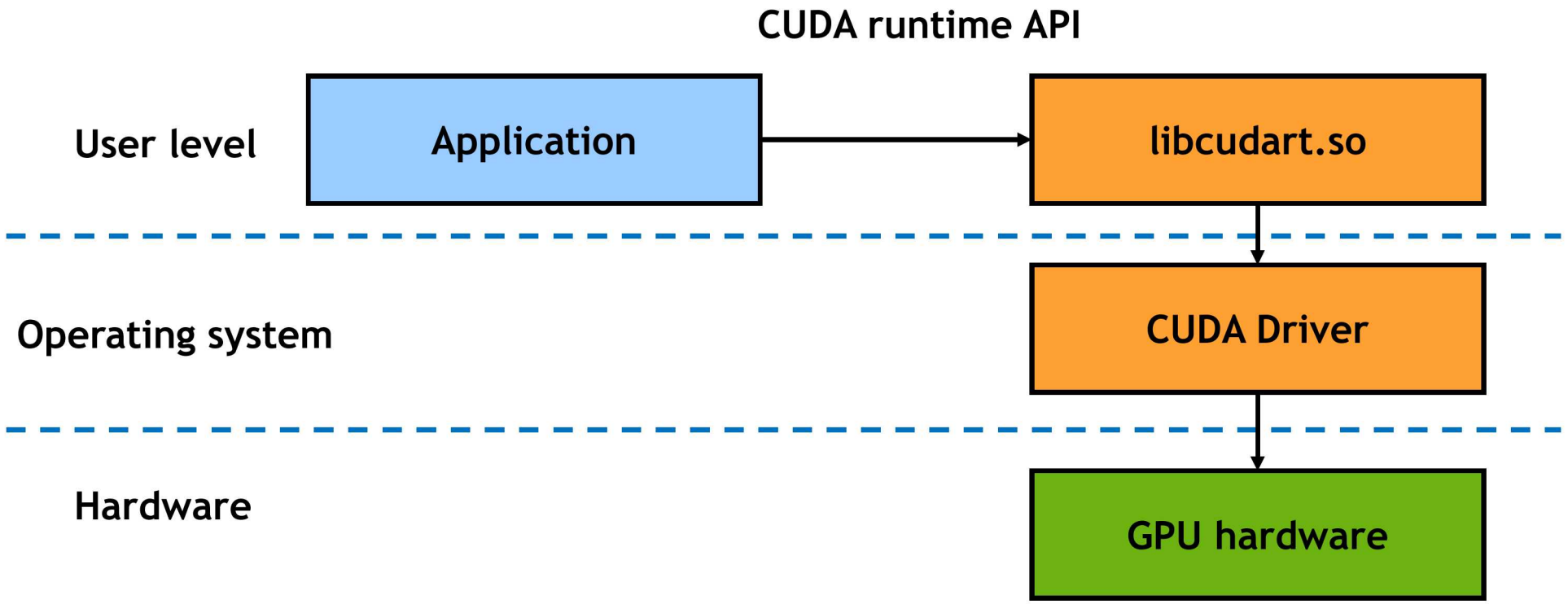
SASS (PTXPlus)

```
l0x00000060:
  add.half.u32        $r7, $r4, 0x00000400;
  ld.global.u32       $r8, [$r4];
  ld.global.u32       $r7, [$r7];
  add.half.u32        $r0, $r5, $r0;
  add.half.u32        $r6, $r8, $r6;
  set.gt.u32.u32      $p0/$o127, s[0x0020], $r0;
  add.half.u32        $r6, $r7, $r6;
  add.half.u32        $r4, $r4, $r3;
  @$p0.ne bra         l0x00000060;
...
  set.gt.u32.u32      $p0/$o127, $r2, const [0x0000];
  @$p0.equ add.u32    $ofs2, $ofs1, 0x00000230;
  @$p0.equ add.u32    $r6, s[$ofs2+0x0000], $r6;
  @$p0.equ mov.u32    s[$ofs1+0x0030], $r6;
  bar.sync  0x00000000;
```

Interfacing GPGPU-Sim to applications

GPGPU-Sim compiles into a shared runtime library and implements the API:

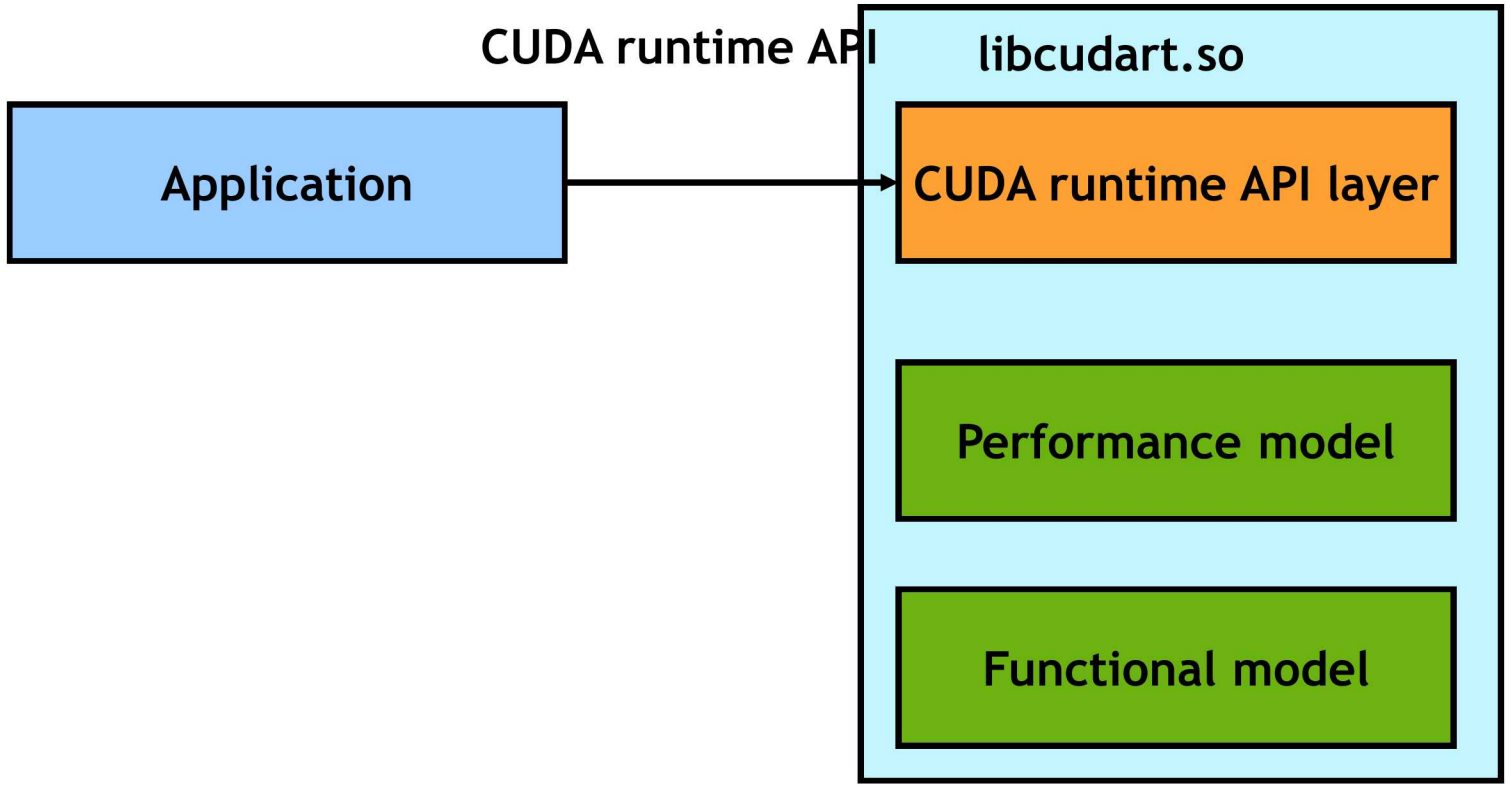
- libcudart.so ← CUDA runtime API
- libOpenCL.so ← OpenCL API



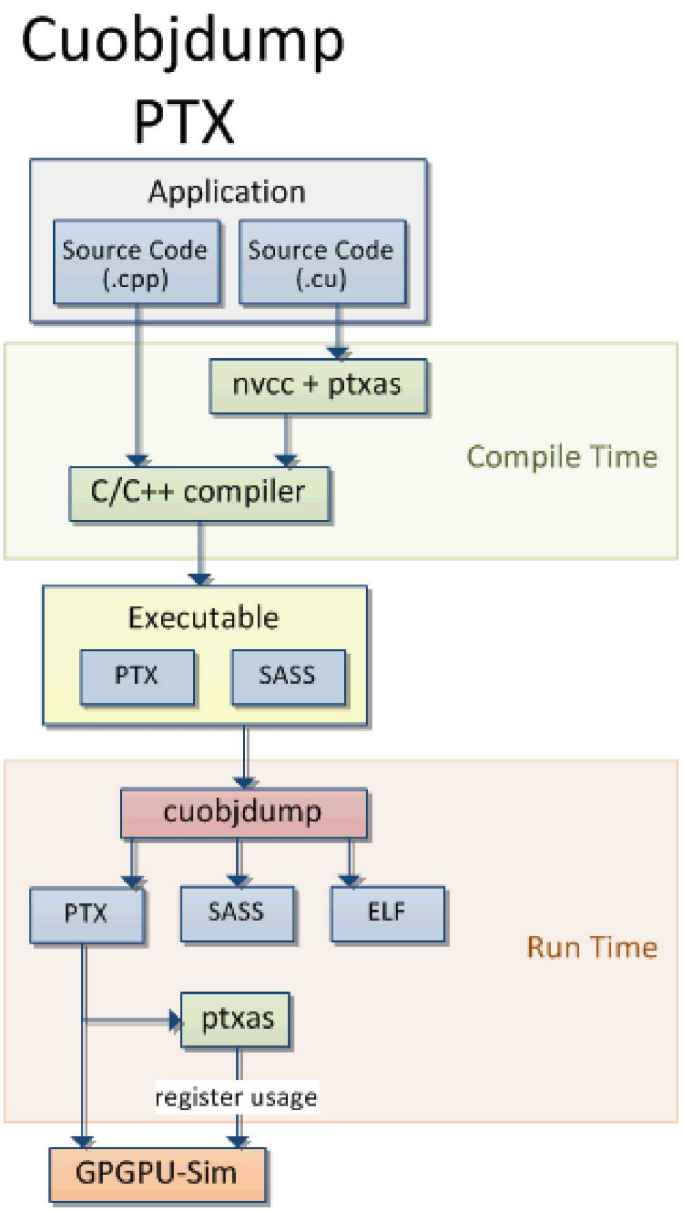
Interfacing GPGPU-Sim to applications

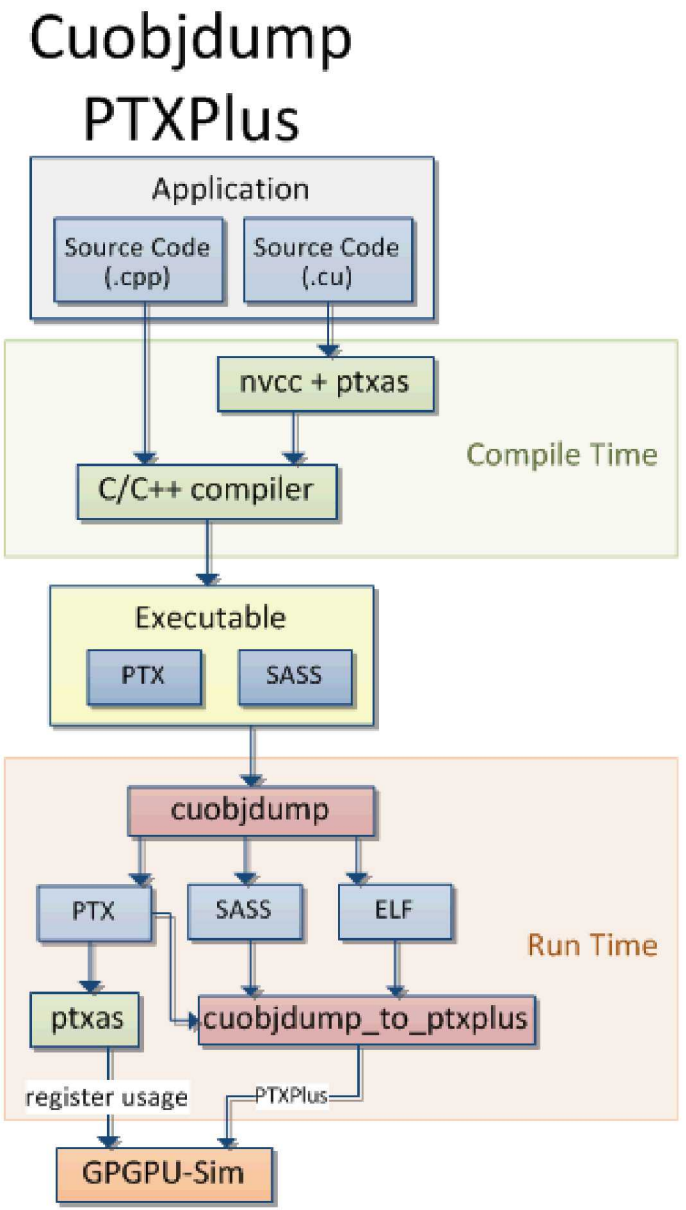
GPGPU-Sim compiles into a shared runtime library and implements the API:

- libcudart.so ← CUDA runtime API
- libOpenCL.so ← OpenCL API



GPGPU-Sim Runtime Flow





CLICK TO EDIT MASTER TITLE STYLE

Slide Left Blank