HPC with the NVIDIA Accelerated Computing Toolkit

Mark Harris, November 16, 2015
Accelerators Surge in World’s Top Supercomputers

- 100+ accelerated systems now on Top500 list
- 1/3 of total FLOPS powered by accelerators
- NVIDIA Tesla GPUs sweep 23 of 24 new accelerated supercomputers
- Tesla supercomputers growing at 50% CAGR over past five years
70% OF TOP HPC APPS NOW ACCELERATED

INTERSECT360 SURVEY OF TOP APPS

Top 10 HPC Apps
90% Accelerated

Top 50 HPC Apps
70% Accelerated

KEY MATERIALS APP NOW ACCELERATED

Typically Consumes 10-25% of HPC System

1 Dual K80 Server
1.3x

4 Dual CPU Servers
1.0x

Dual-socket Xeon E5-2690 v2 3GHz, Dual Tesla K80, FDR InfiniBand Dataset: NiAl-MD Blocked Davidson

Intersect360, Nov 2015 “HPC Application Support for GPU Computing”
Accelerated Computing Available Everywhere

NVIDIA Accelerated Computing Toolkit

Supercomputer  Datacenter - Cloud  Workstation - Notebook  Embedded - Automotive
NVIDIA Accelerated Computing Toolkit
Everything You Need to Accelerate Your Applications

Libraries
- FFTW
- BLAS
- DNN

Language Solutions
- C/C+
- Fortran
- OpenACC

Tools
- NSight IDE
- Visual Profiler

System Interfaces
- Unified Memory
- GPUDirect

Training
- Hands-on Labs
- Docs & Samples

developer.nvidia.com
Get Started Quickly

Machine Learning Tools
Design, Train and Deploy Deep Neural Networks
developer.nvidia.com/deep-learning

OpenACC Toolkit
Simple way to accelerate scientific code
developer.nvidia.com/openacc

CUDA Toolkit
High-Performance C/C++ Applications
developer.nvidia.com/cuda-toolkit
Why is Machine Learning Hot Now?

Big Data Availability

- Facebook: 350 millions images uploaded per day
- Walmart: 2.5 Petabytes of customer data hourly
- YouTube: 300 hours of video uploaded every minute

New Techniques

GPU Acceleration

IMAGENET
cuDNN

Deep Learning Primitives

Igniting Artificial Intelligence

- Drop-in Acceleration for major Deep Learning frameworks: Caffe, Theano, Torch
- Up to 2x faster on AlexNet than baseline Caffe GPU implementation

Applications

Frameworks

Caffe

Theano

cuDNN

NVIDIA GPU

Millions of images trained per day

Train up to 2X faster than cuDNN 2
Tesla M40

World’s Fastest Accelerator for Deep Learning

8x Faster Caffe Performance

Reduce Training Time from 10 Days to 1 Day

- CUDA Cores: 3072
- Peak SP: 7 TFLOPS
- GDDR5 Memory: 12 GB
- Bandwidth: 288 GB/s
- Power: 250W

Caffe Benchmark: AlexNet training throughput based on 20 iterations, CPU: E5-2680v3 @ 2.7GHz. 64GB System Memory, CentOS 6.2
NVIDIA® DIGITS™
Interactive Deep Learning
GPU Training System

Data Scientists and Researchers:
Quickly design the best deep neural network (DNN) for your data
Visually monitor DNN training quality in real-time
Manage training of many DNNs in parallel on multi-GPU systems

developer.nvidia.com/digits
OpenACC

Simple | Powerful | Portable

Fueling the Next Wave of Scientific Discoveries in HPC

```c
main()
{
    <serial code>
    #pragma acc kernels
    //automatically run on GPU
    {
        <parallel code>
    }
}
```

RIKEN Japan
NICAM- Climate Modeling

7-8x Speed-Up
5% of Code Modified

University of Illinois
PowerGrid: MRI Reconstruction

70x Speed-Up
2 Days of Effort

Performance Portability with PGI

- CPU: MPI + OpenMP
- CPU: MPI + OpenACC
- CPU + GPU: MPI + OpenACC

Speedup vs. Single CPU Thread

- miniGhost: CPU: Intel Xeon E5-2698 v3, 2 sockets, 32-cores total, GPU: Tesla K80 (single GPU)
- NEMO: Each socket CPU: Intel Xeon E5-2698 v3, 16 cores; GPU: NVIDIA K80 both GPUs
- CLOVERLEAF: CPU: Dual socket Intel Xeon CPU E5-2690 v2, 20 cores total, GPU: Tesla K80 both GPUs

http://www.cray.com/sites/default/files/resources/OpenACC_213462.12_OpenACC_Cosmo_CS_FNL.pdf
http://www.openacc.org/content/experiences-porting-molecular-dynamics-code-gpus-cray-xk7
OpenACC makes GPU computing approachable for domain scientists. Initial OpenACC implementation required only minor effort, and more importantly, no modifications of our existing CPU implementation.

Large-scale Application for Calculating High-accuracy Molecular Energies

Big Performance

LS-DALTON CCSD(T) Module
Benchmarked on Titan Supercomputer (AMD CPU vs Tesla K20X)

<table>
<thead>
<tr>
<th></th>
<th>Alanine-1</th>
<th>Alanine-2</th>
<th>Alanine-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atoms</td>
<td>13</td>
<td>23</td>
<td>33</td>
</tr>
<tr>
<td>Speedup vs CPU</td>
<td>12.0x</td>
<td>8.0x</td>
<td>4.0x</td>
</tr>
</tbody>
</table>

Janus Juul Eriksen, PhD Fellow
qLEAP Center for Theoretical Chemistry, Aarhus University
The NVIDIA OpenACC Toolkit
Free Toolkit Offers Simple & Powerful Path to Accelerated Computing

Download at developer.nvidia.com/openacc
CUDA Toolkit
Flexibility - High Performance

LIBRARIES

PARALLEL C++

DEVELOPER TOOLS

Drop-In Acceleration

Parallel Algorithms and Custom Parallel C++

Pinpoint Bugs and Performance Opportunities

__global__
void xyzw_frequency(int *count, char *text, int n)
{
    const char letters[] { 'x','y','z','w' };
    count_if(count, text, n, [&](char c) {
        for (const auto x : letters)
            if (c == x) return true;
        return false;
    });
}

Download at developer.nvidia.com/cuda-toolkit
GPU Accelerated Libraries
Drop-in Acceleration for Your Applications

Linear Algebra & Numerical
- cuBLAS
- cuSparse
- cuSOLVER
- AmgX
- NVIDIA Math Lib
- cuRAND
- SuiteSparse CHOLMOD

Parallel Algorithms
- Thrust
- ArrayFire

Signal Processing
- FFT, Image & Video
- cuFFT
- NVENC Video Encode
- NVIDIA NPP
- FFmpeg

Machine Learning
- cuDNN

developer.nvidia.com/gpu-accelerated-libraries
Developer Tools
Pinpoint Performance Bottlenecks and Hard-to-Find Bugs

NVIDIA Visual Profiler
Analyze GPU Performance

CUDA-GDB
Standard Linux Debugger

CUDA-MEMCHECK
Find memory errors, leaks and race conditions

Develop, Debug, & Profile in Eclipse or Visual Studio
Your Next Steps
Many Learning Opportunities

Free Hands-On Self-Paced Labs in the NVIDIA Booth

Sign up for credit for on-line labs

Free In-Depth On-Line Courses

Deep Learning: developer.nvidia.com/deep-learning-courses

OpenACC: developer.nvidia.com/openacc-course

Learn from hundreds of experts at GTC 2016

All about Accelerated Computing: developer.nvidia.com
Backup
THRUST: Productive Parallel C++
High-level parallel algorithms

Resembles C++ STL

Included in CUDA Toolkit

Productive High-level API

CPU/GPU Performance portability

CUDA, OpenMP, and TBB backends

Flexible, extensible

// 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());
C++11 in CUDA

“C++11 Feels like a new language” - Bjarne Stroustrup

Count the number of occurrences of letters x, y, z and w in text

```cpp
__global__
void xyzw_frequency(int *count, char *text, int n) {
    const char letters[] { 'x', 'y', 'z', 'w' };

    count_if(count, text, n, [&](char c) {
        for (const auto x : letters) {
            if (c == x) return true;
        }
        return false;
    });
}
```

Output:

Read 3288846 bytes from "warandpeace.txt" counted 107310 instances of 'x', 'y', 'z', or 'w' in "warandpeace.txt"
Parallel STL: Algorithms + Policies

Higher-level abstractions for ease of use and performance portability

```cpp
int main()
{
    size_t n = 1 << 16;
    std::vector<float> x(n, 1), y(n, 2), z(n);
    float a = 13;
    auto I = interval(0, n);

    std::for_each(std::par, std::begin(I), std::end(I),
                  [&](int i)
                  {
                      z[i] = a * x[i] + y[i];
                  });

    return 0;
}
```

On track for C++17 Standard

Includes algorithms such as for_each, reduce, transform, inclusive_scan ...

Execution policies:
- `std::seq`
- `std::par`
- `std::par_vec`
Instruction-Level Profiling

Pinpoint Performance Problems

New in CUDA 7.5 Visual Profiler

Highlights hot spots in code, helping you target bottlenecks

Quickly locate expensive code lines

Detailed per-line stall analysis helps you understand underlying limiters
Unified Memory
Dramatically Lower Developer Effort

Past Developer View

GPU Memory

System Memory

Developer View With Unified Memory

Unified Memory
Simplified Memory Management Code

**CPU Code**

```c
void sortfile(FILE *fp, int N) {
    char *data;
    data = (char *)malloc(N);
    fread(data, 1, N, fp);
    qsort(data, N, 1, compare);
    use_data(data);
    free(data);
}
```

**CUDA 6 Code with Unified Memory**

```c
void sortfile(FILE *fp, int N) {
    char *data;
    cudaMallocManaged(&data, N);
    fread(data, 1, N, fp);
    qsort<<<...>>>(data,N,1,compare);
    cudaDeviceSynchronize();
    use_data(data);
    cudaFree(data);
}
```
Great Performance with Unified Memory

RAJA: Portable Parallel C++ Framework

RAJA uses Unified Memory for heterogeneous array allocations

Parallel `forall` loops run on CPU or GPU

“Excellent performance considering this is a "generic" version of LULESH with no architecture-specific tuning.”

- Jeff Keasler, LLNL

LULESH Throughput

- CPU: 10-core Haswell
- GPU: Tesla K40

GPU: NVIDIA Tesla K40, CPU: Intel Haswell E5-2650 v3 @ 2.30GHz, single socket 10-core