GPU ACCELERATED DATABASES
Database Driven OpenCL Programming

Tim Child
3DMashUp
CEO
SPEAKERS BIO

- Tim Child
- 35 years experience of software development
- Formerly
  - VP Engineering, Oracle Corporation
  - VP Engineering, BEA Systems Inc.
  - VP Engineering, Informix
  - Leader at Illustra, Autodesk, Navteq, Intuit, ...
- 30+ years experience in 3D, CAD, GIS and DBMS
OUTLINE

- Speakers Biography
- Outline
- Solution Goals
- OpenCL Programming Challenge
- Review of GPU Accelerated Databases
- Swiss Army Knife of Data
- OpenCL Bindings to PostgreSQL
- Challenges
- Example Use Cases
- Benefits of the Approach
- Q&A
GOALS

- Develop New Applications
  - Develop new GPU Accelerated Database Applications that are computationally intensive.
- Ease of Use
  - Make use GPU accelerated code easier to use
  - Make GPU accelerated code more mainstream to Information Technology
- Data Scalability
  - Scale GPU application data size
- Enhance existing database internal operations
OPENCL PROGRAMMING CHALLENGE

- Write an OpenCL Application that:
  - Reads data from DBMS or File
  - Publishes Results as Web Pages
  - Handles Frequent Data Updates
  - Data Size >> System RAM >> GPU RAM

- Possible Solutions
  - C/C++ Binding using Web CGI
  - Java/Perl/Python Bindings in App Server
  - Other Choices ??

or

Database Driven GPU Programming
REVIEW OF GPU ACCELERATED DATABASE ARCHITECTURES
Examples
- 2004 Bandi, Sun, et al
- Many others
**Examples**
- 2008 Bakkum, Skardon
- 2010 Palo OLAP
- 2010 ParStream
- 2011 Kaczmarski
PROCEDURAL LANGUAGE ARCHITECTURE

Examples
- 1995 Illustra/Intel
- 2010 3DMashUp
SWISS ARMY KNIFE OF DATA

- Extensible Types
- SQL (Declarative Language, Set Operations)
- Extensible Procedural Languages (Java, Perl, ...)
- Rules System
- Extensible Indices
- Open Source
- Vibrant Community
- Native API’s
- Remote Data Access
- PostGIS (Vector, Raster)
- OpenCL
SQL OPENCL TYPES

- Vector Types
  - cl_charX
  - cl_ucharX
  - cl_shortX
  - cl_ushortX
  - cl_floatX
  - cl_doubleX

- Images Types
  - image2d_t
  - Image3d_t

Create table opencltypes
(id serial,
 matrix cl_double4[4],
 image image2d)
);
Insert into opencltypes (matrix) values (' { '1,0,0,0', 0,1,0,0', '0,0,1,0', '0,0,0,1' } ' );
DATABASE DRIVEN OPENCL

<table>
<thead>
<tr>
<th>Web Browser</th>
<th>HTTP</th>
<th>Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>App Server</td>
</tr>
<tr>
<td></td>
<td>TCP/IP</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>PostgreSQL Client</td>
<td>TCP/IP</td>
<td>Disk I/O</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data Tables</td>
</tr>
</tbody>
</table>

PostgreSQL Server

SQL Statement

PgOpenCL SQL Procedure

PCIe x2 Bus

GPGPU

HTTP

TCP/IP
OPENCL BINDINGS TO POSTGRES SQL

```sql
CREATE or REPLACE FUNCTION VectorAdd(IN Id int[], IN a real[], IN B real[], OUT C real[] ) AS $BODY$

__kernel void VectorAdd( __global int * id,
   __global float *a,
   __global float *b,
   __global float *c)
{
    int i = get_global_id(0); /* Query OpenCL for the Array Subscript */
    c[i] = a[i] + b[i];
}

$BODY$
Language PgOpenCL;
Select VectorAdd(Id, a, c) from Vectors;
```
## COMPARISON TABLE

<table>
<thead>
<tr>
<th></th>
<th>Co-Process</th>
<th>GPU Hosted</th>
<th>Procedural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>Spatial Joins</td>
<td>OLAP</td>
<td>Generic</td>
</tr>
<tr>
<td>Queries</td>
<td>Specific Queries</td>
<td>Select statements</td>
<td>SQL</td>
</tr>
<tr>
<td>Data Types</td>
<td>SQL</td>
<td>SQL</td>
<td>SQL + OpenCL</td>
</tr>
<tr>
<td>Performance</td>
<td>IPC/RPC</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Scalability</td>
<td>Large</td>
<td>Limited</td>
<td>Large</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>Per Query</td>
<td>ETL database</td>
<td>Per Query</td>
</tr>
</tbody>
</table>
DATABASE DRIVEN OPENCL

Table

Select Table to Array

100's - 1000's of Threads (Kernels)

VectorAdd(A, B) Returns C

Copy

Unnest Array To Table

Table

Copy
**OPENCL TIME SERIES TYPE**

- Specialized OpenCL image2d_t Types
- 1 Sample (Pixel) High
- N Samples Wide
- Single Channel (CL_A)
- Float4 or Int32 Data Type

1 Px

<table>
<thead>
<tr>
<th>Int32</th>
<th>[</th>
<th>[</th>
<th>[</th>
<th>[</th>
</tr>
</thead>
</table>

CL\_UNSIGNED\_INT, CL\_INTENSITY

1 Px

<table>
<thead>
<tr>
<th>Float4</th>
<th>[</th>
<th>[</th>
<th>[</th>
<th>[</th>
</tr>
</thead>
</table>

CL\_FLOAT, CL\_INTENSITY

N Pixels

- Simplified Input Syntax
  - ‘1,2,3,4,5’
- Casts to / from Integer Arrays
### Time Series Data

34 Years IBM data in 3NF = 8734 records

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/11/2003</td>
<td>75.82</td>
<td>76.33</td>
<td>75.2</td>
<td>75.35</td>
<td>8119200</td>
</tr>
<tr>
<td>3/10/2003</td>
<td>77.45</td>
<td>77.45</td>
<td>75.5</td>
<td>75.7</td>
<td>6641300</td>
</tr>
<tr>
<td>3/7/2003</td>
<td>75.71</td>
<td>77.99</td>
<td>75.71</td>
<td>77.9</td>
<td>8129200</td>
</tr>
<tr>
<td>3/6/2003</td>
<td>77</td>
<td>77.78</td>
<td>76.7</td>
<td>77.07</td>
<td>5876300</td>
</tr>
<tr>
<td>3/5/2003</td>
<td>76.7</td>
<td>77.73</td>
<td>76.25</td>
<td>77.73</td>
<td>6658000</td>
</tr>
<tr>
<td>3/4/2003</td>
<td>77.6</td>
<td>77.75</td>
<td>76.53</td>
<td>76.7</td>
<td>5672200</td>
</tr>
<tr>
<td>3/3/2003</td>
<td>78.9</td>
<td>79</td>
<td>77.12</td>
<td>77.33</td>
<td>6618300</td>
</tr>
<tr>
<td>2/28/2003</td>
<td>77</td>
<td>78.47</td>
<td>77</td>
<td>77.95</td>
<td>6585200</td>
</tr>
<tr>
<td>2/27/2003</td>
<td>77.9</td>
<td>78.59</td>
<td>76.75</td>
<td>77.28</td>
<td>9926500</td>
</tr>
</tbody>
</table>

As Time Series = 34 Records, 6 Series Columns (~250 Values/Series)
EXAMPLE USE CASES

- GPU Accelerated Time Series
- 3D Content Management / GIS
  - Spatial Selections
  - Coordinate Transformations
  - Image Processing
- Bioinformatics
  - DNA & Protein Sequence Matching
- Database Internal Operations
  - Joins
  - Sorting
  - Query Planning
DEMO SCREEN 1

```sql
select * from opencl.platform;
--select * from opencl.device;
--select * from opencl.supportedimageformat;
```
DEMO SCREEN 2

```sql
select * from opencl.platform;
select * from opencl.devices;
--select * from opencl.supportedimageformat;
```

<table>
<thead>
<tr>
<th>platform integer</th>
<th>device integer</th>
<th>name text</th>
<th>vendor text</th>
<th>version text</th>
<th>Driver Version text</th>
<th>OpenCL C Version text</th>
<th>type text</th>
<th>Device Available boolean</th>
<th>Compiler Available boolean</th>
<th>Current Device boolean</th>
<th>Host Little Endian boolean</th>
<th>Device Little Endian boolean</th>
<th>profile text</th>
<th>external text</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>Intel(R) Core(TM) i3 CPU 3.3GHz</td>
<td>Intel Corporation</td>
<td>OpenCL 1.1</td>
<td>OpenCL 1.1</td>
<td>GPU</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>PULL_PF_U_X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>Intel(R) Core(TM) i3 CPU 3.3GHz</td>
<td>Intel Corporation</td>
<td>OpenCL 1.1</td>
<td>OpenCL 1.1</td>
<td>GPU</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>PULL_PF_U_X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DEMO SCREEN 3

```sql
select * from opengl.platform;
select * from opengl.device;
select * from opengl.supportedimageformat;
```
Example Time Series data and Kernel Function call:

```sql
CREATE OR REPLACE FUNCTION timeseriesFunc(
    in argIn opencl.timseriesfloat4) returns opencl.timseriesfloat4
AS body

```

```c
Sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE | CLK_ADDRESS_CLAMP | CLK_FILTER_LINEAR;

void timeseriesFunc:
    __read_only image2d_t argIn, __write_only image2d_t argOut
{
    /* OpenCL kernel code to copy each Time Series Element */
    int2 coord = (int2)(get_global_id(0), get_global_id(1));
    write_image(argOut, coord, read_image(argIn, sampler, coord) * 7.1956821);
}
```

Output pane:

| QUERY PLAN | 1 | Seq Scan on timeseries (cost=0.00..119.20 rows=482 width=22) (actual time=0.000..0.008)
| 2 | Total runtime: 0.708 ms

---

24 | GPU Accelerated Databases | June 6, 2011
DEMOSCREEN 5

```c
/* OpenCL Kernel code to copy each Time Series Element */
int2 coord = (int2)( get_global_id(0), get_global_id(1) );
write_imagef( argout, coord, read_imagef( argin, sampler, coord ) , 7.16042E1 );

// C++ language: timeseries

select opencl.GetCurrentKRange("2 0 6 0 8 4 4");

drop table if exists timeseries;

create table timeseries

<table>
<thead>
<tr>
<th>id</th>
<th>serial</th>
<th>-- Sequential Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbol</td>
<td>text,</td>
<td>-- Time Series Name Symbol</td>
</tr>
<tr>
<td>volume</td>
<td>opencl.timeseriesfloat4,</td>
<td>-- Specialized OpenCL Image Type</td>
</tr>
<tr>
<td>coeffs</td>
<td>opencl.cl_double,</td>
<td>-- OpenCL Double Vector</td>
</tr>
<tr>
<td>bitmap</td>
<td>opencl.image2d</td>
<td>-- OpenCL Image 2D</td>
</tr>
</tbody>
</table>

insert into timeseries (volume, symbol, coeffs) values ('1.1.2.2.3.3.6.4.5.6.6.6.6.6', 'AMD', '7.3, 8.1, 9.1, 10.1, 11.1, 12.1, 13.1, 14.1');

explain analyze select timeseriesfunc(volume) from timeseries;
```

**QUERY PLAN**

1. Seq Scan on timeseries (cost=0.00..119.20 rows=50 width=32) (actual time=0.088)
2. Total runtime: 0.098 ms
CHALLENGES

- Type Mapping
  - Extend SQL Types with
    - OpenCL Vectors Types
    - OpenCL Image Types
- Setup -> Runtime
  - Caching kernel info
- Data Transfer
  - CPU ↔ GPU
    - Still present
  - SQL Queries
    - + Δ Overhead (< 4μs)
    - Map – Array
  - Bulk Data Loaders
    - New Task
- Problem Size
  - DBMS Table Size >> GPU RAM
  - # Work Groups / # Work Items
    - Runtime Partitioning
- Device Management
  - CPU vs. GPU
    - Runtime Selection
- Concurrency
  - No Pre-emptive Multi-Tasking
    - Time-out Long Queries
    - Partitioning / Scheduling
BENEFITS OF THE APPROACH

OpenCL

High Performance Computing

- Persistence
- Sharing & Collaboration
- Multi-user & Web Apps

Computational Database Applications

PostgreSQL

- Increased Compute Power
- New Data Types
- Factor of N Speed-ups

Database Internal Operations

Open Source Release
Q&A

- PgOpenCL
  - Twitter @3DMashUp
  - Blog www.scribd.com/3dmashup

- OpenCL

  - www.khronos.org/opencl/
Disclaimer & Attribution
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. There is no obligation to update or otherwise correct or revise this information. However, we reserve the right to revise this information and to make changes from time to time to the content hereof without obligation to notify any person of such revisions or changes.

NO REPRESENTATIONS OR WARRANTIES ARE MADE WITH RESPECT TO THE CONTENTS HEREOF AND NO RESPONSIBILITY IS ASSUMED FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

ALL IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED. IN NO EVENT WILL ANY LIABILITY TO ANY PERSON BE INCURRED FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. All other names used in this presentation are for informational purposes only and may be trademarks of their respective owners.

The contents of this presentation were provided by individual(s) and/or company listed on the title page. The information and opinions presented in this presentation may not represent AMD’s positions, strategies or opinions. Unless explicitly stated, AMD is not responsible for the content herein and no endorsements are implied.