

Performance Analysis Models and Tools

Hyesoon Kim



**Georgia
Tech**



comparch



Performance Impact Factors

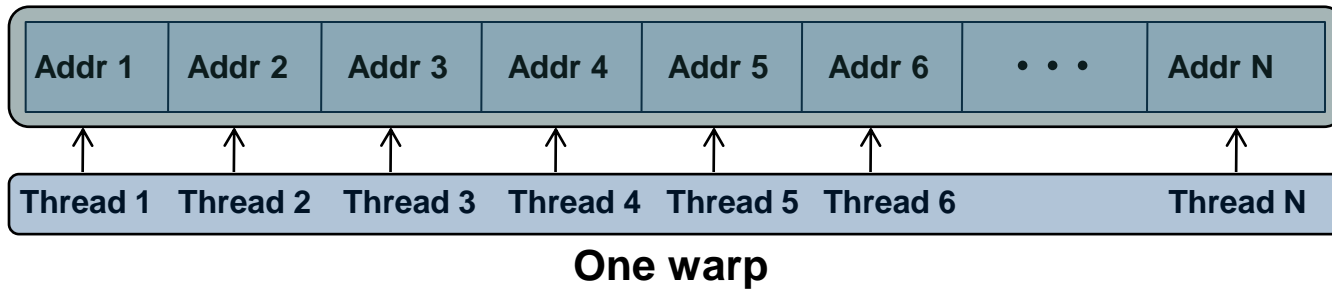
- Communication cost between CPU and GPU
- Occupancy
- Uncoalesced global memory access
- Divergent branches
- Bank conflicts (shared memory/global memory)
- Integer operations



Memory Access Type

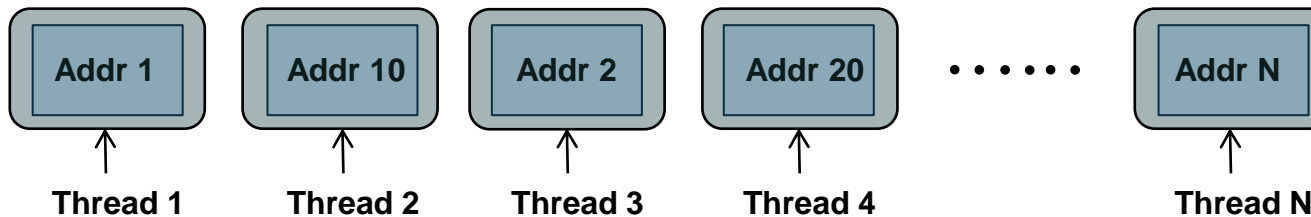
One warp generates a memory request

Coalesced memory access type



One memory transaction

Uncoalesced memory access type

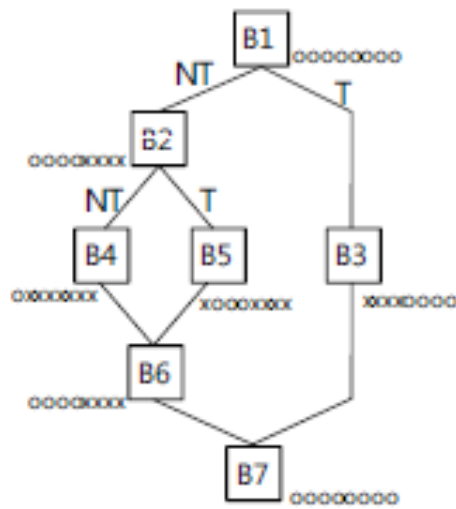


Multiple memory transactions

- More processing cycles for the uncoalesced case



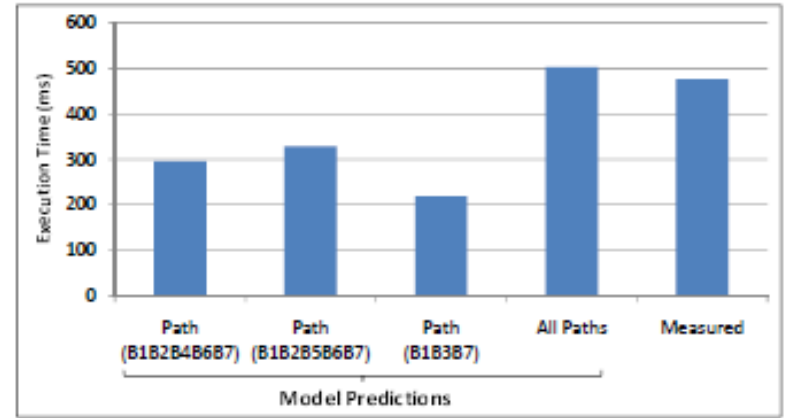
Divergent Branches



O: Active Thread
X: Non-Active Thread

```

int mod = threadIdx.x & 31; // modulus
if (mod < A)
  {
    Memory_load
    FP Operations
  }
else
  {
    Memory_load
    if (mod < B)
      {
        FP Operations
      }
    else
      {
        FP Operations
      }
    FP Operations
  }
Memory_store
  
```



- Divergent branches serialize execution of warps



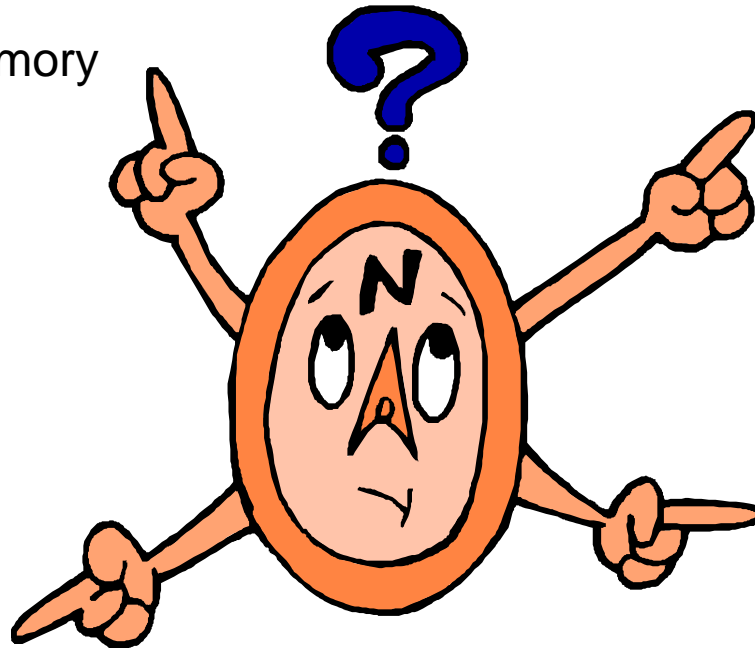
Memory Bank Conflicts

- Shared memory bank conflicts
 - Shared memory accesses within a warp are serialized
 - 4 cycles $\rightarrow 2 * 32 = 64$ cycles
- DRAM memory bank conflicts
 - DRAM memory accesses are serialized
 - 100 cycles $\rightarrow 100$ cycles * # of serialization
 - Higher number of banks, statically, there will be some amount of bank conflicts



What to Optimize for?

Shared memory conflicts



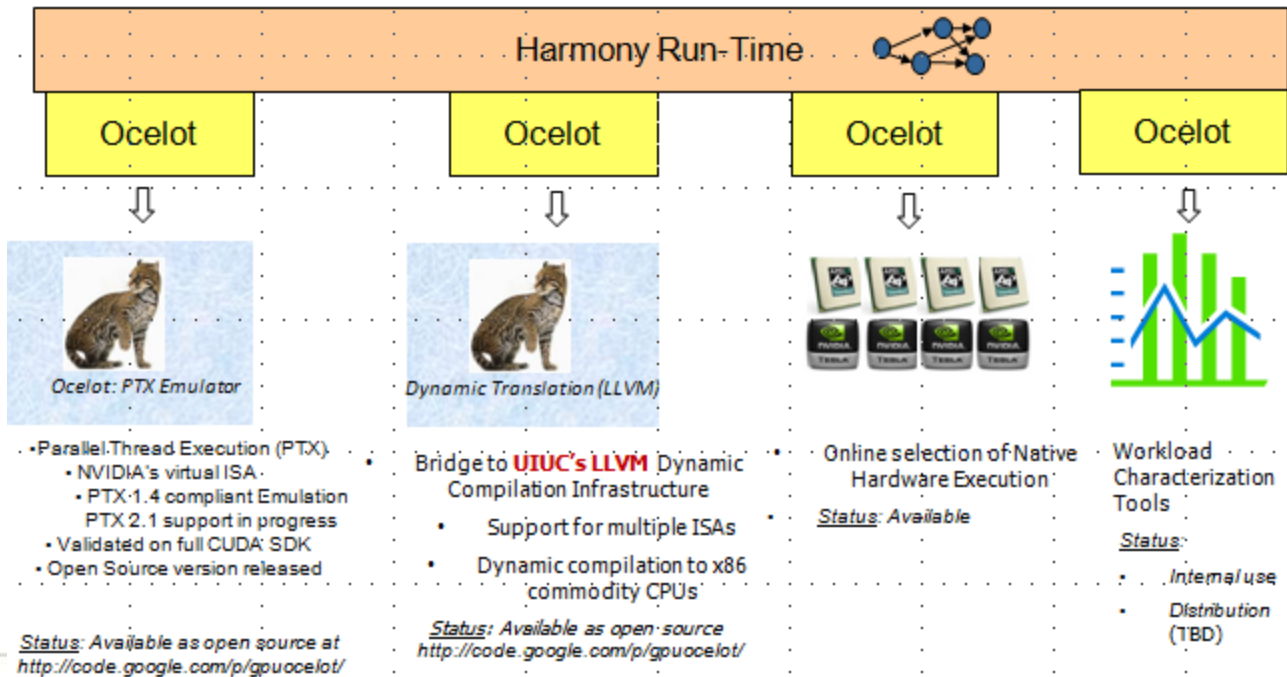
Occupancy

Global memory



Existing Tools

- CUDA Profiler[NVIDIA]
 - coalesced/uncoalesced, global memory throughputs, warp serializations, throughput oriented information
- Ocelot: Dynamic execution environment[GT]





From Ocelot

- Instruction mixtures
- # of instructions
- Shared memory bank conflicts
- Coalesced/uncoalesced memory
- Divergent warps
- Average bandwidth requirements



GPU PERFORMANCE ANALYSIS ANALYTICAL MODEL



How is Performance Determined ?

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

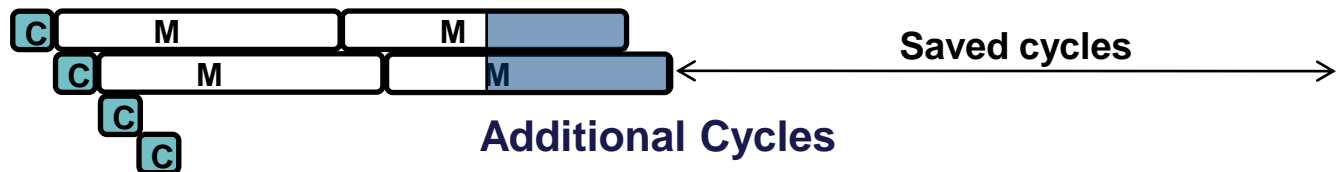
No Parallelism



Infinite Parallelism



Finite Parallelism
(Two)

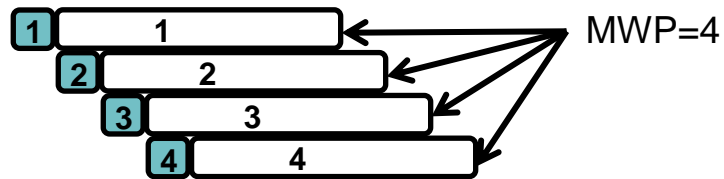


- Performance can be **predicted** by knowing the amount of memory-level parallelism



MWP

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

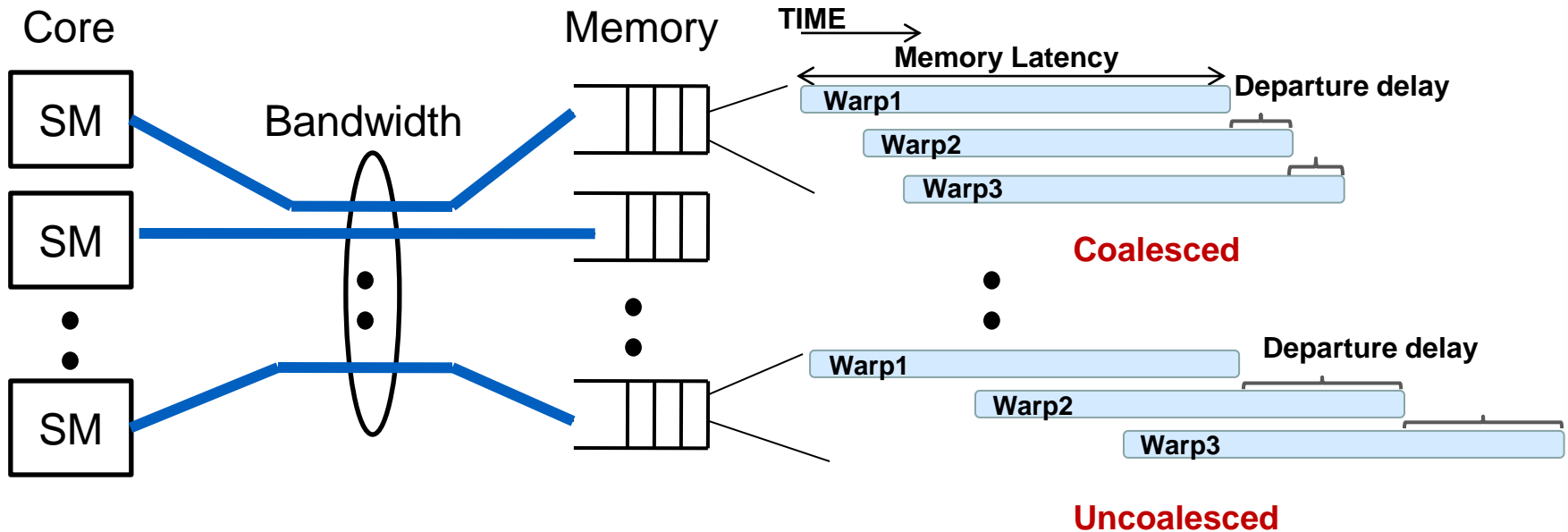


Four warps are overlapped during memory accesses

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



Memory System Model

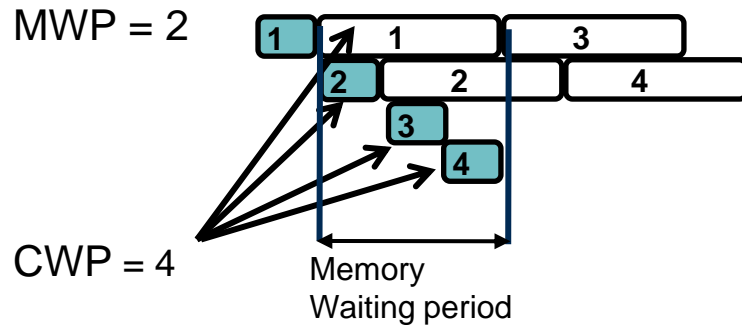


- Each SM has a simple queue and consumes an equal bandwidth
- MWP is determined by **#Active SMs**, **#Active warps**, **Bandwidth**, **Types of memory accesses (Coalesced, Uncoalesced)**



CWP

- Computation Warp Parallelism
- Analogous concept to MWP

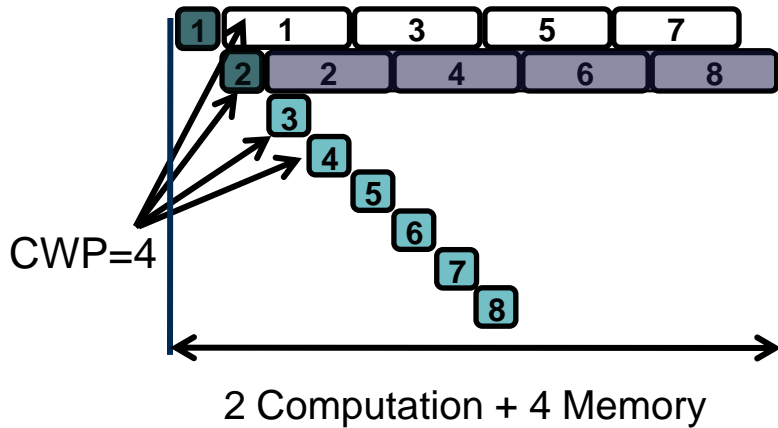


- Number of warps that execute instructions during one memory access period
- **Three scenarios can occur depending on the MWP and CWP relationship**



Case 1: When $MWP \leq CWP$

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



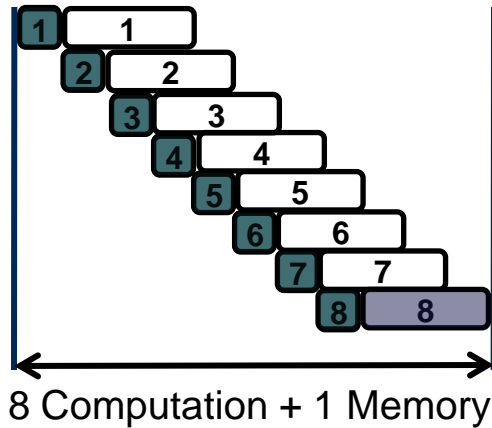
- Computation cycles are hidden by memory waiting periods
- Overall performance is dominated by the memory cycles

$$Exec_cycles = Mem_cycles \times \frac{N}{MWP} + Comp_p \times MWP \quad (MWP=2, N = 8)$$



Case 2: When MWP > CWP

MWP=8 N = 8 (Number of warps)



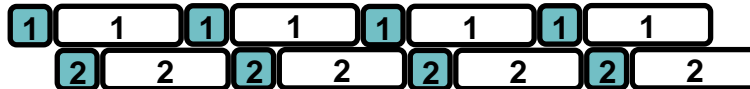
- Memory accesses are mostly hidden due to high MWP
- Overall performance is dominated by the computation cycles

$$Exec_cycles = Mem_p + Comp_cycles \times N \quad (MWP=8, N = 8)$$



Case 3: $N = MWP$ and $N = CWP$ (Not Enough Warps)

Two warps



- Increasing the number of warps will increase the processor utilization
- MWP is limited by the number of active warps per SM

$$\begin{aligned}
 Exec_cycles &= Mem_cycles \times N/MWP + Comp_cycles \times \\
 &N/MWP + Comp_p(MWP - 1) \\
 &= Mem_cycles + Comp_cycles + Comp_p(MWP - 1)
 \end{aligned}$$



MWP Calculation

$$MWP = \text{MIN}(MWP_Without_BW, MWP_peak_BW, N)$$

$$MWP_peak_BW = \frac{Mem_Bandwidth}{BW_per_warp \times \#ActiveSM}$$

$$BW_per_warp = \frac{Freq \times Load_bytes_per_warp}{Mem_L}$$

$$MWP_Without_BW_full = Mem_L / Departure_delay$$

$$MWP_Without_BW = \text{MIN}(MWP_Without_BW_full, N)$$

Putting It All together

$$Mem_L_Uncoal = Mem_LD + (\#Uncoal_per_mw - 1) \times Departure_del_uncoal$$

$$Mem_L_Coal = Mem_LD$$

$$Mem_L = Mem_L_Uncoal \times Weight_uncoal + Mem_L_Coal \times Weight_coal$$

$$Weight_uncoal = \frac{\#Uncoal_Mem_insts}{(\#Uncoal_Mem_insts + \#Coal_Mem_insts)}$$

$$Weight_coal = \frac{\#Coal_Mem_insts}{(\#Coal_Mem_insts + \#Uncoal_Mem_insts)}$$

$$Departure_delay = (Departure_del_uncoal \times \#Uncoal_per_mw) \times Weight_uncoal + Departure_del_coal \times Weight_coal$$

$$MWP_Without_BW_full = Mem_L / Departure_delay$$

$$MWP_Without_BW = MIN(MWP_Without_BW_full, \#Active_warps_per_SM)$$

$$Mem_cycles = Mem_L_Uncoal \times \#Uncoal_Mem_insts + Mem_L_Coal \times \#Coal_Mem_insts$$

$$Comp_cycles = \#Issue_cycles \times (\#total_insts)$$

$$N = \#Active_warps_per_SM$$

$$\#Rep = \frac{\#Blocks}{\#Active_blocks_per_SM \times \#Active_SMs}$$

If (MWP is N warps per SM) and (CWP is N warps per SM)

$$Exec_cycles_app = (Mem_cycles + Comp_cycles + \frac{Comp_cycles}{\#Mem_insts} \times (MWP - 1)) \times \#Rep$$

If (CWP \geq MWP) or (Comp_cycles $>$ Mem_cycles)

$$Exec_cycles_app = (Mem_cycles \times \frac{N}{MWP} + \frac{Comp_cycles}{\#Mem_insts} \times (MWP - 1)) \times \#Rep$$

If (MWP $>$ CWP)

$$Exec_cycles_app = (Mem_L + Comp_cycles \times N) \times \#Rep$$





How to use the model

▪ 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)
Analyzing memory access pattern

▪ Outputs of the model

- Estimated execution time (cycles), CPI per warp granularity

$$CPI = \frac{Exec_cycles_app}{\#Total_insts \times \frac{\#Threads_per_block}{\#Threads_per_warp} \times \frac{\#Blocks}{\#Active_SMs}}$$

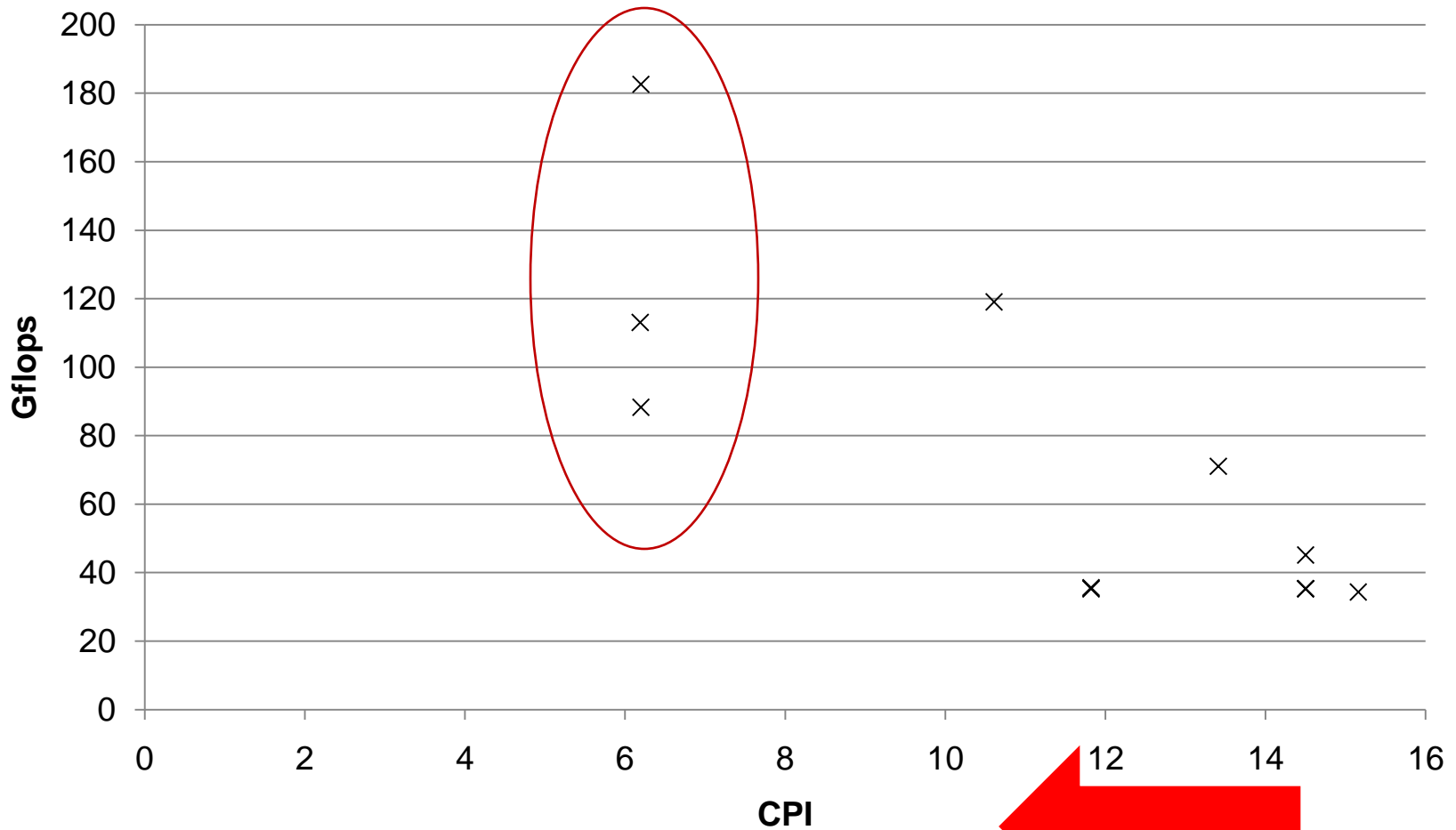


Previous Performance Tuning Example

Name of program	Threads/Block	Blocks On dataset)	Number of Instructions	Coalesced memory Accesses	Uncoalesced memory Accesses	Occupancy	MWP Peak BW	MWP	CWP	Model_CPI	Estimated Performance (Gflops/s)
cuda.cu	256	585	78680	4	14260	0.5	13.62	6	8	14.5	45.16
cuda_shmem.cu	256	585	78756	4	14261	0.5	13.62	6	8	14.5	35.4
cuda_shmem_trans.cu	256	585	78756	4	14261	0.5	13.62	6	8	14.5	35.25
cuda_vecpack.cu	256	585	50158	0	3610	0.5	6	6	4.25	10.61	119.06
cuda_vecpack_ujam.cu	256	585	50169	0	3612	0.25	6	6	3.57	13.41	71.04
cuda_vecpack_rsqr.cu	256	585	46609	0	3611	0.75	13.62	6	8	6.198	182.61
cuda_vecpack_shmem	256	585	46687	0	3612	0.5	13.62	6	8	6.189	113.06
cuda_vecpack_ujam_rsqr	256	585	46619	0	3612	0.25	13.62	6	8	6.198	88.3
cuda_shmem_ujam_rsqr_tight	256	585	96517	4	14261	0.25	13.62	6	8	11.82	35.65
cuda_shmem_trans_ujam_rsqr	256	585	75211	4	14261	0.25	13.62	6	8	15.16	34.33
cuda_shmem_trans_ujam_rsqr_tight	256	585	96517	4	14261	0.25	13.62	6	8	11.82	35.25



CPI vs. Gflop/s



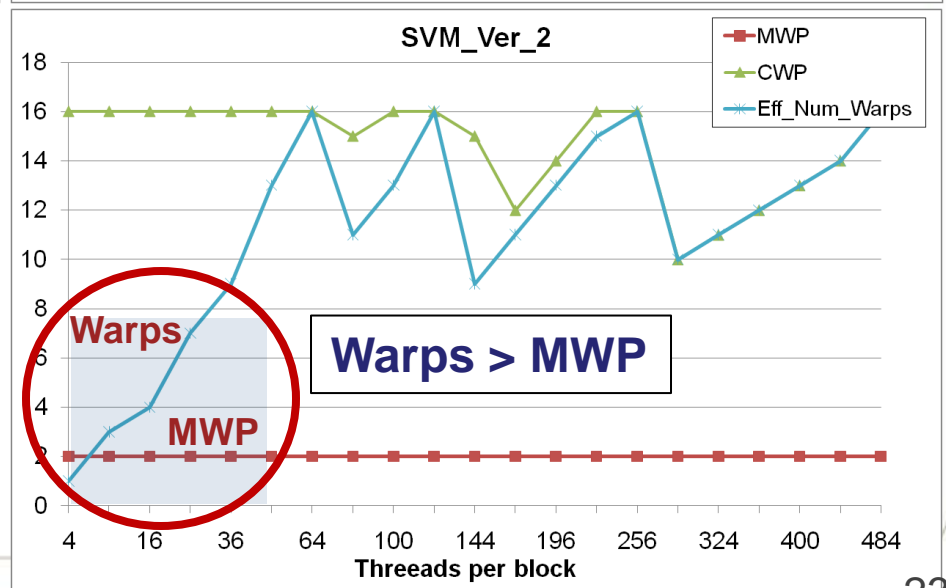
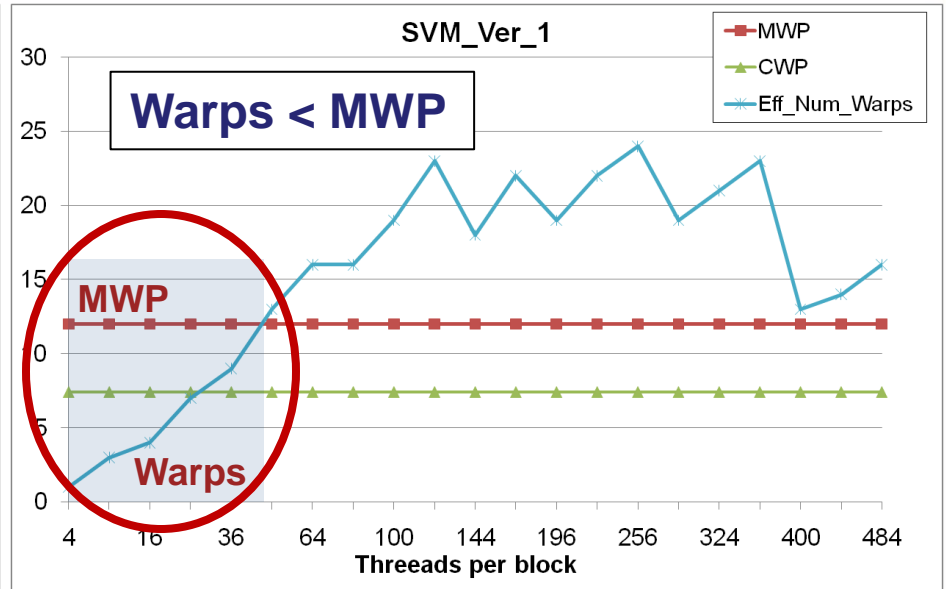
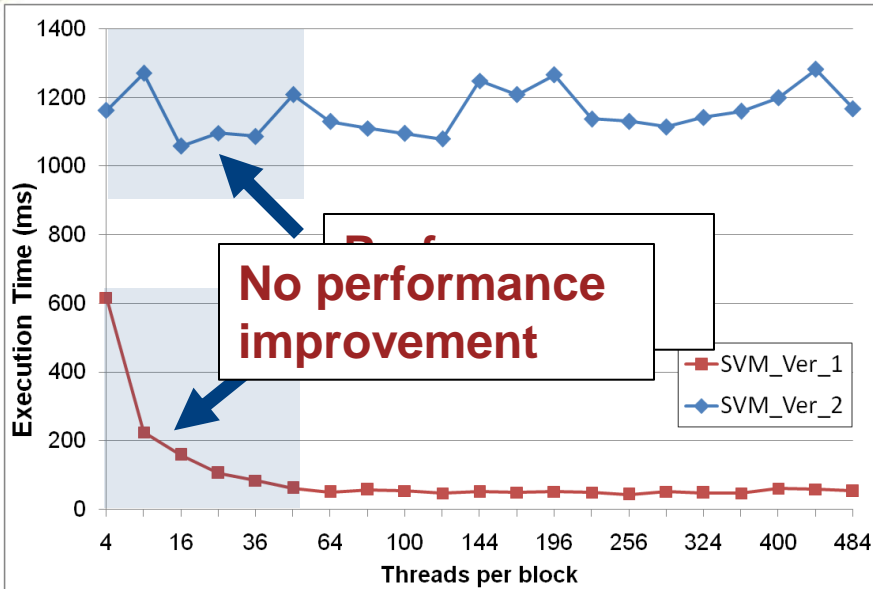


Limitations of the Model

- 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

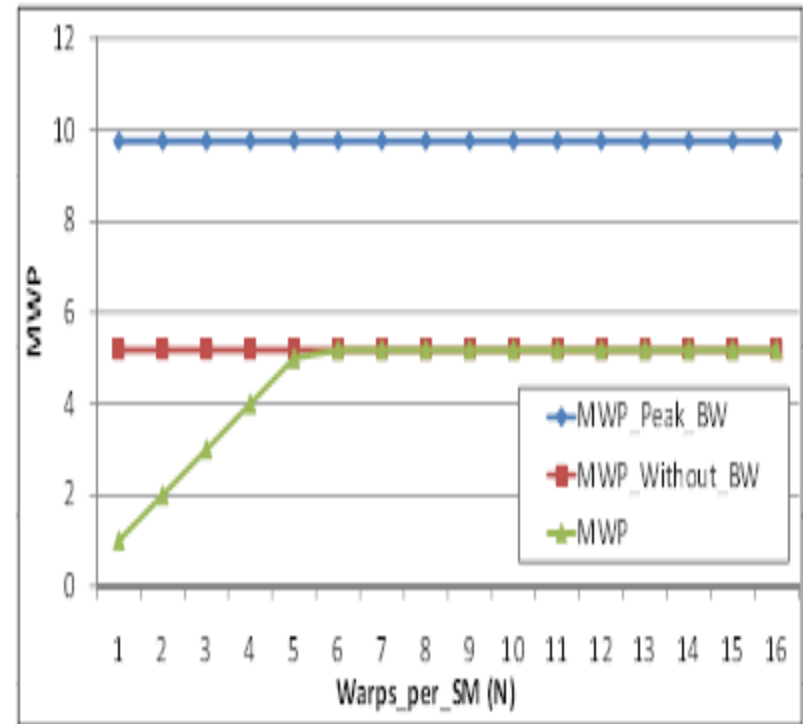
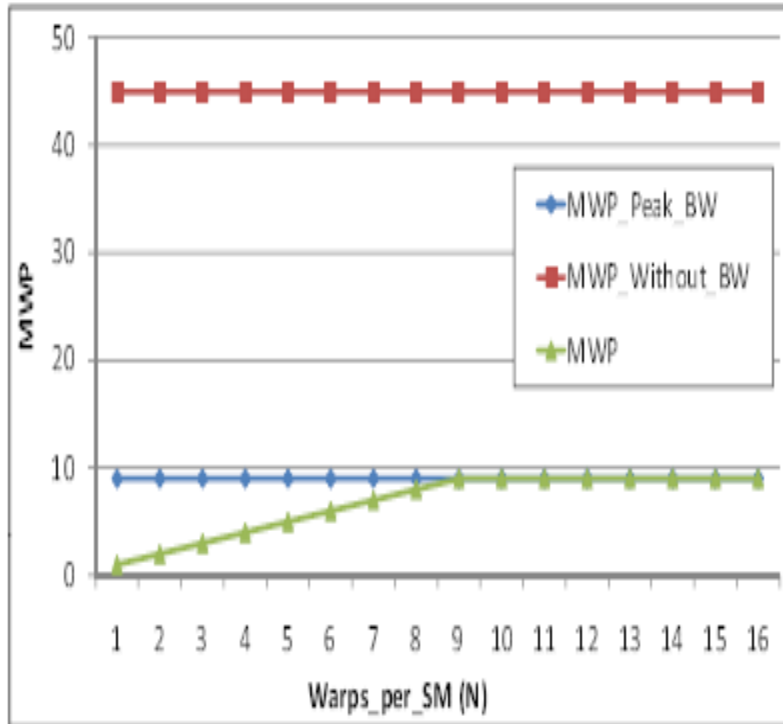


Insights on MWP (Motivation Example)





Increasing occupancy vs. MWP





Other Performance Models

- Work-flow graph analysis (UIUC)
- Auto-tuning (GT)
- Roofline model (Berkeley)



GPU Architecture Simulators

- GPGPU-Sim (UBC)
 - <http://www.ece.ubc.ca/~aamodt/gpgpu-sim/>
- ATILA – fixed graphics (UPC)
- MacSim (GT, Sandia Lab)
- Architecture simulators
 - Detailed memory behavior, memory bank conflicts, buffer conflicts, etc.



Acknowledgement

- Aniruddha Dusgupta
- Sunpyo Hong
- Jaekyu Lee