Should I port my code to a GPU?


*Clusters, Clouds, and Grids for Scientific Computing*
Flat Rock, North Carolina - September 9, 2010
A: It depends. Opportunity cost?

(1) Who are you?
(2) What is your app?
(3) What are your performance, productivity, and portability goals?

For most of us in this room, I’d say, “yes.”
For the “average” apps developer, I’d say “not yet.”
Q: Pay-off of a GPU port?
(Posed to me by Scott Klasky at ORNL)

- Meta-analysis, for **semi-irregular** sci. comp. + data analytics apps
  (sparse iterative + direct solvers; tree-based particle methods)

- A: Given roughly same level of tuning & power*, ...

\[
\text{GPU} \quad \equiv \quad x \ 2 \ \text{CPUs}
\]

Reason 1: The potential is real, but might be less than you expect.
<table>
<thead>
<tr>
<th>Architecture</th>
<th>Intel Nehalem X5550</th>
<th>NVIDIA T10P C1060</th>
<th>NVIDIA GT200 GTX 285</th>
<th>NVIDIA Fermi C2050</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHz</td>
<td>2.66</td>
<td>1.44</td>
<td>1.47</td>
<td>1.15</td>
</tr>
<tr>
<td>Sockets</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cores per socket</td>
<td>4</td>
<td>30</td>
<td>30</td>
<td>15*</td>
</tr>
<tr>
<td>Peak Gflop/s single (double)</td>
<td>170.6 (85.3)</td>
<td>933 (78)</td>
<td>1060 (88.4)</td>
<td>1030 (515)</td>
</tr>
<tr>
<td>Peak GB/s</td>
<td>51.2</td>
<td>102</td>
<td>159</td>
<td>144</td>
</tr>
<tr>
<td>Sys. Watts (sockets only)</td>
<td>375 (200)</td>
<td>200</td>
<td>204</td>
<td>247</td>
</tr>
</tbody>
</table>
Intensity (flop : byte)

Gflop/s

1/8 1/4 1/2 1 2 4 8 16

Platform

- Fermi
- C1060
- Nehalem x 2
- Nehalem

Single-precision

Thursday, September 9, 2010
Platform

- Fermi
- C1060
- Nehalem x 2
- Nehalem

Gflop/s vs. Intensity (flop : byte)

Double-precision

Thursday, September 9, 2010
Reason 2: **Productivity:** Though there is potential, there is also no free lunch.
Parallel Sorting (survey)
(Does not include Merrill & Grimshaw ’10)

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (sec / 10 Melem)</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nesl (16-cores)</td>
<td>20.752</td>
<td>11</td>
</tr>
<tr>
<td>C (1-core)</td>
<td>2.86</td>
<td>25</td>
</tr>
<tr>
<td>Cilk++ (16-cores)</td>
<td>1.17</td>
<td>56</td>
</tr>
<tr>
<td>Tuned library (16-core)</td>
<td>0.19</td>
<td>161</td>
</tr>
<tr>
<td>CUDA (GPU)</td>
<td>0.07</td>
<td>282</td>
</tr>
</tbody>
</table>
Parallel Sorting (survey)
(Does not include Merrill & Grimshaw '10)

- **Time**
  - Nesl (16-cores): 20.752 sec / 10 Melem
  - C (1-core): 2.86
  - Cilk++ (16-cores): 1.17
  - Tuned library (16-core): ~ 0.19
  - CUDA (GPU): 0.07

- **Lines of Code**
  - Nesl (16-cores): 11
  - C (1-core): 25
  - Cilk++ (16-cores): 56
  - Tuned library (16-core): 161
  - CUDA (GPU): 282

Thursday, September 9, 2010
Parallel Sorting (survey)
(Does not include Merrill & Grimshaw ’10)

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (sec / 10 Melem)</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nesl (16-cores)</td>
<td>20.752</td>
<td>11</td>
</tr>
<tr>
<td>C (1-core)</td>
<td>2.86</td>
<td>25</td>
</tr>
<tr>
<td>Cilk++ (16-cores)</td>
<td>1.17</td>
<td>56</td>
</tr>
<tr>
<td>Tuned library (16-core)</td>
<td>0.19</td>
<td>161</td>
</tr>
<tr>
<td>CUDA (GPU)</td>
<td>0.07</td>
<td>282</td>
</tr>
</tbody>
</table>

~ 2.7x
~ 1.8x
Reason 3: It’s a moving target that might end up converging to what we had before.
\[ C = \frac{\text{flops}}{\text{time}} \]

\[ \beta = \frac{\text{words}}{\text{time}} \]
Balance equation for an I/O-optimal matrix multiply:

\[ \frac{C}{\beta} = \Theta \left( \sqrt{M} \right) \]

See, for example, Kung (ISCA 1986).
\[ C = \frac{\text{flops}}{\text{time}} \]

\[ \beta = \frac{\text{words}}{\text{time}} \]

**Balance equation for an I/O-optimal matrix multiply:**

\[ \frac{C}{\beta} = \Theta \left( \sqrt{M} \right) \]

**For comparison-based sort:**

\[ \frac{C}{\beta} = \Theta \left( \log_2 M \right) \]

See, for example, Kung (ISCA 1986).
Having said all that, I am still optimistic about the role GPUs will play in current and future systems!
Sparse direct solvers

Kent Czechowski, M. Efe Guney, R. Vuduc

(Work in progress)

Anatomy of a sparse direct solver

- Sparse Cholesky factorization, $A = L \cdot L^T$, where $A$ & $L$ are sparse
- Mixed compute intensity, average of $\sim 4$ flops : byte for sample problem
All time is in “BLAS 3” calls
Thread capacity

Time

Matrix dimension

% xGEMM peak

Thursday, September 9, 2010
Thread capacity

Time

One-at-a-time
Naive CUDA Streams Code
Optimal Packing

% xGEMM peak

Thursday, September 9, 2010
Thread capacity

Time

% xGEMM peak

One-at-a-time
Naive CUDA Streams Code
Optimal Packing

~ 1.7x

Thursday, September 9, 2010
Some concluding questions...

* What is the right way to think about **opportunity cost**?

* What are the **performance principles** for reducing tuning effort?

* What **applications** will lead to better designs?
Parallelism = Elimination tree

Independent subtrees may be processed in parallel.
Finer-grained dependencies

Colored circles on the right are BLAS calls on operands of varying size.