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

By

Tim Child

Bio

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

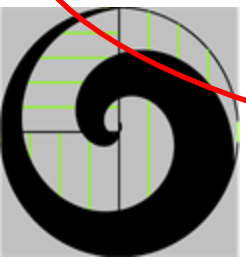
Terminology

Term	Description
Procedure Language	Language for SQL Procedures (e.g. PgPLSQL, Perl, TCL, Java, …)
GPU	Graphics Processing Unit (highly specialized CPU for graphics)
GPGPU	General Purpose **GPU** (non-graphics programming on a GPU)
CUDA	Nvidia's GPU programming environment
APU	Accelerated Processing Unit (AMD's Hybrid CPU & GPU chip)
ISO C99	Modern standard version of the **C** language
OpenCL	Open Compute Language
OpenMP	Open Multi-Processing (parallelizing compilers)
SIMD	Single Instruction Multiple Data (Vector instructions)
SSE	**x86, x64** (Intel, AMD) Streaming SIMD Extensions
xPU	**Any** Processing Unit device (CPU, GPU, APU)
Kernel	Functions that execute on a OpenCL Device
Work Item	Instance of a Kernel
Workgroup	A group of Work Items
FLOP	**Fl**oating Point **Op**eration (single = SQL real type)
MIC	Many Integrated Cores (Intel's 50+ x86 Core chip architecture)

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



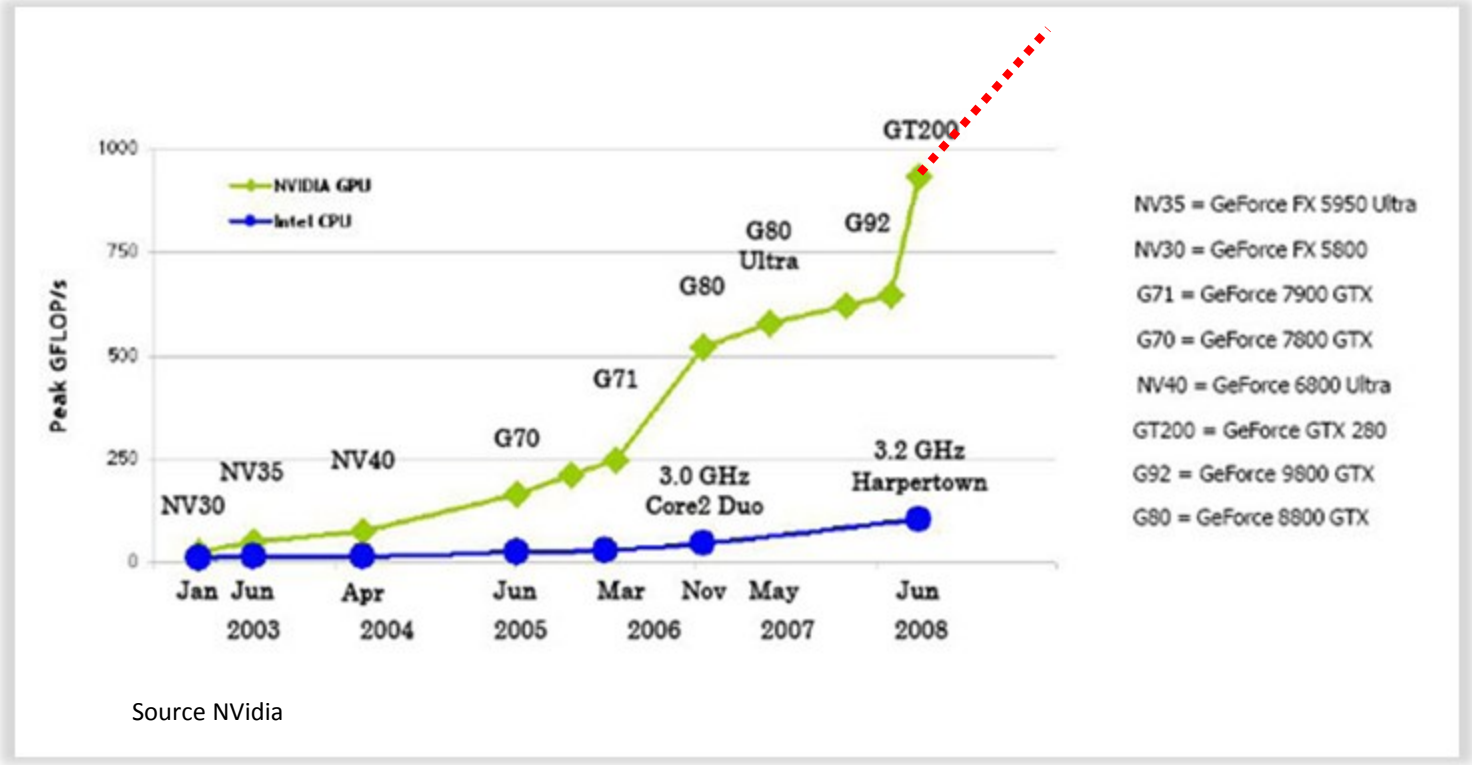
Compute Intensive **xPU** Database Applications

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

GPU vs CPU

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

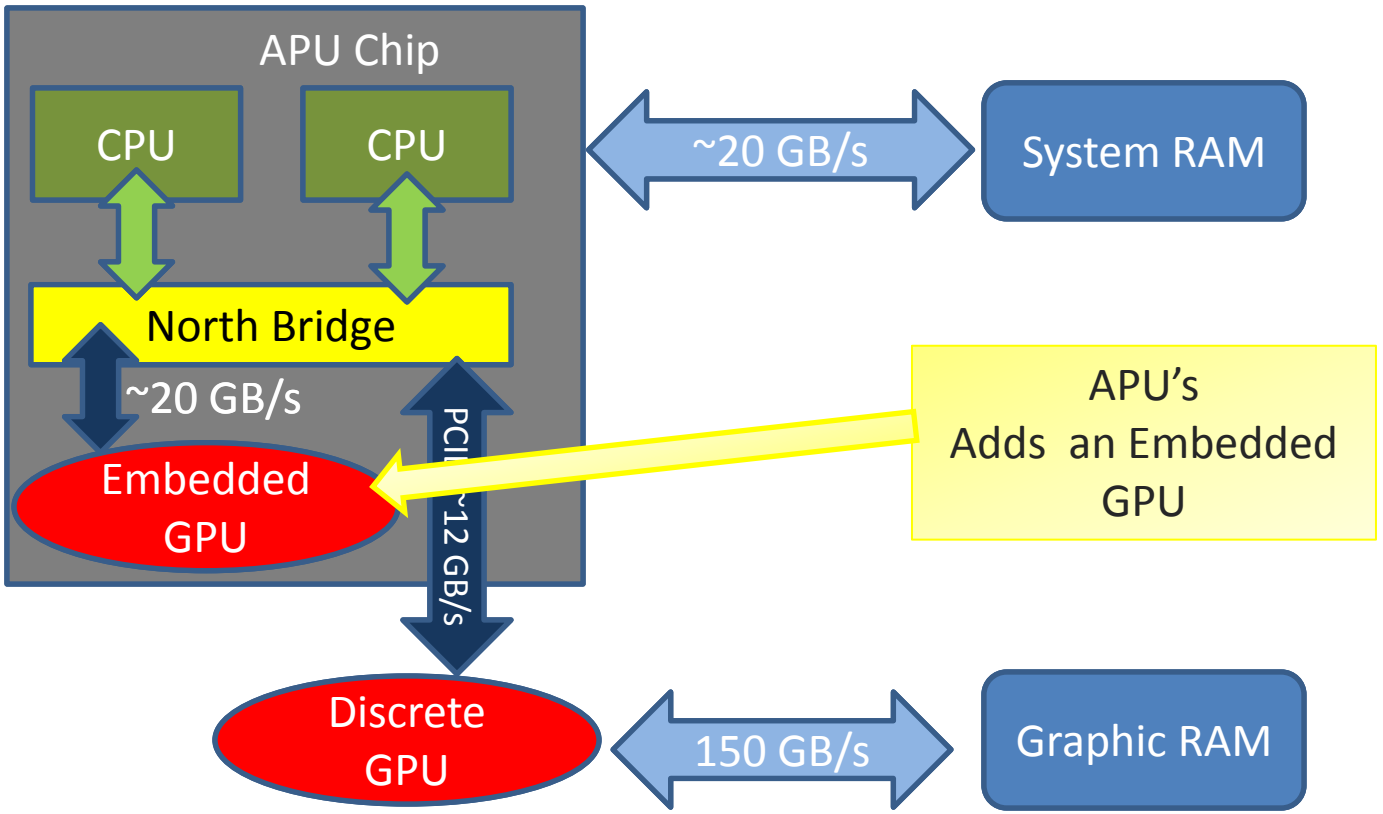
Multi-Core Performance



Source NVidia

Future (Mid 2011) APU Based PC

APU (Accelerated Processing Unit)



Source AMD

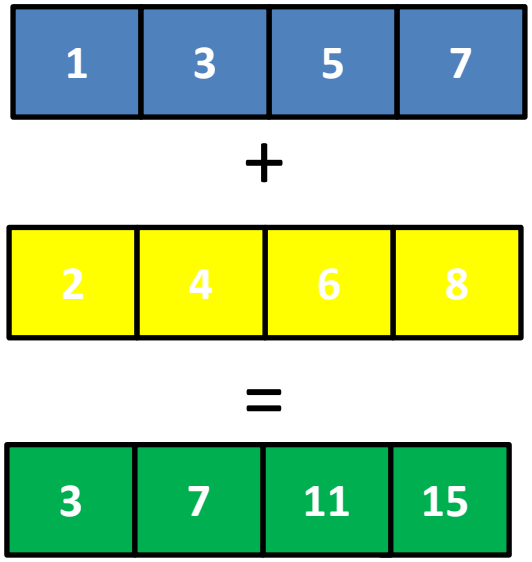
Scalar vs. SIMD

Scalar Instruction

C = A + B

SIMD Instruction

Vector C = Vector A + Vector B



OpenCL

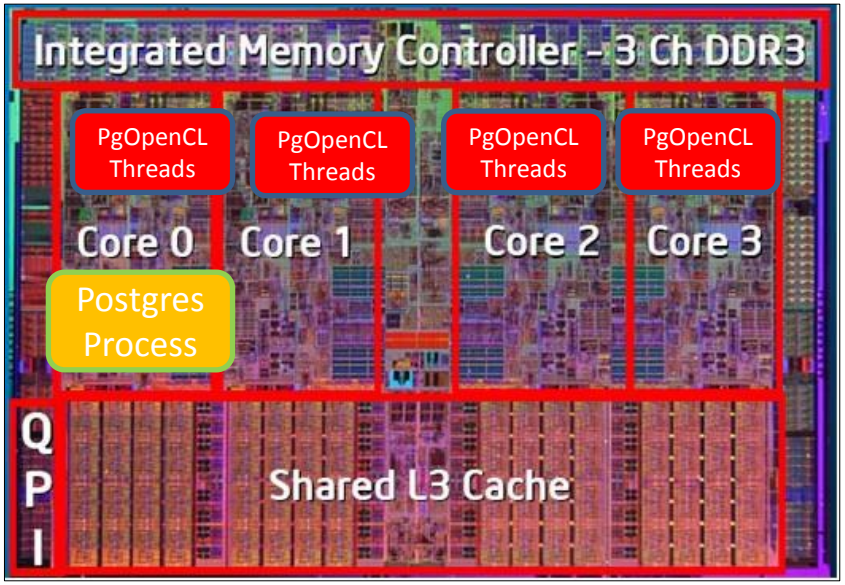
Vector lengths **2,4,8,16** for char, short, int, float, double

Summarizing xPU Trends

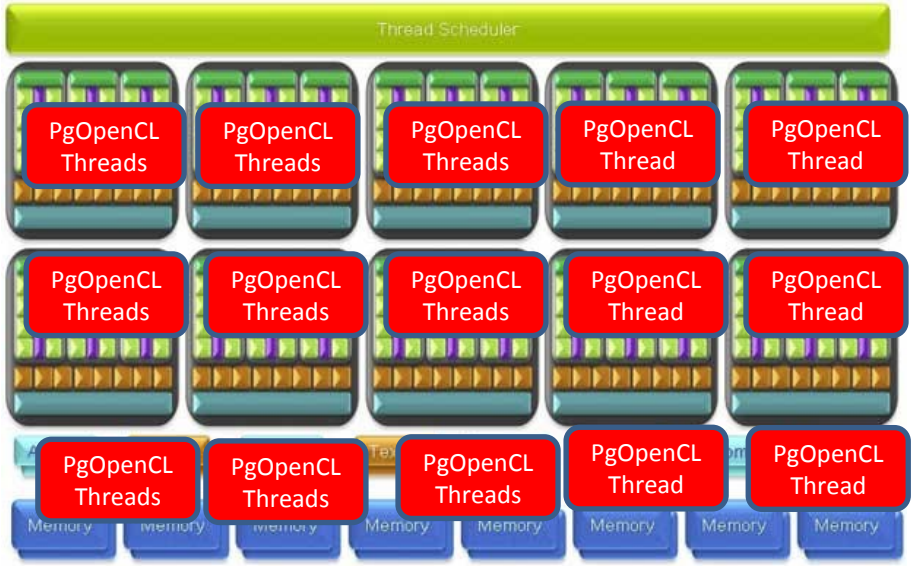
- Many more xPU Cores in our Future
- Compute Environment becoming Hybrid
  - CPU and GPU's
  - Need CPU to give access to GPU power
- GPU Capabilities
  - Lots of cores
  - Vector/SIMD Instructions
  - Fast Memory
- GPU Futures
  - Virtual Memory
  - Multi-tasking / Pre-emption

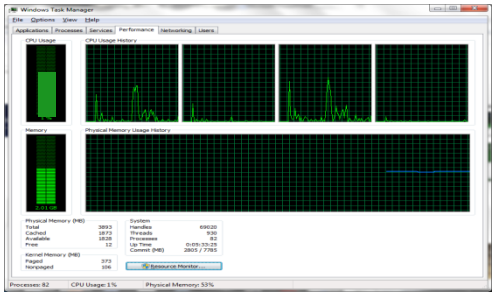
Scaling PostgreSQL Queries on **xPU's**

Multi-Core CPU

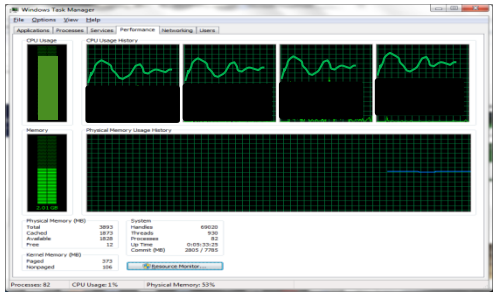


Many Core GPU





Using More Transistors



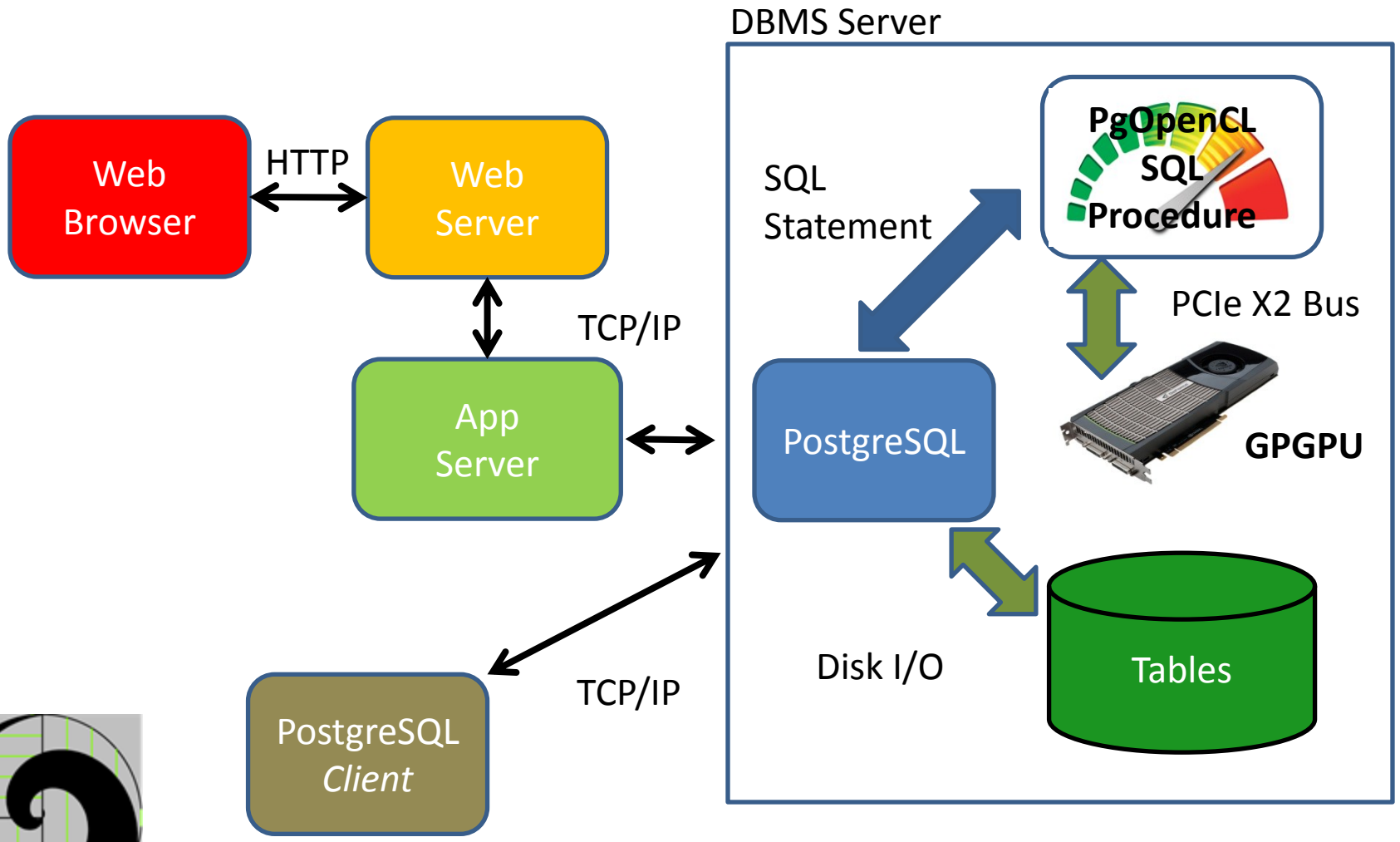
Parallel Programming Systems

Category	CUDA	OpenMP	OpenCL
Language	C	C, Fortran	C
Cross Platform	x	√	√
Standard	Vendor	OpenMP	Khronos
CPU	x	√	√
GPU	√	x	√
Clusters	x	√	x
Compilation / Link	Static	Static	Dynamic

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

System Overview



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, …

PgOpenCL Admin

Query - opencl on postgres@localhost:5433 *

File Edit Query Favourites Macros View Help

opencl on postgres@localhost:5433

SQL Editor | Graphical Query Builder

Delete | Delete All

```
select  * from opencl.platform;
```

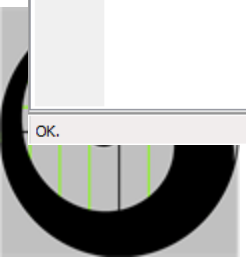
Scratch pad

Output pane

Data Output | Explain | Messages | History

	id integer	name text	version text	vendor text	extensions text	platform_profile text
1	0	ATI Stream	OpenCL 1.1 ATI-Stream-v2.2 (302)	Advanced Micro Devices	FULL_PROFILE	cl_khr_icd cl_amd_event_callback cl_khr_d3d10_sharing
2	1	Intel OpenCL	OpenCL 1.1 WINDOWS	Intel Corporation	FULL_PROFILE	cl_khr_fp64 cl_khr_global_int32_base_atomics cl_khr_global_int32_extend

OK. | Unix | Ln 2 Col 1 Ch 33 | 2 rows. | 15 ms

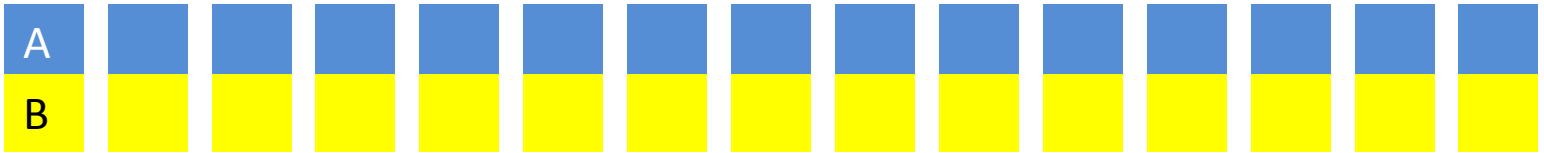


PGOpenCL Function Declaration

```
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;
```

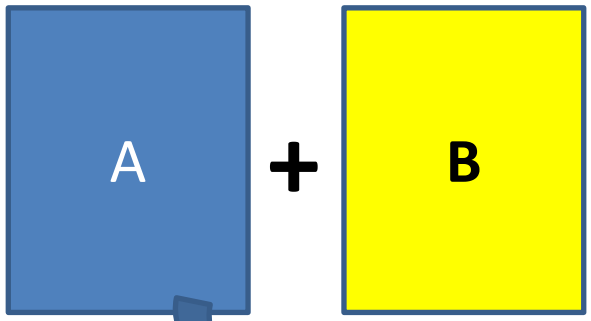
PgOpenCL Execution Model

Table

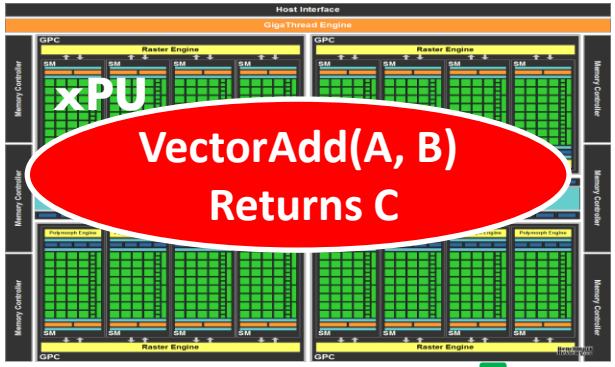


Select Table to Array

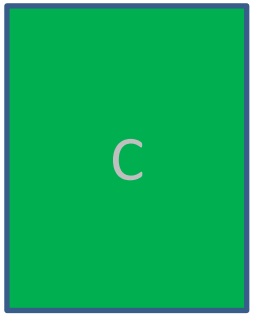
100's - 1000's of Threads (Kernels)



+



=

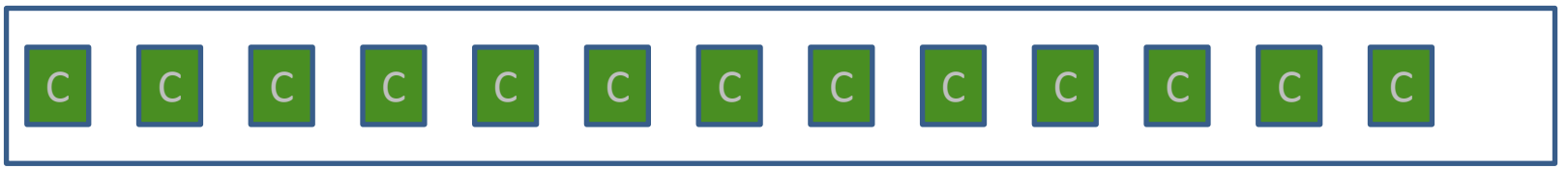


Copy

Copy

Unnest Array To Table

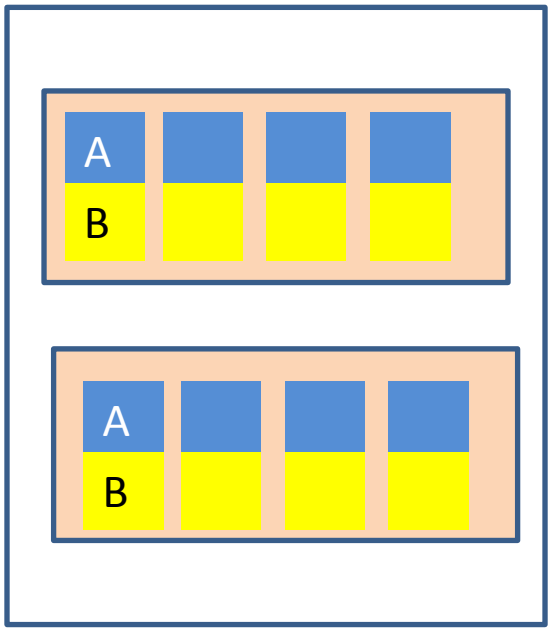
Table



Using Re-Shaped Tables

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

Table of Arrays



A + B = C

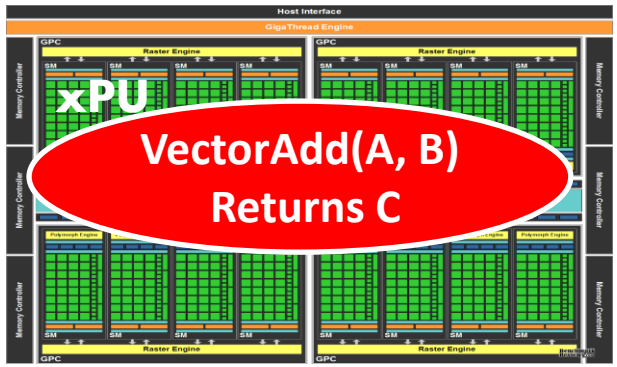
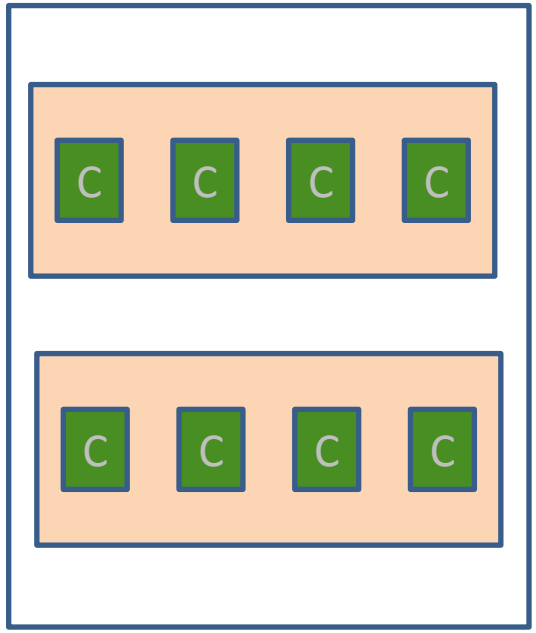


Table of Arrays



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***!

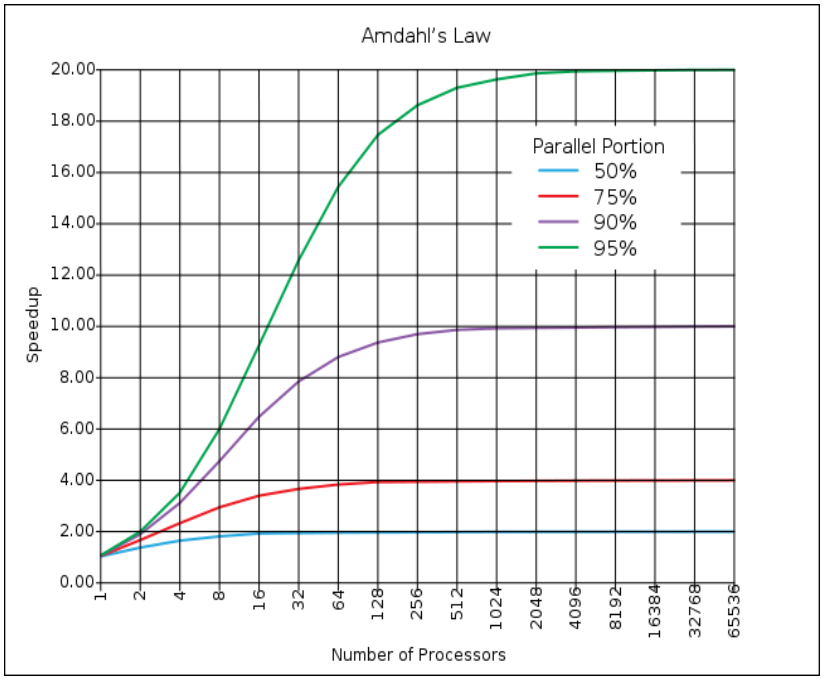
Making a Problem Work for You

- Determine % Parallelism Possible

for (i = 0, i < ∞, i++)

for (j = 0; j < ∞; j++)

for (k = 0; k < ∞; k++)

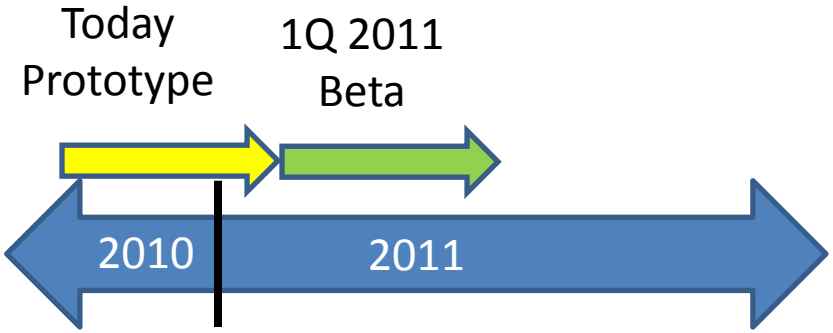


- Arrange data to fit available GPU RAM
- Ensure ***calculation time*** **>>** I/O transfer overhead
- Learn about **Parallel Algorithms** and the **OpenCL** language
- Learn new tools
- Carefully choose Data Types, Organization and Alignments
- Profile and Measure at Every Stage

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

PGOpenCL Status

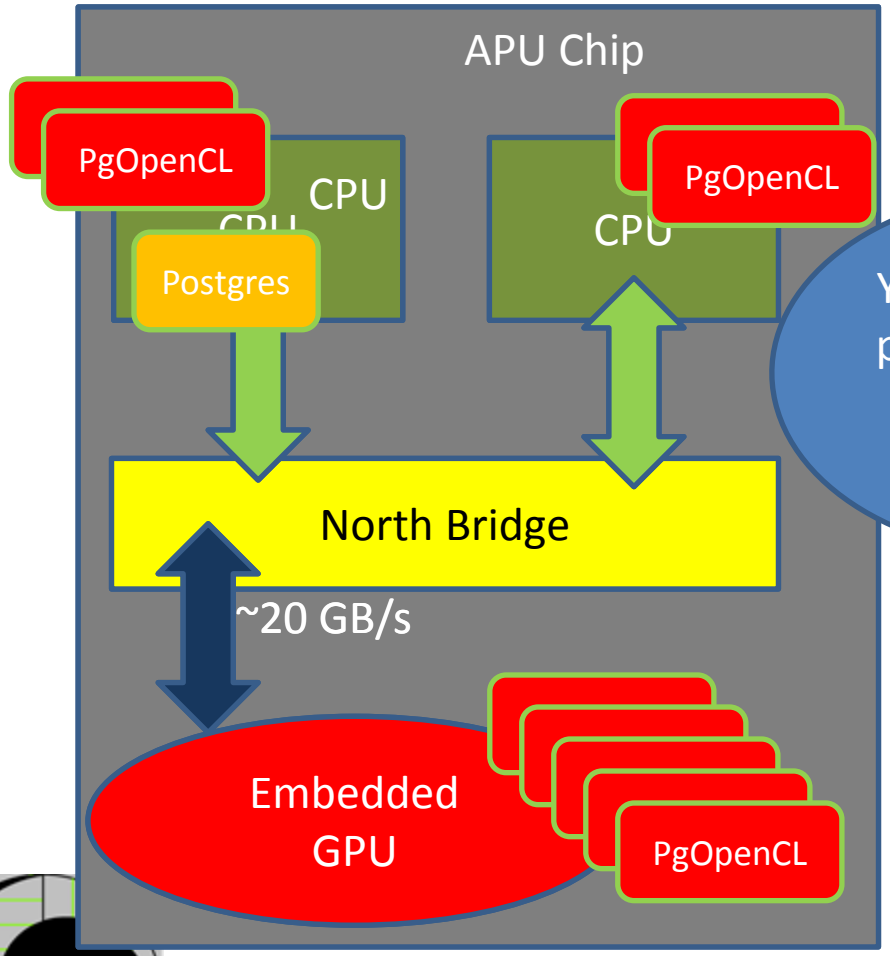


- **Wish List**
  - **Beta Testers**
    - Existing OpenCL App?
    - Have a GPU App?
  - **Contributors**
    - Code server side functions?
  - **Sponsors & Supporters**
    - AMD Fusion Fund?
    - Khronos?

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

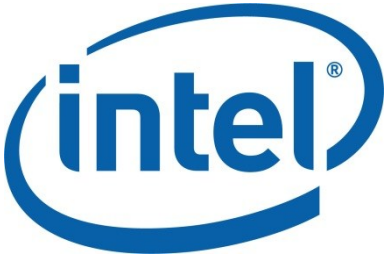


You can't be in a parallel universe with a single brain!

- 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

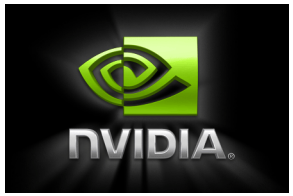


Supports Heterogeneous **Parallel** Compute Environments

- 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/ KHRONOS GROUP

AMD

- www.amd.com/us/products/technologies/stream-technology/opencl/

intel

- http://software.intel.com/en-us/articles/intel-opencl-sdk

NVIDIA

- http://www.nvidia.com/object/cuda_opencl_new.html

- http://developer.apple.com/technologies/mac/snowleopard/opencl.html

Q & A

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