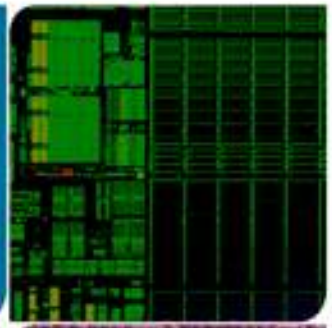
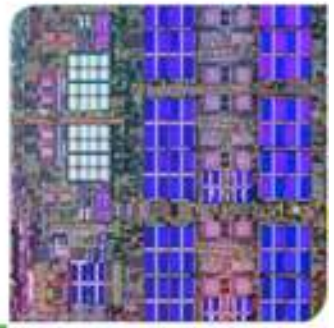
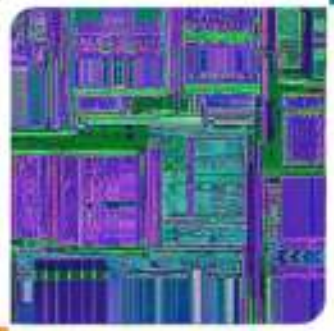





An Analytical Model for a GPU Architecture with Memory-level and Thread-level Parallelism Awareness

Sunpyo Hong, Hyesoon Kim

Georgia Tech comparch

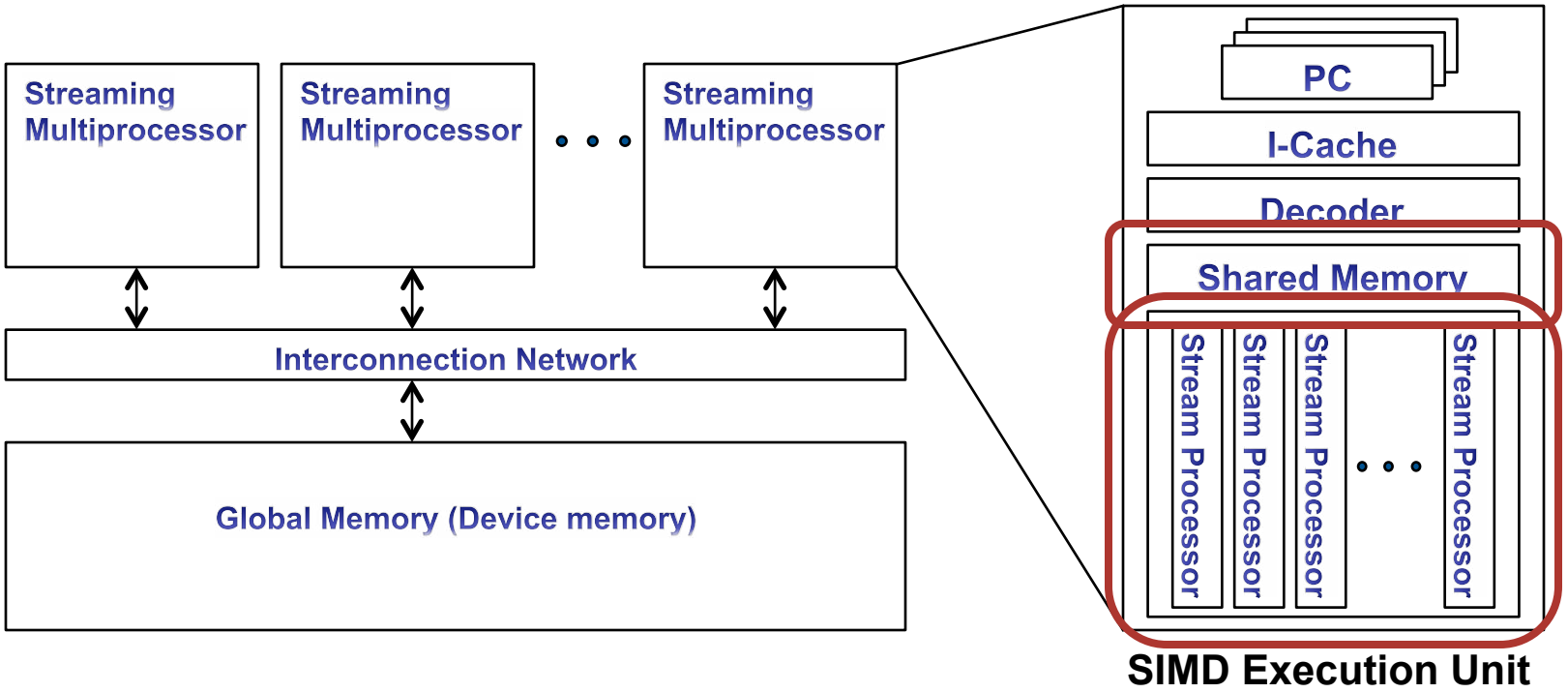


Outline

- **Background**
- Model
- Results
- Conclusion

Georgia Tech comparch

Overview of GPU Architecture



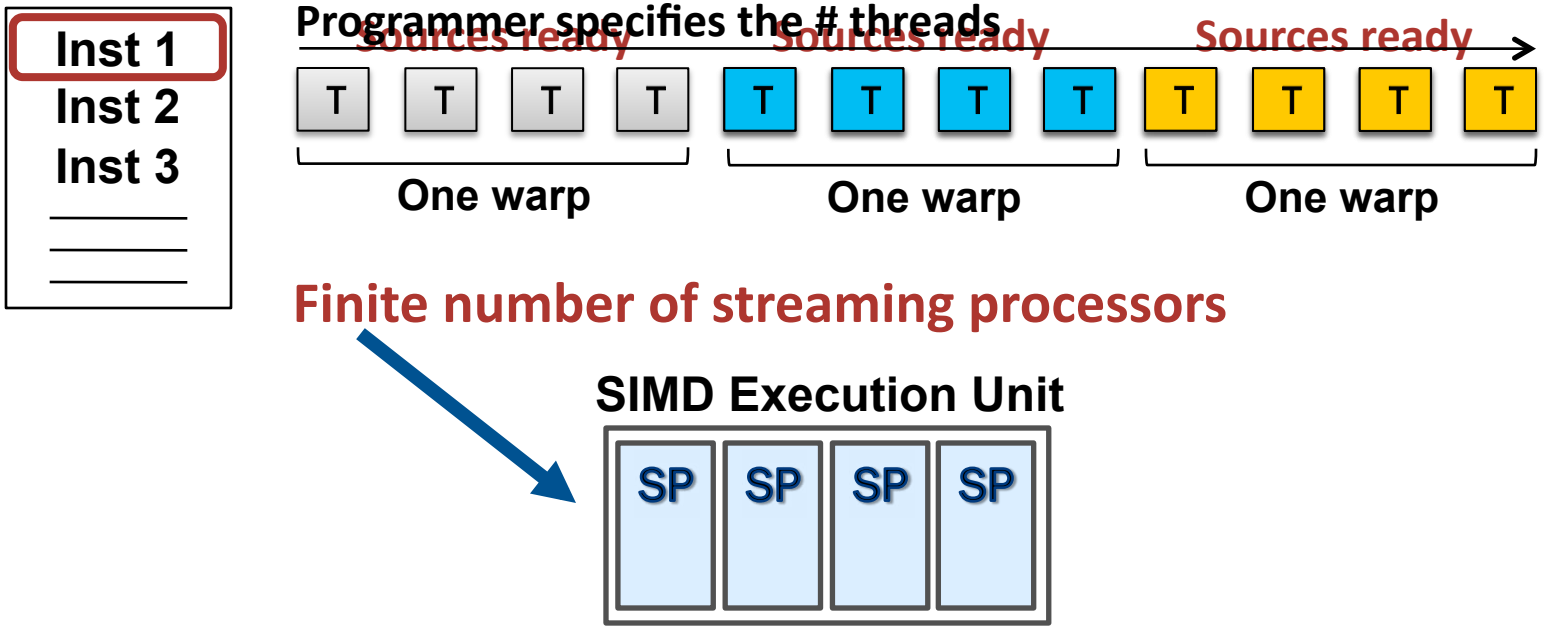
- Software-managed cache
- SIMD Execution Unit inside SM

Georgia Tech comparch

Warp

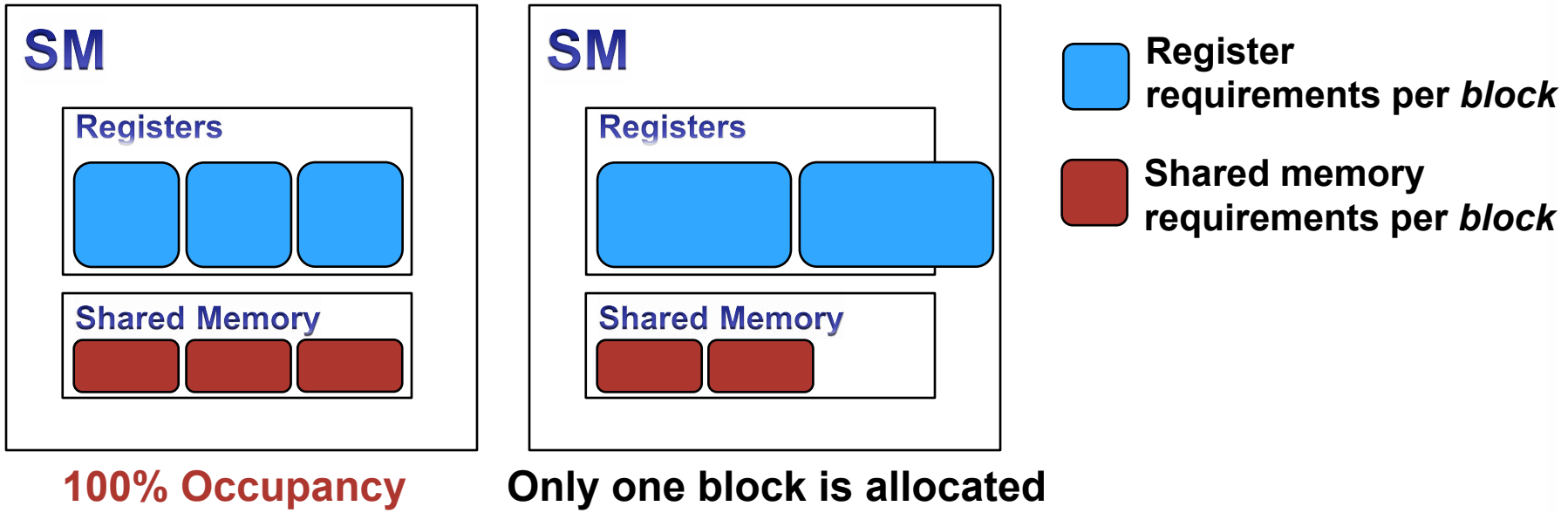
- **Warp is the basic unit of execution**
  - A group of threads (e.g. 32 threads for the Tesla GPU architecture)

Warp Execution



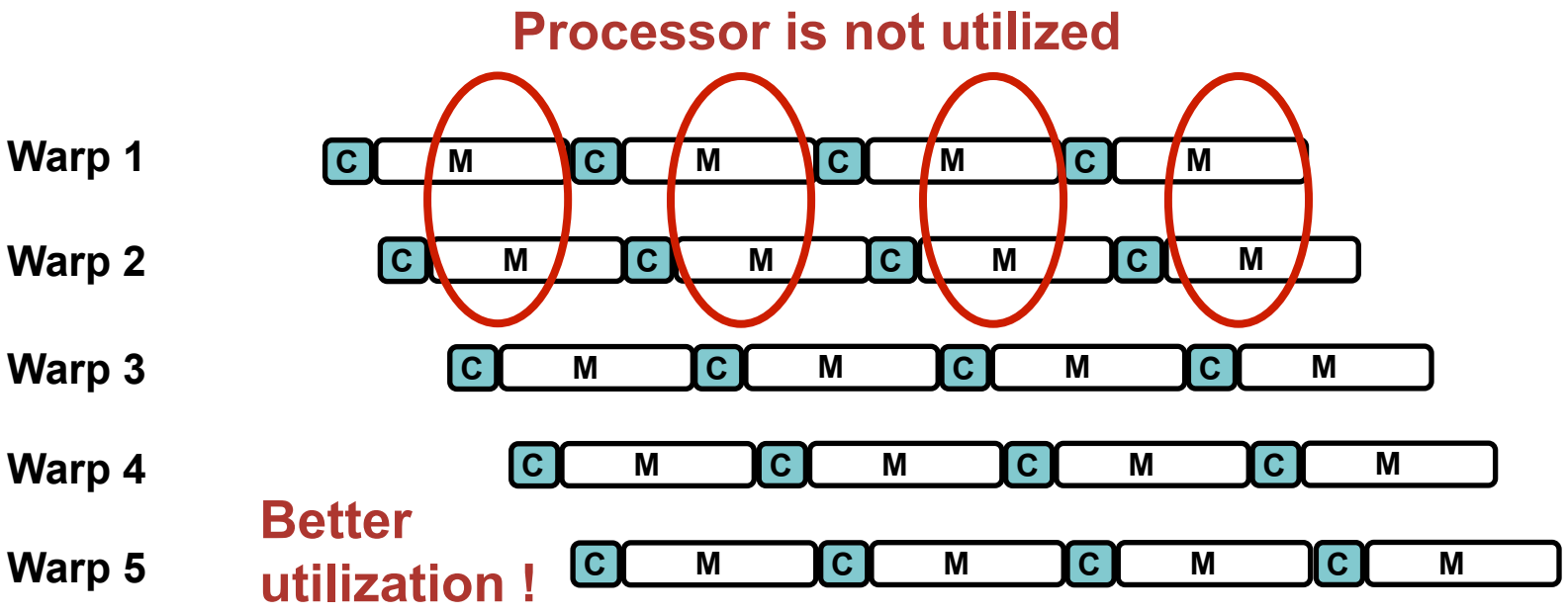
Occupancy

- Shows how many warps are assigned to the SM
- Warps are assigned at block granularity
- Programmer specifies the number of threads per block

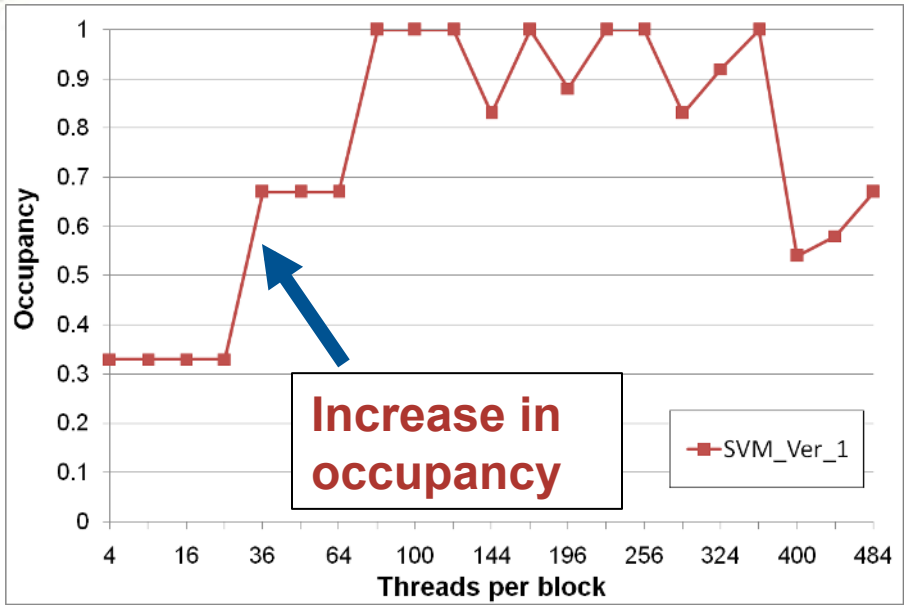


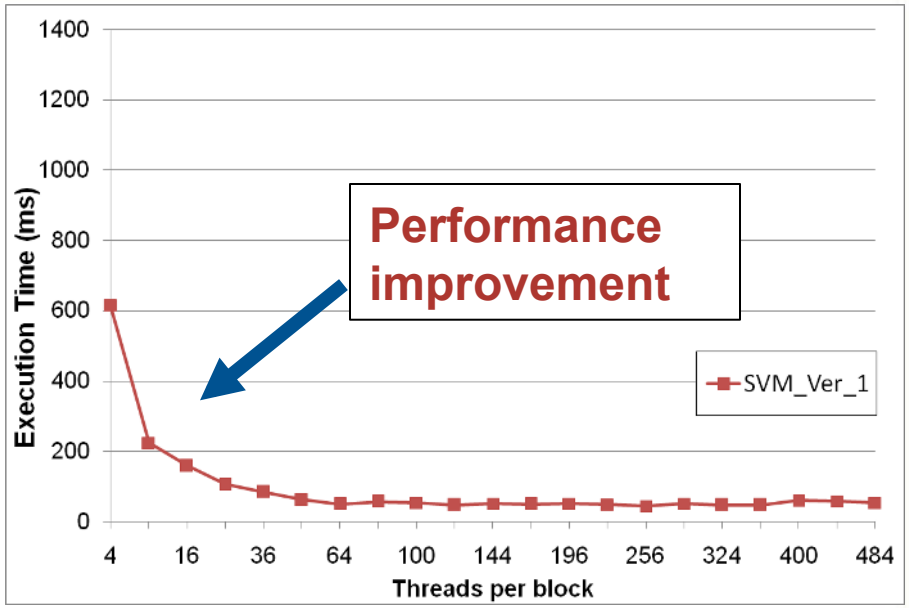
Higher Occupancy

- Better processor utilization
- Hide the memory latency



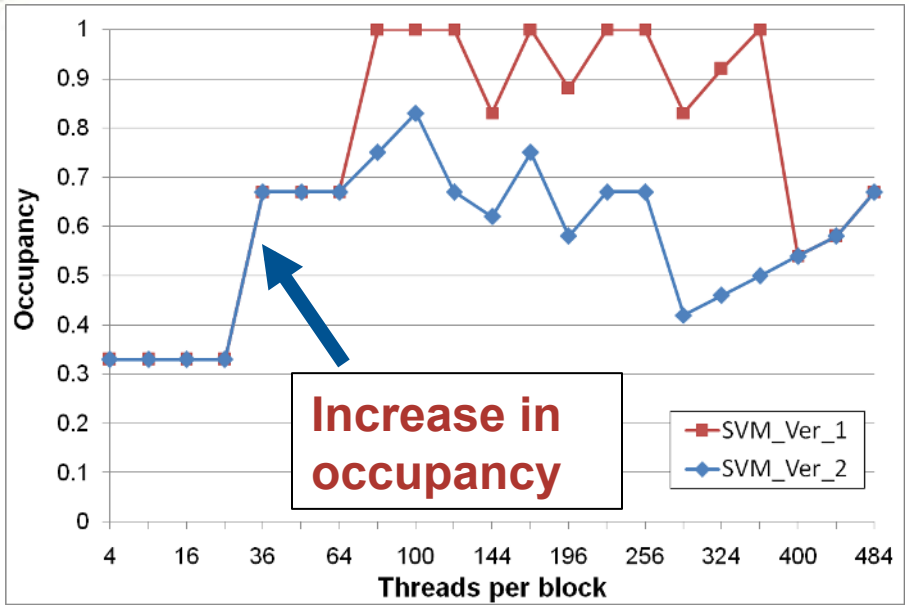
High Occupancy = High Performance ?

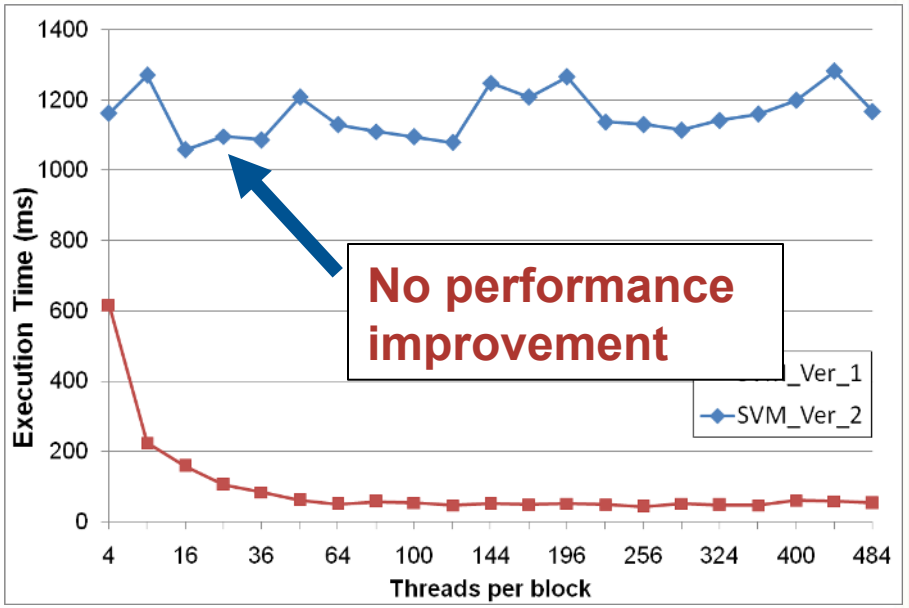




- Programmers try to optimize programs for occupancy

High Occupancy ≠ High Performance





- **Programmers try to optimize programs for occupancy**
- No performance improvement from increased occupancy

Georgia Tech comparch



Motivation of the Work

- **Propose analytical model that can estimate performance**
- Why ?
  - Optimizing for occupancy may not have impact on the performance
  - Occupancy does not consider the application behavior
  - To understand the GPU performance and bottlenecks
- Other benefits
  - Prediction for faster performance simulation

Georgia Tech comparch

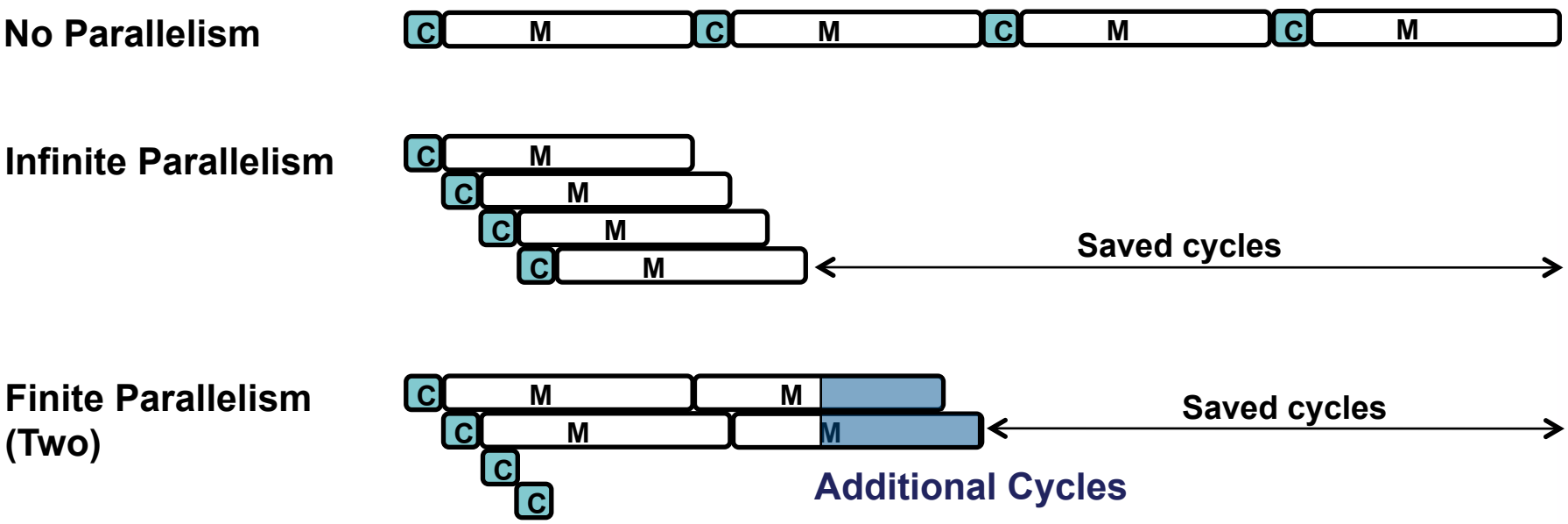


Outline

- Background
- **Model**
- Results
- Conclusion

How is Performance Determined ?

- **Memory accesses can be overlapped between warps**
  - Performance **significantly** depends on the memory-level parallelism

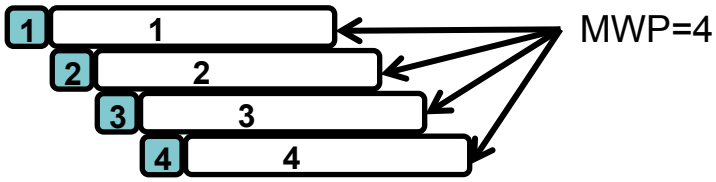


- Performance can be **predicted** by knowing the amount of memory-level parallelism

Georgia Tech comparch

MWP

- Memory Warp Parallelism
- **Metric of memory-level parallelism**



Four warps are overlapped during memory accesses

- Maximum number of warps that can overlap memory accesses
- Tightly coupled with DRAM system
  - Memory latency, bandwidth, memory access type

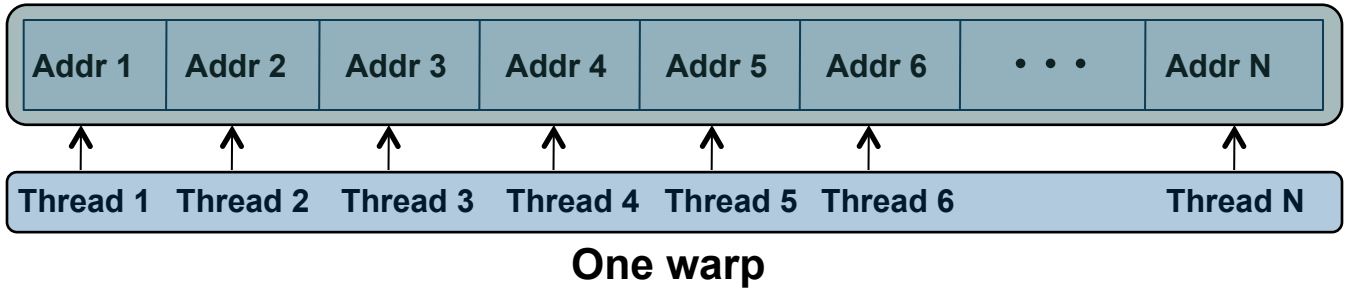
Georgia Tech comparch

Memory Access Type

One warp generates a memory request

One memory transaction

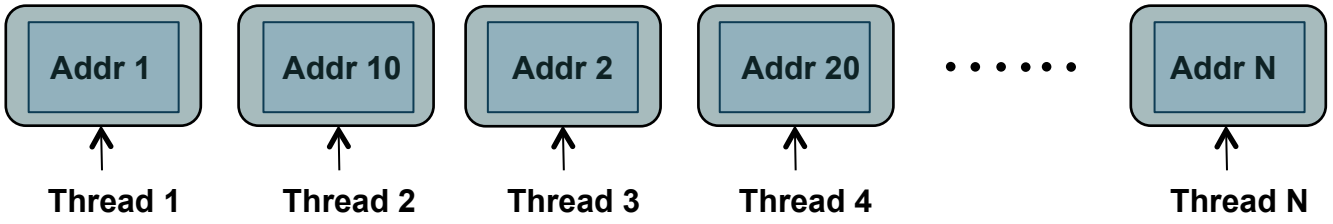
Coalesced memory access type



One warp

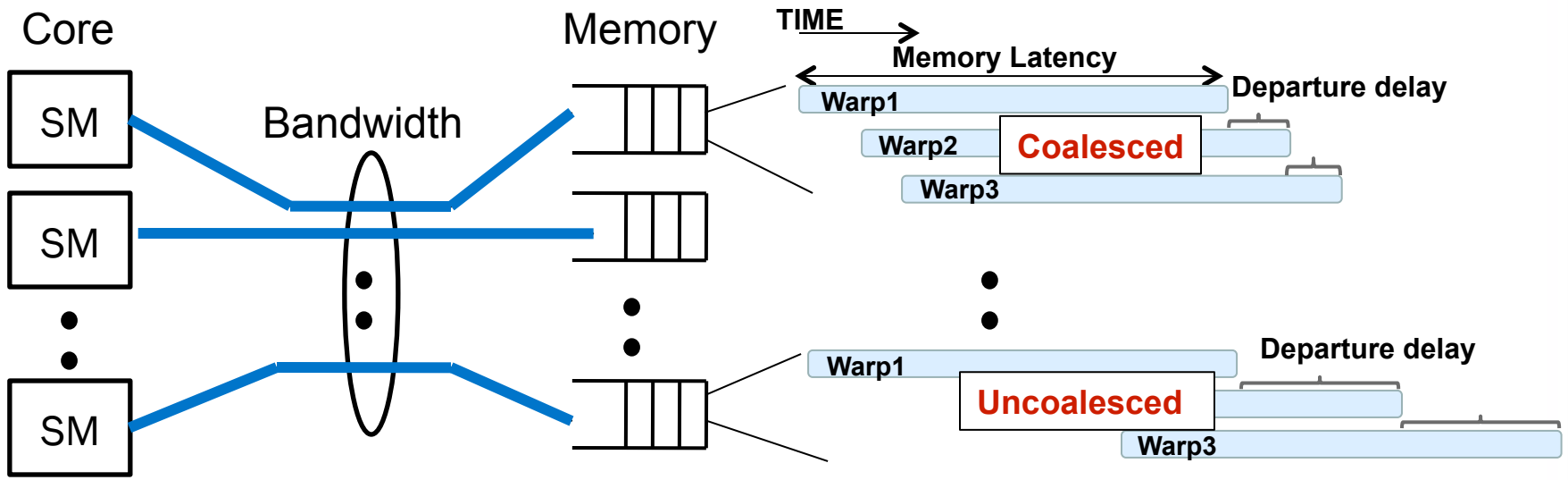
Uncoalesced memory access type

Multiple memory transactions



- More processing cycles for the uncoalesced case

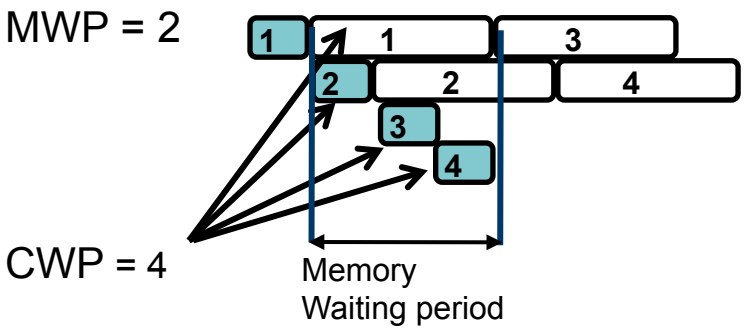
Memory System Model



- Each SM has a simple queue and consumes an equal bandwidth
- MWP is determined by #Active SMs, #Active warps, Bandwidth, Types of memory accesses (Coalesced, Uncoalesced)

CWP

- Computation Warp Parallelism
- Analogous concept to MWP

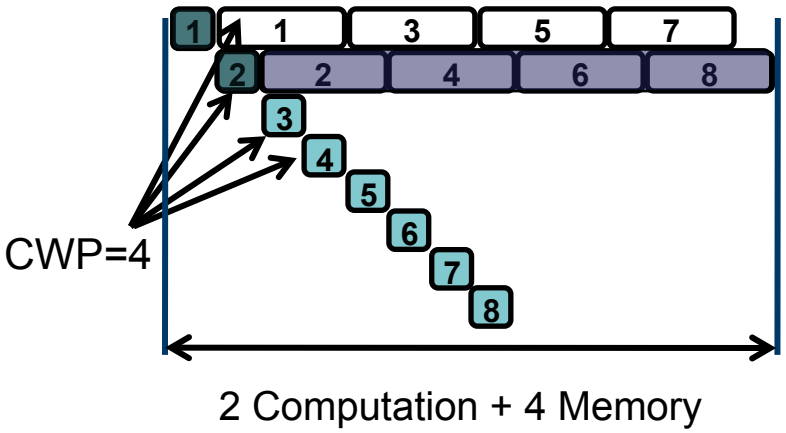


- Number of warps that execute instructions during one memory access period
- **Three scenarios can occur depending on the MWP and CWP relationship**

Georgia Tech comparch

(1) When MWP ≤ CWP

MWP=2, N = 8 (Number of warps)



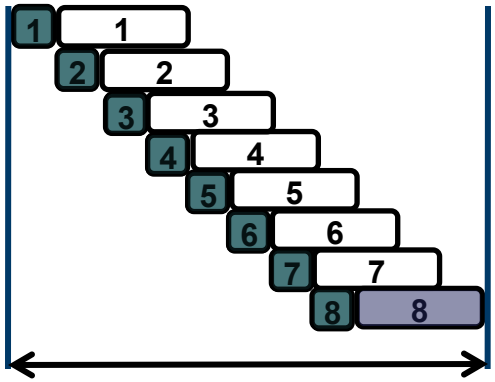
- Computation cycles are hidden by memory waiting periods
- Overall performance is dominated by the memory cycles

$$Exec\_cycles = Mem\_cycles \times \frac{N}{MWP} + Comp\_p \times MWP$$

(MWP=2, N = 8)

(2) When MWP > CWP

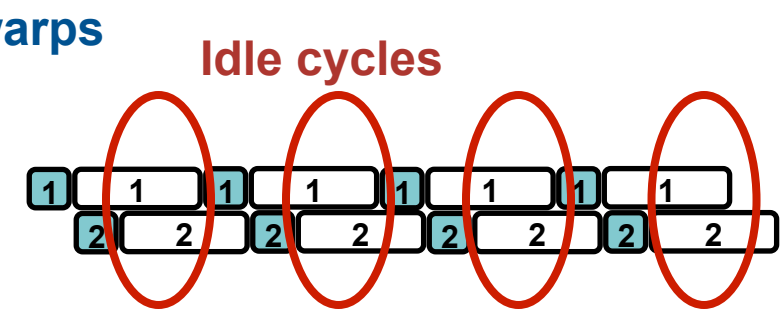
MWP=8 N = 8 (Number of warps)



- Memory accesses are mostly hidden due to high MWP
- Overall performance is dominated by the computation cycles

$$Exec\_cycles = Mem\_p + Comp\_cycles \times N$$ (MWP=8, N = 8)

(3) Not Enough Warps



- Increasing the number of warps will increase the processor utilization
- MWP is limited by the number of active warps per SM
- The analytical model is inside the paper

Georgia Tech comparch



Outline

- Background
- Model
- **Results**
- Conclusion



Evaluation Methodology

- Micro benchmarks are devised to obtain the memory parameters
  - Memory latency, departure delay
- Model inputs
  - Number of instructions, memory type, thread/block configuration, memory parameters
- Merge benchmarks
  - Execution time, CPI compared

Evaluated Systems

GPU Model	8800GTX	FX5600	8800GT	GTX280
Number of SMs	16	16	14	30
(SP) Processor Cores	128	128	112	240
Processor Clock	1.35 GHz	1.35GHz	1.5 GHz	1.3 GHz
Memory Size	768 MB	1.5 GB	512 MB	1 GB
Memory Bandwidth	86.4 GB/s	76.8 GB/s	57.6 GB/s	141.7 GB/s
Computing Version	1	1	1.1	1.3

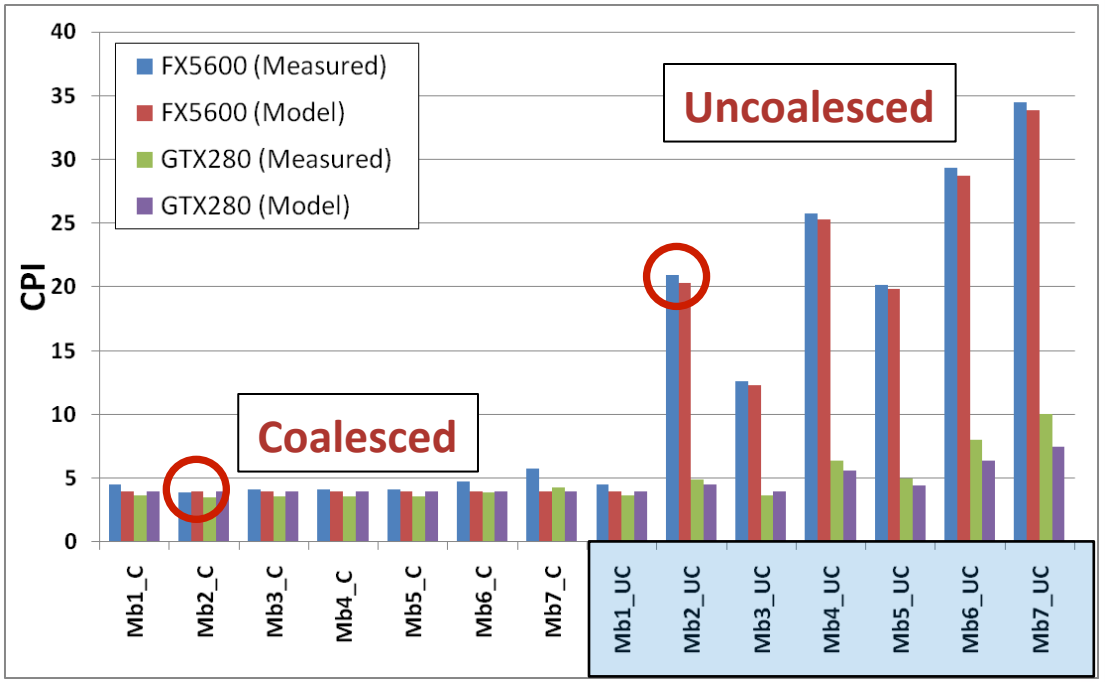
Georgia Tech comparch

Micro Benchmarks

- Ratio of memory to computation instructions is varied
- Coalesced, uncoalesced memory types

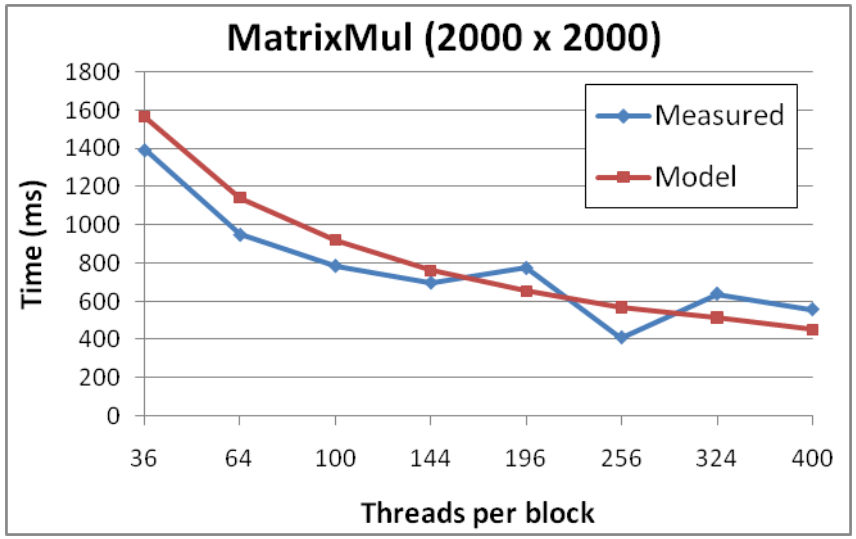
Memory Model Parameters

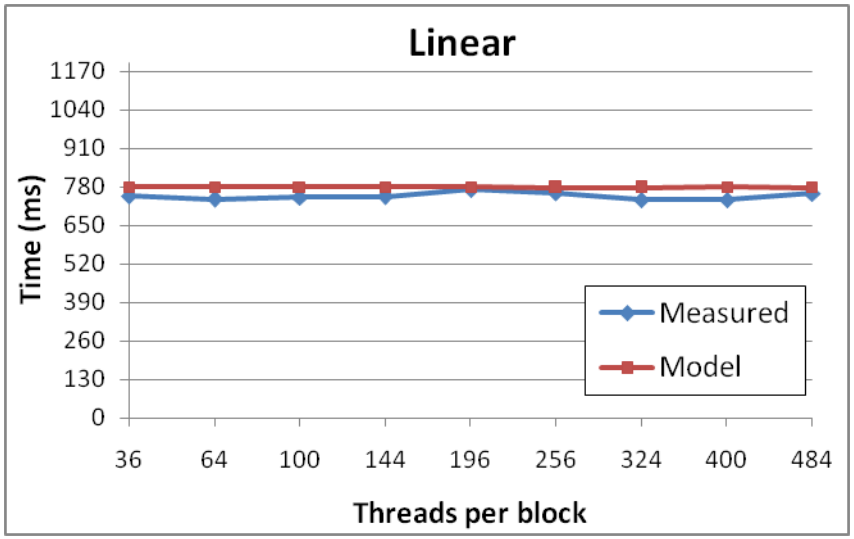
Parameters	FX5600	GTX280
Memory latency	420	450
Departure delay uncoalesced	10	40
Departure delay coalesced	4	4



Georgia Tech comparch

Merge Benchmarks

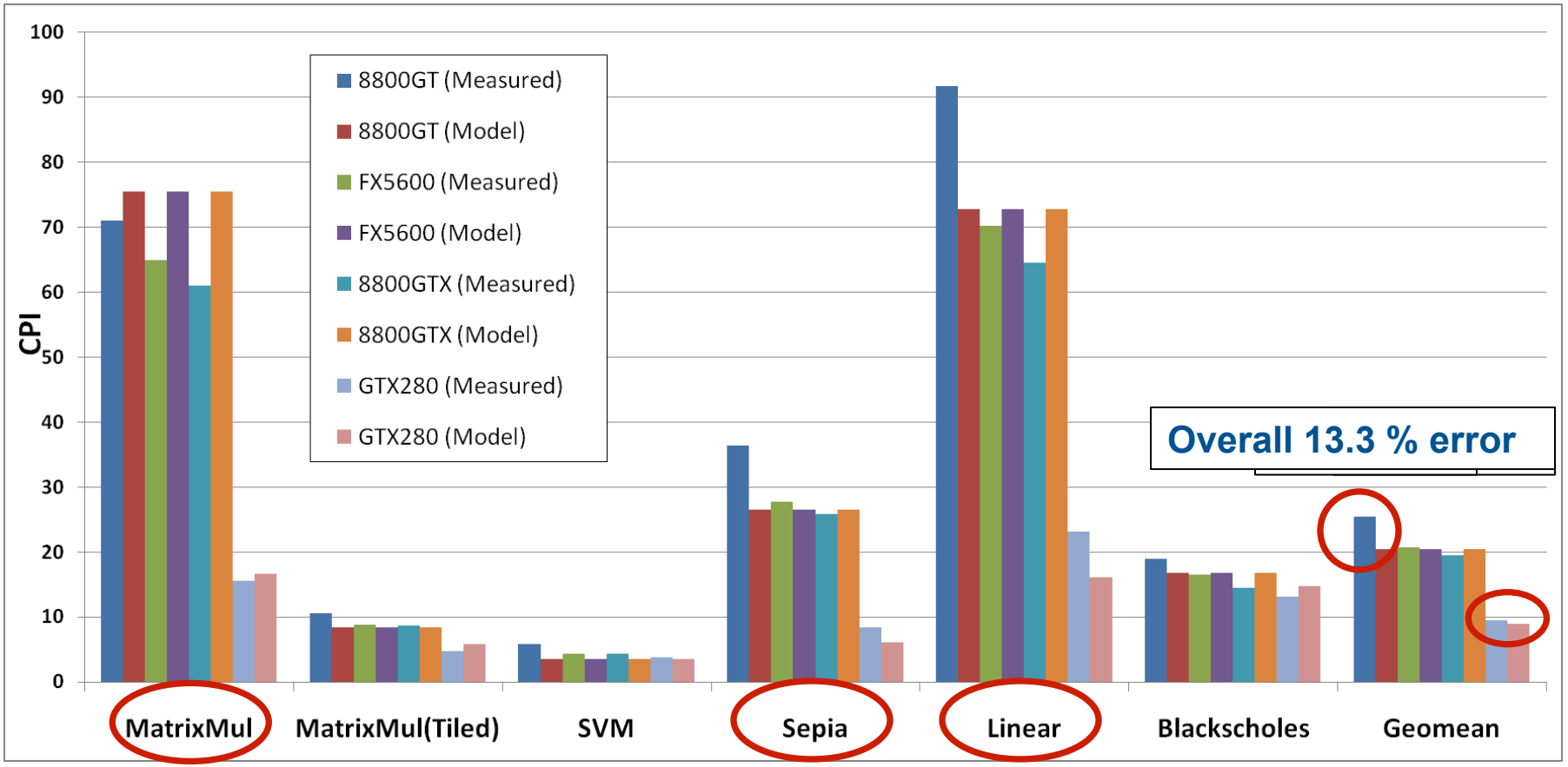




- Merge benchmark performance estimation
- The prediction closely follows the actual execution
  - **Two types** of execution behavior are predicted

Georgia Tech comparch

CPI Comparison



- CPI comparison between the model and the actual execution

Georgia Tech comparch



Outline

- Background
- Model
- Results
- **Conclusion**

Georgia Tech comparch



Conclusions

- Introduced **MWP, CWP** metrics that determine the performance
- **Simplified** the complex memory operations
- Prediction
  - For Micro benchmarks, the prediction error is 5.4%
  - For Merge benchmarks, the prediction error is 13.3%
- First analytical model that calculates the **execution cycles** for GPU
- **Better** understanding of the performance aspects of the GPU architecture
- Future research
  - Help providing more systematic approaches for **optimizing** GPGPU applications

Georgia Tech comparch



Thank you



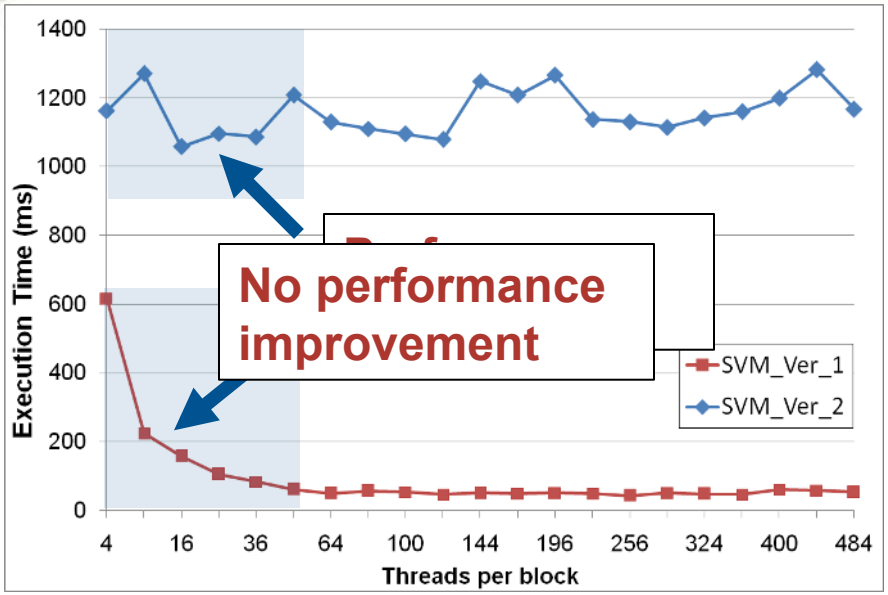
Questions ?

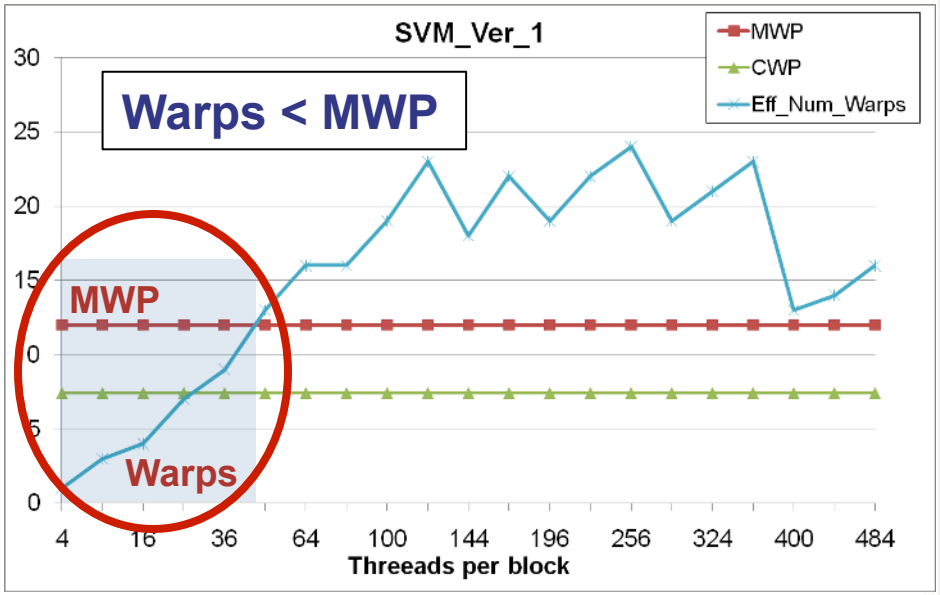
Georgia Tech comparch

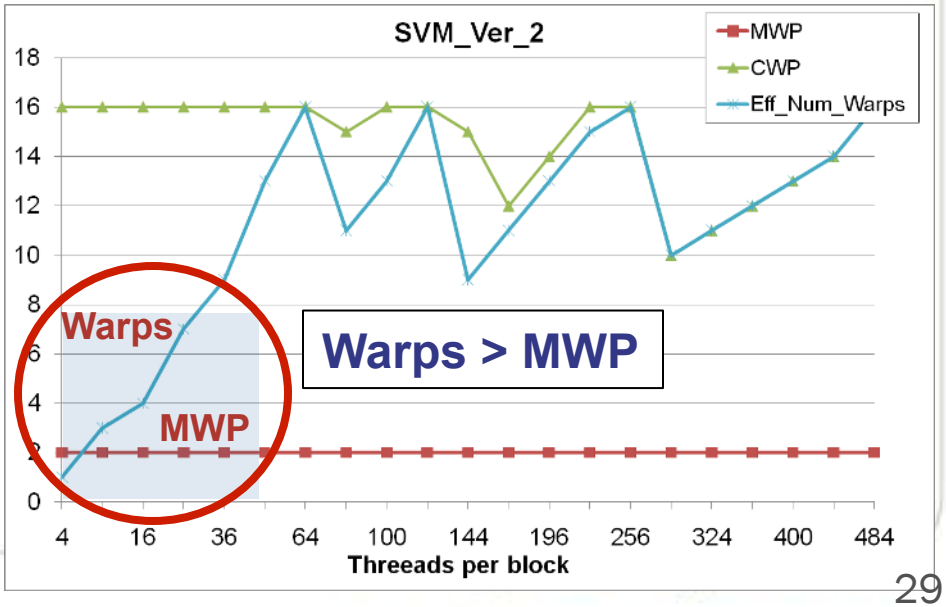


Backup Slides

Insights on MWP (Motivation Example)









Programming

- The model provides the **upper limit** of # of active warps for a given application that **fully utilizes** the processor resources
  - **Increasing** the # of warps when N is **smaller** than MWP, CWP
  - **Trade-off**
    - More register allocation vs. More computation instructions
    - Traditionally, if the optimization on the thread **decreases the occupancy**, that optimization is **unlikely to be performed**
- Calculated CPI value indicates how **optimized** the code is
  - CPI per warp near 4 is the upper-bound
  - However, if the model **predicts** that higher occupancy does not improve the performance, then that optimization can be **applied** without **performance degradation**
- We $CPI = \frac{Exec\_cycles\_app}{\#Total\_insts \times \frac{\#Threads\_per\_block}{\#Threads\_per\_warp} \times \frac{\#Blocks}{\#Active\_SMs}}$ pach
  - By using the metrics that we proposed in this work

Georgia Tech comparch



Limitations of the Model

- Cache misses
  - Current analytical model does not consider cache miss penalties
- Graphics Applications
  - Not modeling texture cache, texture processing
- Divergent branches
  - Double counting the number of instructions in both path
  - Provides the upper limit for the execution time
- Data transfer time between CPU and GPU
  - The analytical work models the GPU kernel execution only
- Considers total average execution time
  - No time-phase behavior



How to use the model (I)

- Inputs to the model
  - Thread/block configuration
  - Register/shared memory usage
  - Number of Instructions
  - Memory access type

Programmer specifies in the source code
Available in the CUDA compiler output (.cubin file)

Source code analysis
PTX file (compiler output)

- Micro benchmarks
  - Exact number of instructions for different arithmetic intensity is known
- Merge benchmarks
  - Source and PTX (virtual ISA) analysis
  - Currently, GPU emulator is available
  - Dynamic number of PTX instructions is calculated

Georgia Tech comparch

How to use the model (II)

- Inputs to the model
  - Thread/block configuration
  - Register/shared memory usage
  - Number of Instructions
  - Memory access type

Analyzing memory access pattern

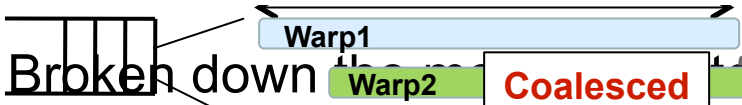
- Outputs of the model
  - Estimated execution time (cycles), CPI per warp granularity
  - D… access type, and the…

$$CPI = \frac{Exec\_cycles\_app}{\#Total\_insts \times \frac{\#Threads\_per\_block}{\#Threads\_per\_warp} \times \frac{\#Blocks}{\#Active\_SMs}}$$

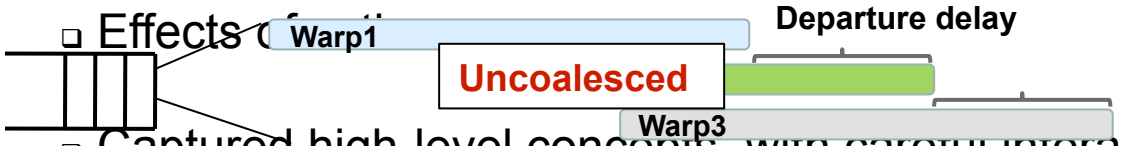
Georgia Tech comparch

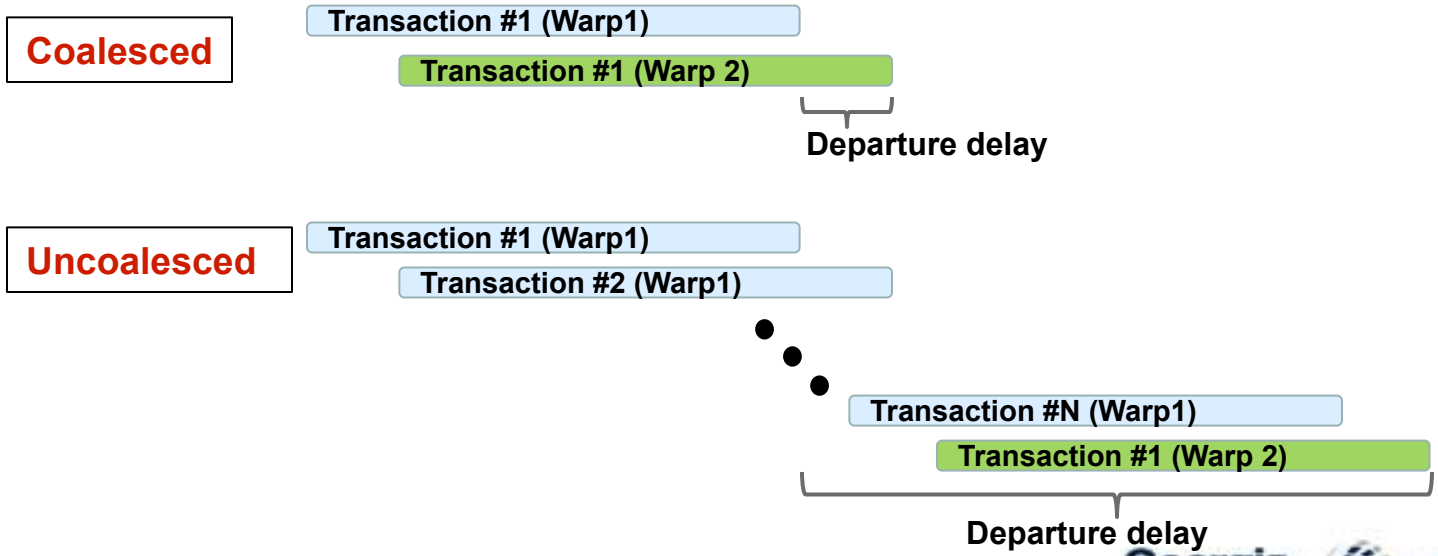
Memory System

$$MWP = MIN(MWP\_Without\_BW, MWP\_peak\_BW, N)$$



- Broken down the [illegible] from high-level view
  - Maximum po[illegible]ndwidth consideration
  - MWP with bandwidth consideration (Considers #warps, # Transactions, …)
  - Effects o[illegible]
  - Captured high-level concepts with careful interactions

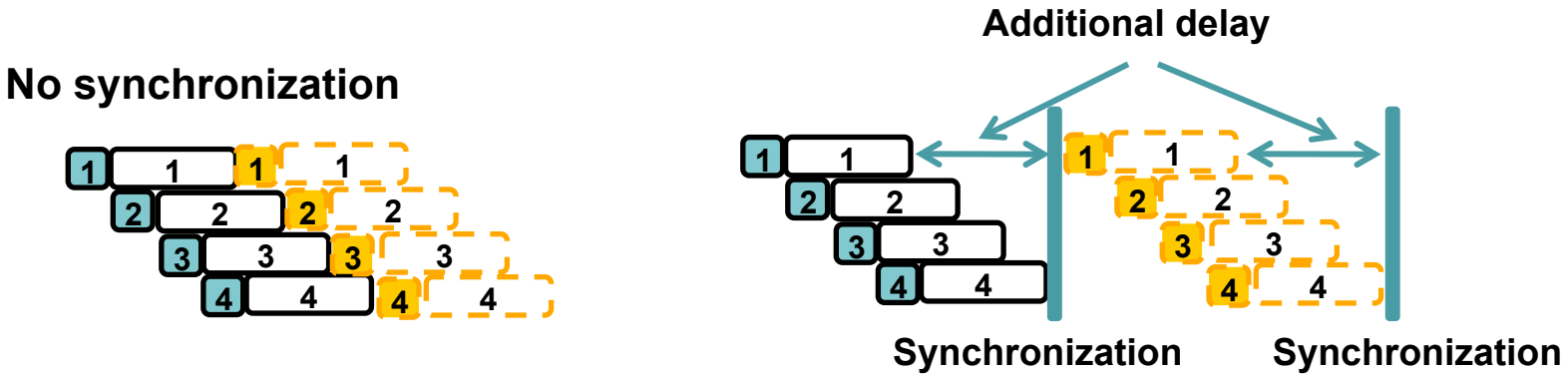




Georgia Tech comparch

Synchronization effects

- Barrier instruction causes extra waiting cycles
- Warps inside one SM are synchronized



- Extra cycles are calculated by knowing the value of MWP