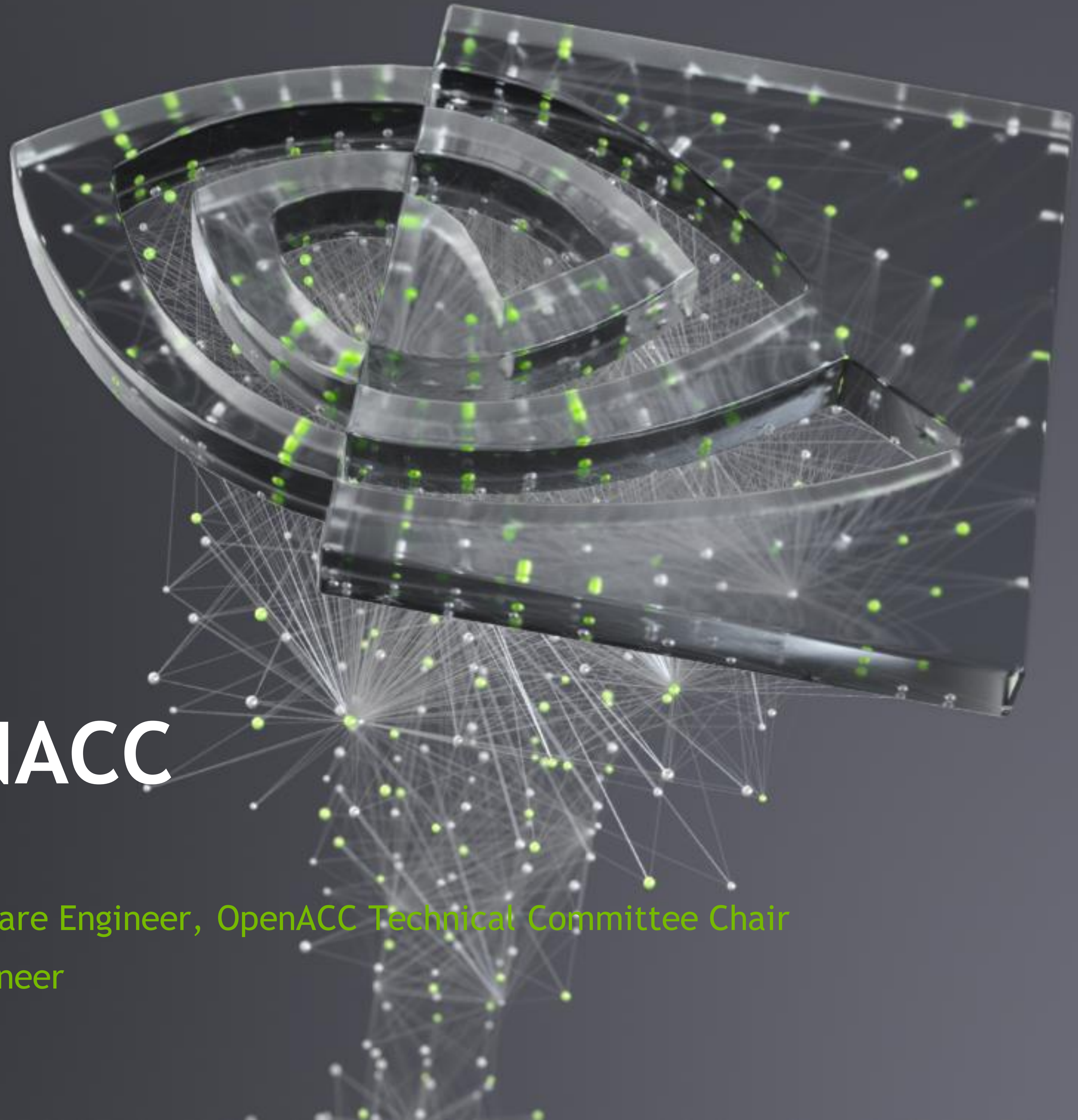
HIGHLY PARALLEL FORTRAN AND OPENACC DIRECTIVES

Jeff Larkin <jlarkin@nvidia.com> Sr. DevTech Software Engineer, OpenACC Technical Committee Chair

Michael Wolfe <mwolfe@nvidia.com> Compiler Engineer



OPENACC DIRECTIVES

a directive-based parallel programming model designed for usability, performance, and portability

3 OF TOP 5 HPC

Intersect360 RESEARCH

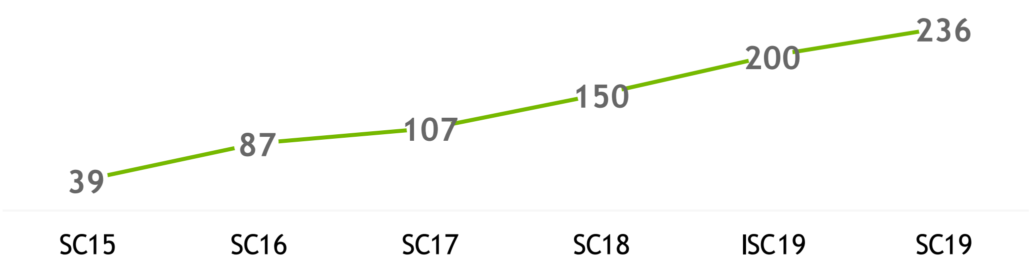
18% OF INCITE AT SUMMIT

summit

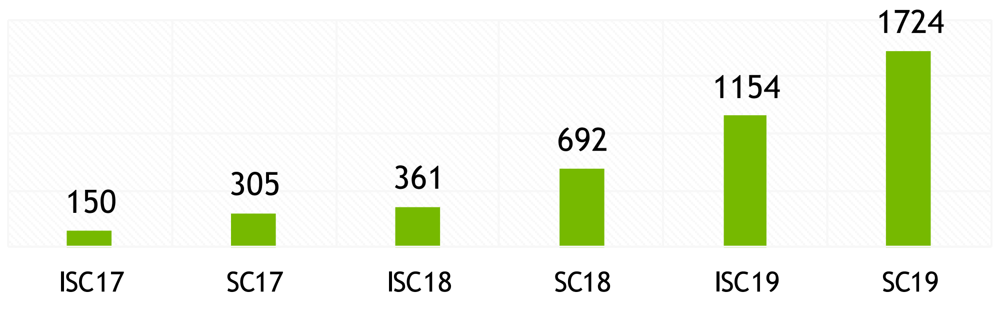
PLATFORMS SUPPORTED

NVIDIA GPU
X86 CPU
POWER CPU
Sunway
ARM CPU
AMD GPU

OPENACC APPS

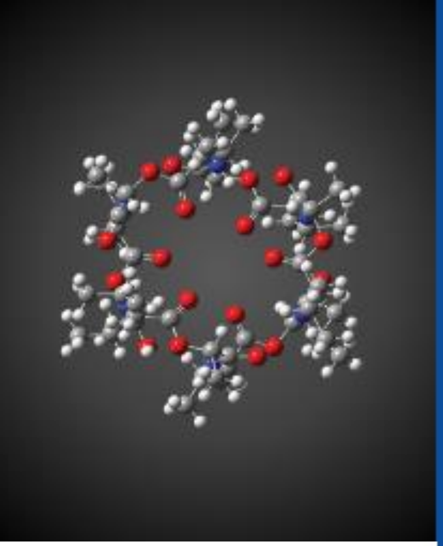


OPENACC SLACK MEMBERS



>200K 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.”

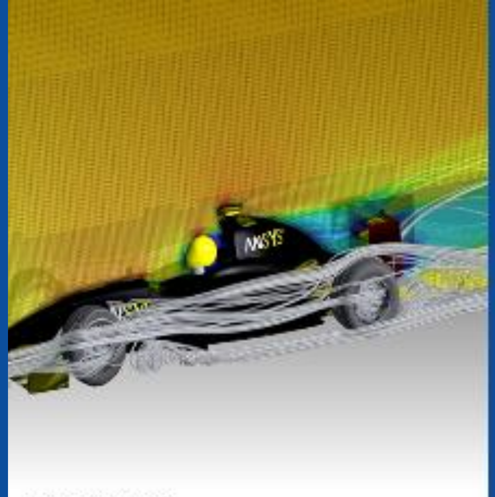
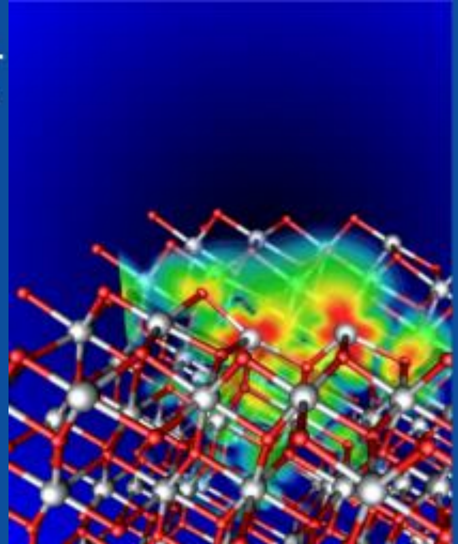


Image courtesy: ANSYS

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.”

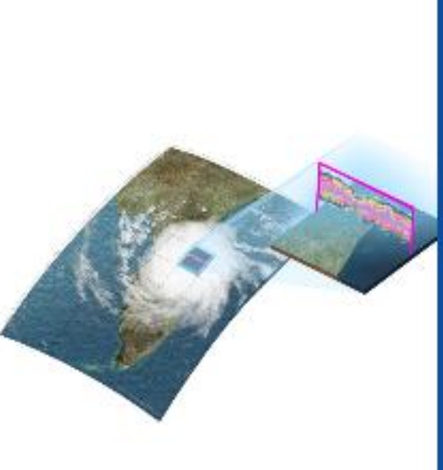


Image courtesy: Oak Ridge National Laboratory

E3SM

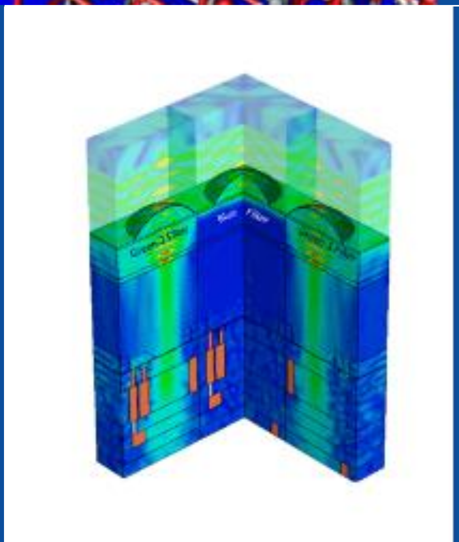
Mark A. Taylor
Multiphysics Applications
Sandia

> “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

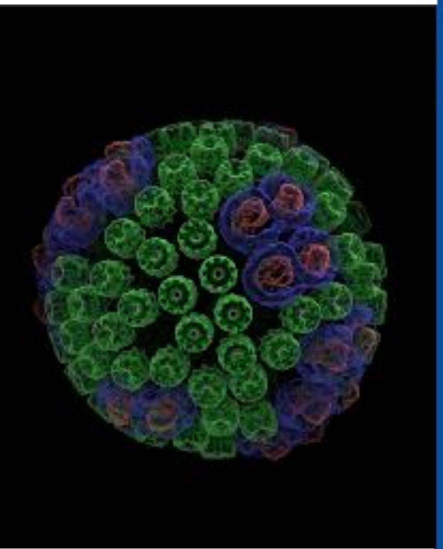
> “Using OpenACC, we've GPU-accelerated the Synopsys TCAD Sentaurus Device EMW simulator to speed up optical simulations of image sensors. GPUs are key to improving simulation throughput in the design of advanced image sensors.”

Image courtesy: NCAR

MPAS-A

Richard Loft
Director, Technology Development
NCAR

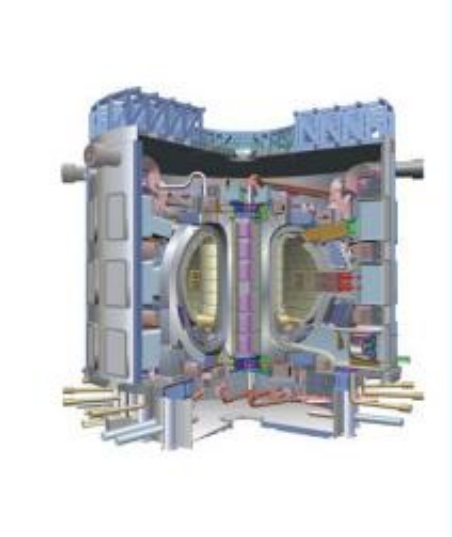
> “Our team has been evaluating OpenACC as a pathway to performance portability for the *Model for Prediction (MPAS)* atmospheric model. Using this approach on the MPAS dynamical core, we have achieved performance on a single P100 GPU equivalent to 2.7 dual socketed Intel Xeon nodes on our new Cheyenne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



VMD

John Stone
Senior Research Programmer
Beckham 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

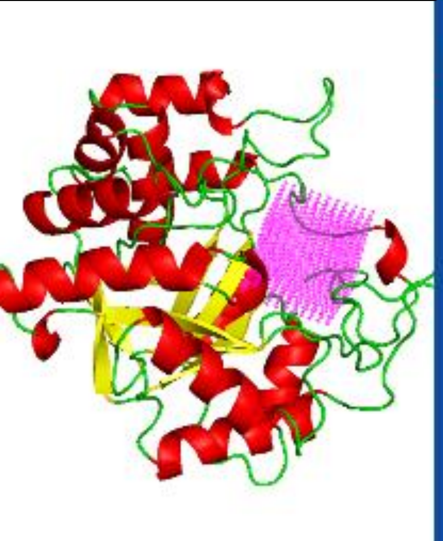
More Science, Less Programming

Map courtesy University of Tokyo

GAMERA

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

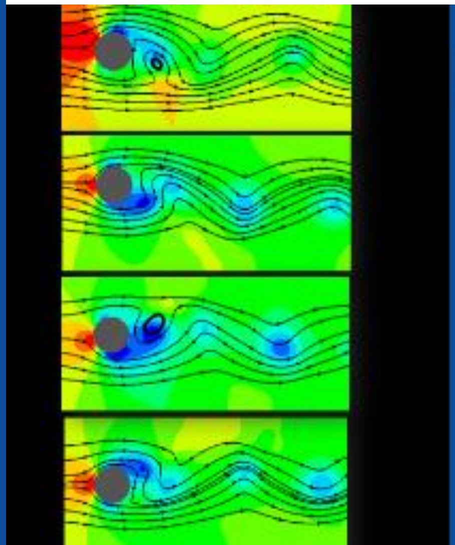
> “With OpenACC and a compute node based on NVIDIA's Tesla P100 GPU, we achieved more than a 14X speed up over a K Computer node running our earthquake disaster simulation code”



SANJEEVINI

Abhilash Jayaraj
Project Scientist
Indian Institute of Technology
New Delhi

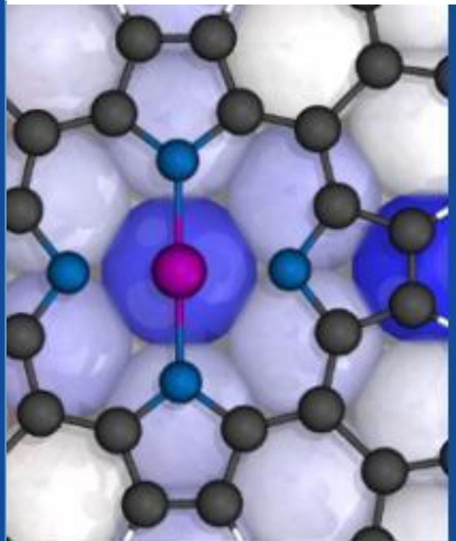
> “In an academic environment maintenance and speedup of existing codes is a tedious task. OpenACC provides a great platform for computational scientists to accomplish both tasks without involving a lot of efforts or manpower in speeding up the entire computational task.”



IBM-CFD

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

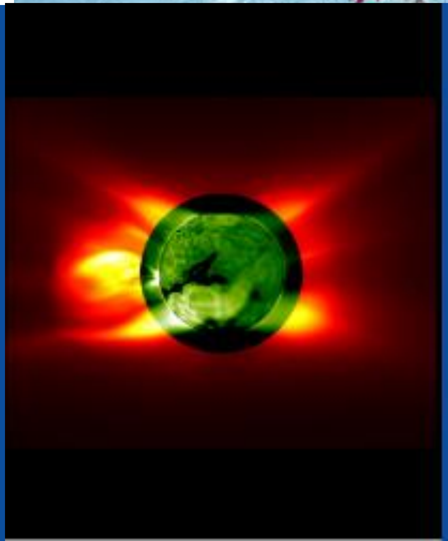
> “OpenACC can prove to be a handy tool for computational engineers and researchers to obtain fast solution of non-linear dynamics problem. In immersed boundary incompressible CFD, we have obtained order of magnitude reduction in computing time by porting several components of our legacy codes to GPU. 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.”

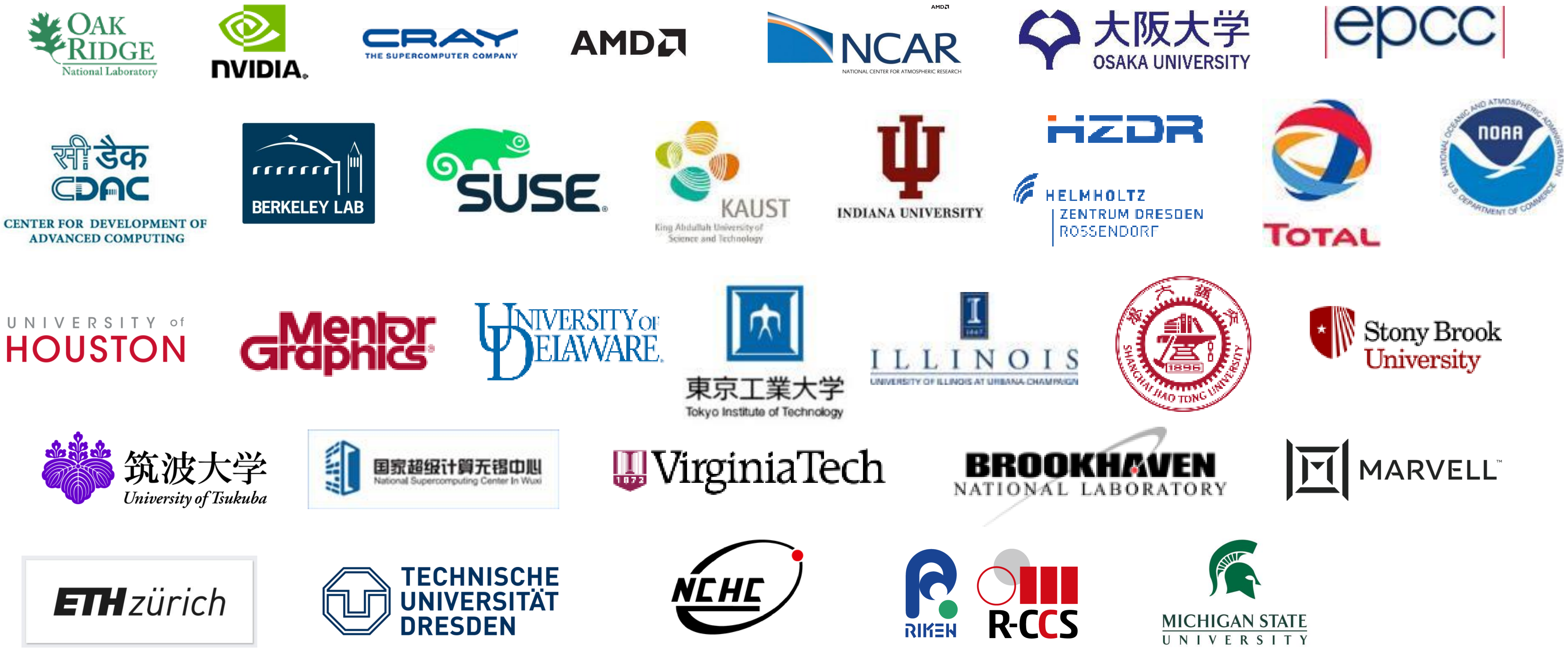


MAS

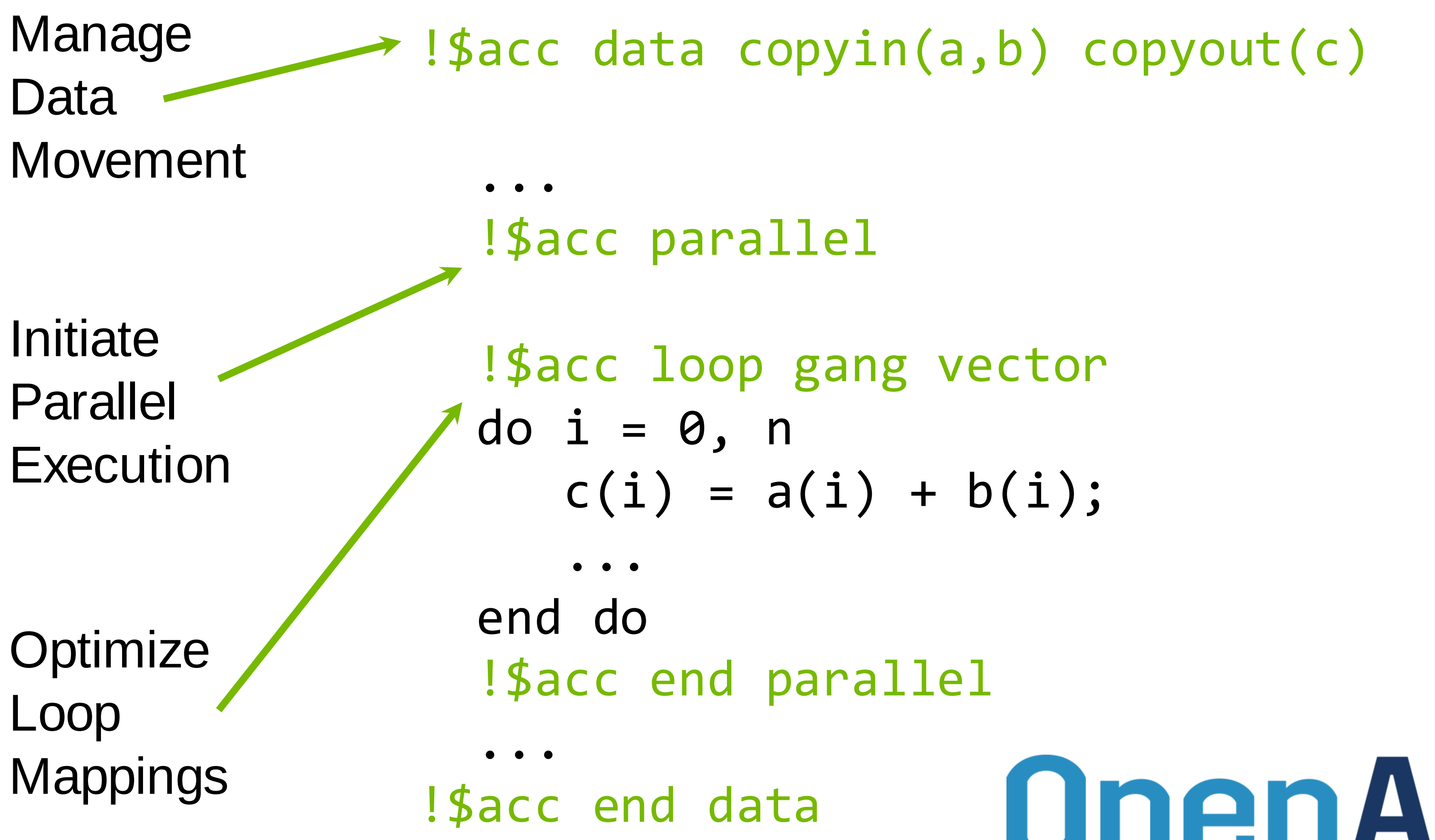
Ronald M. Caplan
Computational Scientist
Predictive Science Inc.

> “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 Members


OAK RIDGE National Laboratory
NVIDIA.
CRAY THE SUPERCOMPUTER COMPANY
AMD
NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
大阪大学 OSAKA UNIVERSITY
epcc
सी-डैक CDAC CENTER FOR DEVELOPMENT OF ADVANCED COMPUTING
BERKELEY LAB
SUSE
KAUST King Abdullah University of Science and Technology
INDIANA UNIVERSITY
HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TOTAL
NOAA NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION U.S. DEPARTMENT OF COMMERCE
UNIVERSITY of HOUSTON
Mentor Graphics
UNIVERSITY OF DELAWARE
東京工業大学 Tokyo Institute of Technology
ILLINOIS UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
SHANGHAI JIAO TONG UNIVERSITY 1896
Stony Brook University
筑波大学 University of Tsukuba
国家超级计算无锡中心 National Supercomputing Center In Wuxi
VirginiaTech 1872
BROOKHAVEN NATIONAL LABORATORY
MARVELL
ETH zürich
TECHNISCHE UNIVERSITÄT DRESDEN
NCHC
RIKEN
R-CCS
MICHIGAN STATE UNIVERSITY

OpenACC Directives



Manage Data Movement

Initiate Parallel Execution

Optimize Loop Mappings

```
!$acc data copyin(a,b) copyout(c)

  ...
  !$acc parallel

  !$acc loop gang vector
  do i = 0, n
     c(i) = a(i) + b(i);
     ...
  end do
  !$acc end parallel
  ...
!$acc end data
```

- Incremental
- Single source
- Interoperable
- Performance portable
- CPU, GPU, and more

OpenACC

Directives for Accelerators

THE ROLE OF DIRECTIVES FOR PARALLEL PROGRAMMING

Directives convey additional information to the compiler.

Serial Programming Languages

Parallel Programming Languages

PARALLEL PROGRAMMING CONCERNS

What	Why
Express Parallelism	What can and should be run in parallel?
Optimize Parallelism	How can I execute faster on the parallel hardware I have?
Manage Compute Locality	Executing on the local compute core isn’t enough. Threading and offloading are now the norm.
Manage Data Locality	Memory is no longer simple and flat. Memory has hierarchies, which are made even more complex when considering heterogenous architectures.
Asynchronous Operations	Asynchronicity is increasingly required to cover various overheads.

EXPRESS PARALLELISM IN OPENACC

What can and should be run in parallel?

```
!$acc parallel (or kernels)
!$acc loop independent
do i=1,N
  C(i) = A(i) + B(i)
end do
```

Not shown here:

- Private Clause
- Reduction Clause
- Atomic directive

- Assert data-independence to the compiler (*can* be run in parallel)
- Identify parallelism-blockers (private, reduction, atomic)
- Hint a desire for the iterations to be run in parallel (*should* be run in parallel)

EXPRESS PARALLELISM IN FORTRAN 2018

What can and should be run in parallel?

```
do concurrent (i=1:N)
  C(i) = A(i) + B(i)
end do

C(:) = A(:) + B(:)
```

Not shown here:

- Local Clause
- Other intrinsics

- Assert ability for iterations to be run in any order (possibly concurrently)
- Identify privatization of variables, if necessary
- Currently no loop-level reduction or atomics

OPTIMIZE PARALLELISM IN OPENACC

How can it run better in parallel?

```
!$acc loop independent vector(128)
do i=1,N
  C(i) = A(i) + B(i)
end do
```

Not shown here:

- Collapse
- Tile
- Gang
- Worker
- Device_type Specification

- Guide the compiler to better parallel decomposition on a given hardware.
- Potentially transform loops to better expose parallelism (collapse) or locality (tile)
- Only analogue in Fortran is re-structuring or changing compiler flags. Good enough?

MANAGE COMPUTE LOCALITY IN OPENACC

Where should it be run in parallel?

```
!$acc set device device_num(0)
!$acc&     device_type(nvidia)
!$acc parallel loop
do i=1,N
  C(i) = A(i) + B(i)
end do
```

Not shown here:

- Self clause
- Serial or Kernels construct

- Instruct the compiler where to execute the region
- Or leave it up to the runtime
- Potentially execute on multiple devices (including the host) simultaneously

MANAGE DATA LOCALITY IN OPENACC

Where should data live and for how long?

```
!$acc data copyin(A,B)
!$acc&     copyout(C)
!$acc parallel loop
do i=1,N
  C(i) = A(i) + B(i)
end do
!$acc end data
```

- Identify data structures that are required when distinct memories exist.
- Reduce data motion between distinct memories
- Give compiler context for data reuse

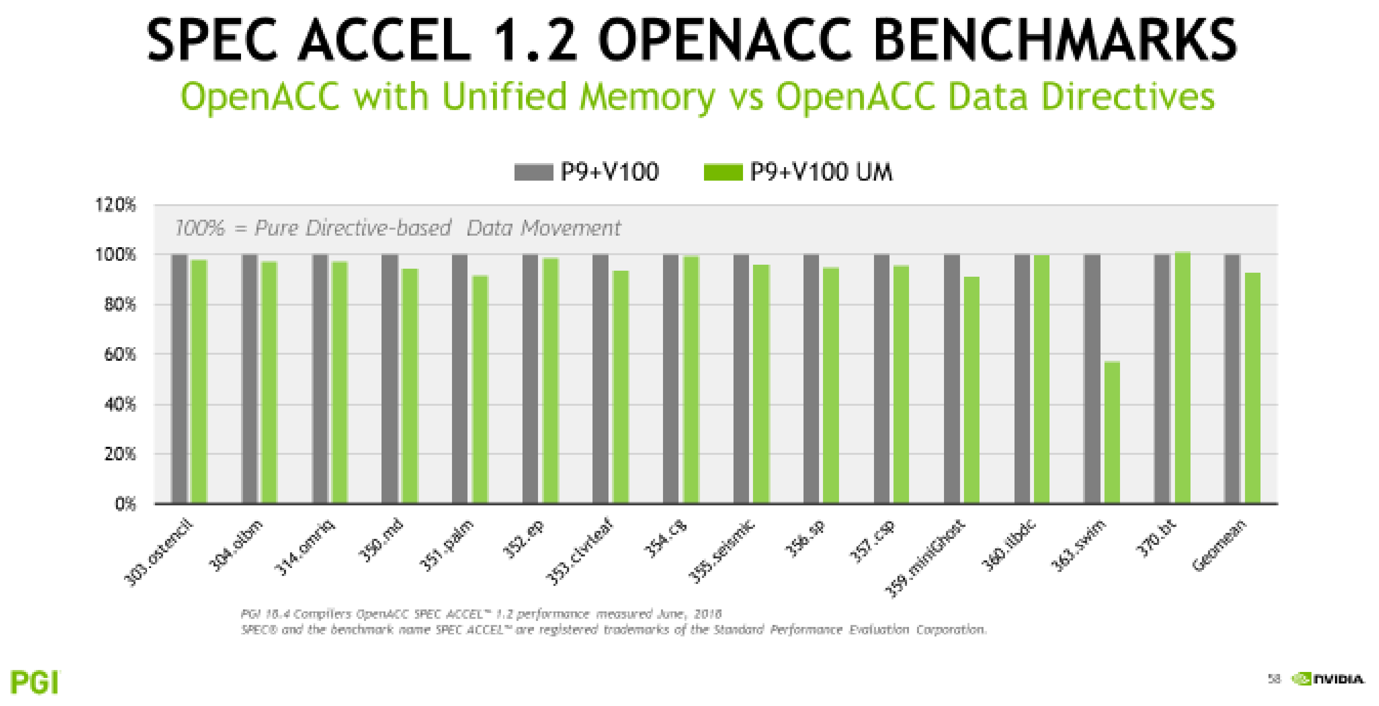
Not shown here:

- Unstructured data management
- Update directive
- Deep-copy
- Device data interoperability
- Cache directive

DATA LOCALITY

Unified Memory and the Heterogenous Memory Manager (HMM)

- Modern GPUs can handle paging memory to/from the GPU automatically (locality matters)
- HMM is a Linux kernel feature for exposing all memory in this way (ubiquity matters)
- Can the 99% work without explicitly optimizing memory locality?

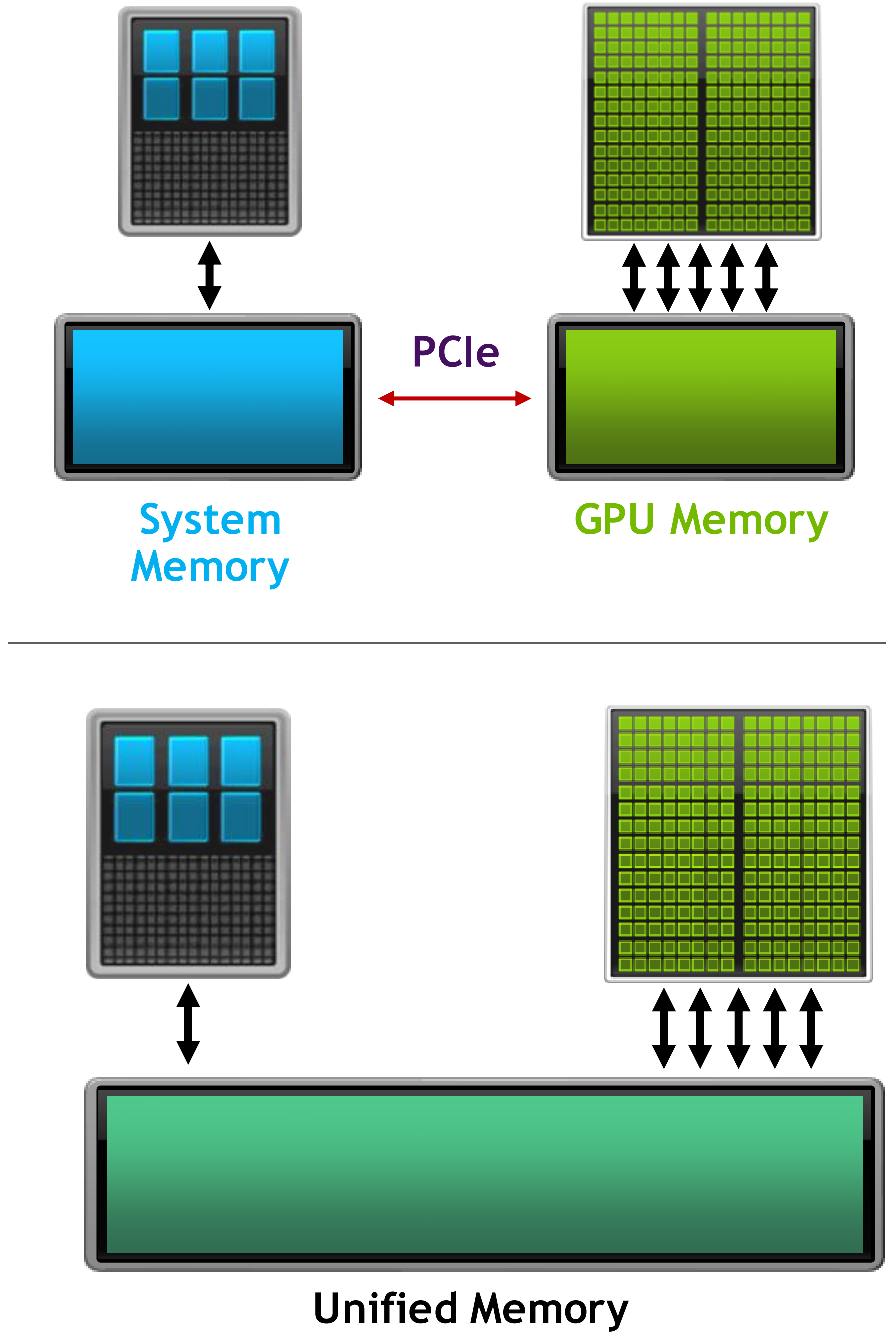


SPEC ACCEL 1.2 OPENACC BENCHMARKS

OpenACC with Unified Memory vs OpenACC Data Directives

PGI 18.4 Compilers OpenACC SPEC ACCEL™ 1.2 performance measured June, 2018
SPEC® and the benchmark name SPEC ACCEL™ are registered trademarks of the Standard Performance Evaluation Corporation.

PGI



PCIe

System Memory

GPU Memory

Unified Memory

ASYNCHRONOUS EXECUTION IN OPENACC

What can be run concurrently?

```
!$acc update device(A) async(1)
!$acc parallel loop async(1)
do i=1,N
  C(i) = A(i) + B(i)
end do
!$acc update self(C) async(1)
!$acc wait(1)
```

- Expose dependence (or lack of) between different regions
- Introduce potential for overlapping operations
- Increase overall system utilization

PARALLEL PROGRAMMING CONCERNS

	Fortran	OpenACC
Identify Parallelism	✓	✓
Optimize Parallelism	⊘ Good Enough?	✓
Manage Compute Locality	⊘	✓
Manage Data Locality	✓	✓
Asynchronous Operations	⊘	✓

CLOVERLEAF V1.3

http://uk-mac.github.io/CloverLeaf

AWE Hydrodynamics mini-app

6500+ lines, !$acc kernels

OpenACC or OpenMP or do concurrent

Source on GitHub

FORTRAN WITH OPENACC DIRECTIVES

```
% pgfortran -fast -ta=tesla:managed -Minfo -c
PdV_kernel.f90
pdv_kernel:
    ...
    77, Loop is parallelizable
    79, Loop is parallelizable
        Accelerator kernel generated
        Generating Tesla code
        77, !$acc loop gang, vector(4) ! blockidx%y
                                       ! threadidx%y
        79, !$acc loop gang, vector(32)! blockidx%x
                                       ! threadidx%x
    ...
```

```
 75 !$ACC KERNELS
 76 !$ACC LOOP INDEPENDENT
 77     DO k=y_min,y_max
 78 !$ACC LOOP INDEPENDENT PRIVATE(right_flux,left_flux,top_flux,bottom_flux,total_flux,
                              min_cell_volume,energy_change,recip_volume)
 79       DO j=x_min,x_max
 80
 81         left_flux=  (xarea(j  ,k  )*(xvel0(j  ,k  )+xvel0(j  ,k+1)                 &
 82                                     +xvel0(j  ,k  )+xvel0(j  ,k+1)))*0.25_8*dt*0.5
 83         right_flux= (xarea(j+1,k  )*(xvel0(j+1,k  )+xvel0(j+1,k+1)                 &
 84                                     +xvel0(j+1,k  )+xvel0(j+1,k+1)))*0.25_8*dt*0.5
 85         bottom_flux=(yarea(j  ,k  )*(yvel0(j  ,k  )+yvel0(j+1,k  )                 &
 86                                     +yvel0(j  ,k  )+yvel0(j+1,k  )))*0.25_8*dt*0.5
 87         top_flux=   (yarea(j  ,k+1)*(yvel0(j  ,k+1)+yvel0(j+1,k+1)                 &
 88                                     +yvel0(j  ,k+1)+yvel0(j+1,k+1)))*0.25_8*dt*0.5
 89         total_flux=right_flux-left_flux+top_flux-bottom_flux
 90
 91         volume_change(j,k)=volume(j,k)/(volume(j,k)+total_flux)
 92
 93         min_cell_volume=MIN(volume(j,k)+right_flux-left_flux+top_flux-bottom_flux &
 94                            ,volume(j,k)+right_flux-left_flux                      &
 95                            ,volume(j,k)+top_flux-bottom_flux)
 97         recip_volume=1.0/volume(j,k)
 99         energy_change=(pressure(j,k)/density0(j,k)+viscosity(j,k)/density0(j,k))*...
101         energy1(j,k)=energy0(j,k)-energy_change
103         density1(j,k)=density0(j,k)*volume_change(j,k)
105       ENDDO
106     ENDDO
107 !$ACC END KERNELS
```

http://uk-mac.github.io/CloverLeaf

FORTRAN 2018 DO CONCURRENT

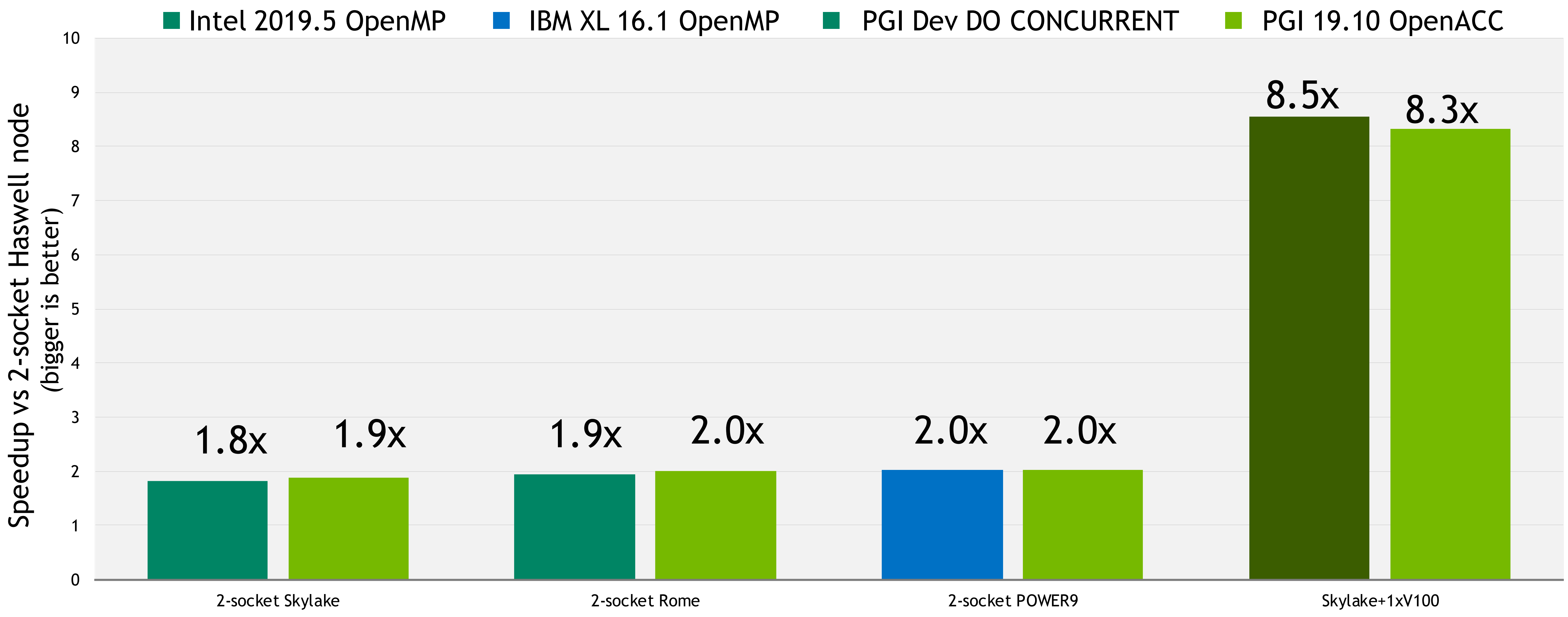
```
% pgfortran –fast –ta=tesla:managed –Minfo -c
PdV_kernel.f90
pdv_kernel:
    ...
    77, Do concurrent is parallelizable
        Accelerator kernel generated
        Generating Tesla code
        77, !$acc loop gang, vector(4) ! blockidx%y
                                       ! threadidx%y
            !$acc loop gang, vector(32)! blockidx%x
                                       ! threadidx%x
    ...
```

```
 75
 76
 77    DO CONCURRENT (k=y_min:y_max, j=x_min:x_max) &
 78                         LOCAL (right_flux,left_flux,top_flux,bottom_flux,total_flux, &
                                   min_cell_volume,energy_change,recip_volume)
 79
 80
 81        left_flux=  (xarea(j  ,k  )*(xvel0(j  ,k  )+xvel0(j  ,k+1)                 &
 82                                    +xvel0(j  ,k  )+xvel0(j  ,k+1)))*0.25_8*dt*0.5
 83        right_flux= (xarea(j+1,k  )*(xvel0(j+1,k  )+xvel0(j+1,k+1)                 &
 84                                    +xvel0(j+1,k  )+xvel0(j+1,k+1)))*0.25_8*dt*0.5
 85        bottom_flux=(yarea(j  ,k  )*(yvel0(j  ,k  )+yvel0(j+1,k  )                 &
 86                                    +yvel0(j  ,k  )+yvel0(j+1,k  )))*0.25_8*dt*0.5
 87        top_flux=   (yarea(j  ,k+1)*(yvel0(j  ,k+1)+yvel0(j+1,k+1)                 &
 88                                    +yvel0(j  ,k+1)+yvel0(j+1,k+1)))*0.25_8*dt*0.5
 89        total_flux=right_flux-left_flux+top_flux-bottom_flux
 90
 91        volume_change(j,k)=volume(j,k)/(volume(j,k)+total_flux)
 92
 93        min_cell_volume=MIN(volume(j,k)+right_flux-left_flux+top_flux-bottom_flux &
 94                           ,volume(j,k)+right_flux-left_flux                      &
 95                           ,volume(j,k)+top_flux-bottom_flux)
 97        recip_volume=1.0/volume(j,k)
 99        energy_change=(pressure(j,k)/density0(j,k)+viscosity(j,k)/density0(j,k))*...
101        energy1(j,k)=energy0(j,k)-energy_change
103        density1(j,k)=density0(j,k)*volume_change(j,k)
105
106    ENDDO
107
```

https://github.com/UoB-HPC/CloverLeaf_doconcurrent

FORTRAN 2018 DO CONCURRENT FOR V100

CloverLeaf AWE hydrodynamics mini-App, bm32 data set



Systems: Skylake 2x20 core Xeon Gold server (perf-sky-6) one thread per core, Rome: Two 24 core AMD EPYC 7451 CPUs @ 2.9GHz w/ 256GB memory; POWER9 DD2.1 server (perf-wsn1) two threads per core
Compilers: Intel 2019.5.281, PGI 19.10, IBM XL 16.1.1.3
Benchmark: CloverLeaf v1.3 OpenACC, OpenMP and DoConcurrent versions downloaded from https://github.com/UoB-HPC the week of June 10, 2019

https://github.com/UoB-HPC/CloverLeaf_doconcurrent

THE FUTURE OF PARALLEL PROGRAMMING

Standard Languages | Directives | Specialized Languages

```
std::for_each_n(POL, idx(0), n,
                [&](Index_t i){
    y[i] += a*x[i];
});
```

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

Drive Base Languages to Better Support Parallelism

```
!$acc data copy(x,y)

...

do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo

...

!$acc end data
```

Augment Base Languages with Directives

```
attribute(global)
subroutine saxpy(n, a, x, y) {
  int i = blockIdx%x*blockDim%x +
          threadIdx%x;
  if (i < n) y(i) += a*x(i)
}
program main
  real        :: x(:), y(:)
  real,device :: d_x(:), d_y(:)
  d_x = x
  d_y = y

  call saxpy
    <<<(N+255)/256,256>>>(...)

  y = d_y
```

Maximize Performance with Specialized Languages & Intrinsics

THE FUTURE OF PARALLEL PROGRAMMING

Standard Languages | Directives | Specialized Languages

```
std::for_each_n(POL, idx(0), n,
                [&](Index_t i){
    y[i] += a*x[i];
});
```

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

Drive Base Languages to Better Support Parallelism

```
!$acc data copy(x,y)

...

do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo

...

!$acc end data
```

Augment Base Languages with Directives

```
attribute(global)
subroutine saxpy(n, a, x, y) {
  int i = blockIdx%x*blockDim%x +
          threadIdx%x;
  if (i < n) y(i) += a*x(i)
}
program main
  real        :: x(:), y(:)
  real,device :: d_x(:), d_y(:)
  d_x = x
  d_y = y

  call saxpy
    <<<(N+255)/256,256>>>(...)

  y = d_y
```

Maximize Performance with Specialized Languages & Intrinsics

THE FUTURE OF PARALLEL PROGRAMMING

Standard Languages | Directives | Specialized Languages

```
std::for_each_n(POL, idx(0), n,
                [&](Index_t i){
    y[i] += a*x[i];
});
```

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

Drive Base Languages to Better Support Parallelism

```
!$acc data copy(x,y)

...

!$acc parallel loop async
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
...

!$acc end data
```

Augment Base Languages with Directives

```
attribute(global)
subroutine saxpy(n, a, x, y) {
  int i = blockIdx%x*blockDim%x +
          threadIdx%x;
  if (i < n) y(i) += a*x(i)
}
program main
  real        :: x(:), y(:)
  real,device :: d_x(:), d_y(:)
  d_x = x
  d_y = y

  call saxpy
    <<<(N+255)/256,256>>>(...)

  y = d_y
```

Maximize Performance with Specialized Languages & Intrinsics

FUTURE DIRECTIONS FOR OPENACC AND FORTRAN

DO CONCURRENT – Is this ready for widespread use? Is it enough?

Fortran reductions – Critical for many OpenACC applications. Significant restructuring otherwise.

Co-Arrays – How do they fit into this picture?

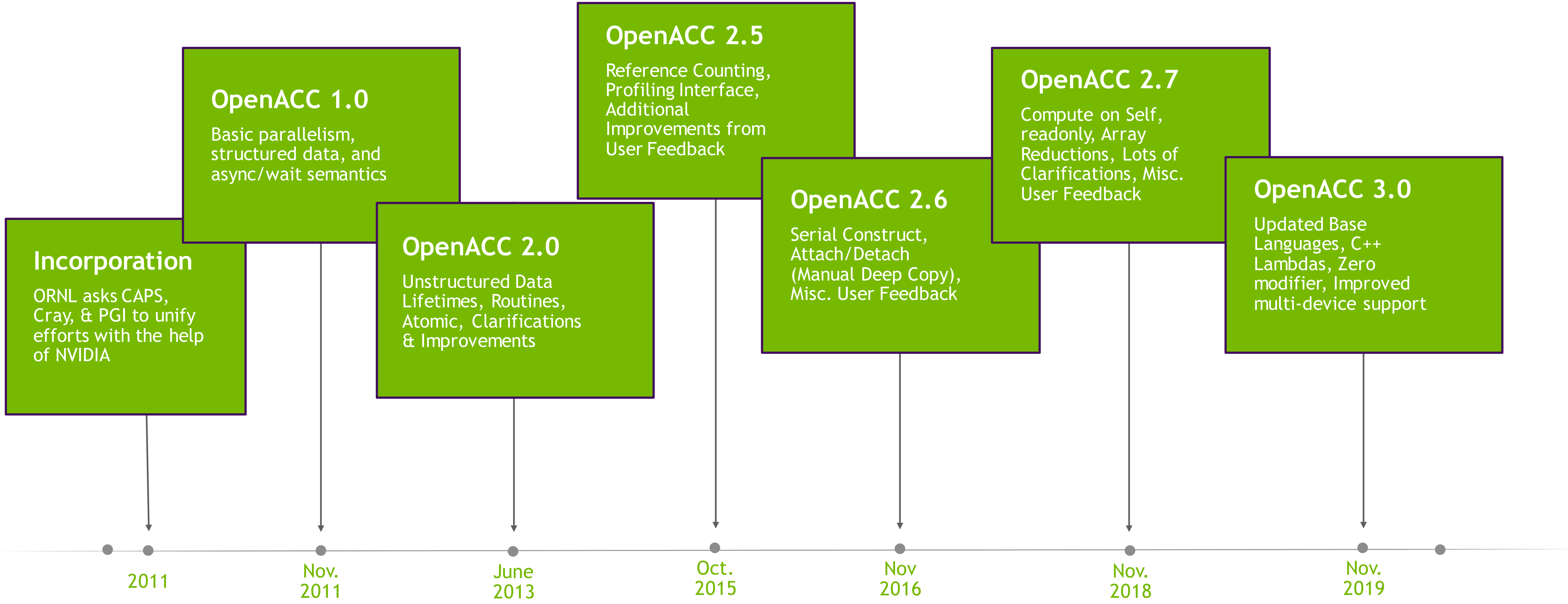
Are there other gaps that need filling? What are our common challenges?

How can the OpenACC and Fortran communities work together more closely?

feedback@openacc.org

BACK-UP

A BRIEF HISTORY OF OPENACC



Incorporation

ORNL asks CAPS, Cray, & PGI to unify efforts with the help of NVIDIA

2011

OpenACC 1.0

Basic parallelism, structured data, and async/wait semantics

Nov. 2011

OpenACC 2.0

Unstructured Data Lifetimes, Routines, Atomic, Clarifications & Improvements

June 2013

OpenACC 2.5

Reference Counting, Profiling Interface, Additional Improvements from User Feedback

Oct. 2015

OpenACC 2.6

Serial Construct, Attach/Detach (Manual Deep Copy), Misc. User Feedback

Nov 2016

OpenACC 2.7

Compute on Self, readonly, Array Reductions, Lots of Clarifications, Misc. User Feedback

Nov. 2018

OpenACC 3.0

Updated Base Languages, C++ Lambdas, Zero modifier, Improved multi-device support

Nov. 2019