Accelerating Database Joins Using a General Purpose GPU

Kevin Angstadt, Ed Harcourt
Department of Mathematics, Computer Science, and Statistics
St. Lawrence University
Canton, NY
Overview

- Terminology
- Computer Architecture
- Database Joins
- GPU Implementation
- Results
Terminology

- **CPU**
  - Central Processing Unit
  - Most computation done here
  - Uses RAM to store information
  - "Host"

- **GPU**
  - Graphics Processing Unit
  - Separate card used for displaying computer graphics
  - Built-in Memory
  - "Device"
Computational Power

Which of these two processors (from the same era) has more computational power?

Intel Core® i7-920 CPU
nVidia Fermi GPU
CPU vs. GPU Architecture

CPU
- Control
- Cache
- DRAM
- MIMD

GPU
- DRAM
- SIMD
## Database Joins

- Common Database operation
- Combines data rows from two tables

### Table A

<table>
<thead>
<tr>
<th>course_number</th>
<th>course_title</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS140</td>
<td>Intro to Computer Programming</td>
</tr>
<tr>
<td>CS220</td>
<td>Computer Organization</td>
</tr>
<tr>
<td>CS321</td>
<td>Computer Networking</td>
</tr>
<tr>
<td>CS340</td>
<td>Software Engineering</td>
</tr>
<tr>
<td>CS364</td>
<td>Programming Languages</td>
</tr>
</tbody>
</table>

### Table B

<table>
<thead>
<tr>
<th>course_number</th>
<th>course_professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS140</td>
<td>Ed Harcourt</td>
</tr>
<tr>
<td>CS140</td>
<td>Lisa Torrey</td>
</tr>
<tr>
<td>CS140</td>
<td>Choong-Soo Lee</td>
</tr>
<tr>
<td>CS220</td>
<td>Ed Harcourt</td>
</tr>
<tr>
<td>CS321</td>
<td>Choong-Soo Lee</td>
</tr>
<tr>
<td>CS340</td>
<td>Lisa Torrey</td>
</tr>
<tr>
<td>CS364</td>
<td>Lisa Torrey</td>
</tr>
</tbody>
</table>
- **Cartesian Product**: the set of all possible pairs \((a, b)\) where \(a \in A\) and \(b \in B\). (cross product)

\[
A = \{1, 2, 3\} \quad B = \{x, y, z\}
\]

\[
(1, x) \quad (1, y) \quad (1, z) \\
(2, x) \quad (2, y) \quad (2, z) \\
(3, x) \quad (3, y) \quad (3, z)
\]
Cartesian Product has superfluous rows
- Restrict result with *predicates*
- \( A\cdot \text{course\_number}=B\cdot \text{course\_number} \)

<table>
<thead>
<tr>
<th>course_number</th>
<th>course_title</th>
<th>course_number</th>
<th>course_professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS140</td>
<td>Intro to Computer Programming</td>
<td>CS140</td>
<td>Ed Harcourt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CS140</td>
<td>Lisa Torrey</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CS140</td>
<td>Choong-Soo Lee</td>
</tr>
<tr>
<td>CS220</td>
<td>Computer Organization</td>
<td>CS220</td>
<td>Ed Harcourt</td>
</tr>
<tr>
<td>CS321</td>
<td>Computer Networking</td>
<td>CS321</td>
<td>Choong-Soo Lee</td>
</tr>
<tr>
<td>CS340</td>
<td>Software Engineering</td>
<td>CS340</td>
<td>Lisa Torrey</td>
</tr>
<tr>
<td>CS364</td>
<td>Programming Languages</td>
<td>CS364</td>
<td>Lisa Torrey</td>
</tr>
</tbody>
</table>

Table A

Table B
nVidia Cuda API
- Provides toolkit needed to run code on GPU
- C-style language

Extends “Virginian” Database
- Demonstrated speedup for single-table queries
- UVA; NEC Labs, New Jersey

Virtual Machine-based Implementation
- Similar to SQLite Database
- Used by programs, such as Firefox
### Step 1: Compile SQL

```sql
SELECT A.course_number, A.course_title, B.course_professor
FROM A, B
WHERE A.course_number = B.course_number;
```

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Table</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Table</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>ResultColumn</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>ResultColumn</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>ResultColumn</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Parallel</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Column</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Integer</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>Le</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Column</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Integer</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Lt</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Invalid</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>Rowid</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Result</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>Converge</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Finish</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Compute for each row
Step 3: Set up GPU Kernel

Rows from TableA

Rows from TableB

Single Kernel Thread

Kernel Thread Block

(CS220, Computer Organization)
(CS220, Ed Harcourt)
GPU Kernels
Step 4: Execute
Test Machine

- Intel Core® i7 920 CPU
  - Eight hardware threads @ 2.66 GHz
  - 12 GB RAM
  - Linux 3.2.0-61
- nVidia GTX460 GPU (x2)
  - 336 Cuda Cores
  - 1 GB Memory
Results

Query Suite

Execution Time

Single Core
Mapped GPU

0 1 2 3 4 5 6 7 8 9
Results

Increase source table size; Small result table size

Memory Test — Restrictive Predicate

![Graph showing Execution Time vs Rows in Source Tables for CPU and GPU.]

- **Execution Time**
- **Rows in Source Tables**
- **CPU**
- **GPU**
Constant source table size; Increase result table size

Memory Test — Proportional Predicate

Execution Time

Proportion of Cross Product in Result

CPU
GPU
Conclusions

- Memory is the bottleneck
  - Small result tables = Faster execution time

- For *reasonable* queries, the GPU is more efficient

- On average, queries execute **TWICE** as fast on GPU
References


Questions?