HPC Symposium Series
September 28, 2016

#HPCMATTERS
@ Computational Research and Programming Lab (CRPL)

Sunita Chandrasekaran
Assistant Professor, Computer and Information Sciences
schandra@udel.edu
430 Smith Hall
UDEL is

• Access to 100 GPU programming labs available at nvidia.qwiklab.com for free, of which you can divide amongst your students.

• Self-paced and hosted in the cloud, a student only needs a web-browser and internet access to participate.

• Access to the GPU Educators Program providing teaching materials and real GPU Resources

• NVIDIA-certified slides, hands-on lab exercises, and additional course content you can use in your classes
What’s my node on Farber?

- (2) x Intel E5-2660v3 (2S x 10C)
- (8) x 16 GB DDR4 ECC DIMMs
- (1) state-of-the-art nVidia Tesla K80 card
Why do I use Farber?

• Two Main goals
  – For research and development of HPC projects and codes
  – To prepare my codes to run on large supercomputers

• Sub Goals
  – To understand state-of-the-art hardware
  – To explore compilation techniques
  – To run applications faster
  – To learn to use evaluation metrics
White House Executive Order to build an Exascale supercomputer by 2023; National Strategic Computing Initiative (NSCI)

Order of processor power of the human brain

“Advancing U.S. leadership in High-Performance Computing”
**TOP 10 Sites for June 2016**

For more information about the sites and systems in the list, click on the links or view the complete list.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</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>National Supercomputing Center in Wuxi China</td>
<td>Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC</td>
<td>10,649,600</td>
<td>93,014.6</td>
<td>125,435.9</td>
<td>15,371</td>
</tr>
<tr>
<td>2</td>
<td>National Super Computer Center in Guangzhou China</td>
<td>Tianhe-2 [MilkyWay-2] - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.20GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT</td>
<td>3,120,000</td>
<td>33,862.7</td>
<td>54,902.4</td>
<td>17,808</td>
</tr>
<tr>
<td>3</td>
<td>DOE/SC/Oak Ridge National Laboratory United States</td>
<td>Titan - Cray XK7, Opteron 6274 16C 2.20GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.</td>
<td>560,640</td>
<td>17,590.0</td>
<td>27,112.5</td>
<td>8,209</td>
</tr>
<tr>
<td>4</td>
<td>DOE/NNSA/LLNI</td>
<td>Sequoia - BlueGene/Q, Power BQC 16C</td>
<td>1,572,864</td>
<td>17,173.2</td>
<td>20,132.7</td>
<td>7,820</td>
</tr>
</tbody>
</table>
We need to prepare ourselves in order to tap into supercomputers.
My CRPL HPC group uses Farber for:

- **Porting Applications**
  - Parallel Benchmarks suites
  - Sparse Fast Fourier Transform
  - Bioinformatics
  - Radiation Transport

- **Using Profilers**
  - TAU
  - Score-P
  - NVProf

- **Exploring Compilers/Programming Models**
  - OpenMP, OpenACC
  - ICC, GCC, PGI, MPI
  - CUDA

- **Other Purposes**
  - Building testsuite
  - Training Materials
  - UDEL Hackathon
  - Stress Test hardware/software
Using GPUs - Minimum change, big speedup!

Application Code

GPU
Compute-Intensive Functions
Use GPU to Parallelize

Rest of Sequential CPU Code

CPU
Parallel Programming Models
Close to the user-level, Industry-standards

• Parallel Programming Models
  – OpenMP, OpenACC
  – High Level, Directives-based, user-friendly
  – Directives describe parallelism in an application code
  – Write code once, use it on any platform (portability)
  – Incrementally improve the code
  – Code can still be compiled and executed even if the compiler doesn’t support the directives
OpenMP – Parallel Programming Model

- High-level Programming Model for shared memory programming (until V 3.1) and accelerators (V 4.x onwards)
  - Widespread vendor support and a large user base
  - User makes strategic decisions; compiler figures out details
  - Mainstream compilers for Fortran, C and C++ support OpenMP
  - Compiler support available in Intel, PGI, Cray, OpenUH, GNU, IBM, Oracle, LLVM
  - Supports X86, Xeon Phi coprocessors, work in progress to support GPUs

```c
#pragma omp parallel
#pragma omp for schedule(dynamic)
for (I=0;I<N;I++){
    NEAT_STUFF(I);
} /* implicit barrier here */
```

```c
#pragma omp target device(0)
map(tofrom:B)
#pragma omp parallel for
for (i=0; i<N; i++)
    B[i] += sin(B[i]);
```
```c
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
    int nthreads, tid;

    /* Fork a team of threads giving them their own copies of variables */
    #pragma omp parallel private (nthreads, tid) {
        /* Obtain thread number */
        tid = omp_get_thread_num();
        printf("Hello World from thread = %d\n", tid);

        /* Only master thread does this */
        if (tid == 0) {
            nthreads = omp_get_num_threads();
            printf("Number of threads = %d\n", nthreads);
        }
    } /* All threads join master thread and disband */
}
```

### OpenMP on Farber

Hello World from thread = 0
Hello World from thread = 3
Hello World from thread = 2
**Number of threads = 8**
Hello World from thread = 6
Hello World from thread = 1
Hello World from thread = 4
Hello World from thread = 7
Hello World from thread = 5
OpenACC – Programming Model for Accelerators

• High Level Programming Model for Heterogeneous Systems and Accelerators
  – A large user base (weather, particle physics, CFD, oil and gas)
  – Programmer offloads compute intensive portions of the program to the accelerator

• OpenACC code is performance **portable** and evolving
  – Compiler support available in PGI, Cray, Pathscale, GCC, OpenUH, OpenARC, accULL
  – Supports X86, NVIDIA and AMD GPUs, ARM, Power + GPUs, (Research Compilers support FPGAs and Xeon Phi coprocessors)

```c
#pragma acc parallel loop
for( i = 0; i < n; ++i )
a[i] = b[i] + c[i];
```
A simple matrix multiplication code

/* dgemm implementation with openACC acceleration*/
static void acc_dgemm( int n, double alpha, const double *A,
        const double *B, double beta, double *C) {

    int i, j, k;

    #pragma acc parallel loop copyin(A[0:(n*n)], B[0:(n*n)]) copy(C[0:(n*n)])
    for (i = 0; i < n; ++i) {

        #pragma acc loop
        for (j = 0; j < n; ++j) {

            double prod = 0;
            for (k = 0; k < n; ++k) prod += A[k * n + i] * B[j * n + k];

            C[j * n + i] = alpha * prod + beta * C[j * n + i];
        }
    }
Code compilation on Farber using OpenACC

```bash
(schandra_crpl:schandra)@farber msap-acc]$ pgcc -acc -Minfo=accel -ta=tesla:managed -fast gemm.c -o gemm
main:
  227, Generating copyin(matrix[:][:],order[:])
msap.c:
msap:
  63, Generating present(order[:],scores[:][:],matrix[:][:],lengths[:],sequences[:][:])
    Accelerator kernel generated
    Generating Tesla code
  63, Generating reduction(max:abs_max)
  65, #pragma acc loop gang, vector(128) collapse(2) /* blockIdx.x threadIdx.x */
  72, /* blockIdx.x threadIdx.x collapsed */
  81, #pragma acc loop seq
  89, #pragma acc loop seq
  95, #pragma acc loop seq
81, Loop is parallelizable
89, Loop carried dependence of h-> prevents parallelization
    Loop carried backward dependence of h-> prevents vectorization
    Loop carried dependence of h-> prevents vectorization
    Loop carried dependence of f-> prevents parallelization
    Loop carried backward dependence of f-> prevents vectorization
    Loop carried scalar dependence for max_score at line 116
    Scalar last value needed after loop for max_score at line 129
  dependence of h-> prevents vectorization
```
#!/bin/bash

for cores in {1,2,4,8,16,20} ; do
    export ACC_NUM_CORES=$cores
    echo $cores "core(s)"
    ./needle 16384 10
done

# getting the arguments
INPUT1=$1
CLASSES=(${INPUT1[@]})

# running on multicore
if [ "$#" -ge "2" ]; then
    ENV_THREADS=$2
    NUM_THREADS=$3
    export $ENV_THREADS=$NUM_THREADS
    echo Running multicore with $ENV_THREADS
    ./needle $CLASSES $NUM_THREADS
fi

export $ENV_THREADS=$NUM_THREADS
echo Running multicore with $ENV_THREADS
.....
FREE OpenACC online tutorial! Sign up!!
Free online course to start accelerating codes with GPUs

1. Lectures
2. Hands-on labs
3. Time with experts

“OpenACC allows me to concentrate on the mathematical research instead of struggling on debugging or optimizing the codes”

Shih-Hau Tan, Researcher at University of Greenwich
OpenACC Course 2015 Graduate
GTC 2016 Speaker

1st session starts October 26th

Hands-on Labs
https://nvidia.qwiklab.com/learning_paths/3/lab_catalogue
Free OpenMP tutorial, Sign up!!

• Course on OpenMP on Xeon Phi Coprocessors
• Recorded and upcoming sessions
• http://colfaxresearch.com/how-16-09/
• https://software.seek.intel.com/ColfaxThreadingWebinar.Reg
More Learning Opportunities

• UD will be a satellite site for XSEDE HPC Monthly Training

• OpenMP on Oct 4th and OpenACC on Dec 6th.

• Mark your calendar!!
Parallelizing different benchmark suites on Farber

• Parallelized ~ 20 Scalable HeterOgeneous Computing (SHOC) HPC mini-benchmark codes using OpenACC on Intel cores
  – Accepted and presented at NVIDIA’s GTC 2016
  – (Poster in the next slide)

• Parallelized ~ 7 NASA’s Parallel benchmarks (NPB) using OpenMP and OpenACC on CPUs and GPUs

• Parallelized ~ 8 Rodinia’s HPC benchmark suite on CPUs and GPUs
Code Profiles

- Code can belong to varying scientific domains
  - Famous Berkeley’s 7-13 dwarves
- Some are Compute intensive
- Some are Memory intensive
- Some are Communication intensive
- Heavy-to-light load imbalances
- Some perform better when communication is overlapped with computation
- Some are I/O bound
- We aim for scalable, performance portable, power-aware, reproducible computing
Work in progress

• Sparse Fast Fourier Transform
  – sparse data and irregular memory accesses
  – Migrate CUDA code to OpenACC for GPUs for better portability

• Bioinformatics (lots of data) – a Big Data problem
  – A DNA sequencing alignment
  – Not entirely GPU-friendly
  – Exploring hybrid model OpenMP for CPUs + OpenACC for GPUs
  – Explore MPI + X where X could be OpenMP or OpenACC
Work in Progress
Legacy Code

- Nuclear Energy code called Denovo Radiation Transport Code
- Consortium for Advanced Simulation of Light Water Reactors (CASL) headquartered at ORNL
- Miniapp consists of 6000 lines of code compared to Denovo’s 200K lines of code
- Written in C
- Currently supports MPI, CUDA, OpenMP and Intel Xeon Phi coprocessor directives;
- Ports to OpenCL, OpenACC, and OpenMP 4 are currently in progress
OpenACC Enabled Benchmark Suite on Intel Ivy Bridge

Joel Bricker and Sunita Chandrasekaran
Department of Computer and Information Sciences, University of Delaware, USA
{jbbricker,schandra}@udel.edu

Introduction

- As time passes, computing architectures become increasingly sophisticated.
- Architectures are expected to further advance.
- Reaching Exascale level of computing will require both advanced technology and sophisticated yet flexible programming models.
- A lot of uncertainty about the programming models for Exascale still prevails.
- However the requirements are quite clear.
- Need for performance portability
- Need for maintaining a single code base
- Preserve legacy code
- Tackling conflicting yet critical goals – a real challenge
- Software abstraction yet machine-specific optimization Programming models are expected to provide multiple device support
- OpenACC[1], a high-level directive-based programming model, has been gaining a lot of traction for the past couple of years for its promising performance on accelerators
- Adoption has grown to over 10,000 + OpenACC developers
- What has made the model even more unique is its recent support for multicore CPUs

Benchmark, Compiler and Evaluation Platform

- Benchmark Suite: Scalable Heterogeneous Computing Benchmark Suite (SHOC)[3]
- Compiler: PGI's OpenMP and OpenACC compilers, Version 15.10 and GNU 4.9.3
- Evaluation Platform: University of Delaware’s cluster consisting of 100 compute nodes which total 2000 Intel “Ivy Bridge” cores, 6.4TB RAM, 266TB Lustre filesystem, and an FDR infiniBand network backbone.

OpenMP Vs OpenACC (SGEMM)

```c
void sgemm(int m, int n, int k, const float *A, const float *B, float *C)
{
    int i, j, b;
    for (i = 0; i < n; i++)
        for (j = 0; j < k; j++)
            for (b = 0; b < m; b++)
                sum = A[i][j] * B[j][b];
    for (i = 0; i < m; i++)
        for (b = 0; b < k; b++)
            C[i][b] += sum;
}
```

Discussion

- Implementation of OpenMP and OpenACC required the same effort for either standard
- Preliminary results are promising and demonstrate the following:
  - Write once and use in multiple places!
  - PGI's OpenACC compiler performed the best for nearly every benchmark.
  - PGI's OpenMP compiler performed significantly better for reduction benchmarks
  - PGI's OpenACC compiler showcases performance portability for multicore CPUs
- Following table shows the speedup of OpenACC over sequential and OpenMP versions:

<table>
<thead>
<tr>
<th></th>
<th>Speedup Over</th>
<th>Reduction</th>
<th>Scan</th>
<th>SPMV</th>
<th>Stencil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>3.48</td>
<td>11.48</td>
<td>16.25</td>
<td>8.21</td>
<td></td>
</tr>
<tr>
<td>OpenMP</td>
<td>0.33</td>
<td>1.67</td>
<td>1.25</td>
<td>47.77</td>
<td></td>
</tr>
</tbody>
</table>

Results

Directions for Future Research

- Further investigation required for two results that seem as outliers: PGI with OpenMP for the stencil benchmarks and PGI with OpenMP for the reduction benchmarks.
- We will further explore the results on the rest of the benchmarks in the SHOC suite.
- The most primitive pragmas were added to get the benchmarks to run in parallel. We will explore the more elegant features of both OpenMP and OpenACC to see how tuning will impact the benchmark performance.
- Our end goal is to further explore scalable performance across multiple CPUs and GPUs yet maintaining a single code base using real-world applications.

References


Motivation & Research Question

- Can it be possible to achieve similar performance to OpenMP[2], a widely popular parallel programming model, on x86 multicore with OpenACC?
- Can a single code base be maintained across X86 and accelerators?
- Would the solution be performance portable?
Profiler Tools on Farber to profile HPC codes

• Tools Analysis and Utilities
  – http://www.paratools.com/
  – Paratools and University of Oregon
  – Visualization tool – “ParaProf”

• Score-P
  – Instrumentation framework
  – Users inserts measurement probes into C/C++ & Fortran codes
  – Collects performance-related data
  – Visualization tool “Cube”

• NVPROF
  – NVIDIA profiling tools and APIs
TAU

Multiple causes of load imbalances

Amount of time spent

Time spent in a routine

Metric: TIME
Value: Exclusive

Std. Dev. | Mean  | Max  | Min  | node 0

Std. Dev. | Mean  | Max  | Min  | node 0

Units: seconds

0.771

0.405

0.007

5.1E-4

2.2E-5

1.6E-5

7.0E-6

Loop double multiply(void) C [{loop_test.c} {23.3}- {30.3}]
Loop double multiply(void) C [{loop_test.c} {39.3}- {46.7}]
Loop double multiply(void) C [{loop_test.c} {61.0}- {68.4}]
Loop double multiply(void) C [{loop_test.c} {34.3}- {38.18}]
double multiply(void) C [{loop_test.c} {10.1}- {30.1}]
int main(int, char **) C [{loop_test.c} {53.1}- {57.1}]

Metric: TIME
Value: Number of Calls
TAU 3D plot

Number of calls by function
Score-P (serial code)

What kind of performance metric?

Source code context

How is it distributed across the processes/threads?
1.1. Kernel Performance Is Bound By Instruction And Memory Latency

This kernel exhibits low compute throughput and memory bandwidth utilization relative to the peak performance of "Tesla K80". These utilization levels indicate that the performance of the kernel is most likely limited by the latency of arithmetic or memory operations. Achieved compute throughput and/or memory bandwidth below 60% of peak typically indicates latency issues.

---

**Graph 1:**
- **Utilization Levels:** High, Med, Low
- **Instruction Types:** Load/Store, Arithmetic, Control-Flow, Texture
- **Percentage Representation:** 100%, 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, 10%
**NVProf Timeline view**

1. **CUDA Application Analysis**
   
   The guided analysis system walks you through the various analysis stages to help you understand the optimization opportunities in your application. Once you become familiar with the optimization process, you can explore the individual analysis stages in an unguided mode. When optimizing your application, it is important to consider...
Validation and Verification Testsuite

• Validate OpenACC compiler implementations
• Check for their conformance and correctness to the standard’s specification
  – Created ~200 unit test codes validating and verifying OpenACC compiler
  – Identified compiler bugs, fixed by vendors (NVIDIA, PGI); improved quality of compilers
  – Fix ambiguities and misinterpretations of the standard
  – Integrate into the official harness testsuite of world’s 3rd largest supercomputer @ Oak Ridge National Lab (ORNL), Tennessee
Standard Performance Evaluation Corporation (SPEC) High Performance Group (HPG)

- A non-profit corporation that develops benchmarks
- Uses industry standard parallel application programming interfaces (APIs), OpenMP and MPI
- Yardstick for hybrid system performance
- Stress tests software (compiler) and hardware
- SPEC ACCEL Kit currently running on Farber
  - 19 application benchmarks running under OpenCL and 15 under OpenACC
- UDEL is a member
  - AMD, Intel, Oracle, NVIDIA, SGI, IU, Argonne National Lab, TU Dresden, RWTH Aachen, UIUC, Univ. of Virginia among others
SPEC(R) ACCEL(TM) ACC Summary

NVIDIA Tesla K80
Farber K80
Mon Mar 7 23:45:48 2016

ACCEL License: 37A
Test date: Mar-2016
Test sponsor: --
Hardware availability: Jan-2015
Tested by: University of Delaware
Software availability: May-2015

<table>
<thead>
<tr>
<th>Estimated Base</th>
<th>Base Ref.</th>
<th>Base Run Time</th>
<th>Base Ratio</th>
<th>Estimated Peak Base Ref.</th>
<th>Peak Run Time</th>
<th>Peak Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>303.ostencil</td>
<td>145</td>
<td>57.1</td>
<td>2.54 S</td>
<td>145</td>
<td>57.7</td>
<td>2.51 S</td>
</tr>
<tr>
<td>303.ostencil</td>
<td>145</td>
<td>57.6</td>
<td>2.52 *</td>
<td>145</td>
<td>57.7</td>
<td>2.52 S</td>
</tr>
<tr>
<td>303.ostencil</td>
<td>145</td>
<td>57.6</td>
<td>2.52 S</td>
<td>145</td>
<td>57.7</td>
<td>2.51 *</td>
</tr>
<tr>
<td>304.olbm</td>
<td>455</td>
<td>205</td>
<td>2.21 S</td>
<td>455</td>
<td>205</td>
<td>2.21 S</td>
</tr>
<tr>
<td>304.olbm</td>
<td>455</td>
<td>207</td>
<td>2.20 S</td>
<td>455</td>
<td>206</td>
<td>2.21 S</td>
</tr>
<tr>
<td>304.olbm</td>
<td>455</td>
<td>206</td>
<td>2.21 *</td>
<td>455</td>
<td>205</td>
<td>2.21 *</td>
</tr>
<tr>
<td>314.omriq</td>
<td>956</td>
<td>396</td>
<td>2.41 S</td>
<td>956</td>
<td>397</td>
<td>2.41 S</td>
</tr>
<tr>
<td>314.omriq</td>
<td>956</td>
<td>397</td>
<td>2.41 *</td>
<td>956</td>
<td>397</td>
<td>2.41 *</td>
</tr>
<tr>
<td>314.omriq</td>
<td>956</td>
<td>397</td>
<td>2.41 S</td>
<td>956</td>
<td>397</td>
<td>2.41 S</td>
</tr>
</tbody>
</table>
| .......
UDEL Hackathon in collaboration with ORNL May 2016

- 6 teams (NASA, Brookhaven National Lab, National Cancer Institute, UDEL CIS, ECE and Chemical & Biomolecular Engg)
- The UDEL Chemical & Biomolecular Engg used Farber’s GPU for preparatory and development work before moving code to TITAN
  - The team was using GPUs and OpenACC for the first time!!
UDEL 2016 Summer Scholar Program

- Collin Clark, Ryan Beneck, Daniel Liang presented posters at the UDEL’s summer program
- All benchmarks were compiled and ran on Farber
- Benchmarks used OpenACC
Other stuff...

- Using Farber to prepare training materials for nvidia.qwiklab.com
- A training site for people to learn OpenACC among other tools
Moving Forward

• Pre-Exascale machines will soon exist 2017-2019
• DOE’s awarded $425M for next-gen supercomputers
  – Collaboration of Oak Ridge, Argonne, and Lawrence Livermore (CORAL) National Laboratories
## AURORA

**Argonne National Lab**

### System Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mira</th>
<th>Aurora</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Nodes</td>
<td>49,152</td>
<td>&gt;50,000</td>
</tr>
<tr>
<td>Processor</td>
<td>PowerPC A2 1600 MHz</td>
<td>3rd Generation Intel Xeon Phi</td>
</tr>
<tr>
<td>System Memory</td>
<td>768 TB</td>
<td>&gt;7 PB DRAM and persistent memory</td>
</tr>
<tr>
<td>System Interconnect</td>
<td>IBM 5D torus interconnect with VCSEL photonics</td>
<td>2nd Generation Intel Omni-Path Architecture with silicon photonics</td>
</tr>
<tr>
<td>File System Capacity</td>
<td>26 PB GPFS</td>
<td>&gt;150 PB Lustre</td>
</tr>
<tr>
<td>Intel Architecture (x86-64)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Compatibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peak Power Consumption</td>
<td>4.8 MW</td>
<td>13 MW</td>
</tr>
</tbody>
</table>
# Oak Ridge National Lab

## SUMMIT

*Scale new heights. Discover new solutions.*

Oak Ridge National Laboratory's next High Performance Supercomputer.

*Coming 2018.*

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>TITAN</th>
<th>SUMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Nodes</td>
<td>18,688</td>
<td>~3,400</td>
</tr>
<tr>
<td>Processor</td>
<td>(1) 16-core AMD Opteron per node</td>
<td>(Multiple) IBM POWER 9s per node</td>
</tr>
<tr>
<td>Accelerator</td>
<td>(1) NVIDIA Kepler K20x per node</td>
<td>(Multiple) NVIDIA Volta GPUs per node</td>
</tr>
<tr>
<td>Memory per node</td>
<td>32GB (DDR3)</td>
<td>&gt;512GB (HBM+DDR4)</td>
</tr>
<tr>
<td>CPU-GPU Interconnect</td>
<td>PCI Gen2</td>
<td>NVLINK (5-12x PCIe3) (/summit-faqs/#f2)</td>
</tr>
<tr>
<td>System Interconnect</td>
<td>Gemini</td>
<td>Dual Rail EDR-IB (23 GB/s)</td>
</tr>
<tr>
<td>Peak Power Consumption</td>
<td>9 MW</td>
<td>10 MW</td>
</tr>
</tbody>
</table>
Sierra at lawrence livermore national lab

- Sierra will be five to seven times Sequoia in workload performance with a 120-150 petaflop/s peak.

- The Sierra system will include compute nodes (POWER Architecture Processor, NVIDIA Volta, NVMe-compatible PCIe 800GB SSD, greater than 512 GB DDR4 + HBM, and coherent shared memory), compute racks (standard 19-inch with warm-water cooling), and the compute system with be 2.1–2.7 PB memory, 120–150 petaflop/s, and 10 MW).
How do we prepare ourselves?
Preparedness is key

• Community clusters are highly critical!
• Equipped with state-of-the-art Hardware and software
• Great platform to learn how to profile large codes
  – Understand application characteristics and their suitability to platforms
  – Parallelize codes to exploit underlying hardware
  – Migrate codes to newer hardware and learn challenges
UDEL is a member of

• OpenMP standard organization
• OpenACC standard organization
• SPEC HPG consortium
• UDEL is awarded the NVIDIA GPU Education Center title
UDEL is

- Access to 100 GPU programming labs available at nvidia.qwiklab.com for free, of which you can divide amongst your students.
- Self-paced and hosted in the cloud, a student only needs a web-browser and internet access to participate.
- Access to the GPU Educators Program providing teaching materials and real GPU Resources
- NVIDIA-certified slides, hands-on lab exercises, and additional course content you can use in your classes
CRPL Group (schandra@udel.edu)

Graduate Students: Kshitij Srivatsav, Rahul Deore, Nikolai Snow

Sergio Pino  Sanhu Li  Arnov Sinha

Undergraduate Students: Ryan Beneck, Collin Clark, Daniel Liang, Kyle Friedline