Finding Vertex Cover: Acceleration Via CUDA

Yang Liu, High Performance Research Computing, Texas A&M University
Jinbin Ju, Electrical Engineering, Texas A&M University
Derek Rodriguez, Computer Engineering, Texas A&M University
Motivation

Phylogeny

http://tolweb.org/tree/learn/concepts/whatisphylogeny.html

Motif Finding

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2286532/figure/f2

Microarray Analysis

https://www.youtube.com/watch?v=9U-9mI0zoZ8
Related Problems: Maximum Clique and Maximum Independent Set

- **Clique**
  - A set of vertices such that there is an edge between any pair of vertices in this set.

- **Independent Set**
  - A set of vertices such that there is no edge between any pair of vertices in this set.

Example of Clique (Blue Vertices):
- $n = 12$
- $k = 5$

Example of Independent Set (Blue Vertices):
- $n = 12$
- $k = 5$
Vertex Cover

• **Vertex Cover:**
  • A subset of vertices such that every edge is incident to at least one vertex in the subset

• **Minimum Vertex Cover:**
  • Find a vertex cover of minimum size.
  • One of Richard Karp’s 21 NP-Complete Problems

• **Parameterized Vertex Cover**
  • Given a parameter k, find a vertex cover of size at most k.

---

Example of Vertex Covers (Red Vertices)

n = 12
k = 7
Related Work

- These problems have been extensively studied (exact algorithms, parameterized algorithms, approximation algorithms, and heuristics).

- **Parameterized Vertex Cover**
  - Best parameterize algorithm: $O^*(1.2738^k)$ (by Chen, et. al. 2010)
  - Parallel implementation scales up to 2400 CPUs (by Weerapurage et. al. 2011).

- **Independent Set**
  - $W[1]$-hard, i.e., unlikely to have algorithms of complexity $O(f(k)p(n))$ where $f(k)$ is independent of $n$.

- **Clique**
Branching Process

- Find max degree vertex \( v \)
- Two branch sets: \( v \) is in vertex cover or \( v \)'s neighbors are in vertex cover

\[
|G'| = |G| - 1 \\
k' = k - 1
\]

\[
|G'| = |G| - |N| - 1 \\
k' = k - |N|
\]
**Branching Process**

- Find max degree vertex $v$
- Two branch sets: $v$ is in vertex cover or $v$’s neighbors are in vertex cover

**Branching Process**

- $|G'| = |G| - 1$
- $k' = k - 1$

**Finding Vertex Cover: Acceleration Via CUDA**

- $n = 12$
- $k = 7$

- $|G'| = |G| - |N| - 1$
- $k = k - |N|$

- $n = 11$
- $k = 6$

- $n = 8$
- $k = 4$

When $n \leq$ Threshold, send to GPU for further processing.
Branching Process

- Find max degree vertex $v$
- Two branch sets: $v$ is in vertex cover or $v$’s neighbors are in vertex cover

$|G'| = |G| - 1$
$k' = k - 1$

Synchronization between CPU and GPU:

Kernel and CPU communicate on solution state (vertex cover is found or not) via memory copy from GPU to CPU.

(We tried mapped memory, but somehow our program is unstable, and very difficult to debug)

**CPU:**

```c
while (there is a graph to branch) {
    branch and create a small graph  // n <= THRESHOLD
    if (a vertex cover is found by kernel)
        return;
    if (SMALL-GRAPH-COUNT small graphs are created) {
        start kernel to process SMALL-GRAPH-COUNT small graphs  // can overlap with creation of small graphs
    }
}
```
GPU Memory Hierarchy

<table>
<thead>
<tr>
<th>Memory</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Memory</td>
<td>~8,000 GB/s</td>
</tr>
<tr>
<td>Shared Memory</td>
<td>~1,600 GB/s</td>
</tr>
<tr>
<td>Global Memory</td>
<td>~177 GB/s</td>
</tr>
<tr>
<td>Mapped Memory</td>
<td>~8 GB/s</td>
</tr>
</tbody>
</table>

CUDA Application Design and Development, Rob Farber, 2012.

https://www.quantalea.net/media/_doc/2/7/manual/index.html?GPUHardwareImplementation.html
Placement of Small Graphs in GPU

- A small sub-graph in shared memory for each block
- Only 48K bytes per SM (Streaming Multiprocessor) → This limits the THRESHOLD for small sub-graphs
- Maximum concurrent blocks per SM is 16 (then each block has only 3k shared memory) → this limits the THRESHOLD for small sub-graphs too.

Memory Bandwidth

<table>
<thead>
<tr>
<th>Memory</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Memory</td>
<td>~8,000 GB/s</td>
</tr>
<tr>
<td>Shared Memory</td>
<td>~1,600 GB/s</td>
</tr>
<tr>
<td>Global Memory</td>
<td>~177 GB/s</td>
</tr>
<tr>
<td>Mapped Memory</td>
<td>~8 GB/s</td>
</tr>
</tbody>
</table>

CUDA Application Design and Development, Rob Farber, 2012.

https://www.quantalea.net/media/_doc/2/7/manual/index.html?GPUHardwareImplementation.html
• Tesla K20: 192 cores per SM, max active 64 warps (32 threads/warp), max resident 16 blocks
• More warps and/or blocks than 192 cores can actually execute → fast context switching to hide memory access latency.
• Occupancy: active warps/64 → higher occupancy is likely hide more memory latency, but not necessary implies better performance (see Better Performance at Lower Occupancy).
• Our program aims to have occupancy of 25%: 16 resident blocks so that 16 small graphs (<3K each) can be processed concurrently in each SM.
Configurations for GPU Processing

- 2000 blocks
  - Allow maximum number of small graphs to be processed concurrently (Tesla K20 allows 13*16=208 blocks resident in shared memory)
  - reduce the overhead of memory copy
  - possibly reduce the impact of imbalanced computations among blocks.
- 32 threads per block
  - A warp for easy synchronization.
  - Number of threads per block determines the amount of shared memory needed by a block \( \rightarrow \) affect the occupancy.
- Each block processes a small graph with vertices \( \leq 80 \) (if available)
  - Number of vertices of a small graph determines the amount of shared memory needed by a block \( \rightarrow \) affect the occupancy.
• Current occupancy of our program is 25% (16 active warps/blocks per SM).

• Our tests show this configuration achieves the best performance in general
  • A particular graph may achieve better performance with different configurations.
  • Some graphs have bottleneck on cpu side while some graphs have bottleneck on gpu side.

• Number of registers per thread and shared memory per block \(\rightarrow\) how many blocks can run concurrently
  • `nvcc --ptxas-options=-v` provides such information
  • Our program: 69 registers per thread, and 2976 bytes shared memory per block
Test Data

- Provided by Michael Langston and Gary Rogers.
- Created from real biological data related to:
  - Folic acid deficiency effect on colon cancer cells (fo30, fo35, fo40, fo45)
  - Low concentrations of 17 beta-estradiol effects on breast cancer cell line (es30, es35, es40, es45)
  - Interferon receptor deficient lymph node B cell response to influenza infection (in30, in35, in40, in45)
- Each test data is tested with two k values:
  - $k = t$ when there is a vertex cover of at most $t$ vertices
  - $k = t-1$ where there are no vertex covers of at most $t-1$ vertices.
### Speedup of CPU+GPU Program Over Serial Program

<table>
<thead>
<tr>
<th>Graph-k</th>
<th>Serial (seconds)</th>
<th>CPU+GPU (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>est30-k981</td>
<td>11679.7356</td>
<td>885.6192</td>
</tr>
<tr>
<td>est30-k982</td>
<td>3501.296</td>
<td>283.3502</td>
</tr>
<tr>
<td>est35-k983</td>
<td>3145.042</td>
<td>254.1531</td>
</tr>
<tr>
<td>est35-k984</td>
<td>327.1932</td>
<td>29.5715</td>
</tr>
<tr>
<td>est40-k984</td>
<td>849.564</td>
<td>55.2437</td>
</tr>
<tr>
<td>est40-k985</td>
<td>113.1256</td>
<td>7.5073</td>
</tr>
<tr>
<td>est45-k986</td>
<td>265.9292</td>
<td>33.6088</td>
</tr>
<tr>
<td>est45-k987</td>
<td>6.523</td>
<td>2.5577</td>
</tr>
<tr>
<td>fo30-k982</td>
<td>31337.2464</td>
<td>1895.7333</td>
</tr>
<tr>
<td>fo30-k983</td>
<td>1782.824</td>
<td>124.2507</td>
</tr>
<tr>
<td>fo30-k984</td>
<td>6.709</td>
<td>1.0532</td>
</tr>
<tr>
<td>fo35-k984</td>
<td>7110.7688</td>
<td>649.9995</td>
</tr>
<tr>
<td>fo35-k985</td>
<td>277.7758</td>
<td>27.9412</td>
</tr>
<tr>
<td>fo40-k985</td>
<td>2208.4914</td>
<td>200.178</td>
</tr>
<tr>
<td>fo40-k986</td>
<td>156.8424</td>
<td>12.987</td>
</tr>
<tr>
<td>fo45-k986</td>
<td>574.8578</td>
<td>54.863</td>
</tr>
<tr>
<td>fo45-k987</td>
<td>44.4756</td>
<td>6.2211</td>
</tr>
<tr>
<td>inf30-k883</td>
<td>1574.9104</td>
<td>136.34</td>
</tr>
<tr>
<td>inf30-k884</td>
<td>366.0082</td>
<td>22.1952</td>
</tr>
<tr>
<td>inf35-k884</td>
<td>426.7548</td>
<td>23.6902</td>
</tr>
<tr>
<td>inf35-k885</td>
<td>404.494</td>
<td>21.9723</td>
</tr>
<tr>
<td>inf40-k885</td>
<td>156.3758</td>
<td>16.5731</td>
</tr>
<tr>
<td>inf40-k887</td>
<td>16.1454</td>
<td>3.2252</td>
</tr>
<tr>
<td>inf45-k887</td>
<td>75.0088</td>
<td>12.4427</td>
</tr>
<tr>
<td>inf45-k888</td>
<td>0.9464</td>
<td>1.3148</td>
</tr>
</tbody>
</table>
Test Environment

• Ada X86-64 Cluster at Texas A&M University (862 nodes)
  • Intel E5-2670 v2 (peak performance: 400 GFLOPS)
  • Nvidia K20 (peak performance: 3.52 TFLOPS)

• CUDA 6.5.14 and Intel Compiler 2013_sp.1.3.174
Acknowledgements

• Support from NSF grant 1442734.
• Support from High Performance Research Computing at Texas A&M University
• Test data provided by Michael Langston and Gary Rogers
• Discussions on CUDA with Robert Crovella
Thank You!