Satisfying Data-Intensive Queries Using GPU Clusters
Haicheng Wu, Jeffrey Young, and Sudhakar Yalamanchili
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

Our Solution: Red Fox & Oncilla

Combine Global Physical Address & GPUs

- **Red Fox**: Compilation and optimization of queries for GPUs
  - Remove need for application developer to optimize applications to run on GPUs
- **Oncilla**: Global Address Space (GAS)
  - Commodity Interconnects (HT, QPI, IB, PCIe)
  - HW/SW for global address space
  - Support for large in-core database
  - SW layer for optimized data movements

Oncilla: GAS GPU Cluster Model

- Oncilla supports multiple types of data transfer and CUDA-based optimizations under a simplified runtime.
- Uses EXTOLL NIC to enable high-performance data transfers [6].

Oncilla Runtime and API Example

- Runtime handles allocation of remote memory and keeps track of connection state for different network layers.
- Oncilla API has a concept of “opaque” and “transparent” calls that allow for either simplified or more granular control of data movement between host and accelerator memories.

Red Fox: Execution Environment for the Enterprise

- Execution time = PCIe + GPU Computation
- Problem size is restricted by GPU memory capacity
- No data movement optimizations
- Unoptimized query plan

Raw Performance of RA Primitives(Tesla C2050) [3]

<table>
<thead>
<tr>
<th>Scale Factor</th>
<th>Execution Time (seconds)</th>
<th>Input Size (MB)</th>
<th>#Operations</th>
<th>#CUDA Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>7.32</td>
<td>528</td>
<td>33</td>
<td>146</td>
</tr>
<tr>
<td>Query 2</td>
<td>0.28</td>
<td>38</td>
<td>48</td>
<td>164</td>
</tr>
<tr>
<td>Query 3</td>
<td>25.6</td>
<td>413</td>
<td>48</td>
<td>164</td>
</tr>
<tr>
<td>Query 4</td>
<td>1.62</td>
<td>340</td>
<td>34</td>
<td>95</td>
</tr>
<tr>
<td>Query 5</td>
<td>0.61</td>
<td>264</td>
<td>18</td>
<td>56</td>
</tr>
<tr>
<td>Query 6</td>
<td>1.23</td>
<td>277</td>
<td>42</td>
<td>111</td>
</tr>
<tr>
<td>Query 7</td>
<td>2.52</td>
<td>384</td>
<td>28</td>
<td>75</td>
</tr>
<tr>
<td>Query 8</td>
<td>1.16</td>
<td>349</td>
<td>53</td>
<td>124</td>
</tr>
<tr>
<td>Query 9</td>
<td>3.01</td>
<td>334</td>
<td>61</td>
<td>162</td>
</tr>
<tr>
<td>Query 10</td>
<td>19.50</td>
<td>437</td>
<td>29</td>
<td>93</td>
</tr>
<tr>
<td>Query 11</td>
<td>3.54</td>
<td>300</td>
<td>46</td>
<td>143</td>
</tr>
<tr>
<td>Query 12</td>
<td>3.48</td>
<td>192</td>
<td>21</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>0.66</td>
<td>438</td>
<td>33</td>
<td>107</td>
</tr>
</tbody>
</table>

References