UNIVERSITY OF DELAWARE

Using the Parallel Programming Model, OpenACC, to do More Science and Less Programming

Sunita Chandrasekaran

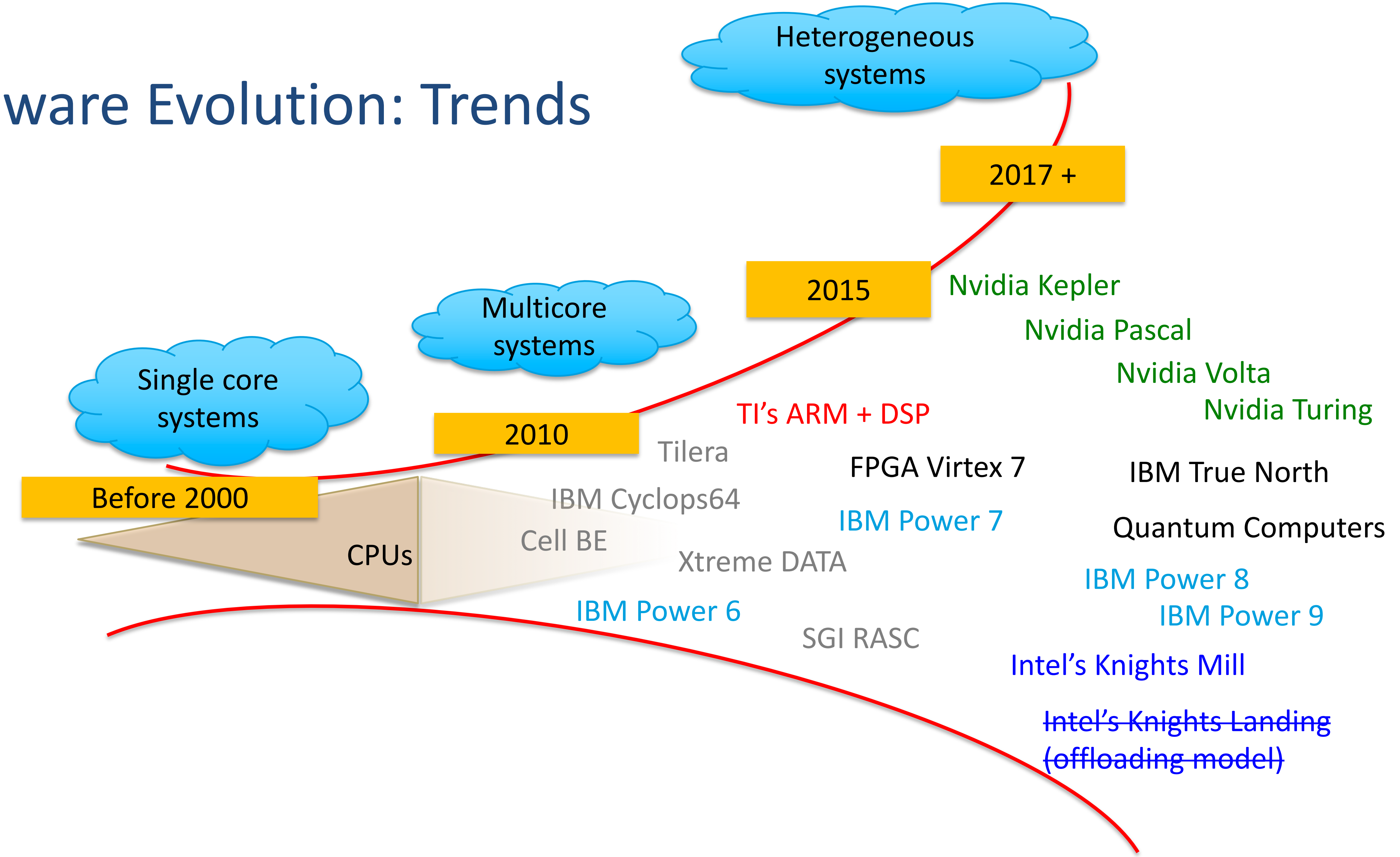
Asst. Professor, Dept. of Computer & Information Sciences

University of Delaware

Oct 27, 2018 Princeton University Bootcamp

CRPL Computational Research and Programming Lab

Hardware Evolution: Trends



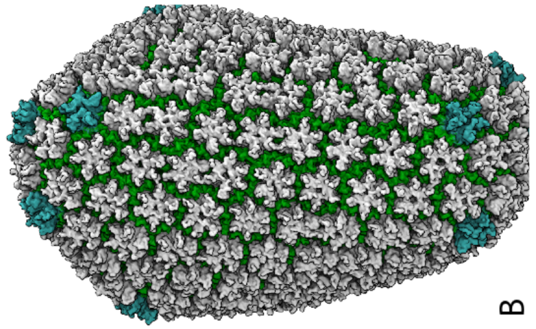
2012 World’s fastest supercomputer. AMD processor + NVIDIA K20

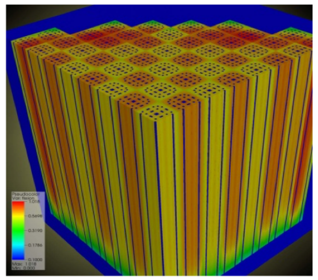
2018 World’s fastest supercomputer. IBM Processor + NVIDIA V100

Has the software caught up yet?

- On-node programming has become even more a challenge
- Need to explore billion way concurrency
- Challenges
  - Migrating/Porting legacy code to current and upcoming platforms
  - Write once and reuse multiple times
  - Maintainable software

Several ways to program





Applications

Libraries

Programming Languages

Directives

Drop in acceleration

Maximum Flexibility

Used for easier acceleration

OpenACC is a directives-based programming approach to **parallel computing** designed for **performance** and **portability** on CPUs and GPUs for HPC.

Add Simple Compiler Directive

```
main()
 {
  <serial code>
  #pragma acc kernels
  {
    <parallel code>
  }
}
```

DIRECTIVE-BASED HPC PROGRAMMING

Who’s Using OpenACC

3 OF TOP 5 HPC APPS

Intersect360 RESEARCH

5 OF 13 CAAR CODES

summit

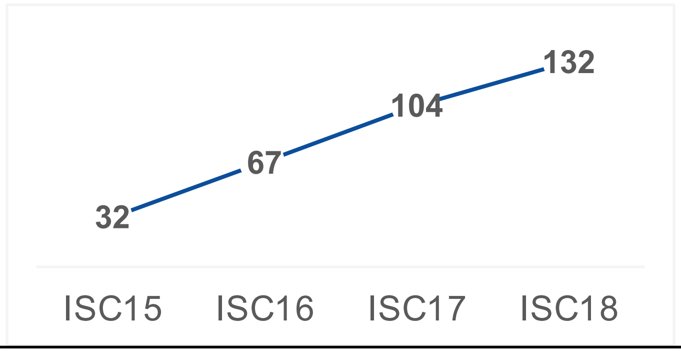
2 OF LAST 9 FINALISTS

acm AWARDS Gordon Bell Prize

450 DOMAIN EXPERTS

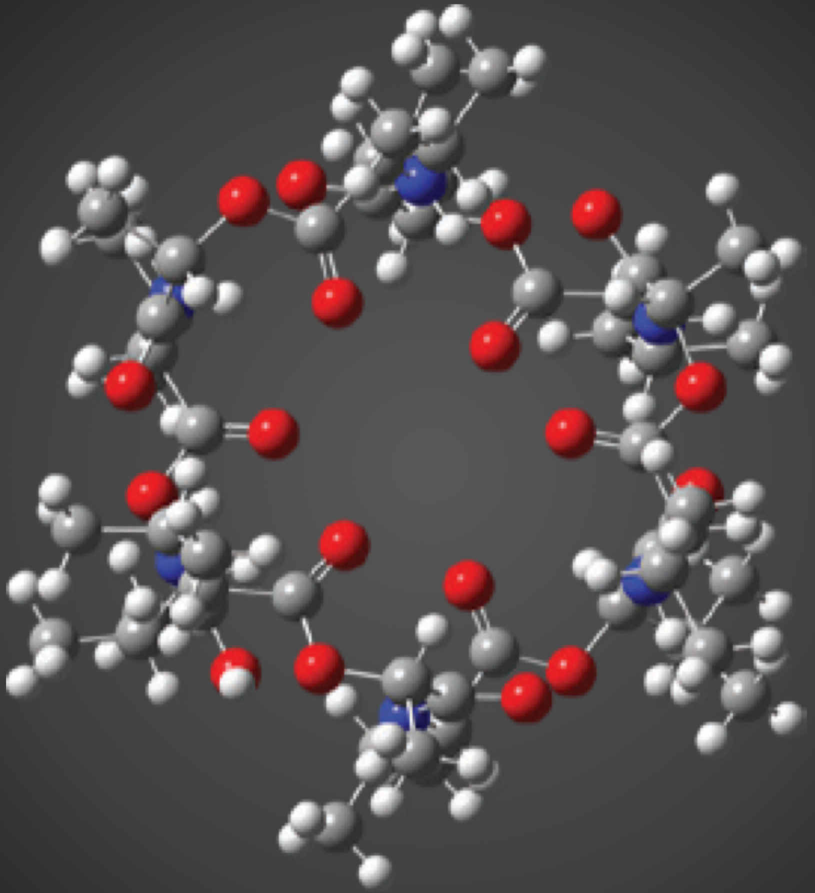
OpenACC HACKATHON Advancing High Performance Computing

ACCELERATED APPS



100,000 DOWNLOADS

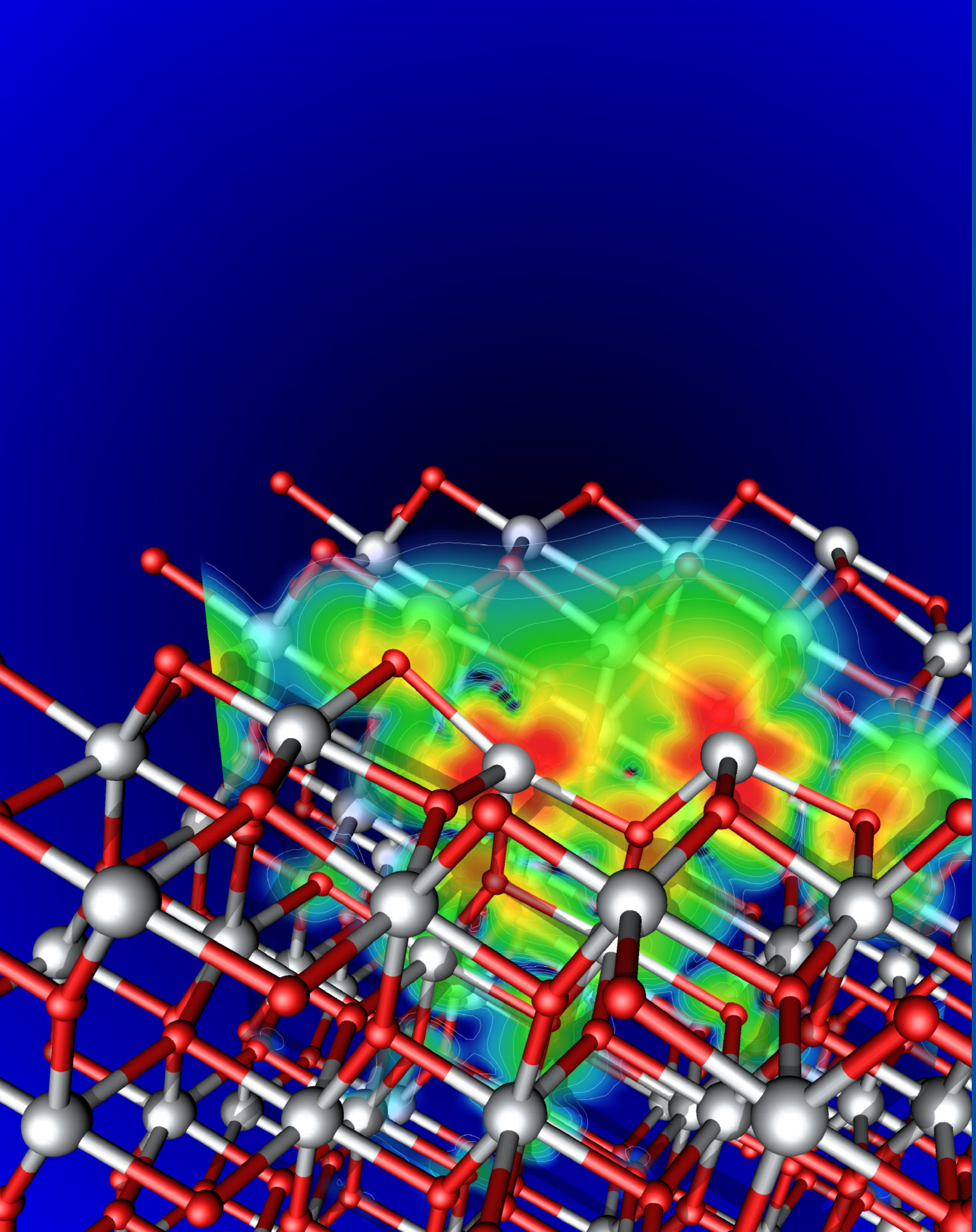
PGI® Community EDITION



GAUSSIAN 16

Mike Frisch, Ph.D.
President and
CEO
Gaussian, Inc.

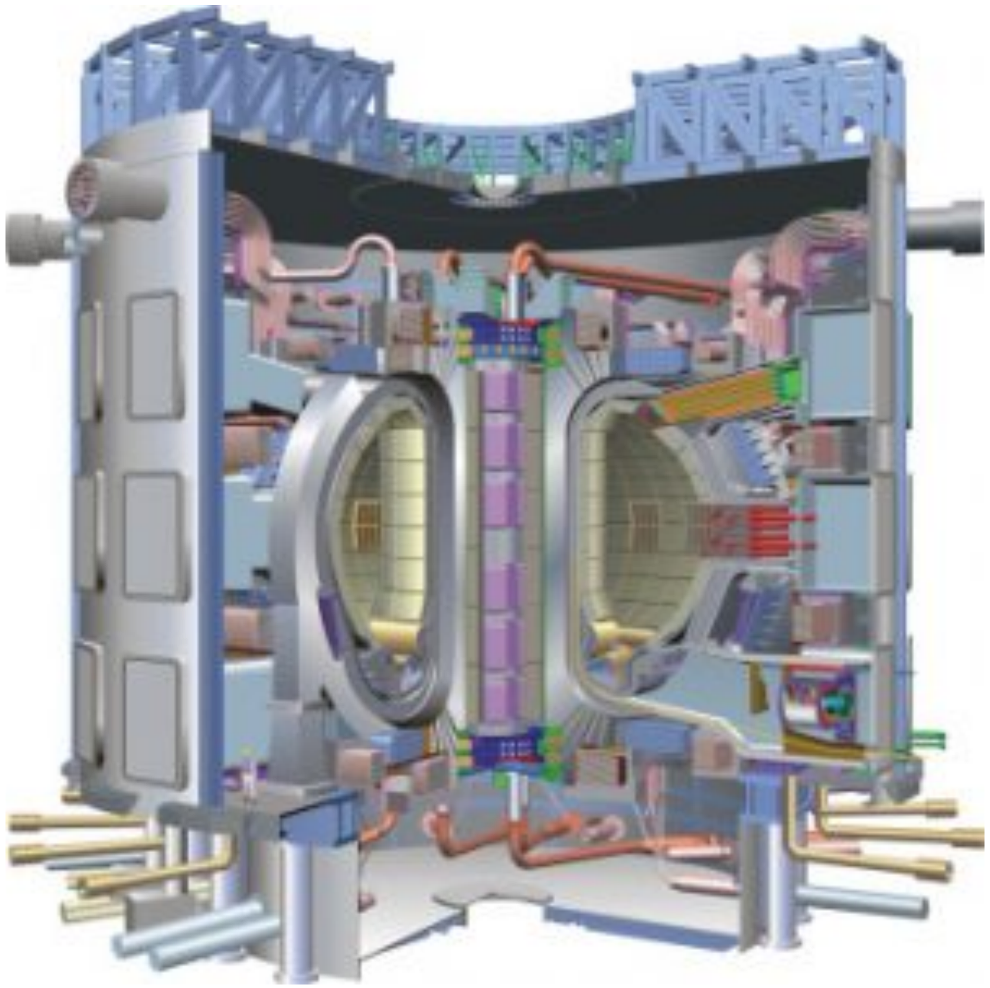
“Using OpenACC allowed us to continue development of our fundamental algorithms and software capabilities simultaneously with the GPU-related work. In the end, we could use the same code base for SMP, cluster/ network and GPU parallelism. PGI's compilers were essential to the success of our efforts.”



VASP

Prof. Georg Kresse
Computational Materials Physics
University of Vienna

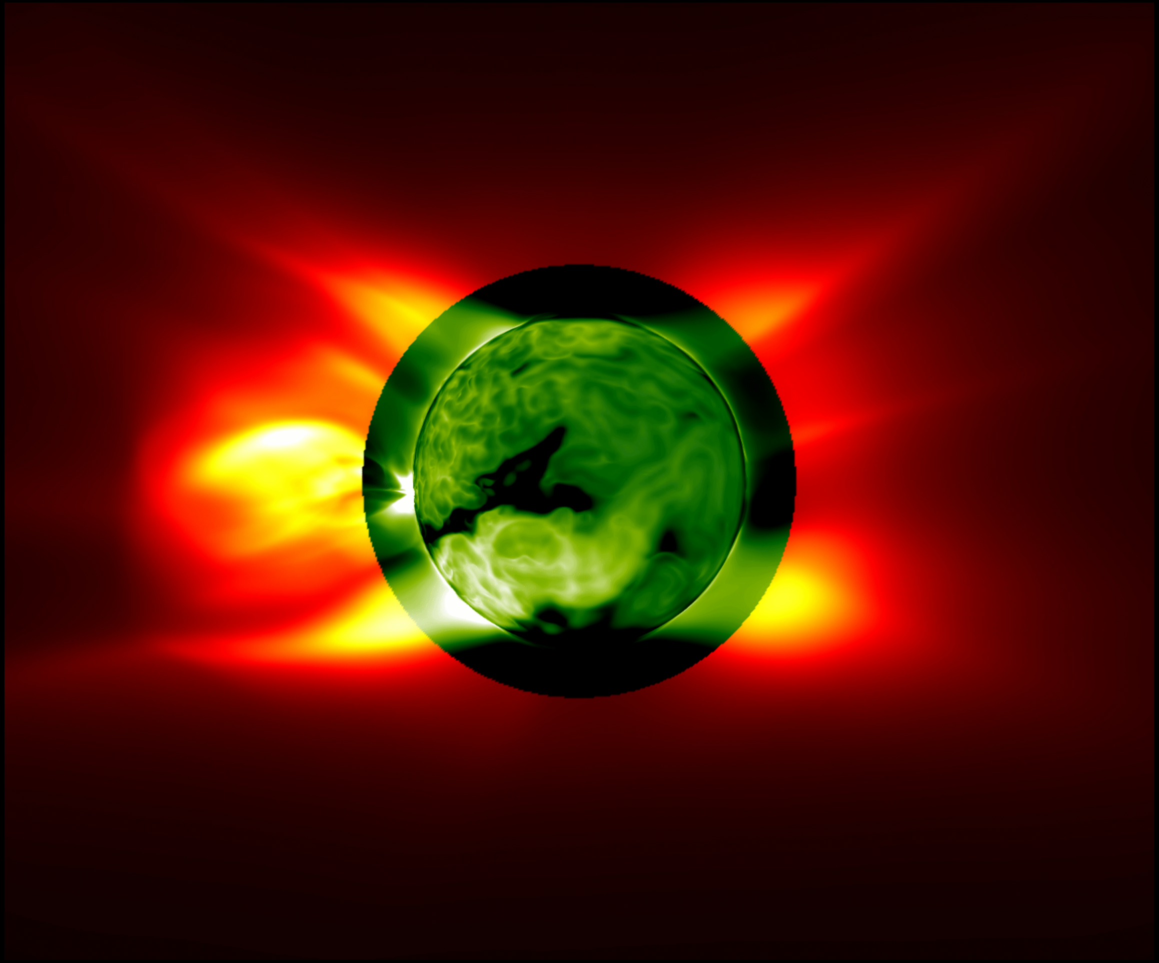
“For VASP, OpenACC is ***the*** way forward for GPU acceleration. Performance is similar and in some cases better than CUDA C, and OpenACC dramatically decreases GPU development and maintenance efforts. We’re excited to collaborate with NVIDIA and PGI as an early adopter of CUDA Unified Memory.”



GTC

Zhihong Lin
Professor and Principal Investigator
UC Irvine

“Using OpenACC our scientists were able to achieve the acceleration needed for integrated fusion simulation with a minimum investment of time and effort in learning to program GPUs.”

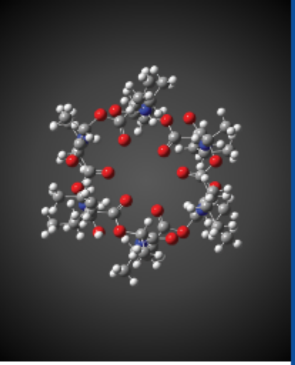


MAS

Ronald M. Caplan
Computational Scientist
Predictive Science Inc.

"Adding OpenACC into MAS has given us the ability to migrate medium-sized simulations from a multi-node CPU cluster to a single multi-GPU server. The implementation yielded a portable single-source code for both CPU and GPU runs. Future work will add OpenACC to the remaining model features, enabling GPU-accelerated realistic solar storm modeling."

https://devblogs.nvidia.com/solar-storm-modeling-gpu-openacc/



GAUSSIAN 16

Mike Frisch, Ph.D.
President and
CEO
Gaussian, Inc.

"Using OpenACC allowed us to continue development of our fundamental algorithms and software capabilities simultaneously with the GPU-related work. In the end, we could use the same code base for SMP, cluster/ network and GPU parallelism. PGI's compilers were essential to the success of our efforts."

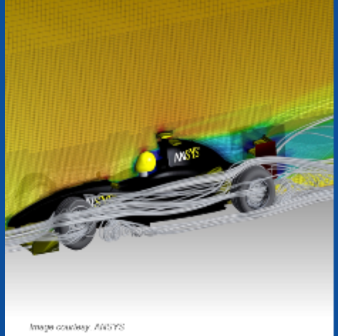
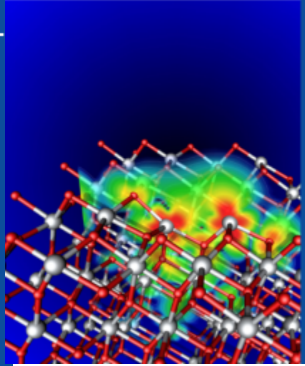


Image courtesy: ANSYS

ANSYS FLUENT

Sunil Sathe
Lead Software Developer
ANSYS Fluent

"We've effectively used OpenACC for heterogeneous computing in ANSYS Fluent with impressive performance. We're now applying this work to more of our models and new platforms."



VASP

Prof. Georg Kresse
Computational Materials Physics
University of Vienna

"For VASP, OpenACC is *the* way forward for GPU acceleration. Performance is similar and in some cases better than CUDA C, and OpenACC dramatically decreases GPU development and maintenance efforts. We're excited to collaborate with NVIDIA and PGI as an early adopter of CUDA Unified Memory."

COSMO

Dr. Oliver Fuhrer
Senior Scientist
Meteoswiss

"OpenACC made it practical to develop for GPU-based hardware while retaining a single source for almost all the COSMO physics code."

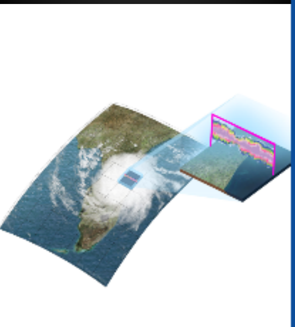


Image courtesy: Oak Ridge National Laboratory

E3SM

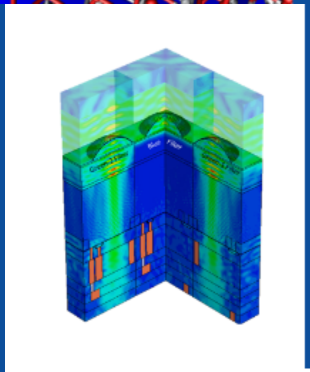
Mark A. Taylor
Multiphysics Applications
Sandia

"The CAAR project provided us with early access to Summit hardware and access to PGI compiler experts. Both of these were critical to our success. PGI's OpenACC support remains the best available and is competitive with much more intrusive programming model approaches."

NUMECA FINE/Open

David Gutzwiller
Lead Software Developer
NUMECA

"Porting our unstructured C++ CFD solver FINE/Open to GPUs using OpenACC would have been impossible two or three years ago, but OpenACC has developed enough that we're now getting some really good results."



SYNOPSYS

Dr. Lutz Schneider
Senior R&D Engineer
Synopsys Inc

"Using OpenACC, we've GPU-accelerated the Synopsys TCAD Sentaurus Device EMW simulator to speed up optical simulations of image sensors. GPUs are key to improving simulation throughput in the design of advanced image sensors."

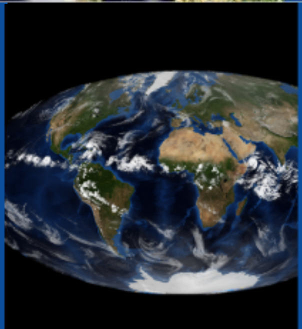
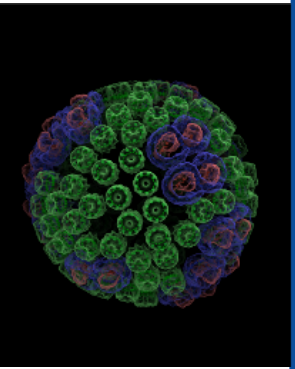


Image courtesy: NCAR

MPAS-A

Richard Loft
Director, Technology Development
NCAR

"Our team has been evaluating OpenACC as a pathway to performance portability for the Model for Prediction (MPAS) atmospheric model. Using this approach on the MPAS dynamical core, we have achieved performance on a single P100 GPU equivalent to 2.7 dual socketed Intel Xeon nodes on our new Cheyenne supercomputer."



VMD

John Stone
Senior Research Programmer
Beckman Institute
University of Illinois

"Due to Amdahl's law, we need to port more parts of our code to the GPU if we're going to speed it up. But the sheer number of routines poses a challenge. OpenACC directives give us a low-cost approach to getting at least some speed-up out of these second-tier routines. In many cases it's completely sufficient because with the current algorithms, GPU performance is bandwidth-bound."

GTC

Zhihong Lin
Professor and Principal Investigator
UC Irvine

"Using OpenACC our scientists were able to achieve the acceleration needed for integrated fusion simulation with a minimum investment of time and effort in learning to program GPUs."

OpenACC
More Science, Less Programming

Map courtesy University of Tokyo

GAMERA

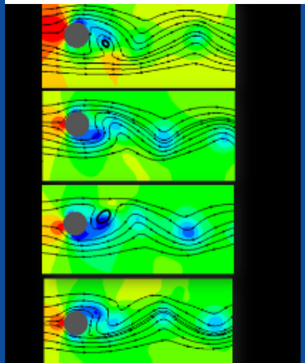
Takuma Yamaguchi, Kohei Fujita, Tsuyoshi Ichimura, Muneo Hori, Lalith Wijerathne
The University of Tokyo

"With OpenACC and a compute node based on NVIDIA's Tesla P100 GPU, we achieved more than a 14X speed up over a K Computer node running our earthquake disaster simulation code"

SANJEEVINI

Abhilash Jayaraj
Project Scientist
Indian Institute of Technology
New Delhi

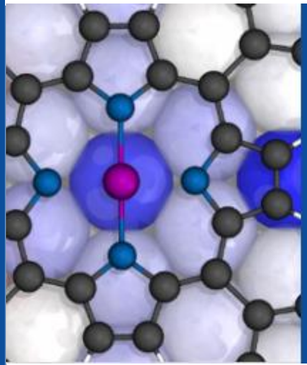
"In an academic environment maintenance and speedup of existing codes is a tedious task. OpenACC provides a great platform for computational scientists to accomplish both tasks without involving a lot of efforts or manpower in speeding up the entire computational task."



IBM-CFD

Somnath Roy
Assistant Professor
Mechanical Engineering Department
Indian Institute of Technology Kharagpur

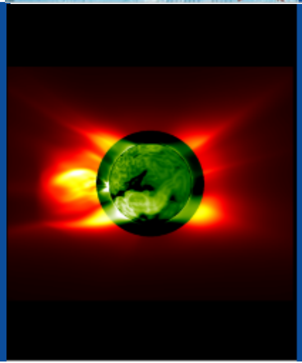
"OpenACC can prove to be a handy tool for computational engineers and researchers to obtain fast solution of non-linear dynamics problem. In immersed boundary incompressible CFD, we have obtained order of magnitude reduction in computing time by porting several components of our legacy codes to GPU. Especially the routines involving search algorithm and matrix solvers have been well-accelerated to improve the overall scalability of the code."



PWscf (Quantum ESPRESSO)

Filippo Spiga
Senior Contributor
Quantum ESPRESSO group

"CUDA Fortran gives us the full performance potential of the CUDA programming model and NVIDIA GPUs. While leveraging the potential of explicit data movement, !$CUF KERNELS directives give us productivity and source code maintainability. It's the best of both worlds."



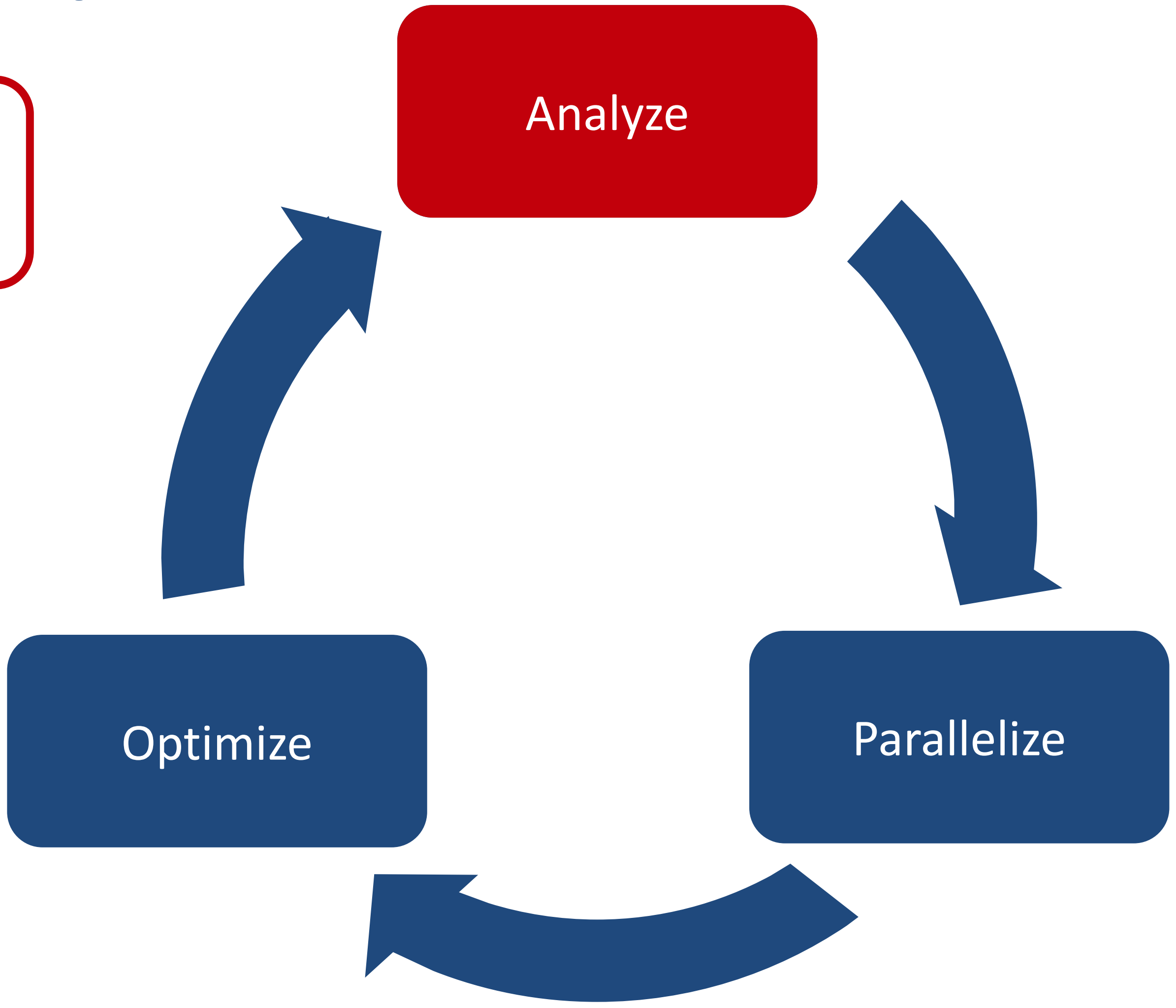
MAS

Ronald M. Caplan
Computational Scientist
Predictive Science Inc.

"Adding OpenACC into MAS has given us the ability to migrate medium-sized simulations from a multi node CPU cluster to a single multi-GPU server. The implementation yielded a portable single-source code for both CPU and GPU runs. Future work will add OpenACC to the remaining model features, enabling GPU-accelerated realistic solar storm modeling."

OpenACC development CYCLE

- **Analyze** your code to determine most likely places needing parallelization or optimization.
- **Parallelize** your code by starting with the most time consuming parts and check for correctness.
- **Optimize** your code to improve observed speed-up from parallelization.



OpenACC

Incremental

- Maintain existing sequential code
- Add annotations to expose parallelism
- After verifying correctness, annotate more of the code

Single Source

- Rebuild the same code on multiple architectures
- Compiler determines how to parallelize for the desired machine
- Sequential code is maintained

Low Learning Curve

- OpenACC is meant to be easy to use, and easy to learn
- Programmer remains in familiar C, C++, or Fortran
- No reason to learn low-level details of the hardware.

OpenACC syntax

Syntax for using OpenACC directives in code

C/C++

```
#pragma acc directive clauses
<code>
```

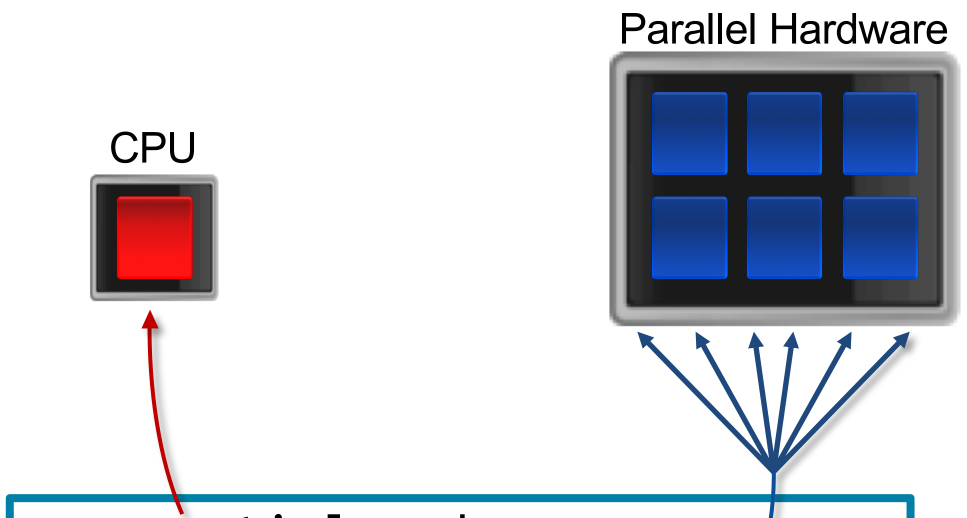
Fortran

```
!$acc directive clauses
<code>
```

- A ***pragma*** in C/C++ gives instructions to the compiler on how to compile the code. Compilers that do not understand a particular pragma can freely ignore it.
- A ***directive*** in Fortran is a specially formatted comment that likewise instructions the compiler in it compilation of the code and can be freely ignored.
- "***acc***" informs the compiler that what will come is an OpenACC directive
- ***Directives*** are commands in OpenACC for altering our code.
- ***Clauses*** are specifiers or additions to directives.

OpenACC parallel directive

Explicit programming



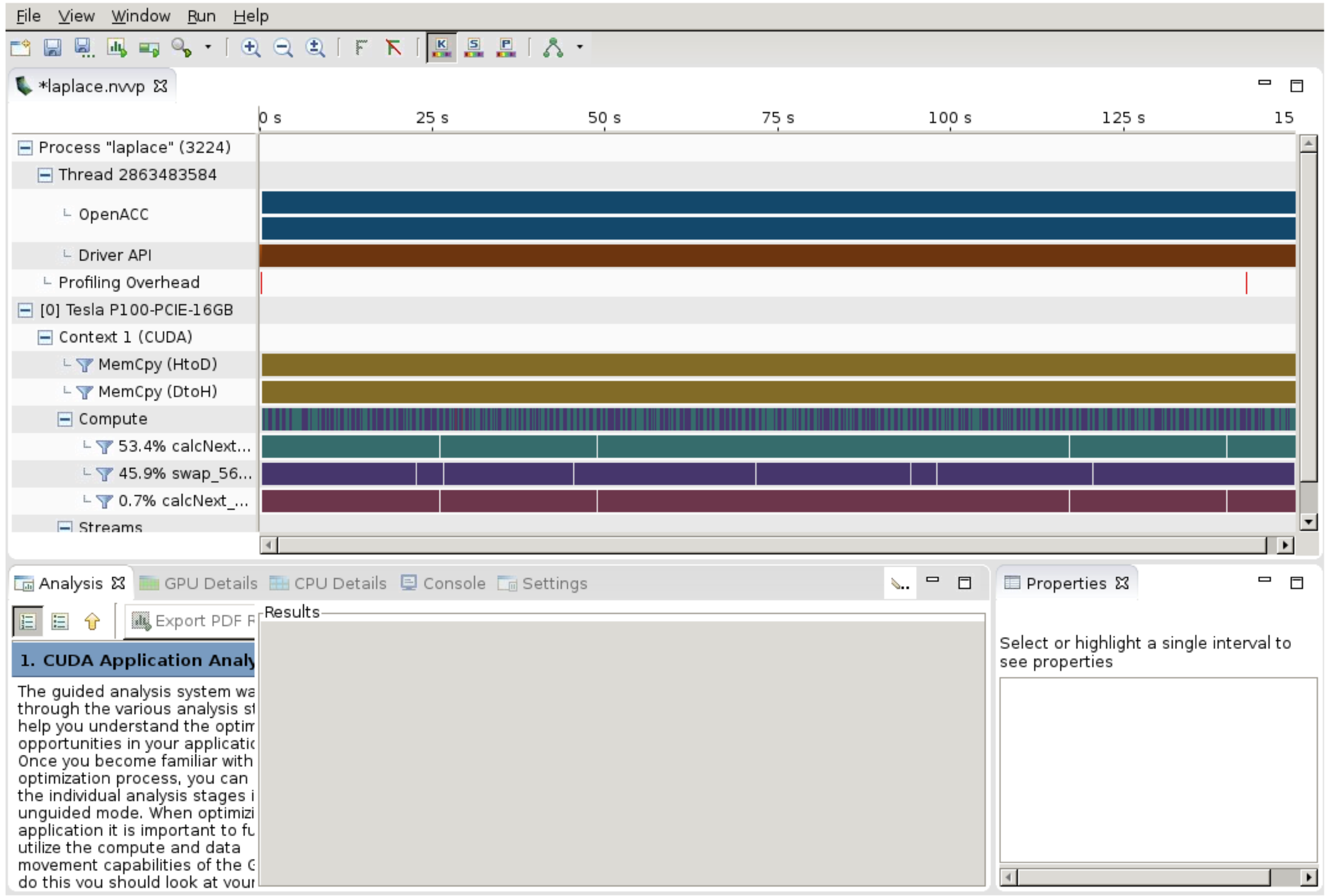
```
<sequential code>

#pragma acc parallel
{
<sequential code>
}
```

- The parallel directive instructs the compiler to create parallel *gangs* on the accelerator
- Gangs are independent groups of worker threads on the accelerator
- The code contained within a parallel directive is executed redundantly by all parallel gangs

Profiling gpu code (PGPROF)

Using PGPROF to profile GPU code

- PGPROF presents far more information when running on a GPU
- We can view CPU Details, GPU Details, a Timeline, and even do Analysis of the performance



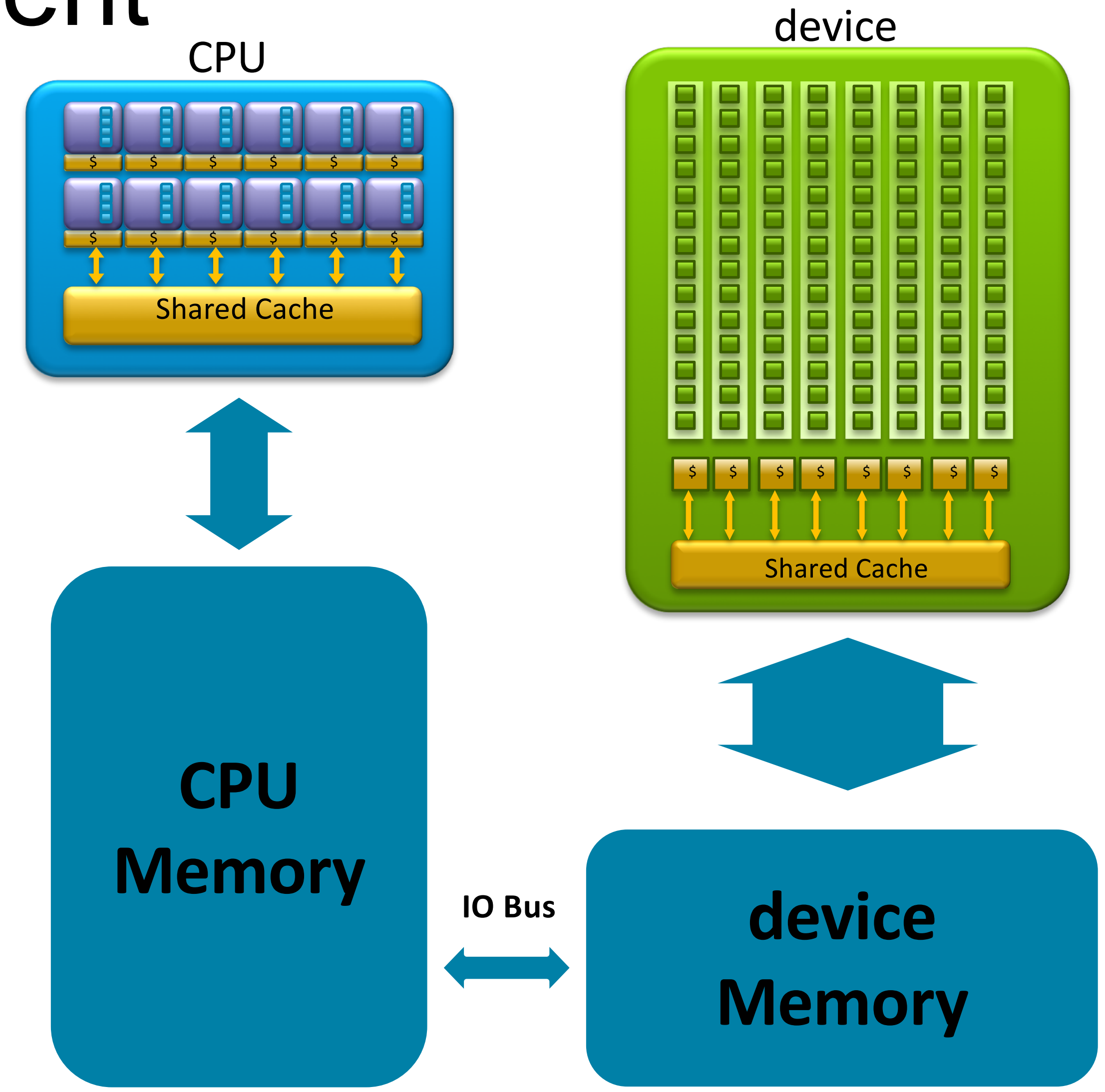
The guided analysis system wa
through the various analysis st
help you understand the optim
opportunities in your applicatio
Once you become familiar with
optimization process, you can
the individual analysis stages i
unguided mode. When optimizi
application it is important to fu
utilize the compute and data
movement capabilities of the G
do this you should look at your

Select or highlight a single interval to see properties

Explicit memory management

Key problems

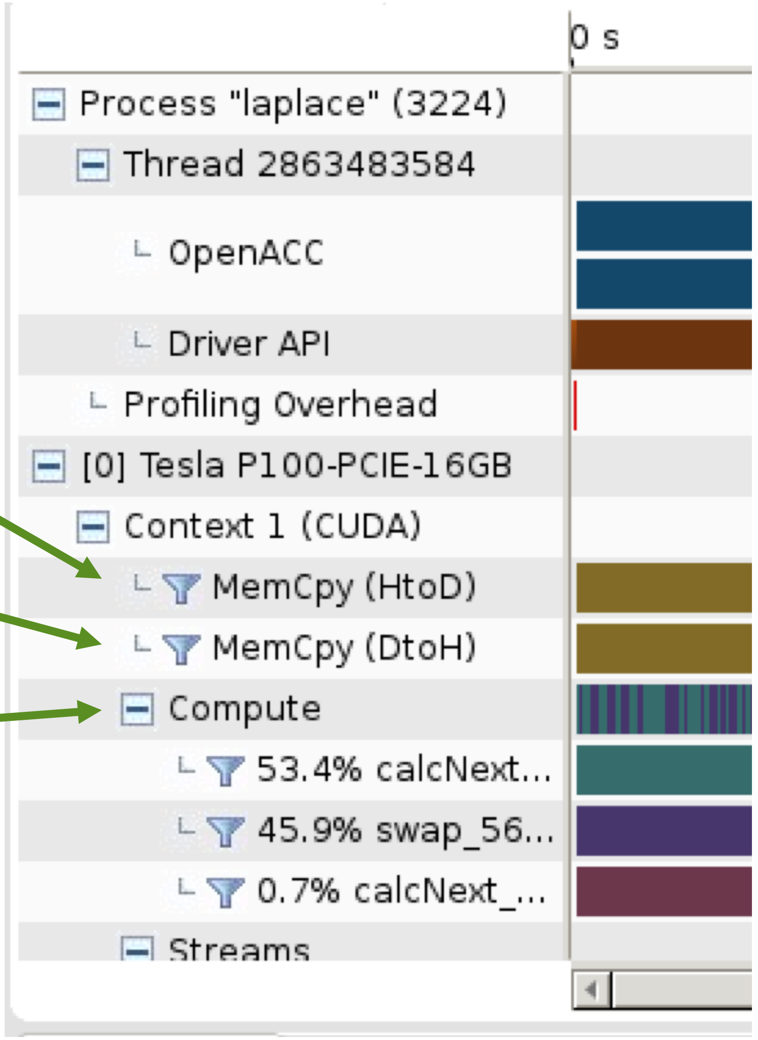
- Many parallel accelerators (such as devices) have a separate memory pool from the host
- These separate memories can become out-of-sync and contain completely different data
- Transferring between these two memories can be a very time consuming process



Profiling gpu code (PGPROF)

Using PGPROF to profile GPU code

- **MemCpy(HtoD):** This includes data transfers from the Host to the Device (CPU to GPU)
- **MemCpy(DtoH):** These are data transfers from the Device to the Host (GPU to CPU)
- **Compute:** These are our computational functions. We can see our calcNext and swap function



```
                                        0 s
⊟ Process "laplace" (3224)
  ⊟ Thread 2863483584
      └ OpenACC
      └ Driver API
    └ Profiling Overhead
⊟ [0] Tesla P100-PCIE-16GB
  ⊟ Context 1 (CUDA)
      └ MemCpy (HtoD)
      └ MemCpy (DtoH)
    ⊟ Compute
        └ 53.4% calcNext...
        └ 45.9% swap_56...
        └ 0.7% calcNext_...
    ⊟ Streams
```

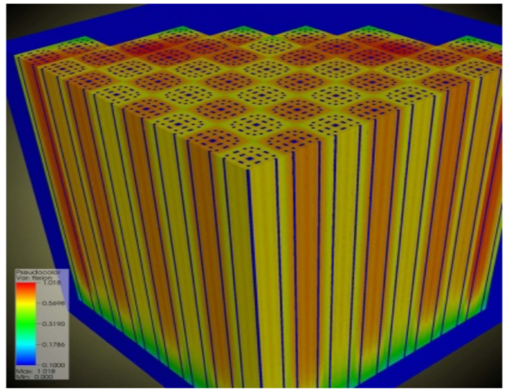
How do you compile an OpenACC code?

- PGI-OpenACC compiler
  - PGI Community Edition, licensed but FREE to download
  - Await the most latest version 18.10 (to be released this week)
  - https://www.pgroup.com/products/community.htm
  - pgcc -fast -ta=tesla:cc60 -Minfo=accel -o laplace laplace.c
- GNU-OpenACC compiler (developed by Mentor Graphics)
  - Available online for download and use

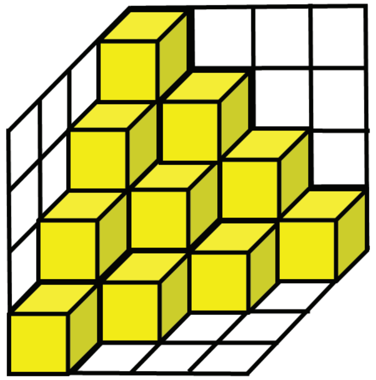
Scientific Codes using OpenACC in my research group

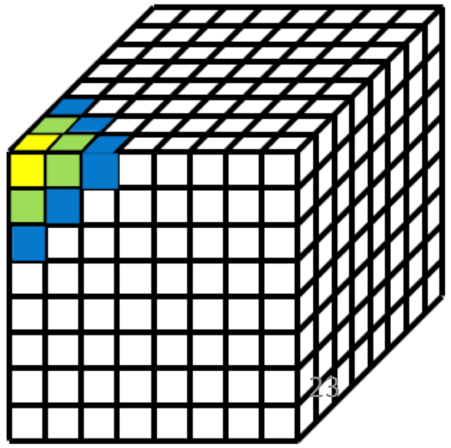
- Minisweep, a miniapp, represents (80-99%) of Denovo S_n code
  - Nuclear Reactor Modeling code
  - Code of interest to Oak Ridge National Lab
- Acceleration of Chemical Shift
  - A code called within NAMD, VMD a 100 times
  - Dept. of Chemistry
- Acceleration of MURaM (Max Planck University of Chicago Radiative MHD)
  - National Center for Atmospheric Research (NCAR)

Nuclear reactor modeling proxy code : Minisweep



- Minisweep, a miniapp, represents (80-99%) of Denovo S_n code
- Denovo S_n (discrete ordinate), part of DOE INCITE project, is used to model fusion reactor – CASL, ITER
  - **Impact:** By running Minisweep faster, experiments with more configurations can be performed directly impacting the determination of accuracy of radiation shielding
- Poses a six dimensional problem
  - 3D in space, 2D in angular particle direction and 1D in particle energy
- The parallel pattern observed is wavefront-based

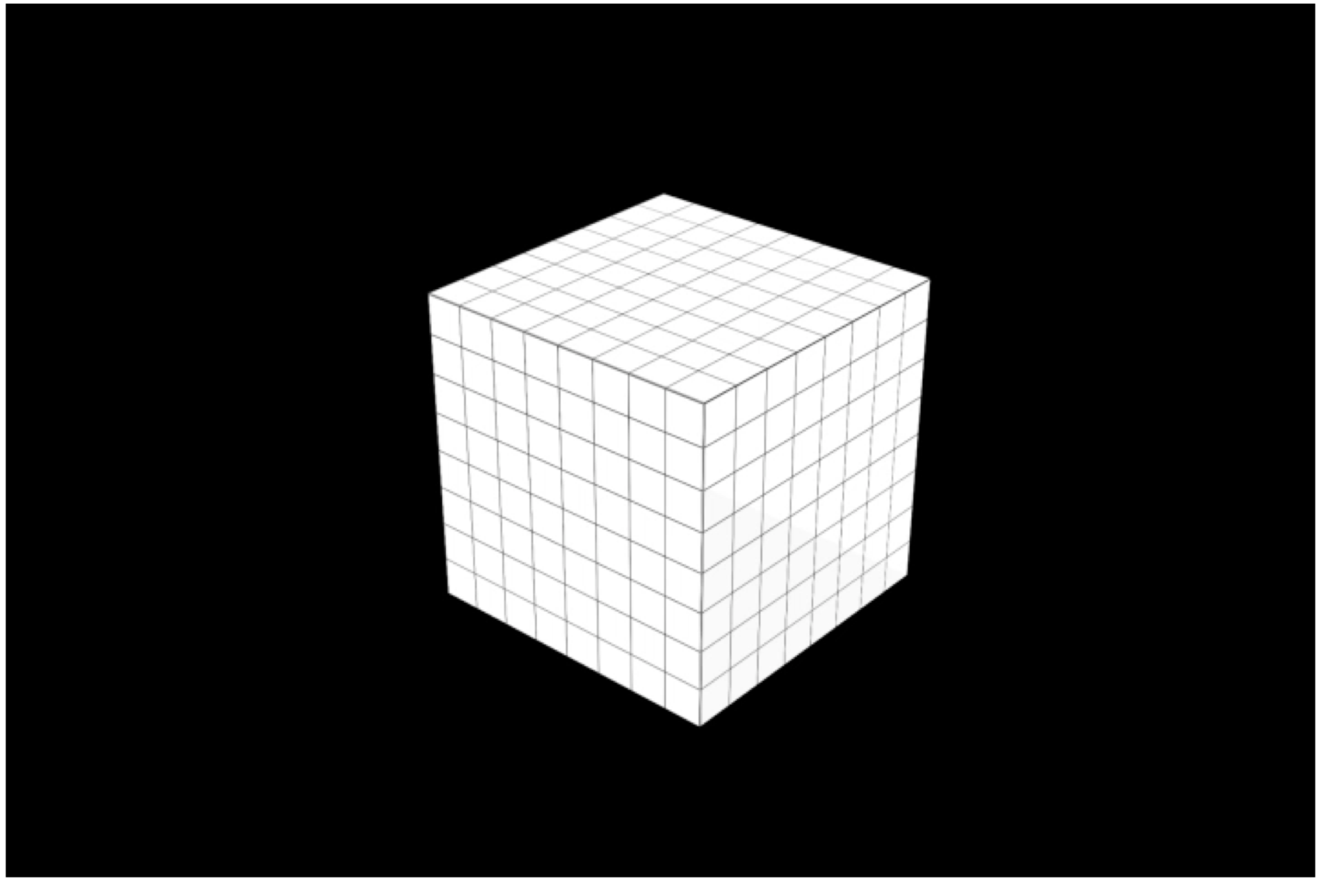




Minisweep code status

- Originally used CUDA and OpenMP 3.1 targeting Beacon and TITAN at ORNL (one node of the Percival Cray XC40 KNL system)
- Has been used for TITAN acceptance testing and now currently being used for SummitDev and Summit acceptance testing at ORNL

Sweep Algorithm



Parallelizing Sweep Algorithm: KBA

- Koch-Baker-Alcouffe (KBA)
- Algorithm developed in 1992 at Los Alamos
- Parallel sweep algorithm that overcomes some of the dependencies in the algorithm

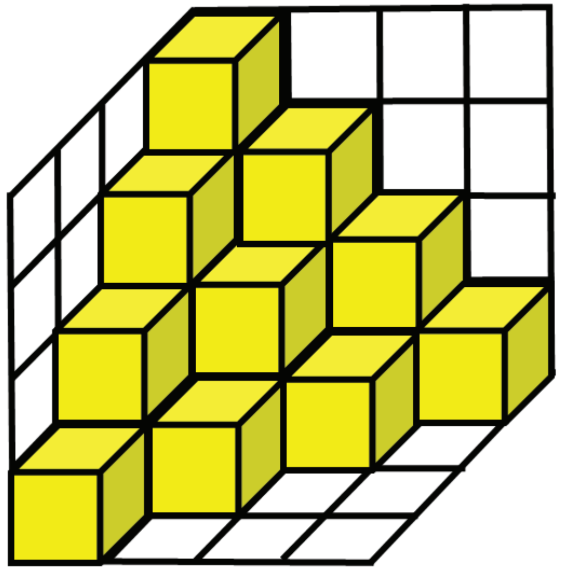


Image credit: High Performance Radiation Transport Simulations: Preparing for TITAN
C. Baker, G. Davidson, T. M. Evans, S. Hamilton, J. Jarrell and W. Joubert
ORNL, USA

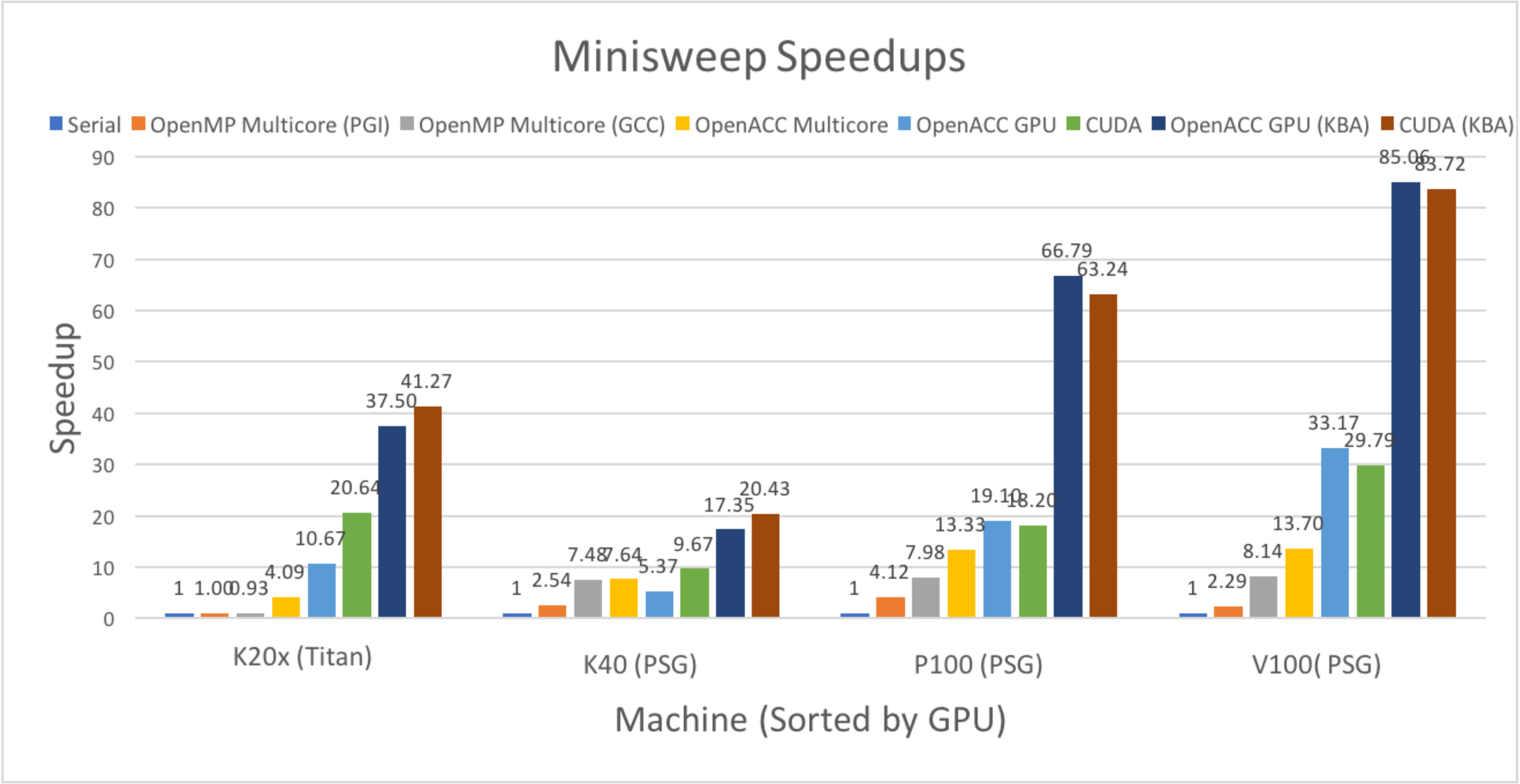
Programming Challenges

- Parallelizing wavefront-based parallel code
  - Manual loop restructuring
  - Applying spatial decomposition
  - Storing previous wavefronts
  - Analyzing upstream dependencies
- Sweeping along 8 directions and avoiding race directions
- Need to address multiple layers of parallelism (minisweep 5-levels)

Experimental Setup

- NVIDIA PSG Cluster
  - CPU: Intel Xeon E5-2698 v3 (16-core)
  - GPU: NVIDIA Tesla P100, Tesla V100, and Tesla K40 (4 GPUs per node)
- ORNL Titan
  - CPU: AMD Opteron 6274 (16-core) & GPU: NVIDIA Tesla K20x
- Software
  - PGI OpenACC Compiler 17.10
  - OpenMP – GCC 6.2.0 (we used Intel 17.0 compiler too but GCC performed better)
- Input size
  - X/Y/Z = 64; number of energy groups = 64 and number of angles = 32

Minisweep Results



Summary

- Parallelized the in-grid cell computations (Wavefront)
- Performing multidirectional sweep
- Using Volta GPU, OpenACC implementation shows 85.06x over serial code Vs CUDA implementation of 83.72x over the same serial implementation
- Maintained a single code base for multicore and GPUs
- Run across nodes with multiple GPUs per node

- 14 months effort
- Papers published in PASC 2018 and Journal CPC 2018

Robert Searles, Sunita Chandrasekaran, Wayne Joubert, Oscar Hernandez. 2018. Abstractions and Directives for Adapting Wavefront Algorithms to Future Architectures. In ACM proceedings of 5th Platform for Advanced Scientific Computing (PASC). DOI: 10.1145/ 3218176.3218228
Robert Searles, Sunita Chandrasekaran, Wayne Joubert, Oscar Hernandez. 2018. Abstractions and Directives for Adapting Wavefront Algorithms to Future Architectures. Journal of Computer Physics Communication (CPC). DOI: 10.1016/j.cpc.2018.10.007

Scientific Codes using OpenACC in my research group

- Minisweep, a miniapp, represents (80-99%) of Denovo S_n code
  - Nuclear Reactor Modeling code
  - Code of interest to Oak Ridge National Lab
- Acceleration of Chemical Shift
  - A code called within NAMD, VMD a 100 times
  - Dept. of Chemistry and other Chemistry packages
- Acceleration of MURaM (Max Planck University of Chicago Radiative MHD)
  - National Center for Atmospheric Research (NCAR)

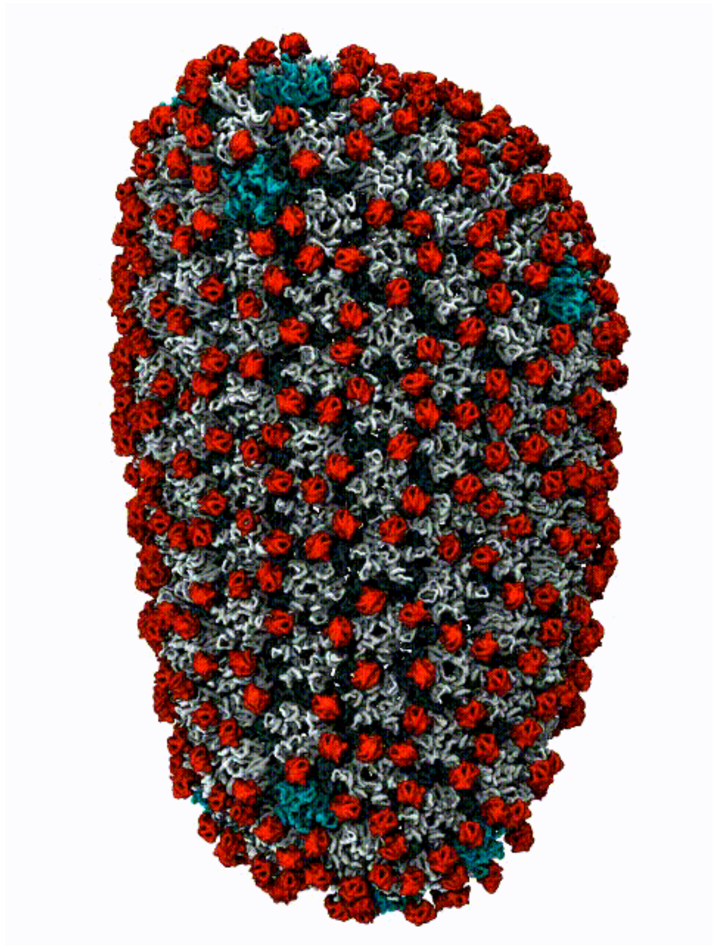
Accelerating chemical shift problem

Project Motivation

- Nuclear Magnetic Resonance (NMR) is a vital tool in the biocomputational space
- Chemical shift gives insight into the physical structure of the protein
- Predicting chemical shift has important uses in scientific areas such as drug discovery

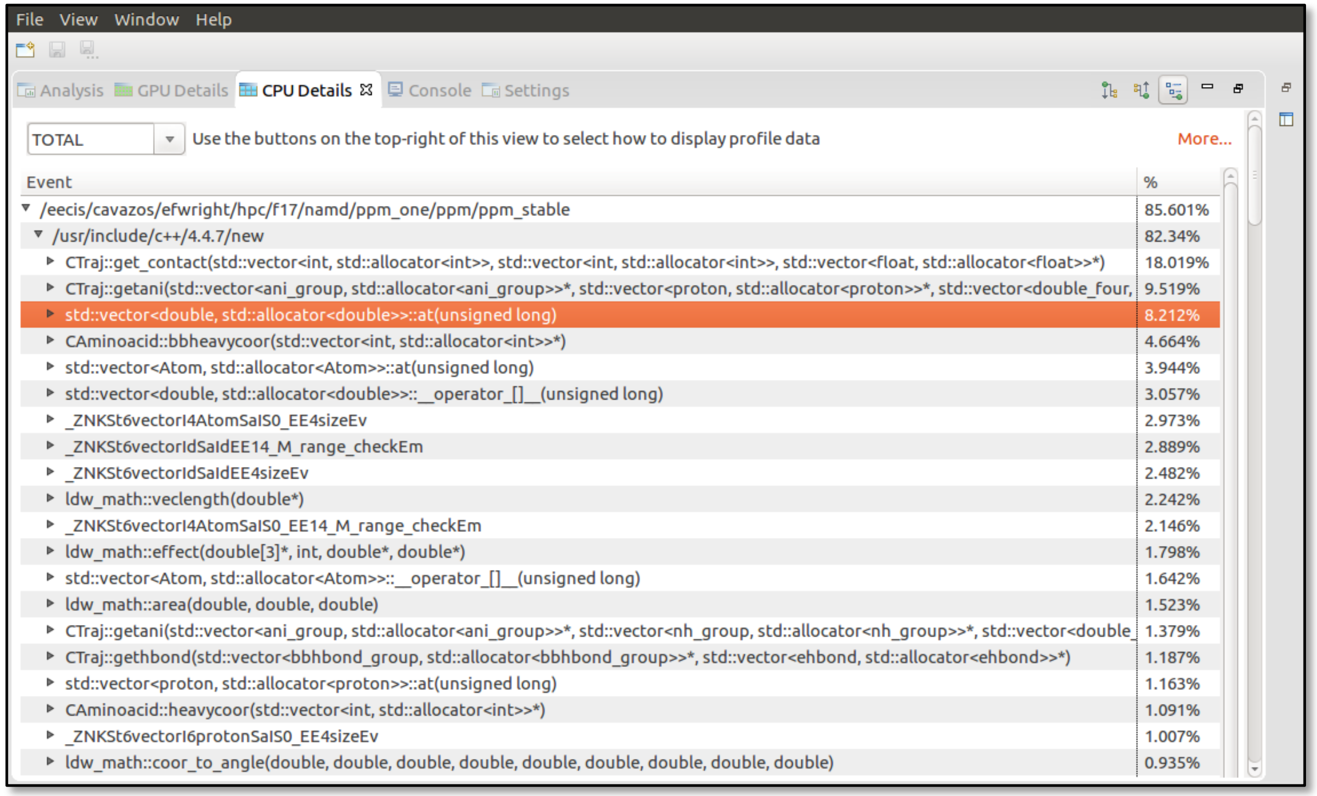
Goal

- To enable execution of multiple chemical shift predictions repeatedly
- To allow chemical shift predictions for larger scale structures



Serial Code Profile PPM_ONE

- Profiled code using PGPROF
  - Without any optimizations
- Gave a baseline snapshot of the code
  - Identified hotspots within the code
  - Identified functions that are potential bottlenecks
- Obtained large overview without needing to read thousands of lines of code

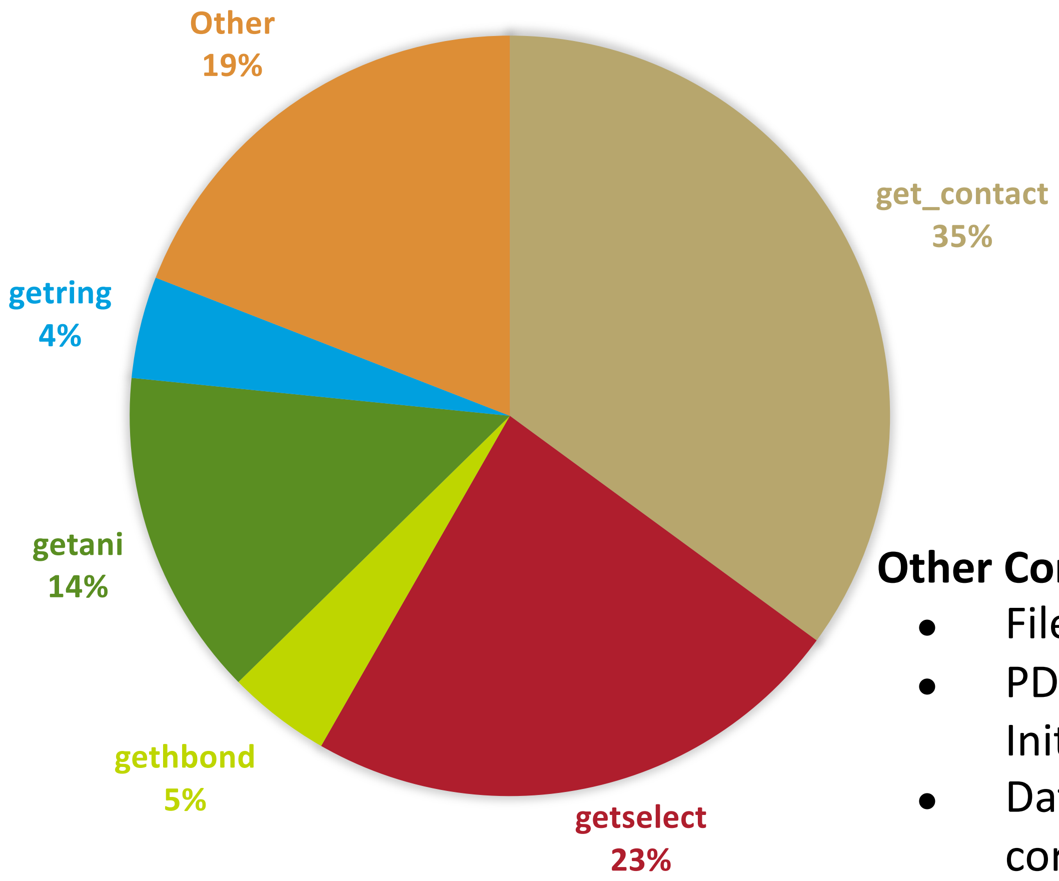


Use the buttons on the top-right of this view to select how to display profile data

Event	%
/eecis/cavazos/efwright/hpc/f17/namd/ppm_one/ppm/ppm_stable	85.601%
/usr/include/c++/4.4.7/new	82.34%
CTraj::get_contact(std::vector<int, std::allocator<int>>, std::vector<int, std::allocator<int>>, std::vector<float, std::allocator<float>>*)	18.019%
CTraj::getani(std::vector<ani_group, std::allocator<ani_group>>*, std::vector<proton, std::allocator<proton>>*, std::vector<double_four,	9.519%
std::vector<double, std::allocator<double>>::at(unsigned long)	8.212%
CAminoacid::bbheavycoor(std::vector<int, std::allocator<int>>*)	4.664%
std::vector<Atom, std::allocator<Atom>>::at(unsigned long)	3.944%
std::vector<double, std::allocator<double>>::__operator_[]__(unsigned long)	3.057%
_ZNKSt6vectorI4AtomSaIS0_EE4sizeEv	2.973%
_ZNKSt6vectorIdSaIdEE14_M_range_checkEm	2.889%
_ZNKSt6vectorIdSaIdEE4sizeEv	2.482%
ldw_math::veclength(double*)	2.242%
_ZNKSt6vectorI4AtomSaIS0_EE14_M_range_checkEm	2.146%
ldw_math::effect(double[3]*, int, double*, double*)	1.798%
std::vector<Atom, std::allocator<Atom>>::__operator_[]__(unsigned long)	1.642%
ldw_math::area(double, double, double)	1.523%
CTraj::getani(std::vector<ani_group, std::allocator<ani_group>>*, std::vector<nh_group, std::allocator<nh_group>>*, std::vector<double_	1.379%
CTraj::gethbond(std::vector<bbhbond_group, std::allocator<bbhbond_group>>*, std::vector<ehbond, std::allocator<ehbond>>*)	1.187%
std::vector<proton, std::allocator<proton>>::at(unsigned long)	1.163%
CAminoacid::heavycoor(std::vector<int, std::allocator<int>>*)	1.091%
_ZNKSt6vectorI6protonSaIS0_EE4sizeEv	1.007%
ldw_math::coor_to_angle(double, double, double, double, double, double, double, double, double)	0.935%

Serial Code Profile (predict_bb_static_ann)

Main Function	% Runtime
main()	100%
predict_bb_static_ann(void)	81.226%
predict_proton_static_new(void)	16.276%
load(string)	1.921%



Other Contains:

- File I/O
- PDB Structure Initialization
- Data error correction

Serial Optimization (getselect)

```
// Pseudocode for getselect function

for( ... )    // Large loop
{
    c2=pdb->getselect(":1-%@allheavy");
    traj->get_contact(c1,c2,&r
}
```

getselect originally accounted for **25%** of the codes runtime. After optimization, it takes less than **1%**.

```
// Pseudocode for getselect function

c2=pdb->getselect(":1-%@allheavy");
for( ... )    // Large loop
{
    traj->get_contact(c1,c2,&result);
}
```

Accelerating get_contact

```
#pragma acc parallel loop private(...) \
 present(..., results[0:results_size]) copyin(...)
for(i=1;i<index_size-1;i++)
{
    ...

    #pragma acc loop reduction(+:contact1, +:contact2, \
     +:contact3) private(...)
    for(j=0;j<c2_size;j++)
    {
        // Calculate contact1, contact2, contact3
    }
    ...
    results[((i-1)*3)+0]=contact1;
    results[((i-1)*3)+1]=contact2;
    results[((i-1)*3)+2]=contact3;
}
```

- Large outer-loop covers all individual get_contact calls
- Inner-loop still iterates over all atoms
- Now calculating 3 different contacts simultaneously
- Writing contacts to one large results array to be used later

Acceleration of gethbond

```
#pragma acc parallel
{
#pragma acc loop gang
for(i=0;i<_hbond_size;i++)
{

    #pragma acc loop vector
    for(j=0;j<hbond_size;j++)
    {
        ...
        #pragma acc loop seq
        for(k=0;k<nframe;k++)
        {
            ...
        }
    }
}
} // end parallel region
```

Gang and vector directives allow us to implement multiple levels of loop parallelism.

The innermost loop is typically very small, and would provide no benefit in parallelizing, so we mark it as “sequential”

Code Checklist

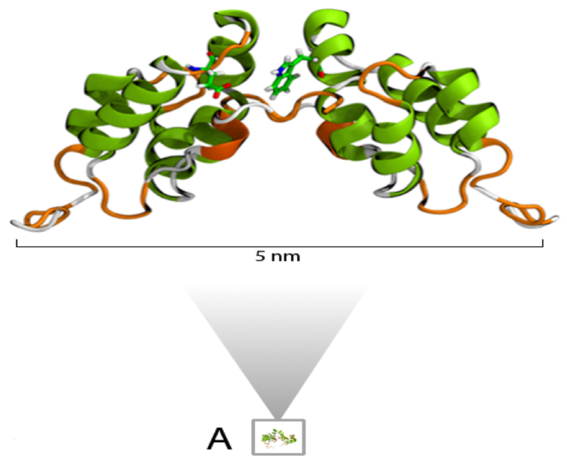
get_contact()	45.652%	✓
getselect()	23.211 %	✓
getani()	18.147%	✓
gethbond()	5.718%	✓
getring()	5.633%	✓

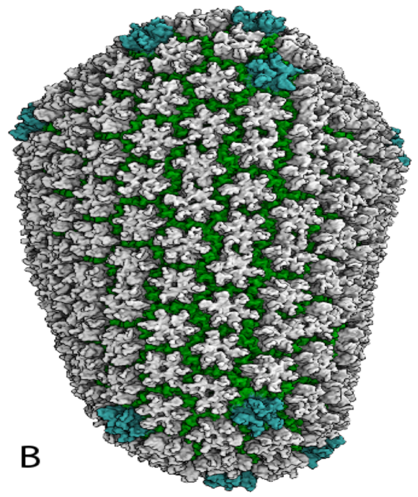
	Before	After
get_contact	2505s	15s
gethbond	337s	1.24s
getani	29s	0.09s
getring	19s	0.09s

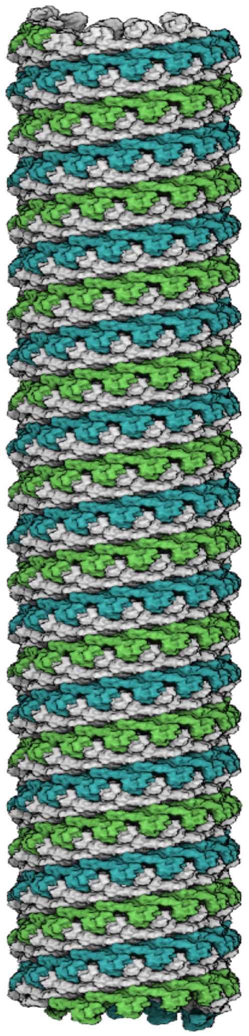
Selective functions - Using Large 5.8M Atom Dataset on V100

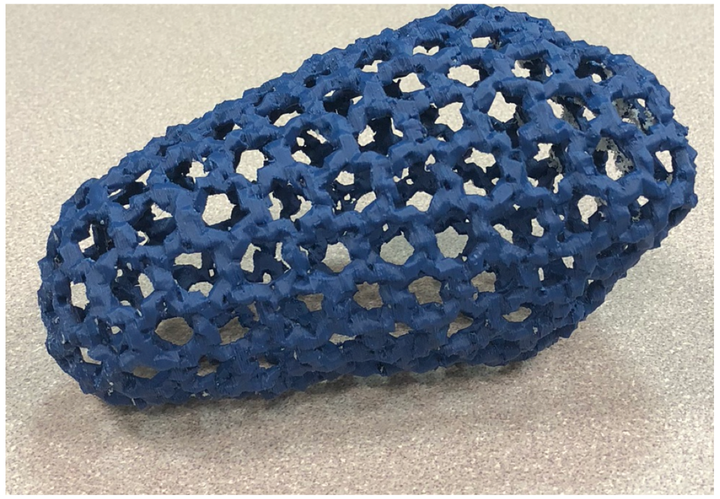
Experimental Datasets

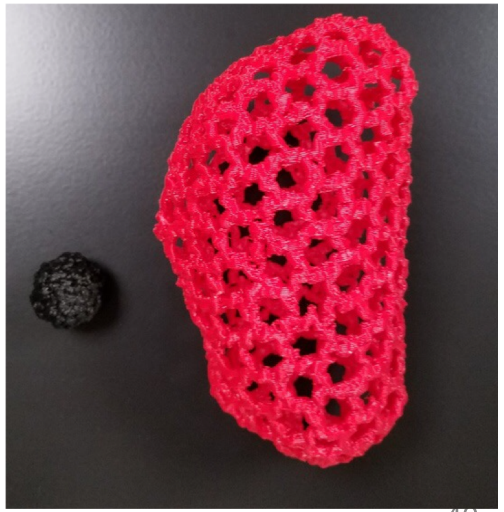
Structure A the first 100,000 atoms of the Dynamin GTPase were isolated and written to their own PDB file.
Structure B The next dataset tested was the HIV-1 capsid assembly (CA) without Hydrogens.
Structure C in Figure 2 is a 6.8 million atom model of 14 turns of the Dynamin GTPase.











Experimental Setup

- NVIDIA PSG Cluster
  - CPU: Intel Xeon E5-2698 v3 (16-core)
  - GPU: NVIDIA Tesla P100, Tesla V100, and Tesla K40 (4 GPUs per node)
- Software
  - PGI OpenACC Compiler 18.4

Results

	Very Small (100K) Atoms	Medium (2.1M) Atoms	Large (6.8M) Atoms	Very Large (11M) Atoms
Serial (Unoptimized)	167.11s	3547.07 (1 hour)	7 hours *approx.*	14 hours *approx.*
Serial (Optimized)	32s	2209.64s (37 min)	2939s (48 min)	9035s (2.5 hours)
Multicore (32 cores)	2.93s	109s	172s	427s
NVIDIA PASCAL P100 GPU	1.72s	36s	69s	170s
NVIDIA VOLTA V100 GPU	1.68s	29s	56s	134s

PPM_ONE Summary

- Performance of 67x on NVIDIA V100 compared to a single core
- Performance of 21x on multicore, dual socket, 32 cores, using OpenACC
- Incorporate the GPU accelerated PPM_One chemical shift prediction into
  - NAMD (Nanoscale Molecular Dynamics) enabling protein structure refinement combined with other experimental techniques
  - VMD (Visual Molecular Dynamics) enabling scientists to perform structure validation

- 4 undergrad+2PhD students
- 12 months effort
- Submitting to Cell Biophysics Journal
- Won the mid-Atlantic Research poster Competition

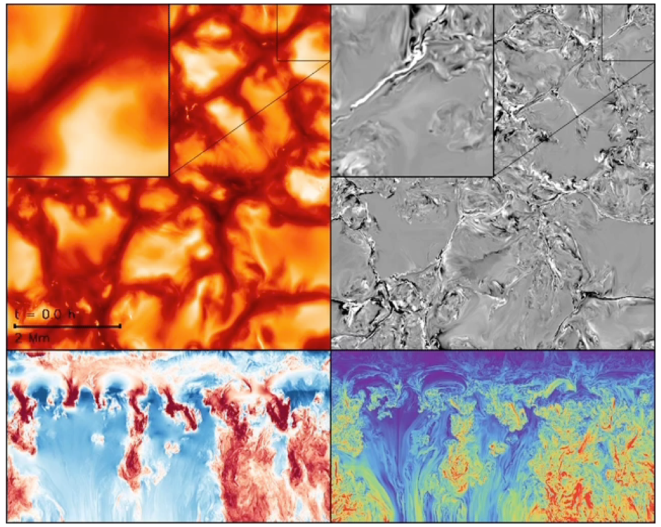
Scientific Codes using OpenACC in my research group

- Minisweep, a miniapp, represents (80-99%) of Denovo S_n code
  - Nuclear Reactor Modeling code
  - Code of interest to Oak Ridge National Lab
- Acceleration of Chemical Shift
  - A code called within NAMD, VMD a 100 times
  - Dept. of Chemistry and other Chemistry packages
- Acceleration of MURaM (Max Planck University of Chicago Radiative MHD)
  - National Center for Atmospheric Research (NCAR)

MURaM (Max Planck University of Chicago Radiative MHD)

- The primary solar model used for simulations of the upper convection zone, photosphere and corona.
- Jointly developed and used by HAO, the Max Planck Institute for Solar System Research (MPS) and the Lockheed Martin Solar and Astrophysics Laboratory (LMSAL).
- MURaM has contributed substantially to our understanding of solar phenomena.
- MURaM also plays a key role in interpreting high resolution solar observations.

The Daniel K. Inouye Solar Telescope (DKIST), a ~$300M NSF investment, is expected to advance the resolution of ground based observational solar physics by an order of magnitude.


MURaM simulation of solar granulation

Roadmap

- Profile the original source code
  - Profiler ranking of top functions consuming most wall time. We focused on optimizing these functions, such as the "init" function.
- Factor in long-term solar science goals
  - From the input of solar physicists, we identified radiative transport (RTS) as the key routine to focus on to enable future science.
- Apply OpenACC programming model
  - We added the OpenACC directives to move most of the intensive computation to GPU. Accelerated mhd function. The function will be further accelerated after the code is re-profiled.
- Optimize the CPU/GPU data movement
  - Optimizations to avoid data transfer between CPU and GPU and keep most of the computations on the GPU.

Results

Function Names	**Runtime %**	**Speedup (V100**
RTS (Radiative Transport)	23%	
MHD (Magnetohydrodynamics)	24%	13x
TVD (Total Variation Diminishing)	34%	39x
EOS (Equation of state)	9%	10x
INT (Integrate Tcheck)	7%	2x
OTHER	3%	

We have the option of computing the RTS several times per iteration. This will increase accuracy and compute time.

The other functions are also being accelerated, but will be less impactful than RTS under full load.

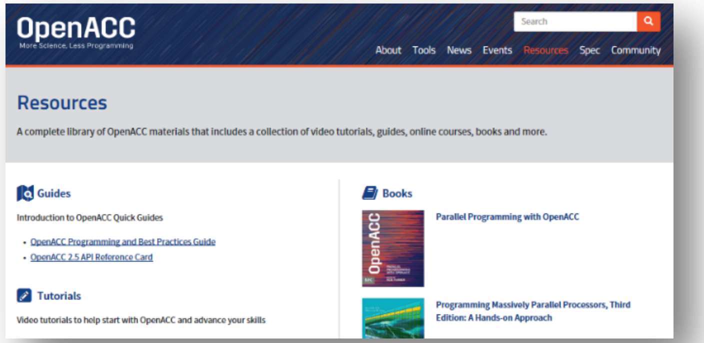
These results are gathered from NVIDIA PSG cluster.
Single V100 GPU
Intel Haswell, dual socket, 32 cores

OPENACC Resources

Guides ● Talks ● Tutorials ● Videos ● Books ● Spec ● Code Samples ● Teaching Materials ● Events ● Success Stories ● Courses ● Slack ● Stack Overflow

FREE Compilers

https://www.openacc.org/resources



Resources

A complete library of OpenACC materials that includes a collection of video tutorials, guides, online courses, books and more.

Guides

Introduction to OpenACC Quick Guides

- OpenACC Programming and Best Practices Guide
- OpenACC 2.5 API Reference Card

Tutorials

Video tutorials to help start with OpenACC and advance your skills

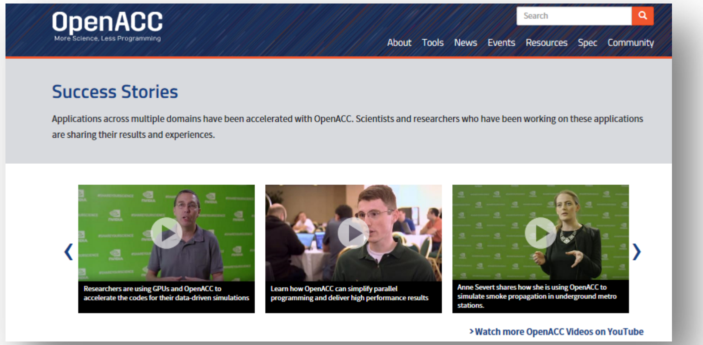
Books

Parallel Programming with OpenACC

Programming Massively Parallel Processors, Third Edition: A Hands-on Approach

Success Stories

https://www.openacc.org/success-stories



Success Stories

Applications across multiple domains have been accelerated with OpenACC. Scientists and researchers who have been working on these applications are sharing their results and experiences.

Researchers are using GPUs and OpenACC to accelerate the codes for their data-driven simulations

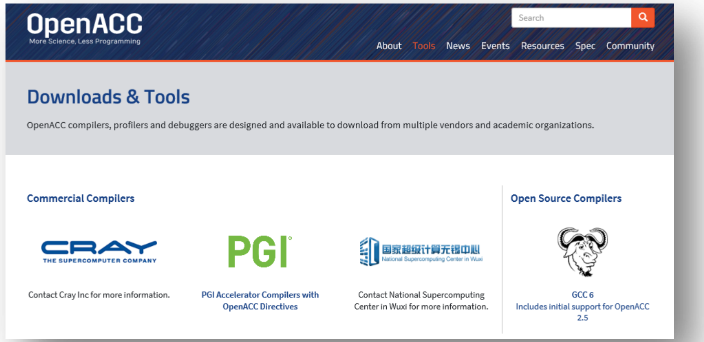
Learn how OpenACC can simplify parallel programming and deliver high performance results

Anne Severt shares how she is using OpenACC to simulate smoke propagation in underground metro stations.

> Watch more OpenACC Videos on YouTube

Compilers and Tools

https://www.openacc.org/tools



Downloads & Tools

OpenACC compilers, profilers and debuggers are designed and available to download from multiple vendors and academic organizations.

Commercial Compilers

Contact Cray Inc for more information.

PGI Accelerator Compilers with OpenACC Directives

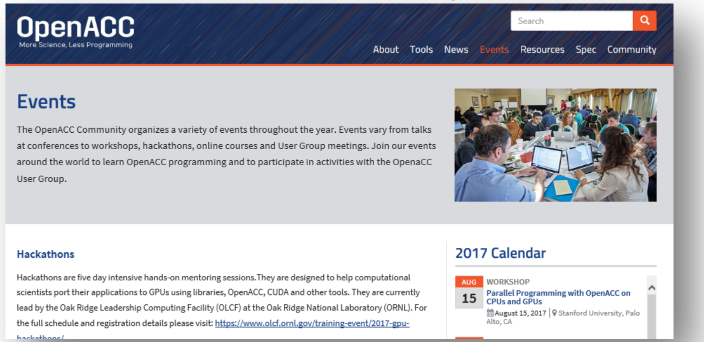
Contact National Supercomputing Center in Wuxi for more information.

Open Source Compilers

GCC 6
Includes initial support for OpenACC 2.5

Events

https://www.openacc.org/events



Events

The OpenACC Community organizes a variety of events throughout the year. Events vary from talks at conferences to workshops, hackathons, online courses and User Group meetings. Join our events around the world to learn OpenACC programming and to participate in activities with the OpenaCC User Group.

Hackathons

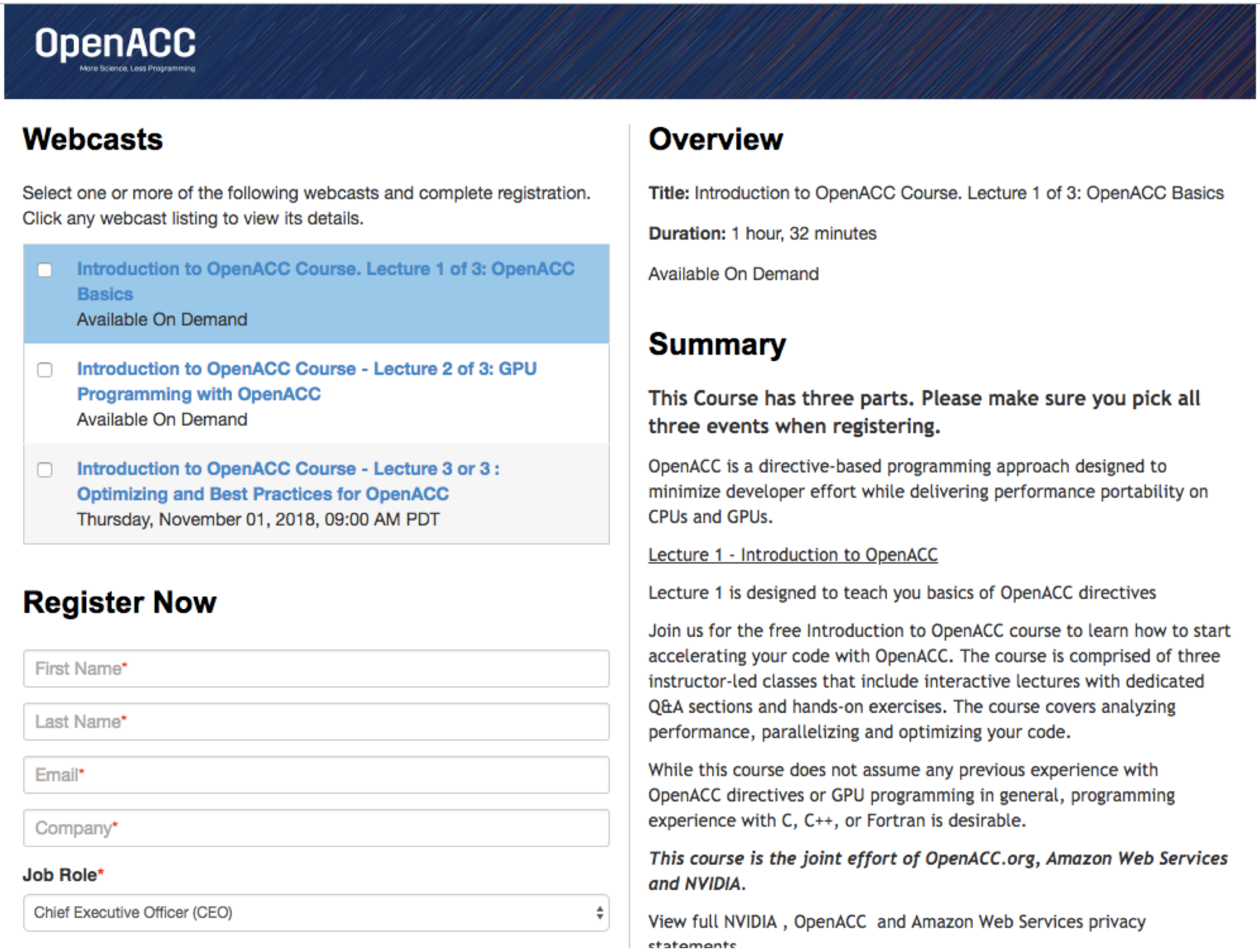
Hackathons are five day intensive hands-on mentoring sessions.They are designed to help computational scientists port their applications to GPUs using libraries, OpenACC, CUDA and other tools. They are currently lead by the Oak Ridge Leadership Computing Facility (OLCF) at the Oak Ridge National Laboratory (ORNL). For the full schedule and registration details please visit: https://www.olcf.ornl.gov/training-event/2017-gpu-hackathons/

2017 Calendar

AUG 15 WORKSHOP
Parallel Programming with OpenACC on CPUs and GPUs
August 15, 2017 | Stanford University, Palo Alto, CA

- An on-going OpenACC online course.
- 3 Modules
- 90 Minutes
- Recorded

https://event.on24.com/wcc/r/1821570/D79EB142A48182C8FF360FBCECE80D3E/155003?partnerref=Sunita



Webcasts

Select one or more of the following webcasts and complete registration. Click any webcast listing to view its details.

- [ ] Introduction to OpenACC Course. Lecture 1 of 3: OpenACC Basics
  Available On Demand
- [ ] Introduction to OpenACC Course - Lecture 2 of 3: GPU Programming with OpenACC
  Available On Demand
- [ ] Introduction to OpenACC Course - Lecture 3 or 3 : Optimizing and Best Practices for OpenACC
  Thursday, November 01, 2018, 09:00 AM PDT

Register Now

First Name*

Last Name*

Email*

Company*

Job Role*: Chief Executive Officer (CEO)

Overview

Title: Introduction to OpenACC Course. Lecture 1 of 3: OpenACC Basics

Duration: 1 hour, 32 minutes

Available On Demand

Summary

This Course has three parts. Please make sure you pick all three events when registering.

OpenACC is a directive-based programming approach designed to minimize developer effort while delivering performance portability on CPUs and GPUs.

Lecture 1 - Introduction to OpenACC

Lecture 1 is designed to teach you basics of OpenACC directives

Join us for the free Introduction to OpenACC course to learn how to start accelerating your code with OpenACC. The course is comprised of three instructor-led classes that include interactive lectures with dedicated Q&A sections and hands-on exercises. The course covers analyzing performance, parallelizing and optimizing your code.

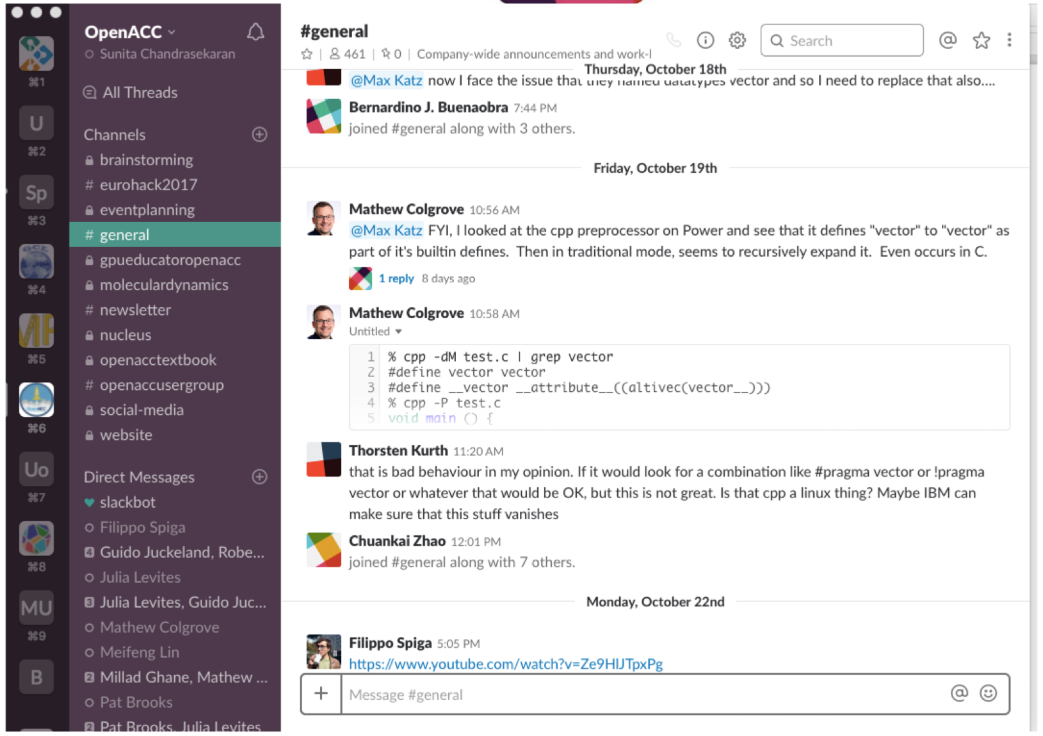
While this course does not assume any previous experience with OpenACC directives or GPU programming in general, programming experience with C, C++, or Fortran is desirable.

This course is the joint effort of OpenACC.org, Amazon Web Services and NVIDIA.

View full NVIDIA , OpenACC and Amazon Web Services privacy statements

Join OpenACC slack community

- https://www.openacc.org/community#slack
- Got technical questions?
- Want to promote any OpenACC related activity? Let us know!



Thursday, October 18th

@Max Katz now I face the issue that they named datatypes vector and so I need to replace that also....

Bernardino J. Buenaobra 7:44 PM
joined #general along with 3 others.

Friday, October 19th

Mathew Colgrove 10:56 AM
@Max Katz FYI, I looked at the cpp preprocessor on Power and see that it defines "vector" to "vector" as part of it's builtin defines. Then in traditional mode, seems to recursively expand it. Even occurs in C.
1 reply 8 days ago

Mathew Colgrove 10:58 AM
Untitled

```
% cpp -dM test.c | grep vector
#define vector vector
#define __vector __attribute__((altivec(vector__)))
% cpp -P test.c
void main () {
```

Thorsten Kurth 11:20 AM
that is bad behaviour in my opinion. If it would look for a combination like #pragma vector or !pragma vector or whatever that would be OK, but this is not great. Is that cpp a linux thing? Maybe IBM can make sure that this stuff vanishes

Chuankai Zhao 12:01 PM
joined #general along with 7 others.

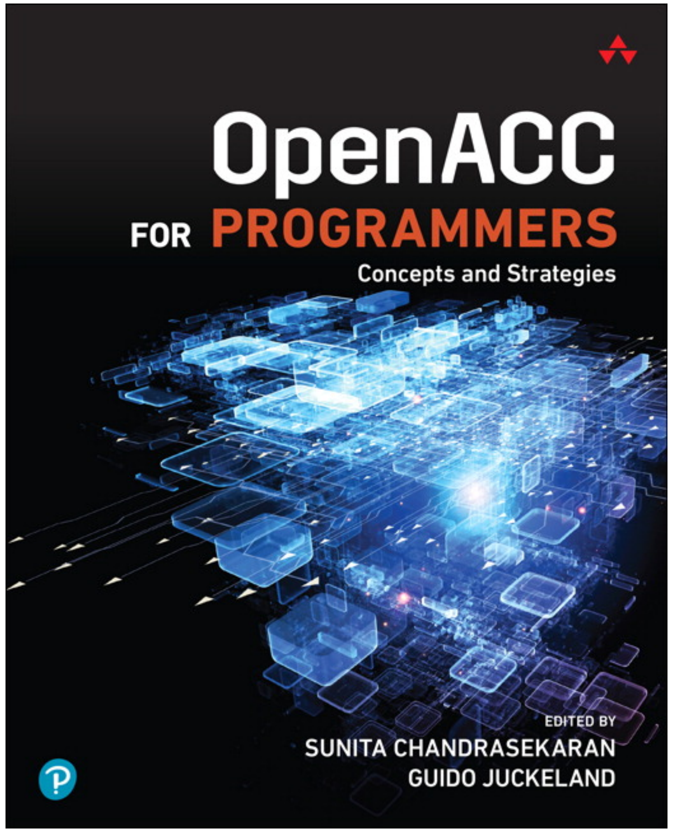
Monday, October 22nd

Filippo Spiga 5:05 PM
https://www.youtube.com/watch?v=Ze9HIJTpxPg

Message #general

OpenACC Textbook

- Recently (November 2017) published textbook
- Exercises from the textbook and Solution: https://github.com/OpenACCUserGroup/openacc_concept_strategies_book
- Jupyter notebooks for exercises also will be soon available



CRPL Computational Research and Programming Lab

- Thank you to all my wonderful collaborators

  NSF, ECP, NCAR, OpenACC, NVIDIA and Nemours