A GPU Database Architecture
How do you...

Use an interface you already know?

Accelerate things you’re already doing?

SQL on the GPU
Peter Bakkum

@pbbakkum

Research at NEC Labs
Algorithmic Trading at ITG
Engineer at Hyperpublic
Data Infrastructure at Groupon
Virginian

Developed at NEC Labs

SQL SELECT scans

Choose execution on the CPU or the GPU with CUDA

Common data format
SQL database

API

SQL parser/compiler

Common query format

CPU VM  GPU VM

Common data format
How do we...

Decompose SQL queries into discrete units?

Interface with the virtual machine?

Fit this into both the CPU and the GPU?

Opcodes
Opcodes

Simple operations
Control query flow
Use the context of the virtual machine
Access VM registers
Type agnostic
Abstract away parallelism
Execute within a kernel, not as a kernel
Opcode example

Add 2, 1, 0, null

 Opcode program

```
select col0 from test where (col0 = 2)
```

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Type</th>
<th>Parameters</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Table</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>ResultColumn</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Parallel</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Column</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Integer</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Eq</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>Invalid</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Result</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Converge</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Finish</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
How do we...

Store our data?
Handle fixed AND variable widths?
Ensure efficiency on both platforms?
Move to a new memory space?

Tablets
Tablets

Split the data into workable chunks

Use column major form to exploit both CPU cache locality and GPU coalescing
Tablets

Use relative pointers so variable width data can be moved

Flexible in terms of fixed vs variable width areas
Tablet management

Data tablets streamed or mapped into GPU memory
Result tablets streamed or mapped off
Effectively circumvent GPU memory limitation
Sequential memory copies

Entire data and results tablets must be moved
Can use streaming to overlap

![Diagram showing data transfer and query execution](image-url)
Mapped memory

Coalesced reads efficiently transfer data as it is needed. Results buffered in GPU memory, then moved in chunks.
Mapped Memory Performance

- **Serial**
  - Results Transfer
  - Kernel Execution
  - Data Transfer
  - Mapped Kernel

The graph shows the running time (in seconds) for different memory performance scenarios: Serial, Mapped, and Cached. The performance metrics include results transfer, kernel execution, data transfer, and mapped kernel execution.
Query Suite Running Times

Running Time (s)

Query

Multi-Core
Mapped GPU
Cached GPU
Rows Processed vs. Running Time

- Multi
- Mapped
- Cached

Running Time (s)

Data Rows (M)
Future work

Some database workloads are a good candidate for GPUs

Implementing an entire database on both platforms is difficult

Future hardware changes will shift the balance towards GPUs
Questions

Peter Bakkum
pbb7c@virginia.edu
@pbbakkum

github.com/bakks/virginian