Performance Analysis Models and Tools

Hyesoon Kim

GeorgiTech 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

![Diagram showing coalesced memory access type]

- One warp generates a memory transaction

Uncoalesced memory access type

![Diagram showing uncoalesced memory access type]

- Multiple memory transactions
- More processing cycles for the uncoalesced case
Divergent Branches

- Divergent branches serialize execution of warps
Memory Bank Conflicts

- Shared memory bank conflicts
  - Shared memory accesses within a warp are serialized
  - 4 cycles $\rightarrow$ $2 \times 32 = 64$ 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

Hong & Kim ISCA’09
MWP

- Memory Warp Parallelism

- Metric of memory-level parallelism

  - Maximum number of warps that can overlap memory accesses

  - Tightly coupled with DRAM system
    - Memory latency, bandwidth, memory access type

Four warps are overlapped during memory accesses

Hong & Kim ISCA’09
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

MWP = 2

CWP = 4

Number of warps that execute instructions during one memory access period

Three scenarios can occur depending on the MWP and CWP relationship

Hong & Kim ISCA’09
Case 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

\[
\text{Exec\_cycles} = \frac{N}{MWP} \times \text{Mem\_cycles} + \text{Comp\_p} \times MWP
\]  

(MWP = 2, \( N = 8 \))
Case 2: When MWP > CWP

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

\[ \text{Exec}_{-}\text{cycles} = \text{Mem}_p + \text{Comp}_{-}\text{cycles} \times N \]

(MWP=8, \hspace{1em} N = 8)

Hong & Kim ISCA'09
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

\[
\text{Exec\_cycles} = \text{Mem\_cycles} \times \frac{N}{MWP} + \text{Comp\_cycles} \times \frac{N}{MWP} + \text{Comp\_p}(MWP - 1)
\]

\[
= \text{Mem\_cycles} + \text{Comp\_cycles} + \text{Comp\_p}(MWP - 1)
\]

Hong & Kim ISCA’09
MWP Calculation

\[ MWP = \text{MIN}(MWP_{\text{Without BW}}, MWP_{\text{peak BW}}, N) \]

\[ MWP_{\text{peak BW}} = \frac{\text{Mem Bandwidth}}{BW_{\text{per warp}} \times \#\text{ActiveSM}} \]

\[ BW_{\text{per warp}} = \frac{\text{Freq} \times \text{Load bytes per warp}}{\text{Mem L}} \]

\[ MWP_{\text{Without BW full}} = \frac{\text{Mem L}}{\text{Departure delay}} \]

\[ MWP_{\text{Without BW}} = \text{MIN}(MWP_{\text{Without BW full}}, N) \]
Putting It All together

\[ Mem_{L}_{\text{Uncoal}} = Mem_{L_{\text{LD}}} + (#Uncoal_{\text{per_mw}} - 1) \times \text{Departure}_{\text{del_uncoal}} \]

\[ Mem_{L_{\text{Coal}}} = Mem_{L_{\text{LD}}} \]

\[ Mem_{L} = Mem_{L_{\text{Uncoal}}} \times Weight_{\text{uncoal}} + Mem_{L_{\text{Coal}}} \times Weight_{\text{coal}} \]

\[ Weight_{\text{uncoal}} = \frac{\#Uncoal_{\text{Mem_insts}}}{(#Uncoal_{\text{Mem_insts}} + #Coal_{\text{Mem_insts}})} \]

\[ Weight_{\text{coal}} = \frac{#Coal_{\text{Mem_insts}}}{(\#Coal_{\text{Mem_insts}} + #Uncoal_{\text{Mem_insts}})} \]

\[ \text{Departure}_{\text{delay}} = (\text{Departure}_{\text{del_uncoal}} \times #Uncoal_{\text{per_mw}}) \times Weight_{\text{uncoal}} + \text{Departure}_{\text{del_coal}} \times Weight_{\text{coal}} \]

\[ MW \_P \text{ Without BW full} = Mem_{L} / \text{Departure}_{\text{delay}} \]

\[ MW \_P \text{ Without BW} = \text{MIN}(MW \_P \text{ Without BW full, } \#Active_{\text{warps per SM}}) \]

\[ Mem_{\text{cycles}} = Mem_{L_{\text{Uncoal}}} \times #Uncoal_{\text{Mem_insts}} + Mem_{L_{\text{Coal}}} \times #Coal_{\text{Mem_insts}} \]

\[ Comp_{\text{cycles}} = \#Issue_{\text{cycles}} \times (#total_{\text{insts}}) \]

\[ N = \#Active_{\text{warps per SM}} \]

\[ #Rep = \frac{#Blocks}{Active_{\text{blocks per SM}} \times Active_{\text{SMs}}} \]

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

\[ Exec_{\text{cycles app}} = (Mem_{\text{cycles}} + Comp_{\text{cycles}} + \frac{Comp_{\text{cycles}}}{\#Mem_{\text{insts}}} \times (MWP - 1)) \times #Rep \]

If (CWP >= MWP) or (Comp_{\text{cycles}} > Mem_{\text{cycles}})

\[ Exec_{\text{cycles app}} = (Mem_{\text{cycles}} \times \frac{N}{MWP} + \frac{Comp_{\text{cycles}}}{\#Mem_{\text{insts}}} \times (MWP - 1)) \times #Rep \]

If (MWP > CWP)

\[ Exec_{\text{cycles app}} = (Mem_{L} + Comp_{\text{cycles}} \times N) \times #Rep \]
How to use the model

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

- Outputs of the model
  - Estimated execution time (cycles), CPI per warp granularity

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

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
## Previous Performance Tuning Example

<table>
<thead>
<tr>
<th>Name of program</th>
<th>Threads/Block</th>
<th>Blocks On dataset</th>
<th>Number of Instructions</th>
<th>Coalesced memory Accesses</th>
<th>Uncoalesced memory Accesses</th>
<th>Occupancy</th>
<th>MWP Peak BW</th>
<th>MWP</th>
<th>CWP</th>
<th>Model_CPI</th>
<th>Estimated Performance (Gflops/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cuda.cu</td>
<td>256</td>
<td>585</td>
<td>78680</td>
<td>4</td>
<td>14260</td>
<td>0.5</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>14.5</td>
<td>45.16</td>
</tr>
<tr>
<td>cuda_shmem.cu</td>
<td>256</td>
<td>585</td>
<td>78756</td>
<td>4</td>
<td>14261</td>
<td>0.5</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>14.5</td>
<td>35.4</td>
</tr>
<tr>
<td>cuda_shmem_trans.cu</td>
<td>256</td>
<td>585</td>
<td>78756</td>
<td>4</td>
<td>14261</td>
<td>0.5</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>14.5</td>
<td>35.25</td>
</tr>
<tr>
<td>cuda_vecpack.cu</td>
<td>256</td>
<td>585</td>
<td>50158</td>
<td>0</td>
<td>3610</td>
<td>0.5</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>4.25</td>
<td>10.61</td>
</tr>
<tr>
<td>cuda_vecpack_ujam.cu</td>
<td>256</td>
<td>585</td>
<td>50169</td>
<td>0</td>
<td>3612</td>
<td>0.25</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>3.57</td>
<td>13.41</td>
</tr>
<tr>
<td>cuda_vecpack_rsqrt.cu</td>
<td>256</td>
<td>585</td>
<td>46609</td>
<td>0</td>
<td>3611</td>
<td>0.75</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>6.198</td>
<td>182.61</td>
</tr>
<tr>
<td>cuda_vecpack_shmem</td>
<td>256</td>
<td>585</td>
<td>46687</td>
<td>0</td>
<td>3612</td>
<td>0.5</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>6.189</td>
<td>113.06</td>
</tr>
<tr>
<td>cuda_vecpack_ujam_rsqrt</td>
<td>256</td>
<td>585</td>
<td>46619</td>
<td>0</td>
<td>3612</td>
<td>0.25</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>6.198</td>
<td>88.3</td>
</tr>
<tr>
<td>cuda_shmem_ujam_rsqrt_tight</td>
<td>256</td>
<td>585</td>
<td>96517</td>
<td>4</td>
<td>14261</td>
<td>0.25</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>11.82</td>
<td>35.65</td>
</tr>
<tr>
<td>cuda_shmem_trans_ujam_rsqrt</td>
<td>256</td>
<td>585</td>
<td>75211</td>
<td>4</td>
<td>14261</td>
<td>0.25</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>15.16</td>
<td>34.33</td>
</tr>
<tr>
<td>cuda_shmem_trans_ujam_rsqrt_tight</td>
<td>256</td>
<td>585</td>
<td>96517</td>
<td>4</td>
<td>14261</td>
<td>0.25</td>
<td>13.62</td>
<td>6</td>
<td>8</td>
<td>11.82</td>
<td>35.25</td>
</tr>
</tbody>
</table>
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)

- Warps < MWP: No performance improvement
- Warps > MWP: Performance improvement
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