MIC-GPU: High-Performance Computing for Medical Imaging on Programmable Graphics Hardware (GPUs)

Parallel Programming Primer

Klaus Mueller  
Stony Brook University  
Computer Science  
Stony Brook, NY

Fang Xu  
Siemens USA Research  
Princeton, NJ

Recommended Literature

- Textbook
- Reference book
- Programming guides available from nvidia.com
- More general books on parallel programming

Speedup Curves

GPU Performance Trends

but wait, there is more to this…..
Amdahl’s Law

Governs theoretical speedup

\[ S = \frac{1}{(1-P) + \frac{P}{S_{parallel}}} = \frac{1}{(1-P) + \frac{P}{N}} \]

- \( P \): parallelizable portion of the program
- \( S \): speedup
- \( N \): number of parallel processors

- Example (assuming infinite \( N \)):
  - \( P=90\% \) → \( S=10 \)
  - \( P=99\% \) → \( S=100 \)

Focus Efforts on Most Beneficial

How many processors to use
- when \( P \) is small → a small number of processors will do
- when \( P \) is large (embarrassingly parallel) → high \( N \) is useful

Optimize program portion with most ‘bang for the buck’
- look at each program component
- don’t be ambitious in the wrong place
Focus Efforts on Most Beneficial

Optimize program portion with most ‘bang for the buck’
  • look at each program component
  • don’t be ambitious in the wrong place

Example:
  • program with 2 independent parts: A, B (execution time shown)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B sped up 5×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A sped up 2×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  • sometimes one gains more with less

Beyond Theory....

Limits from mismatch of parallel program and parallel platform
  • man-made ‘laws’ subject to change with new architectures

Memory access patterns
  • data access locality and strides vs. memory banks
Beyond Theory....

Limits from mismatch of parallel program and parallel platform
- man-made ‘laws’ subject to change with new architectures

Memory access patterns
- data access locality and strides vs. memory banks

Memory access efficiency
- arithmetic intensity vs. cache sizes and hierarchies

Enabled granularity of program parallelism
- MIMD vs. SIMD

---

Device Transfer Costs

Transferring the data to the device is also important
- computational benefit of a transfer plays a large role
- transfer costs are (or can be) significant

Hardware support for specific tasks → on-chip ASICS
Support for hardware access → drivers, APIs
Device Transfer Costs

Transferring the data to the device is also important
- computational benefit of a transfer plays a large role
- transfer costs are (or can be) significant

Adding two \( (N \times N) \) matrices:
- transfer back and from device: \( 3 N^2 \) elements
- number of additions: \( N^2 \)
  \[ \text{operations-transfer ratio} = \frac{1}{3} \text{ or } O(1) \]

Multiplying two \( (N \times N) \) matrices:
- transfer back and from device: \( 3 N^2 \) elements
- number of multiplications and additions: \( N^3 \)
  \[ \text{operations-transfer ratio} = O(N) \text{ grows with } N \]

Programming Strategy

Use GPU to complement CPU execution
- recognize parallel program segments and only parallelize these
- leave the sequential (serial) portions on the CPU

parallel portions (enjoy)
sequential portions (do not bite)

PPP (Peach of Parallel Programming – Kirk/Hwu)

Course Schedule

1:30 – 1:45: Introduction (KM)
1:45 – 2:15: Introductory code examples (KM)
2:15 – 2:30: Parallel programming primer (KM)
2:30 – 3:00: Parallelism in CT reconstruction (FX)
  Coffee Break
3:30 – 3:45: GPU hardware (KM)
3:45 – 4:30: CUDA API, threads, memory, performance optimization (KM)
4:30 – 4:45: CUDA programming environment (FX)
4:45 – 5:25: CT reconstruction examples (FX, KM)
5:25 – 5:30: Closing remarks (KM, FX)