GPU H/W and S/W

Hyesoon Kim

Georgia Tech

GPU simulation

Performance Modeling Techniques

- Cycle level simulation
- Event driven simulation
- Analytical Model
- Sampling based techniques
- Data based statistical/ML modeling
- FPGA based emulation

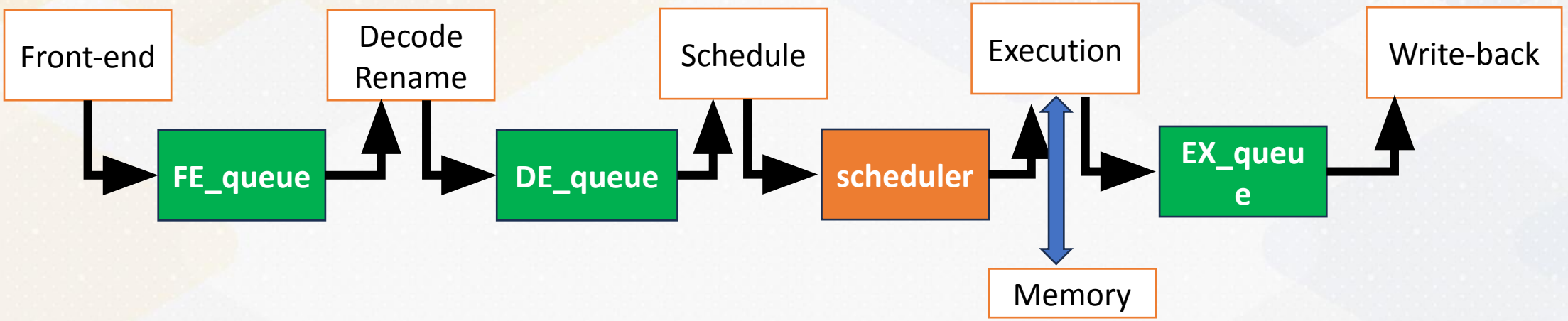
Cycle Level Simulation

- Commonly used in many architecture simulators
- Typically, a global clock exists.
- Each cycle, events, such as instruction fetch and decode are modeled.
- Multiple clock domains can exist. (E.g. Memory clock, processor clock, NoC clock)

Execution Driven vs. Trace Driven Simulation

- Execution driven: instructions are executed during the simulation
  - Execute-at-fetch vs execute-at-execute
  - Execute-at-fetch: instruction is executed when an instruction is fetched
  - Execute-at-execute: instruction is executed at the execution stage
- Trace driven: traces are collected; simulation and execution are decoupled
- Benefits of execution driven: no need to store traces
- Trace driven cannot model run-time dependent behaviors: e.g.) lock acquire, barriers
- Trace-driven simulators are simpler and often lighter and easier to develop
- E.g.) Memory traces only for memory simulation or cache

Queue Based Modeling



- Instructions are moving between queues.
- Scheduler selects instructions that will be sent to the execution stage among ready instructions; not implemented as a queue structure.
- Other queues are FIFO.
- When instruction is complete, the dependent instructions are ready. The dependency chain needs to be modeled and broadcasting also needs to be modeled.
- Cache and memory are modeled to provide memory instruction latency.

Georgia Tech

Modeling Parameters with Queue Based Modeling

- Number of cycles in each pipeline stage □ depth of the queue
- How many instructions can move between queues represent pipeline width (E.g., Issue/execution bandwidth)
- Questions: How do you know the latency of each instruction?
- Instruction latency assumptions:
  - Instruction latency is given as a parameter (e.g., ADD takes 1 cycle, MUL takes 3 cycles).
  - Latency can be obtained from literature or simulators like CACTI or RTL simulation.

5-stage CPU Processor Modeling

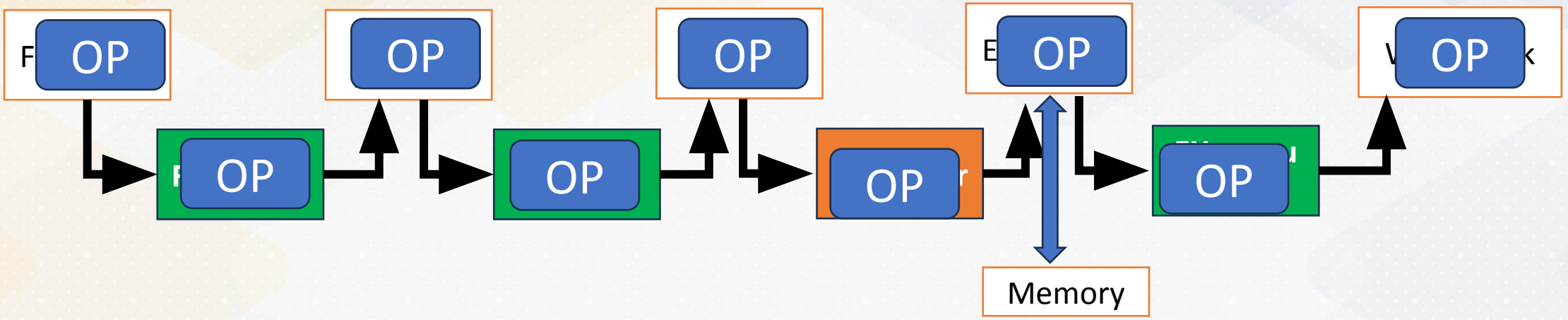
```
int main(int argc, char** argv) {
  macsim_c* sim;

  // Instantiate
  sim = new macsim_c();

  // Initialize Simulation State
  sim->initialize(argc, argv);

  // Run simulation
  // report("run core (single threads)");
  while (sim->run_a_cycle())
    ;

  // Finialize Simulation State
  sim->finalize();

  return 0;
}
```

```
macsim::run_a_cycle (…) {
cycle++;
Cpu->run_a_cycle();
Mem->run_a_cycle();
Noc->run_a_cycle();
}
Cpu->run_a_cycle(…) {
 Wb->run_a_cycle();
Exec->run_a_cycle();
Sch->run_a_cycle();
De->run_a_cycle();
Fe->run_a_cycle();
 }
```

Kim, H. (n.d.). gthparch/macsim. GitHub. https://github.com/gthparch/macsim

Georgia Tech

Example



- Each modeled instruction has an op data structure.
- Op data structure tracks instruction progress and cycle.
- E.g.) Op->done_cycle = op->schedule_cycle + latency
- (Q) what if latency is not fixed?
- E.g.) Cache misses.
- After cache simulation (cache hit/miss hierarchy), we can know the memory instruction latency.

Georgia Tech

Scheduler

- Scheduler checks ready instructions.
- Scheduler handles resource constraints.
- E.g.) # of FP instructions, # of load and store queues, execution units, ports in cache etc.
- Ensuring resources are available for multiple cycles

GPU Cycle-Level Modeling

- Similar to CPU modeling
- The simulation modeling unit is a warp.
  - Warp instruction fetched/decoded/scheduled/executed
- Scheduler chooses instructions from the head of each warp.
- Differences from CPUs:
  - In-order scheduling within a warp
  - Out-of-order across warps
- Major differences between CPU vs. GPU
  - Handling divergent warps
  - Warp, thread block, and kernel concepts
  - Scheduler

End of Simulation

```
int main(int argc, char** argv) {
  macsim_c* sim;

  // Instantiate
  sim = new macsim_c();

  // Initialize Simulation State
  sim->initialize(argc, argv);

  // Run simulation
  // report("run core (single threads)");
  while (sim->run_a_cycle())
    ;

  // Finialize Simulation State
  sim->finalize();

  return 0;
}
```

- Entire thread block scheduled to one SM
- Tracking complete threads
- All threads within a cuda block, the corresponding cuda block completes
- When all thread block is completed, the kernel ends.
- When all kernel ends, the application ends.

Georgia Tech

slido

Please download and install the Slido app on all computers you use

In the queue-based simulation, if we want to increase the execution width, what change do we need to make? Please refer to the diagram in the lecture for the module names. Choose the most relevant one?

ⓘ Start presenting to display the poll results on this slide.

slido

Please download and install the Slido app on all computers you use

Which of the modules cannot be implemented with queue-based modeling?

ⓘ Start presenting to display the poll results on this slide.

Summary

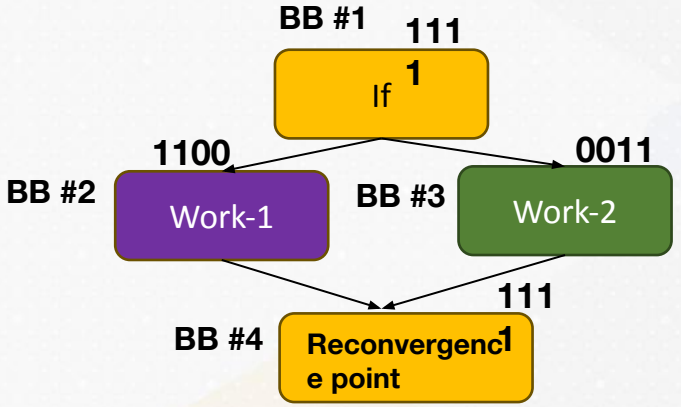
- CPU and GPU cycle-level-simulation techniques reviewed
- Review how to model latency and bandwidth using queues

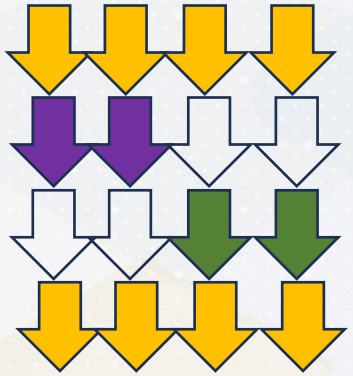
Modeling Unit

- Instructions are modeled at a warp level.
- Accessing I-cache, PC registers
- Reflecting microarchitecture access behavior
- Mask bits are needed to keep track of resource constraints.
- Question: How to model divergent warps and memory coalescing?

Recap: Divergent Branches

- Within a warp, instructions take different paths.
- Simulator also needs to model the SIMT stack.
- Execution driven simulation:
  - Faithfully model SIMT stack





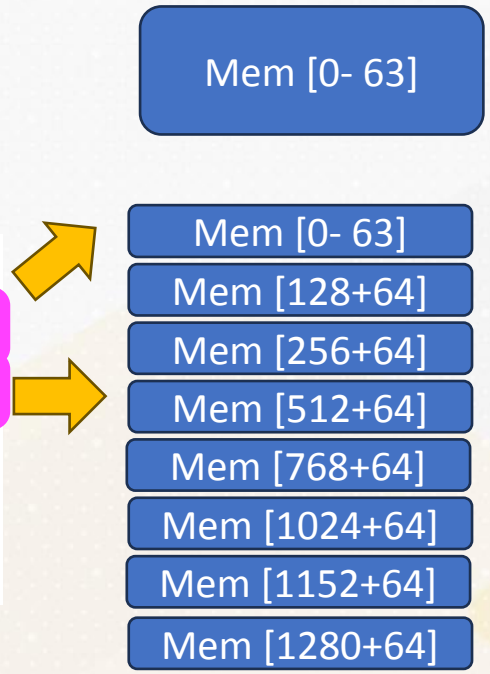
- Trace driven simulation
  - Follow how the traces are collected
  - It's challenging to simulate different divergent branch handling mechanisms than the trace collection's machine.
  - The trace should contain all the paths already. E.g., BB #1 <1111>, BB#2 <1100>, BB #3 <0011>, BB #4 <1111>.
  - The trace needs to have the contents of mask bits.

Georgia Tech

Memory Coalescing Modeling

- Modeling memory coalescing is critical.
- Memory requests need to be merged.
- Typically, this follows cache line sizes.
- A 64 B cache line size is already assumed.

	Th1	Th2	Th3	Th4	Th5	Th6	Th7	Th8
R1	0	4	8	12	16	18	20	24
R2	0	128	256	512	768	1024	1152	1280
R3								
R4								



Georgia Tech

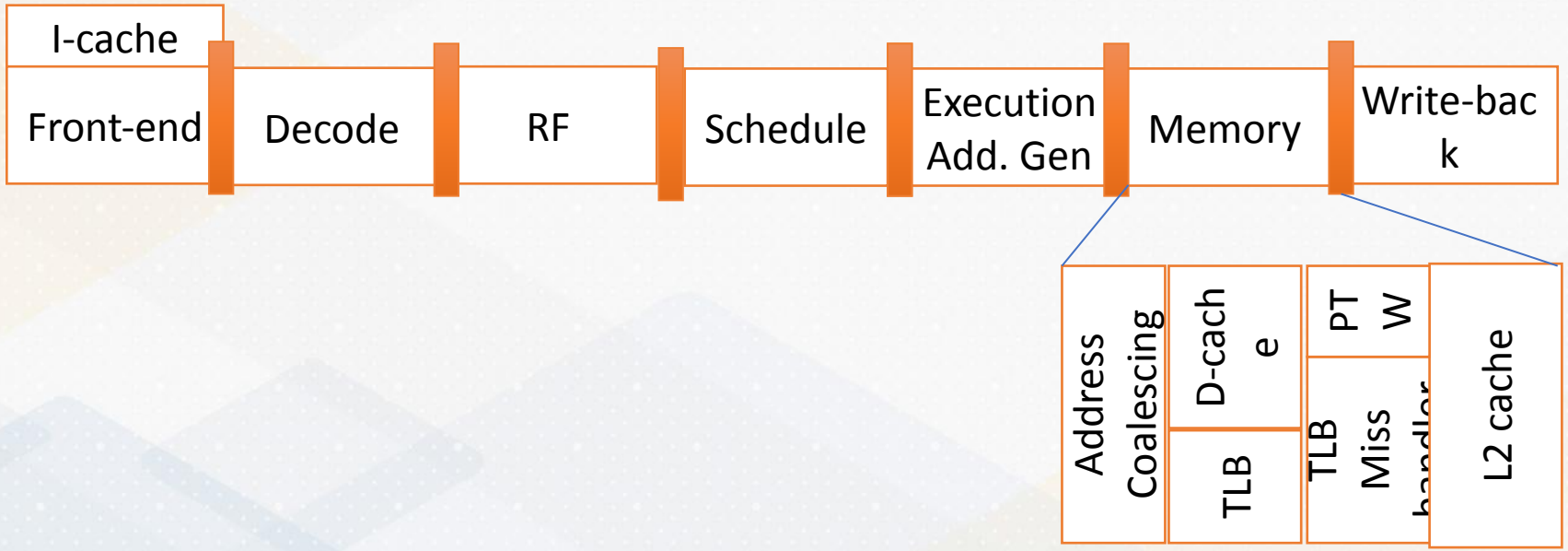
Modeling Memory Coalescing with Trace

- The trace should contain all the memory addresses from each warp.
- The trace generator can insert all memory instructions individually. E.g., va1, va2, va3, etc.
- Or trace generator already coalesces memory requests can reduce the trace size: e.g.) 0x0, 0x4, 0x8, 0x12, 0x16 etc. vs. 0x0 and size 28

Georgia Tech

Cache Hierarchy Modeling

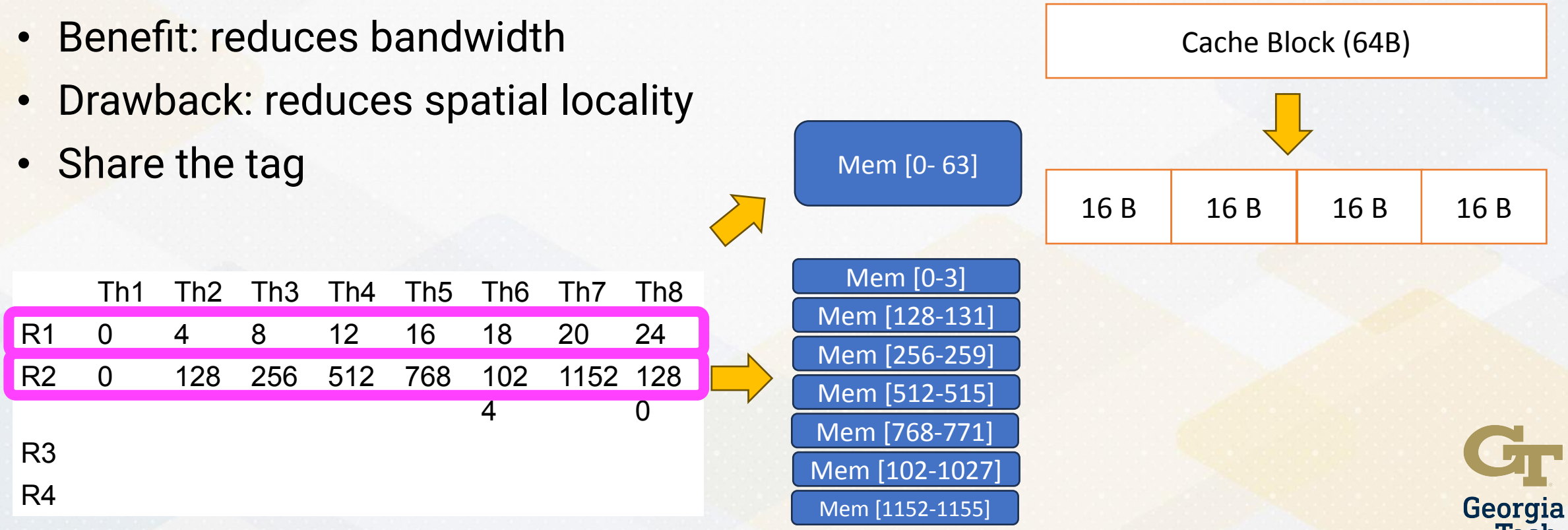
- After addresses are coalesced, memory requests access TLB, L1, L2 caches depending on GPU microarchitecture.



Georgia Tech

Sectored Cache Modeling

- Modern GPUs adopt sectored cache.
- Sectored cache allows bringing a sector of the cache block instead of the entire cache block.
- Benefit: reduces bandwidth
- Drawback: reduces spatial locality
- Share the tag



	Th1	Th2	Th3	Th4	Th5	Th6	Th7	Th8
R1	0	4	8	12	16	18	20	24
R2	0	128	256	512	768	1024	1152	1280
R3								
R4								

Georgia Tech

GPU Simulators

- Several open-source GPU simulators are available.
- GPU simulators for different ISAs

Name	ISA	Type	Architecture	Open Source
GPGPU-Sim	NVIDIA PTX/SASS	Execution driven	GPGPU only	http://www.gpgpu-sim.org/
Accel-Sim	NVIDIA PTX/SASS	Trace driven	GPGPU and accelerator	https://accel-sim.github.io/
MGPU-Sim	AMD GPU	Execution driven	Multi GPUS are supported	[ISCA2019]
Macsim	NVIDIA/Intel GPU	Trace driven	Heterogeneous computing	https://github.com/gthparch/macsim
Gem5-GPGPU-Sim	AMD GPU or NVIDIA PTX	Execution driven	Heterogeneous computing	https://cpu-gpu-sim.ece.wisc.edu/
Vortex-Simx	RISC-V ISA (extensions)	Execution, RTL	3D graphics, GPU	https://vortex.cc.gatech.edu/

slido

Please download and install the Slido app on all computers you use

Which statement most accurately describes sectored cache and small block size of cache?

ⓘ Start presenting to display the poll results on this slide.

Summary

- Review of CPU and GPU Cycle-Level Simulation Techniques
- To support divergent warps, it is necessary to either model the SIMT stack or include mask bits inside the trace.
- Modeling memory coalescing is also the most crucial in the simulation.

GPU Analytical Models

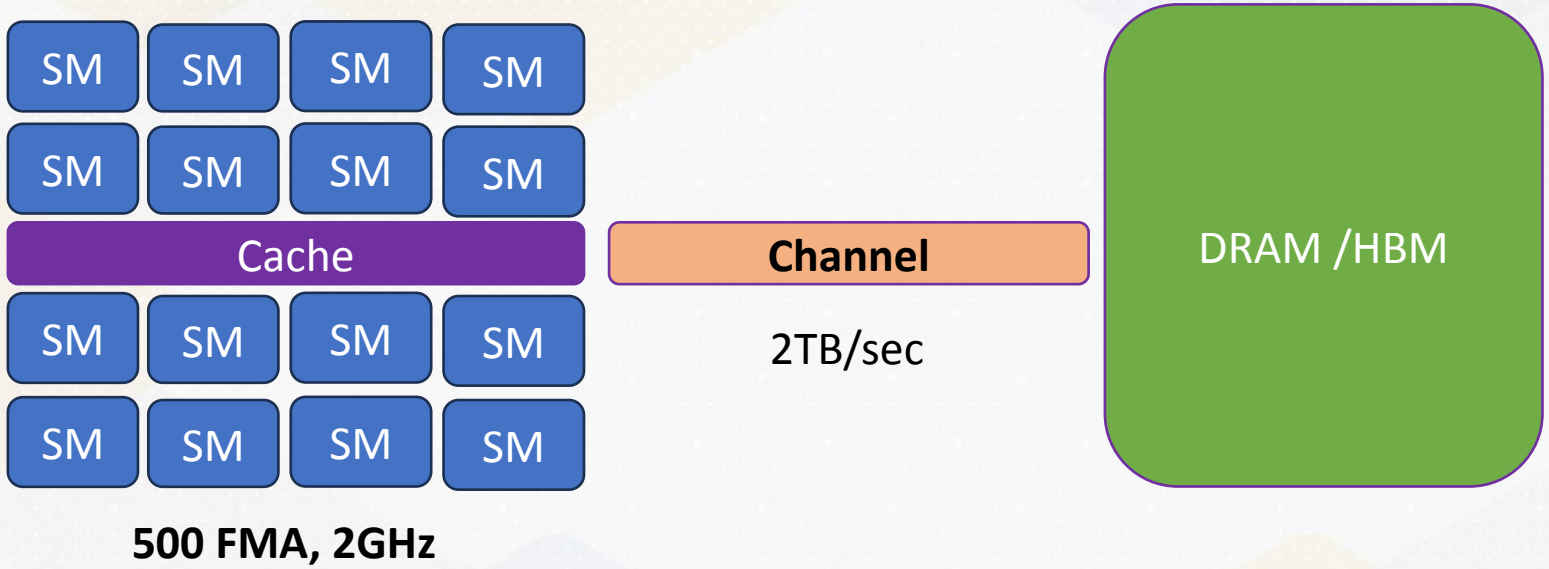
Analytical Models

- Analytical models do not require the execution of the entire program.
- Analytical models are typically simple and capture the first order of performance modeling.
- Analytical models often provide insights to understand performance behavior.

First Order of GPU Architecture Design (1)

- Let's consider accelerating a vector dot product with a goal of 1T vector dot products per second (sum +=x[i] *y[i])
- For compute units, we need to achieve 2T FLOPS operations (multiply and ADD) or 1T FMA/sec.
- If GPU operates at 1GHz, 1000 FMA units are needed; at 2GHz, 500 FMA units are needed.
- Memory units need to supply 2 memory bytes with a 2TB/sec memory bandwidth.

First Order of GPU Architecture Design (2)



- 500 FMA units are approximately equal to 16 warps (warp with 32).
- If each SM can execute 1 warp per cycle at 2GHz and there are 16 SMs, it can compute 1T vector dot products.
- Alternatively, 8 SMs with 2 warps per cycle can also achieve this.

Georgia Tech

slido

Please download and install the Slido app on all computers you use

Based on what we have discussed, if we want to design a processor for 1T vector dot products per second with a 4GHz GPU frequency, what's the memory bandwidth requirement?

ⓘ Start presenting to display the poll results on this slide.

slido

Please download and install the Slido app on all computers you use

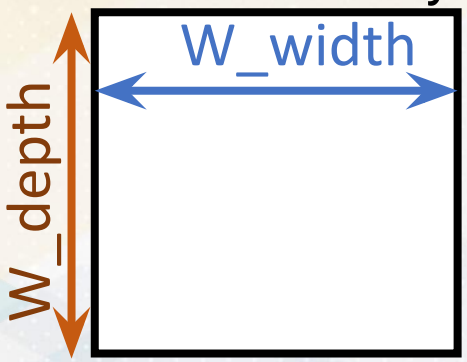
To have 1000 FMA units with a 32-thread width of warps, how many warps need to be executed in one cycle with a 4GHz processor?

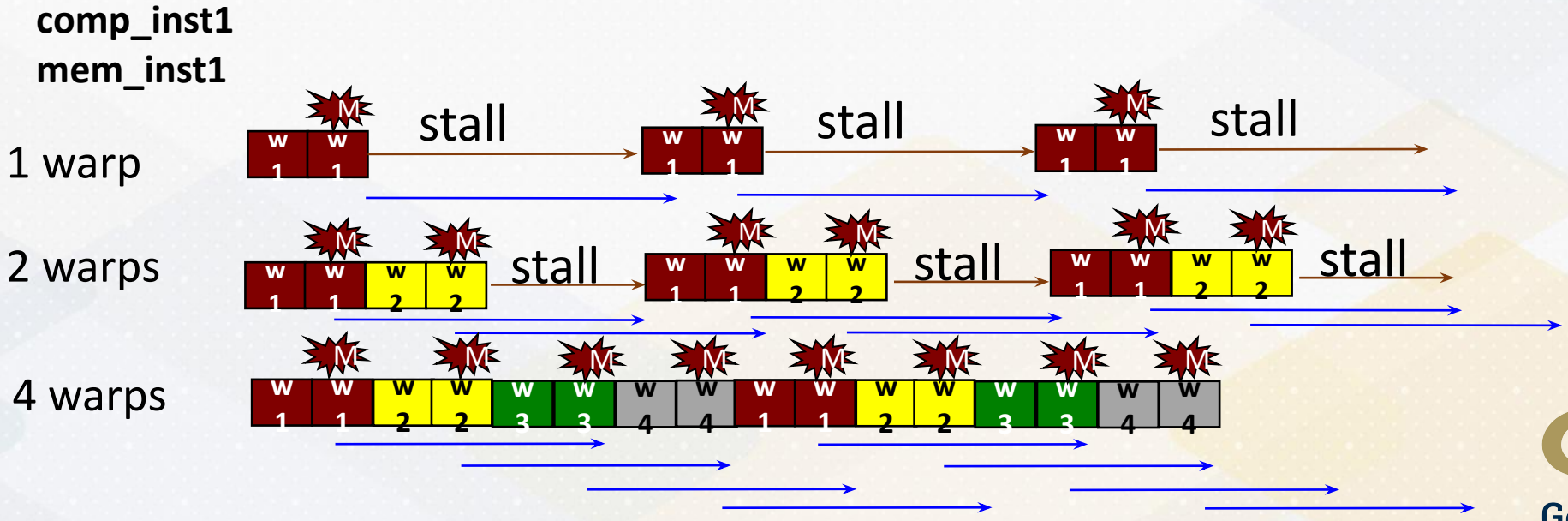
ⓘ Start presenting to display the poll results on this slide.

First Order of GPU Architecture Design (3)

- Multithreading: How about the total number of active warps in each SM?
- W_width : the number of threads (warps) that can run in one cycle

  (W_width x W_depth) number of warps are resident in one SM
- W_depth: the number of threads (warps) that can be scheduled during
- one stall cycle

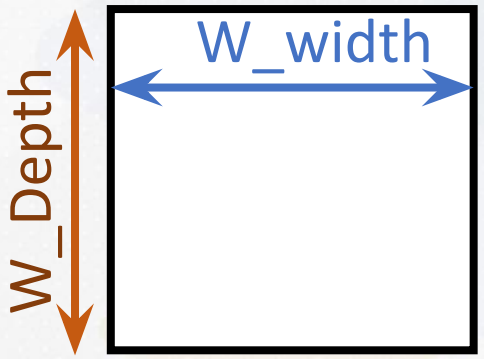




Georgia Tech

W_depth and W_width H/W Constraints

- W_width is determined by the number of ALU units (along with the width of the scheduler)
- W_depth is determined by the number of registers (along with the number of PC registers)
- W_depth 20 means 20 x 32 (W_width) x (# register per thread) number of registers are needed
- W_depth 20 also means at least 20 x PC registers is needed.



Georgia Tech

Finding W_depth

- Strong correlation factor of W_depth is memory latency.
- In the dot product example, assume memory latency is 200 cycles.
- Case 1) 1 comp, 1 memory (dot product):
  - To hide 200 cycles, 200/ (1 comp + 1 memory) = 100 warps are needed
- Case 2) If we have 1 memory instruction per 4 compute instructions
  - To hide 200 cycles 200 / (1+4) = 40 warps are needed

Decision Factors for the Number of SMs

- Previous example: 500 FMA units
- 1 warp x 16 SMs vs. 2 warps x 8 SMs
- Large and fewer SMs vs. small and many SMs
- Cache and registers also need to be split.
- Large cache with fewer SMs vs. small cache with many SMs
- Large cache increases cache access time, but large cache can increase cache hits among multiple CUDA blocks
- Sub-core can also be a design decision factor

Many of these decisions require the analysis of trade-off between size vs. time

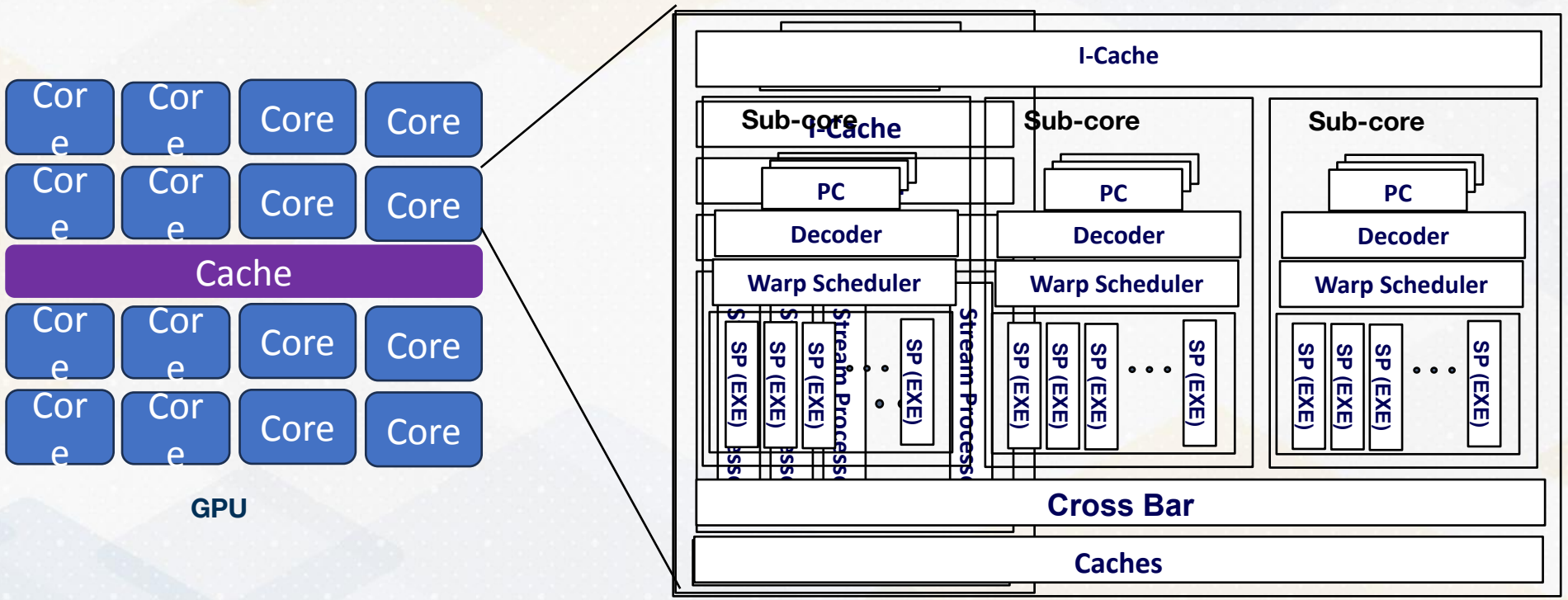
slido

Please download and install the Slido app on all computers you use

What is most strongly correlated with deciding W_width for the register file sizes

ⓘ Start presenting to display the poll results on this slide.

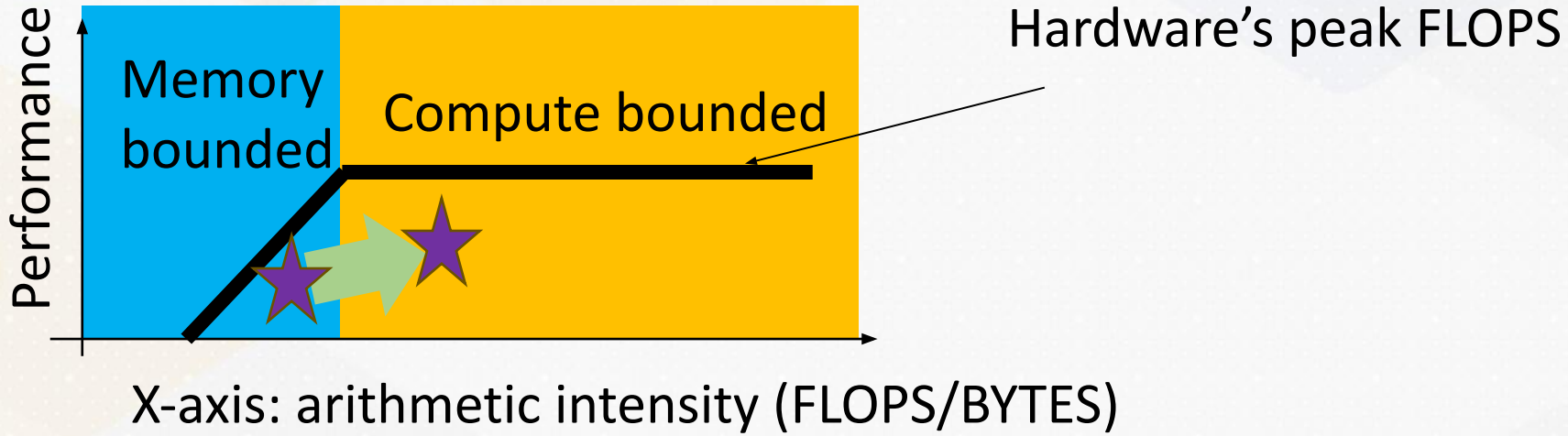
Sub-core



Resource management

Georgia Tech

Roofline Model



- A Visual performance model to determine whether an application (or a processor) is limited by the compute bandwidth or memory bandwidth
- Vector sum example: 2 Bytes per 1 FLOPS □ arithmetic intensity : 0.5
- Another example: sum +=x[i] *x[i]*y[i]*y[i]; □ 2 Bytes per 4 FLOPS □ arithmetic intensity: 2

Williams, S., Waterman, A., & Patterson, D. (2009). Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM, 52*(4), 65–76. https://doi.org/10.1145/1498765.1498785

Georgia Tech

CPI (Cycle per Instruction) Computation

- CPI = CPI<steady state > + CPI<event1> + CPI<event2> + CPU<event3> …
- CPI<steady state>: sustainable performance without any miss events
- Example: 5-stage in-order processor

Assumption: CPI<stead state> = 1. CPI<br misprediction> =3 , CPI<cache_miss> = 5

2% instructions has branch misprediction. 5% instructions has cache misses. Average CPI?

Answer: 1 + 0.02*3 + 0.05*5 = 1.31

Easy to compute the average performance.

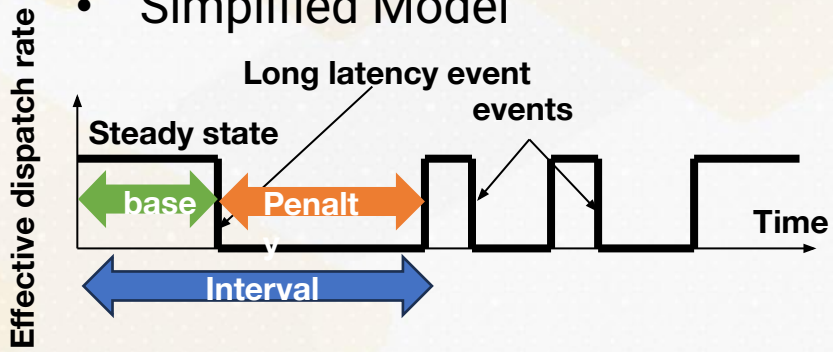
All penalties are assumed to be serialized.

CPI Computation for Multi-threading

- CPI<ideal multi threading> = CPI<single thread>/W_depth
- W_depth: the number of warps that can be scheduled during the stall cycles
- CPI<multi threading> = CPI<ideal multi threading> + CPI <resource contention>
- Resource contention: MSHR (# of memory misses), busy states of execution units, DRAM bandwidth etc.
- GPU is modeled for multi-threading

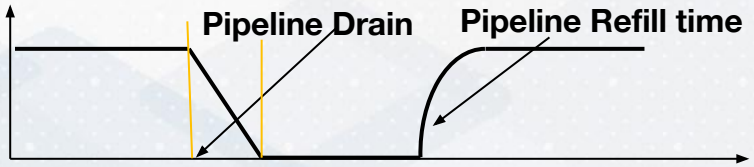
Interval Based Modeling: Extension of CPI Modeling

- Simplified Model



- Interval: one steady state and one long latency event

- First-ordering Modeling [ISCA 2004] : Model the effect of pipeline drain and refill time
- Depending on the events (branch misprediction vs. cache miss): pipeline drain shape is different



Karkhanis, T. S., & Smith, J. E. (2004). A first-order superscalar processor model. In *Proceedings of the 31st Annual International Symposium on Computer Architecture* (pp. 338-349). Munich, Germany. https://doi.org/10.1109/ISCA.2004.1310786

Georgia Tech

Applying Interval Analysis on GPUs

- Naïve approach: consider GPU as just a multi-threading processor
- Major performance differences between GPU and multi-threading processor
  - Branch divergence: not all warps are active; some part of branch code is serialized.
  - Memory divergence: memory latency can be significantly different depending on memory is coalesced or uncoalesced
- Newer models improve the performance models by modeling sub-core, sectored cache, and other resource contentions more accurately.

slido

Please download and install the Slido app on all computers you use

Using the roofline model, we want to guide the performance optimization directions. After plotting into the roofline model, it turns out that the application is compute bounded. What would be a better approach to improve the application?

ⓘ Start presenting to display the poll results on this slide.

Summary

- CPU and GPU cycle level simulation techniques are reviewed.
- Analytical model provides the first order of processor design parameters or application performance analysis.
- CPI computation was reviewed.
- Roofline was introduced.
- Compute bounded or memory bandwidth bounded is the most critical factor.

Accelerating Simulation

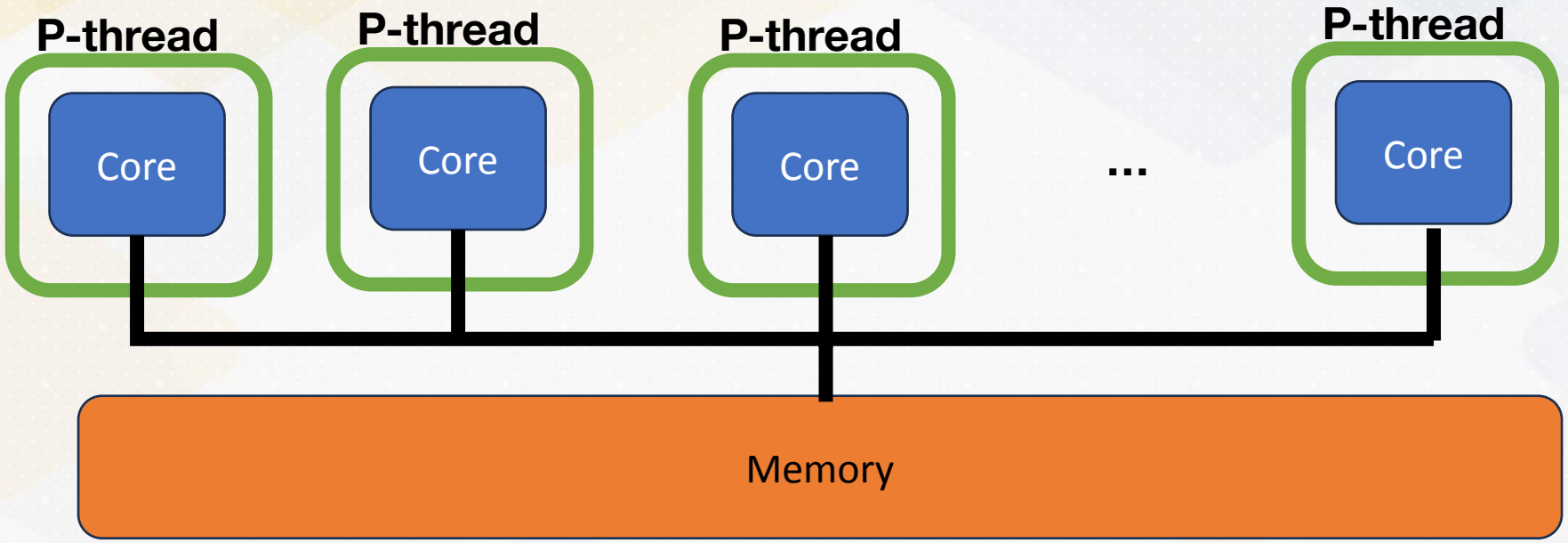
Challenges of Cycle Level Simulation

- Cycle level simulation takes too long.
- If a simulation speed is 10KIPS (10 instructions per sec), simulating 1B instructions (1 sec in a real machine) takes ~= 28 hours
- ML workload takes hours: 10 hours of ML workloads □ 100 years of simulation

Accelerating Simulations

- Accelerating simulation itself
  - Parallelizing the simulator
  - Event driven simulation
  - Simplifying the model
  - Sampling
  - Statistical Modeling
  - ML based Modeling
- Reducing the workloads
  - Micro benchmarks
  - Reducing the workload size
  - Create small representative workloads

Parallelizing Simulator

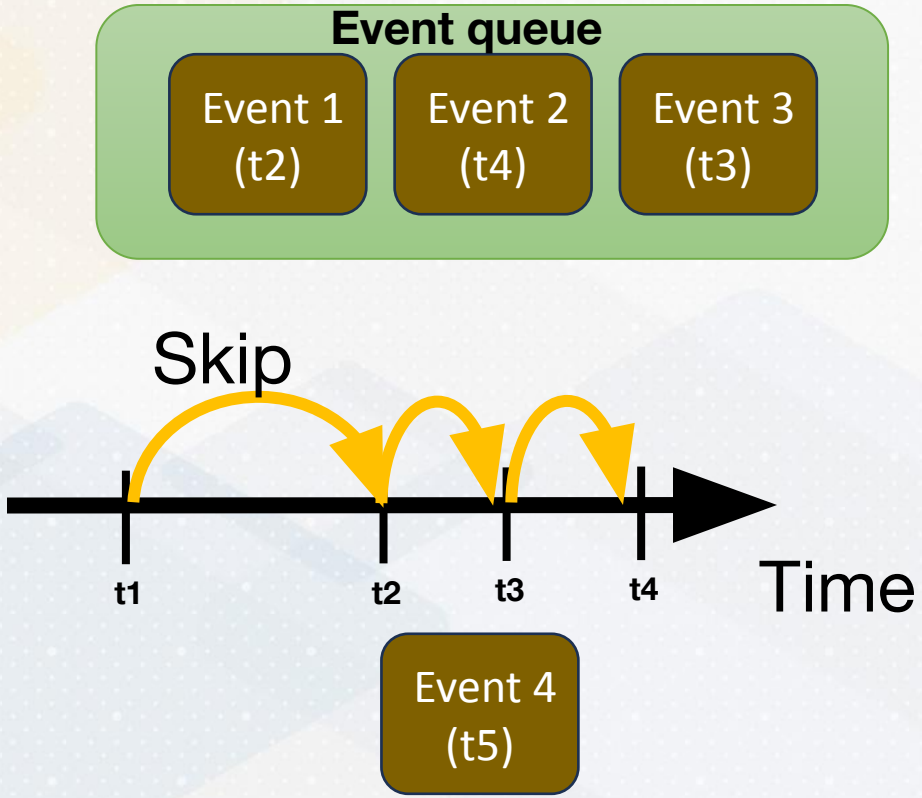


- Each P-thread simulates a core. Cores are typically run in parallel so it can be parallelized.
- Memory/Noc (network on chip) needs to be communicated.
- Bottlenecks on the memory

Georgia Tech

Event Driven Simulation

- Instead of operating by cycle, simulator is operated with events.
- Speeding up long latency operation simulations is possible.
- New event is inserted into the event queue.



Georgia Tech

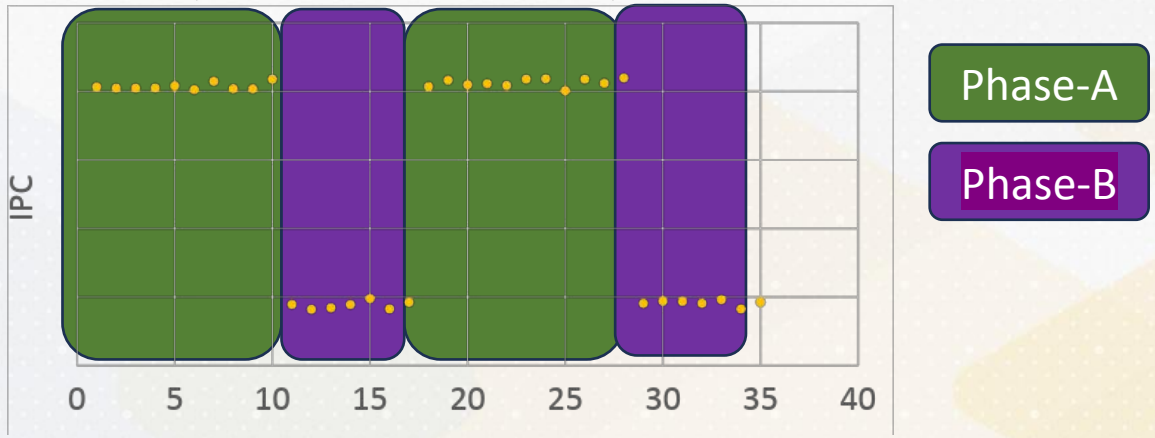
Simplifying Models

- Depending on which one to model, we could simplify the modeling.
- Non-critical components are modeled based on the average throughputs and average latency.
- Detailed modeling is needed for modeling any resource contentions.
- Option #1: simplify the pipeline: instead of modeling instruction fetch/decode/execution, we could assume IPC = issue width. Model only caches and memory. E.g.) z-sim[1] CPU simulator simplifies the core pipeline.
- Option #2: simplify the memory model: assume memory system has a fixed latency

[1] Sánchez, D., & Kozyrakis, C. (2013). ZSim: Fast and accurate microarchitectural simulation of thousand-core systems. In *Proceedings of the 40th Annual International Symposium on Computer Architecture, ISCA 2013* (pp. 475-486). ACM.

Sampling Techniques

- Random Sampling: simple; randomly choose where to simulate.
  - Execution driven: fast forward the part that won't be simulated or use a check-point based method
  - Trace driven: generate traces only for simulating sessions
- Handling state information: e.g.) cache, branch predictors □ warm up time is needed; or running time is sufficiently long enough to overcome
- Program phase based sampling (e.g., Simpoint[1])



[1] Sherwood, T., Perelman, E., Hamerly, G., & Calder, B. (2002). Automatically characterizing large scale program behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*.

Georgia Tech

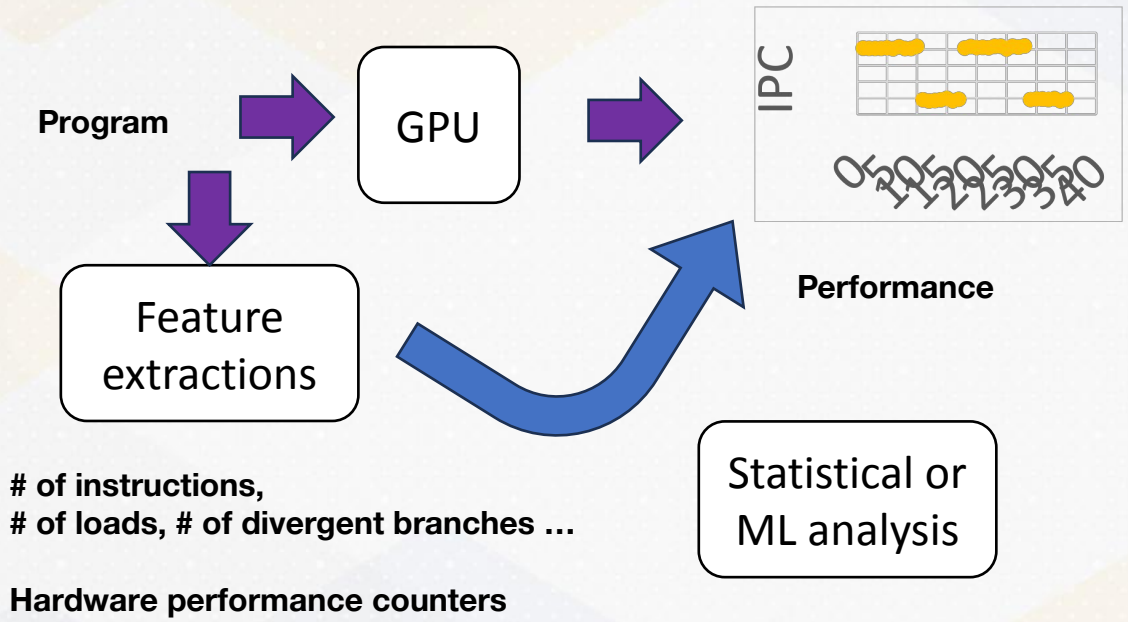
GPU Sampling Techniques

- Program phase based sampling is non-trivial
  - Execution length for one single thread is too short.
- Solutions:
  - CUDA block level sampling: simulate 100 CUDA blocks instead of 1000s of CUDA blocks
  - Kernel level sampling: simulate 1-2 kernels instead of 10s of kernels
  - Warp level sampling: reduce the number of warps to simulate

Reducing Workloads

- Method #1: reducing iteration counts
  - E.g.) Machine learning workloads run 1000 iterations. Each iteration shows very similar characteristics; instead of 1000 iteration, iterates only once.
- Method #2: reducing input sizes
  - E.g.) Graph algorithm traverses for 1B nodes graph algorithm traverses for 1M nodes
- Method #3: identify dominant kernels
  - E.g.) A program has 100s of functions. One or two functions dominate 90% of application time; change the application to run only those two functions.

Data-Driven Modeling



- E.g.) Cycle count = C1 x # instructions ^(exp1) + C2 x # FP instructions^(exp2) + C3 x # memory instructions^(exp3) etc.

Georgia Tech

slido

Please download and install the Slido app on all computers you use

What techniques are the most helpful if we simulate Project #2's homework solution for GPU architecture simulation? Choose all apply.

ⓘ Start presenting to display the poll results on this slide.

Summary

- Recap of techniques for Accelerating Simulation Speed
- Emphasis on Sampling and Workload Reduction