Relational Algorithms for Multi-Bulk-Synchronous Processors

Gregory Diamos*, Haicheng Wu^, Jin Wang^, Ashwin Lele^, and Sudhakar Yalamanchili^

* NVIDIA Research
^ Georgia Institute of Technology

Application Space: Data Warehousing

- On-line and off-line analysis
  - Retail analysis
  - Forecasting
  - Pricing
  - Etc...
- Combination of relational data queries and computational kernels
- Current applications process 1 to 50 TBs of data [1]
- Not a traditional domain for GPU acceleration, but:
  - Parallel queries experience good speedup on GPUs [2]
  - GPU-related techniques can be applied to other “Big Data” problems like irregular graphs, sorting

Relational Algorithm Design

- Strategy: Increase core utilizations until the computation becomes memory bound, and then achieve near peak utilization of the memory interface
- Hybrid multi-stage algorithm (partition, compute, gather) to make trade-offs between computation complexity and memory access efficiency

Data Structure: weakly-sorted densely packed array of tuples

Example of Merge Join

Red Fox: Execution Environment for the Enterprise

Kernel Weaver: Automatically Perform Kernel Fusion
Optimization to reduce data movements [4,5]

Red Fox TPC-H Benchmark Performance
(Tesla C2075)

- Execution time = PCIe + GPU Computation
- Problem size is restricted by GPU memory capacity
- No data movement optimizations
- Unoptimized query plan
- RA operators are designed by us, others are from Thrust library
- Each Operators maps to several CUDA Kernels

Breakdown

<table>
<thead>
<tr>
<th>Query</th>
<th>Scale Factor</th>
<th>Execution Time (second)</th>
<th>Input Size (MB)</th>
<th>#Operators</th>
<th>#CUDA Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>1</td>
<td>7.32</td>
<td>528</td>
<td>33</td>
<td>146</td>
</tr>
<tr>
<td>Q2</td>
<td>1</td>
<td>0.28</td>
<td>38</td>
<td>48</td>
<td>164</td>
</tr>
<tr>
<td>Q3</td>
<td>1</td>
<td>2.56</td>
<td>413</td>
<td>48</td>
<td>164</td>
</tr>
<tr>
<td>Q4</td>
<td>1</td>
<td>1.62</td>
<td>340</td>
<td>34</td>
<td>95</td>
</tr>
<tr>
<td>Q5</td>
<td>1</td>
<td>0.51</td>
<td>264</td>
<td>18</td>
<td>56</td>
</tr>
<tr>
<td>Q6</td>
<td>1</td>
<td>1.23</td>
<td>277</td>
<td>42</td>
<td>111</td>
</tr>
<tr>
<td>Q7</td>
<td>1</td>
<td>2.52</td>
<td>384</td>
<td>28</td>
<td>75</td>
</tr>
<tr>
<td>Q8</td>
<td>1</td>
<td>1.16</td>
<td>349</td>
<td>53</td>
<td>124</td>
</tr>
<tr>
<td>Q9</td>
<td>1</td>
<td>3.01</td>
<td>334</td>
<td>61</td>
<td>162</td>
</tr>
<tr>
<td>Q10</td>
<td>1</td>
<td>19.50</td>
<td>437</td>
<td>29</td>
<td>93</td>
</tr>
<tr>
<td>Q11</td>
<td>1</td>
<td>3.54</td>
<td>300</td>
<td>46</td>
<td>143</td>
</tr>
<tr>
<td>Q12</td>
<td>1</td>
<td>3.48</td>
<td>192</td>
<td>21</td>
<td>53</td>
</tr>
<tr>
<td>Q13</td>
<td>1</td>
<td>0.66</td>
<td>438</td>
<td>33</td>
<td>107</td>
</tr>
</tbody>
</table>

Reference