Introduction to Accelerated/Hybrid Computing with GPGPU Architectures

Carlos J. Barrios H., PhD
@carlosjaimebh

www.red-ricap.org
GPU Computing is Powerful…

… but it’s not simple.
Who is Who in GPU Computing in LATAM?
Fastest Performance on Scientific Applications

Comparing Tesla K20X Speed-Up over Sandy Bridge CPUs

<table>
<thead>
<tr>
<th>Category</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering</td>
<td>MATLAB (FFT)*</td>
</tr>
<tr>
<td>Physics</td>
<td>Chroma</td>
</tr>
<tr>
<td>Earth Science</td>
<td>SPECFEM3D</td>
</tr>
<tr>
<td>Molecular Dynamics</td>
<td>AMBER</td>
</tr>
</tbody>
</table>

CPU results: Dual socket E5-2687w, 3.10 GHz, GPU results: Dual socket E5-2687w + 2 Tesla K20X GPUs
*MATLAB results comparing one i7-2600K CPU vs with Tesla K20 GPU
Disclaimer: Non-NVIDIA implementations may not have been fully optimized

© NVIDIA 2013
Interesting @SC3UIS Experiences

Processing and Visualization for Oil Reservoirs (3D seismic modelling in isotropic and heterogeneous media)

For 10 Millons of bases
0. Original App 3 Months
1. App (3 Weeks)
2. App (2- Days)
3. App (4 Minutes)

Processing Genomic Data for Mexican Flu AHN1 Discovering
### About Top500 List - 2018 June Ranking

- **9/10 Powerful Machines are MPP Clusters**
- **7/10 are Hybrid Machines with Accelerators**
  - **5 NVIDIA GPU Technology**
  - **3 Different Generations (Kepler, Pascal and Volta)**
  - **2 Intel Xeon Phi Technology**
  - **1 Combines GPUs + MICs**

<table>
<thead>
<tr>
<th>Rank</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (TFlop/s)</th>
<th>Rpeak (TFlop/s)</th>
<th>Power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Summit</td>
<td>3,292</td>
<td>122,200</td>
<td>197,459</td>
<td>6,004</td>
</tr>
<tr>
<td>2</td>
<td>TaihakuLight</td>
<td>10,649</td>
<td>93,014</td>
<td>125,435</td>
<td>15,371</td>
</tr>
<tr>
<td>3</td>
<td>Sierra</td>
<td>1,972</td>
<td>71,610</td>
<td>119,193</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Tianhe-2A</td>
<td>4,961</td>
<td>61,444</td>
<td>100,678</td>
<td>18,082</td>
</tr>
<tr>
<td>5</td>
<td>Al Bridging Cloud Infrastructure</td>
<td>391</td>
<td>19,080</td>
<td>22,976</td>
<td>1,649</td>
</tr>
<tr>
<td>6</td>
<td>Piz Daint</td>
<td>361</td>
<td>19,590</td>
<td>25,286</td>
<td>2,272</td>
</tr>
<tr>
<td>7</td>
<td>Titan</td>
<td>560</td>
<td>17,100</td>
<td>27,122</td>
<td>8,209</td>
</tr>
<tr>
<td>8</td>
<td>Sequoia</td>
<td>1,572</td>
<td>17,173</td>
<td>20,132</td>
<td>7,890</td>
</tr>
<tr>
<td>9</td>
<td>Trinity</td>
<td>979</td>
<td>16,127</td>
<td>23,908</td>
<td>3,844</td>
</tr>
<tr>
<td>10</td>
<td>Carri</td>
<td>622</td>
<td>16,014</td>
<td>27,880</td>
<td>3,991</td>
</tr>
</tbody>
</table>
Why Computing Perf/Watt Matters?

Traditional CPUs are not economically feasible

2.3 PFlops
7.0 Megawatts

7000 homes
7.0 Megawatts

CPU
Optimized for Serial Tasks

GPU Accelerator
Optimized for Many Parallel Tasks

10x performance/socket
> 5x energy efficiency

Era of GPU-accelerated computing is here

© NVIDIA 2013
World's Fastest, Most Energy Efficient Accelerator

Tesla K20X vs Xeon CPU
- 8x Faster SGEMM
- 6x Faster DGEMM

Tesla K20X vs Xeon Phi
- 90% Faster SGEMM
- 60% Faster DGEMM
10 Years NVIDIA GPUs Development

- Volta
- Pascal
- Maxwell
- Kepler
- Fermi
- Tesla
Remember Architectural Systems Facts (From Flynn’s Taxonomy)

**SPMD:** Parallel Processing Units execute the same program on multiple parts of data

**SIMD:** All processors units are executing the same instructions in any instant.
Massive Parallel Processing (MPP)

- Computer system with many independent arithmetic units or entire microprocessors, that run in parallel
- MPPA is a MIMD (Multiple Instruction streams, Multiple Data) architecture, with distributed memory accessed locally, not shared globally
CPUs and GPUs Architecture
Add GPUs: Accelerate Science Applications
Small Changes, Big Speed-up

Application Code

Compute-Intensive Functions

Use GPU to Parallelize

Rest of Sequential CPU Code

© NVIDIA 2013
NVIDIA TESLA® Architecture
NVIDIA TESLA™ Graphics and Computing Architecture Features

- TESLA™ shader processors are fully programmable
  - Large instructions memory
  - Cache Instructions
  - Logic Sequence Instructions
- TESLA™ to non-graphics programs:
  - Hierarchical Parallel Threads
  - Barrier Synchronization
  - Atomic Operators (Manage Highly Parallel Computing Work)
Heterogeneous Computing

- **Terminology:**
  - *Host*  The CPU and its memory (host memory)
  - *Device* The GPU and its memory (device memory)
GPGPU Accelerate Computing

Latency Processor + Throughput processor
Low Latency or High Throughput?

CPU
- Optimized for low-latency access to cached data sets
- Control logic for out-of-order and speculative execution

GPU
- Optimized for data-parallel, throughput computation
- Architecture tolerant of memory latency
- More transistors dedicated to computation
1. Copy input data from CPU memory to GPU memory
1. Copy input data from CPU memory to GPU memory
2. Load GPU program and execute, caching data on chip for performance
1. Copy input data from CPU memory to GPU memory
2. Load GPU program and execute, caching data on chip for performance
3. Copy results from GPU memory to CPU memory
CUDA Parallel Computing Platform

Programming Approaches
- Libraries
  - “Drop-in” Acceleration
- OpenACC Directives
  - Easily Accelerate Apps
- Programming Languages
  - Maximum Flexibility

Development Environment
- Nsight IDE
  - Linux, Mac and Windows
  - GPU Debugging and Profiling
- CUDA-GDB debugger
- NVIDIA Visual Profiler

Open Compiler Tool Chain
- Enables compiling new languages to CUDA platform, and CUDA languages to other architectures

Hardware Capabilities
- SMX
- Dynamic Parallelism
- HyperQ
- GPUDirect

© NVIDIA 2013
3 Ways to Accelerate Applications

- Libraries
  - “Drop-in” Acceleration

- OpenACC Directives
  - Easily Accelerate Applications

- Programming Languages
  - Maximum Flexibility

© NVIDIA 2013
3 Ways to Accelerate Applications

Applications

Libraries

OpenACC Directives

Programming Languages

“Drop-in” Acceleration

Easily Accelerate Applications

Maximum Flexibility

© NVIDIA 2013
Libraries: Easy, High-Quality Acceleration

- **Ease of use:** Using libraries enables GPU acceleration without in-depth knowledge of GPU programming
- **“Drop-in”:** Many GPU-accelerated libraries follow standard APIs, thus enabling acceleration with minimal code changes
- **Quality:** Libraries offer high-quality implementations of functions encountered in a broad range of applications
- **Performance:** NVIDIA libraries are tuned by experts
Some GPU-accelerated Libraries

- NVIDIA cuBLAS
- NVIDIA cuRAND
- NVIDIA cuSPARSE
- NVIDIA NPP

- GPU VSIPL
- CULA tools
- MAGMA
- NVIDIA cuFFT

- Rogue Wave Software
- ArrayFire
- CUSP
- Thrust

- IMSL Library
- Matrix Algebra on GPU and Multicore
- Sparse Linear Algebra
- C++ STL Features for CUDA
3 Steps to CUDA-accelerated application

- **Step 1:** Substitute library calls with equivalent CUDA library calls
  
  ```
  saxpy ( ... )  \rightarrow \text{cublasSaxpy} ( ... )
  ```

- **Step 2:** Manage data locality
  
  - with CUDA: `cudaMalloc()`, `cudaMemcpy()`, etc.
  - with CUBLAS: `cublasAlloc()`, `cublasSetVector()`, etc.

- **Step 3:** Rebuild and link the CUDA-accelerated library
  
  ```
  nvcc myobj.o -l cublas
  ```

© NVIDIA 2013
Explore the CUDA (Libraries) Ecosystem

Ways to Accelerate Applications

Applications

Libraries

OpenACC Directives

Programming Languages

“Drop-in” Acceleration

Easily Accelerate Applications

Maximum Flexibility

© NVIDIA 2013
OpenACC Directives

Program myscience
... serial code ...
!$acc kernels
  do k = 1,n1
    do i = 1,n2
      ... parallel code ...
    enddo
  enddo
!$acc end kernels
... end Program myscience

Simple Compiler hints

Compiler Parallelizes code

Works on many-core GPUs & multicore CPUs

Your original Fortran or C code
OpenACC
The Standard for GPU Directives

• **Easy:** Directives are the easy path to accelerate compute intensive applications

• **Open:** OpenACC is an open GPU directives standard, making GPU programming straightforward and portable across parallel and multi-core processors

• **Powerful:** GPU Directives allow complete access to the massive parallel power of a GPU
Start Now with OpenACC Directives

Free trial license to PGI Accelerator Tools for quick ramp

https://developer.nvidia.com/openacc
Ways to Accelerate Applications

Applications

Libraries

“Drop-in” Acceleration

OpenACC Directives

Easily Accelerate Applications

Programming Languages

Maximum Flexibility + Best Performance
// generate 32M random numbers on host
thrust::host_vector<int> h_vec(32 << 20);
thrust::generate(h_vec.begin(), h_vec.end(), rand);

// transfer data to device (GPU)
thrust::device_vector<int> d_vec = h_vec;

// sort data on device
thrust::sort(d_vec.begin(), d_vec.end());

// transfer data back to host
thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());

Rapid Parallel C++ Development

• Resembles C++ STL
• High-level interface
  • Enhances developer productivity
  • Enables performance portability between GPUs and multicore CPUs
• Flexible
  • CUDA, OpenMP, and TBB backends
  • Extensible and customizable
  • Integrates with existing software
• Open source

http://developer.nvidia.com/thrust
Learn More

These languages are supported on all CUDA-capable GPUs.
You might already have a CUDA-capable GPU in your laptop or desktop PC!

CUDA C/C++

Thrust C++ Template Library
http://developer.nvidia.com/thrust

CUDA Fortran

PyCUDA (Python)
http://mathema.tician.de/software/pycuda

GPU.ORG Different Ressources
http://gpgpu.org

MATLAB
http://www.mathworks.com/discovery/matlab-gpu.html

Mathematica
http://www.wolfram.com/gridmathematica/
Getting Started

• Download CUDA Toolkit & SDK: https://developer.nvidia.com/cuda-downloads
• Nsight IDE (Eclipse or Visual Studio): http://www.nvidia.com/object/nsight.html

• General GPU Computing Community: http://gpgpu.org/
• Programming Guide/Best Practices:
  • docs.nvidia.com

• Questions:
  • NVIDIA Developer forums: devtalk.nvidia.com
  • Search or ask on: www.stackoverflow.com/tags/cuda

• Developer Community: https://developer.nvidia.com/ (Join Now!)
The OpenACC Toolkit
Introducing the New OpenACC Toolkit

- **PGI Compiler**
  Free OpenACC compiler for academia

- **NVProf Profiler**
  Easily find where to add compiler directives

- **GPU Wizard**
  Identify which GPU libraries can jumpstart code

- **Code Samples**
  Learn from examples of real-world algorithms

- **Documentation**
  Quick start guide, Best practices, Forums

http://developer.nvidia.com/openacc
Download the OpenACC Toolkit

Go to: https://developer.nvidia.com/openacc
Download the OpenACC Toolkit

- Go to https://developer.nvidia.com/openacc
- Register for the toolkit
  - If you are an academic developer, be sure to click the check box at the bottom.
Download the OpenACC Toolkit

- Go to https://developer.nvidia.com/openacc
- Register for the toolkit
  - If you are an academic developer, be sure to click the check box at the bottom.
- You will receive an email from NVIDIA
  - Be sure to read the Quick Start Guide
Windows/Mac Developers

- The OpenACC Toolkit is only available on Linux, however...
- The PGI compiler is available on Mac and Windows from [http://www.pgroup.com/support/trial.htm](http://www.pgroup.com/support/trial.htm)
  - You should still register for the OpenACC Toolkit to get the 90 day license.
- The CUDA Toolkit contains the libraries and profiling tools that will be used in this course.
  - Obtaining all examples and guides from the toolkit will still require downloading the full OpenACC toolkit.
Install the OpenACC Toolkit

- Go to developer.nvidia.com/openacc
- Register for the OpenACC Toolkit
- Install on your personal machine. (Linux Only)
Where to find help

- OpenACC on StackOverflow - http://stackoverflow.com/questions/tagged/openacc

Additional Resources:
- OpenACC Website - http://openacc.org/
Acknowledgements

- Sunita Chandrasekaran, University of Delaware
- Guido Juckeland, Helmholtz-Zentrum Dresden-Rossendorf (HZDR)
- Fernanda Foertter, Oak Ridge National Laboratory
- Joe Bongo, NVIDIA Deep Learning Institute
Thank you!

@carlosjaimebh