Introducing PgOpenCL
A New PostgreSQL Procedural Language
Unlocking the Power of the GPU!

By
Tim Child
Tim Child

- 35 years experience of software development

Formerly

- VP Oracle Corporation
- VP BEA Systems Inc.
- VP Informix
- Leader at Illustra, Autodesk, Navteq, Intuit, ...

- 30+ years experience in 3D, CAD, GIS and DBMS
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure Language</td>
<td>Language for SQL Procedures (e.g. PgPLSQL, Perl, TCL, Java, ... )</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit (highly specialized CPU for graphics)</td>
</tr>
<tr>
<td>GPGPU</td>
<td>General Purpose <strong>GPU</strong> (non-graphics programming on a GPU)</td>
</tr>
<tr>
<td>CUDA</td>
<td>Nvidia’s GPU programming environment</td>
</tr>
<tr>
<td>APU</td>
<td>Accelerated Processing Unit (AMD’s Hybrid CPU &amp; GPU chip)</td>
</tr>
<tr>
<td>ISO C99</td>
<td>Modern standard version of the <strong>C</strong> language</td>
</tr>
<tr>
<td>OpenCL</td>
<td>Open Compute Language</td>
</tr>
<tr>
<td>OpenMP</td>
<td>Open Multi-Processing (parallelizing compilers)</td>
</tr>
<tr>
<td>SIMD</td>
<td>Single Instruction Multiple Data (Vector instructions)</td>
</tr>
<tr>
<td>SSE</td>
<td><strong>x86, x64</strong> (Intel, AMD) Streaming SIMD Extensions</td>
</tr>
<tr>
<td><strong>xPU</strong></td>
<td>Any Processing Unit device (CPU, GPU, APU)</td>
</tr>
<tr>
<td>Kernel</td>
<td>Functions that execute on a OpenCL Device</td>
</tr>
<tr>
<td>Work Item</td>
<td>Instance of a Kernel</td>
</tr>
<tr>
<td>Workgroup</td>
<td>A group of Work Items</td>
</tr>
<tr>
<td>FLOP</td>
<td>Floating Point <strong>Operation</strong> (single = SQL real type)</td>
</tr>
<tr>
<td>MIC</td>
<td>Many Integrated Cores (Intel’s 50+ x86 Core chip architecture)</td>
</tr>
</tbody>
</table>
Some Technology Trends Impacting DBMS

- **Solid State Storage**
  - Reduced Access Time, Lower Power, Increasing in capacity
- **Virtualization**
  - Server consolidation, Specialized VM’s, lowers direct costs
- **Cloud Computing**
  - EC2, Azure, … lowers capital requirements
- **Multi-Core**
  - 2,4,6,8, 12, …. Lots of benefits to multi-threaded applications
- **xPU (GPU/APU)**
  - GPU >1000 Cores
  - > 1T FLOP /s @ €2500
  - APU = CPU + GPU Chip Hybrids due in Mid 2011
  - 2 T FLOP /s for $2.10 per hour (AWS EC2)
  - Intel MIC “Knights Corner “ > 50 x86 Cores
Compute Intensive **xPU** Database Applications

- Bioinformatics
- Signal/Audio/Image Processing/Video
- Data Mining & Analytics
- Searching
- Sorting
- Spatial Selections and Joins
- Map/Reduce
- Scientific Computing
- Many Others ...
## GPU vs CPU

<table>
<thead>
<tr>
<th>Vendor Architecture</th>
<th>NVidia Fermi</th>
<th>ATI Radeon Evergreen</th>
<th>Intel Nehalem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores</td>
<td>448 Simple</td>
<td>1600 Simple</td>
<td>4 Complex</td>
</tr>
<tr>
<td>Transistors</td>
<td>3.1 B</td>
<td>2.15 B</td>
<td>731 M</td>
</tr>
<tr>
<td>Clock</td>
<td>1.5 G Hz</td>
<td>851 M Hz</td>
<td>3 G Hz</td>
</tr>
<tr>
<td>Peak Float Performance</td>
<td>1500 G FLOP / s</td>
<td>2720 G FLOP / s</td>
<td>96 G FLOP / s</td>
</tr>
<tr>
<td>Peak Double Performance</td>
<td>750 G FLOP / s</td>
<td>544 G FLOP / s</td>
<td>48 G FLOP / s</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>~ 190 G / s</td>
<td>~ 153 G / s</td>
<td>~ 30 G / s</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>250 W</td>
<td>&gt; 250 W</td>
<td>80 W</td>
</tr>
<tr>
<td>SIMD / Vector Instructions</td>
<td>Many</td>
<td>Many</td>
<td>SSE4+</td>
</tr>
</tbody>
</table>
Multi-Core Performance

Source NVidia
Future (Mid 2011)
APU Based PC

APU (Accelerated Processing Unit)

- APU Chip
  - CPU
  - CPU
  - North Bridge
    - Embedded GPU
      - ~20 GB/s
  - PCIE ~12 GB/s

- System RAM
  - ~20 GB/s

- Graphic RAM
  - 150 GB/s

- APU’s
  - Adds an Embedded GPU

Source AMD
Scalar vs. SIMD

Scalar Instruction
\[ C = A + B \]

SIMD Instruction
Vector \( C = \text{Vector } A + \text{Vector } B \)

OpenCL
Vector lengths \(2,4,8,16\) for char, short, int, float, double
Summarizing xPU Trends

• Many more xPU Cores in our Future
• Compute Environment becoming Hybrid
  – CPU and GPU’s
  – Need CPU to give access to GPU power
• GPU Capabilities
  – Lots of cores
  – Vector/SIMD Instructions
  – Fast Memory
• GPU Futures
  – Virtual Memory
  – Multi-tasking / Pre-emption
Scaling PostgreSQL Queries on xPU’s

Multi-Core CPU

Many Core GPU

Using More Transistors
## Parallel Programming Systems

<table>
<thead>
<tr>
<th>Category</th>
<th>CUDA</th>
<th>OpenMP</th>
<th>OpenCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>C</td>
<td>C, Fortran</td>
<td>C</td>
</tr>
<tr>
<td>Cross Platform</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Standard</td>
<td>Vendor</td>
<td>OpenMP</td>
<td>Khronos</td>
</tr>
<tr>
<td>CPU</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GPU</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Clusters</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Compilation / Link</td>
<td>Static</td>
<td>Static</td>
<td>Dynamic</td>
</tr>
</tbody>
</table>
What is OpenCL?

• OpenCL - Open Compute Language
  – Subset of C 99
  – Open Specification
  – Proposed by Apple
  – Many Companies Collaborated on the Specification
  – Portable, Device Agnostic
  – Specification maintained by Khronos Group

• PgOpenCL
  – OpenCL as a PostgreSQL Procedural Language
OpenCL Language

- **A subset of ISO C99**
  - But *without* some C99 features such as standard **C99 headers**,
  - function pointers, recursion, variable length arrays, and bit fields

- **A superset of ISO C99 with additions for:**
  - Work-items and Workgroups
  - Vector types
  - Synchronization
  - Address space qualifiers

- **Also includes a large set of built-in functions**
  - Image manipulation
  - Work-item manipulation,
  - Specialized math routines, etc.
PgOpenCL
Components

• New PostgreSQL Procedural Language
  – Language handler
    • Maps arguments
    • Calls function
    • Returns results
  – Language validator
    • Creates Function with parameter & syntax checking
    • Compiles Function to a Binary format

• New data types
  – cl_double4, cl_double8, ...

• System Admin Pseudo-Tables
  – Platform, Device, Run-Time, ...
```sql
select * from opencl.platform;
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>version</th>
<th>vendor</th>
<th>extensions</th>
<th>platform_profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ATI Stream</td>
<td>OpenCL 1.1 ATI-Stream v2.2</td>
<td>Advanced Micro Devices</td>
<td>FULL_PROFILE</td>
<td>d_ihr_jcd d_ihr_event_callback d_ihr_d3d10_sharing</td>
</tr>
<tr>
<td>2</td>
<td>Intel OpenCL</td>
<td>OpenCL 1.1 WINDOWS</td>
<td>Intel Corporation</td>
<td>FULL_PROFILE</td>
<td>d_ihr_fs54 d_ihr_global_int132 base_externals d_ihr__global_int32_extend</td>
</tr>
</tbody>
</table>
CREATE or REPLACE FUNCTION VectorAdd(IN a float[], IN B float[], OUT c float[]) AS $BODY$

#pragma PGOPENCL Platform : ATI Stream
#pragma PGOPENCL Device : CPU

__kernel __attribute__((reqd_work_group_size(64, 1, 1)))
void VectorAdd(__global const float *a, __global const float *b, __global float *c)
{
    int i = get_global_id(0);
    c[i] = a[i] + b[i];
}

$BODY$
Language PgOpenCL;
PgOpenCL Execution Model

Select Table to Array

Table
A
B

100’s - 1000’s of Threads (Kernels)

VectorAdd(A, B) Returns C

Copy

A  +  B  =  C

Copy

Unnest Array To Table

Table
C  C  C  C  C  C  C  C  C  C  C  C  C  C  C

Copy
Using Re-Shaped Tables

Table of Arrays

A
B

A
B

VectorAdd(A, B) Returns C

Copy

100’s - 1000’s of Threads (Kernels)

Table of Arrays

C
C
C
C
C

C
C
C
C
C

Copy

Copy
Today’s GPGPU Challenges

• No Pre-emptive Multi-Tasking
• No Virtual Memory
• Limited Bandwidth to discrete GPGPU
  – 1 – 8 G/s over PCIe Bus
• Hard to Program
  – New Parallel Algorithms and constructs
  – “New” C language dialect
• Immature Tools
  – Compilers, IDE, Debuggers, Profilers - early years
• Data organization really matters
  – Types, Structure, and Alignment
  – SQL needs to Shape the Data
• Profiling and Debugging is not easy

Solves Well for Problem Sets with the Right Shape!
Making a Problem Work for You

- Determine % Parallelism Possible

\[
\text{for } (i = 0, i < \infty, i++) \\
\text{for } (j = 0; j < \infty; j++) \\
\text{for } (k = 0; k < \infty; k++)
\]

- Arrange data to fit available GPU RAM
- Ensure \textit{calculation time} >> I/O transfer overhead
- Learn about \textbf{Parallel Algorithms} and the \textbf{OpenCL} language
- Learn new tools
- Carefully choose Data Types, Organization and Alignments
- Profile and Measure at Every Stage
PgOpenCL
System Requirements

• PostgreSQL 9.x

• For GPU’s
  – AMD ATI OpenCL Stream SDK 2.x
  – NVidia CUDA 3.x SDK
  – Recent Macs with O/S 11.6

• For CPU’s (Pentium M or more recent)
  – AMD ATI OpenCL Stream SDK 2.x
  – Intel OpenCL SDK Alpha Release (x86)
  – Recent Macs with O/S 11.6
• **Wish List**
  
  • **Beta Testers**
    – Existing OpenCL App?
    – Have a GPU App?
  
  • **Contributors**
    – Code server side functions?
  
  • **Sponsors & Supporters**
    – AMD Fusion Fund?
    – Khronos?
PgOpenCL
Future Plans

- Increase Platform Support
- Scatter/Gather Functions
- Additional Type Support
  - Image Types
  - Sparse Matrices
- Run-Time
  - Asynchronous
  - Events
  - Profiling
  - Debugging
Using the Whole Brain

You can’t be in a parallel universe with a single brain!

• Heterogeneous Compute Environments
  • CPU’s, GPU’s, APU’s
  • Expect 100’s – 1000’s of cores

The Future Is **Parallel**: What's a Programmer to Do?
Summarizing

PgOpenCL

Supports Heterogeneous **Parallel** Compute Environments

• CPU’s, GPU’s, APU’s

OpenCL

• **Portable** and high-performance framework
  – Ideal for computationally intensive algorithms
  – Access to all compute resources (CPU, APU, GPU)
  – Well-defined computation/memory model

• **Efficient** parallel programming language
  – C99 with extensions for **task** and **data parallelism**
  – Rich set of built-in functions

• **Open standard** for heterogeneous parallel computing

• **PgOpenCL**
  • Integrates PostgreSQL with OpenCL
  • Provides **Easy SQL Access** to xPU’s
    • APU, CPU, GPGPU
  • Integrates OpenCL
    • **SQL + Web Apps** (PHP, Ruby, ...)
More Information

- PGOpenCL
  - Twitter @3DMashUp

- OpenCL

- [www.khronos.org/opencl/](http://www.khronos.org/opencl/)


Q & A

- Using Parallel Applications?
- Benefits of OpenCL / PgOpenCL?
- Want to Collaborate on PgOpenCL?