# PERFORMANCE PORTABILITY WITH OPENACC

Jeff Larkin, NVIDIA, November 2015

# TWO TYPES OF PORTABILITY

## FUNCTIONAL PORTABILITY

The ability for a single code to run anywhere.

## PERFORMANCE PORTABILITY

The ability for a single code to run well anywhere.

# TWO TYPES OF PORTABILITY

| FUNCTIONAL PORTABILITY | PERFORMANCE PORTABILITY |
| --- | --- |
| The ability for a single code to run anywhere. | The ability for a single code to run well anywhere. |

Many programming models provide functional portability, but can we provide performance portability too?

# OPENACC

## Directives for Portable Parallelism

### OPEN

CRAY THE SUPERCOMPUTER COMPANY

PathScale High Performance Compilers

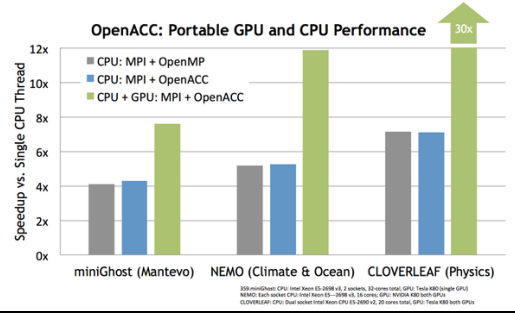
PGI®

### SINGLE SOURCE

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

### PORTABLE


### HIGH PERFORMANCE



“ [The] focus on mythical ‘Performance Portability’ misses the point. The issue is ‘maintainability’.”

*Enabling Application Portability across HPC Platforms: An Application Perspective*, OpenMPCon 2015

“ Having a common code base using a portable programming environment… even if you must fill the code with if-defs or have architecture specific versions of key kernels… is the only way to support maintainability.”

*Enabling Application Portability across HPC Platforms: An Application Perspective*, OpenMPCon 2015

# MAINTAINABLE CODE

## Portable parallelism is easier to maintain.

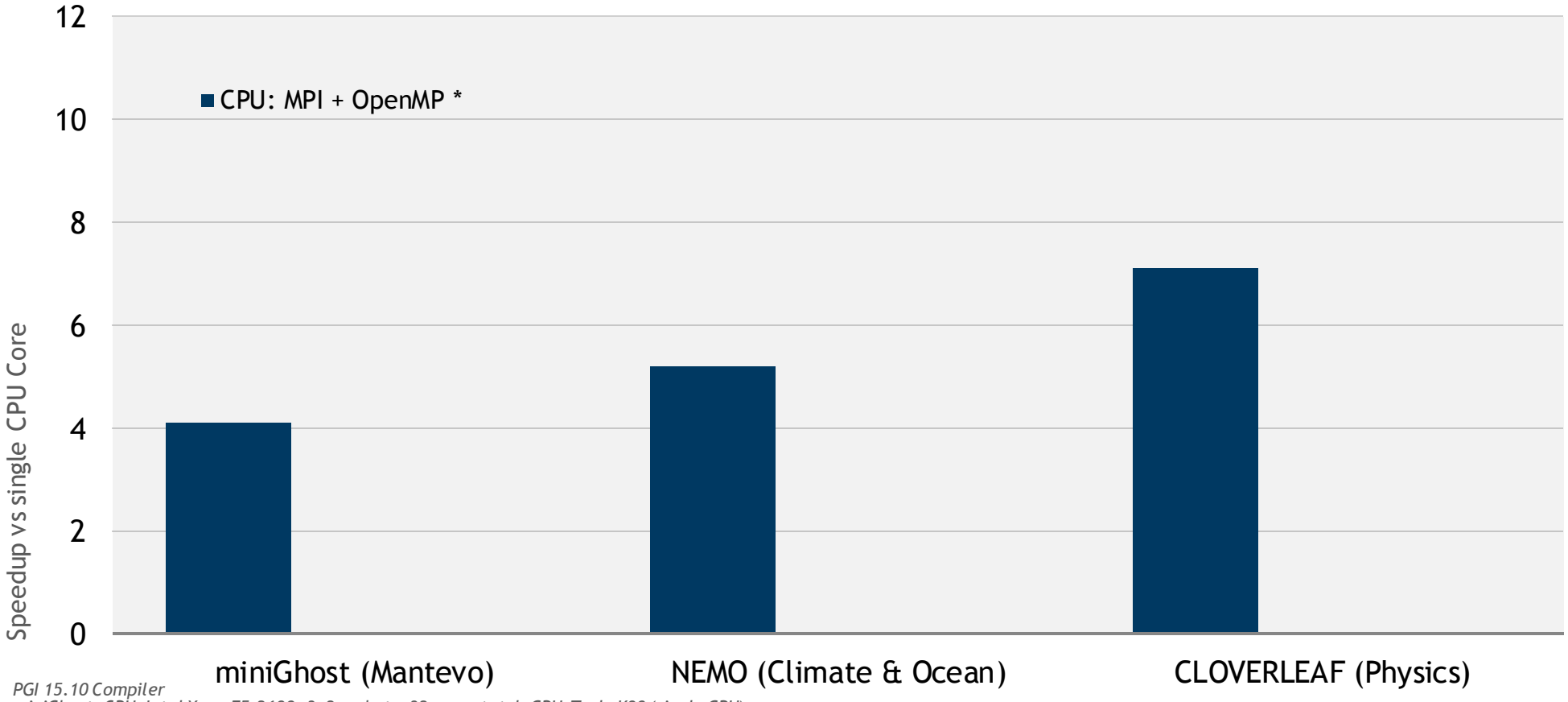
```
#if defined(CPU)
#pragma omp parallel for simd
#elif defined(MIC)
#pragma omp target teams distribute \
       parallel for simd
#elif defined(OMP_GPU)
#pragma omp target teams distribute \
       parallel for schedule(static,1)
#elif defined(SOMETHING_ELSE)
#pragma omp target ...
#endif
for(int i = 0; i < N; i++)
```

```
#pragma acc parallel loop
for(int i = 0; i < N; i++)
```

Performance portability enables maintainable code.

# But is performance portability a myth?

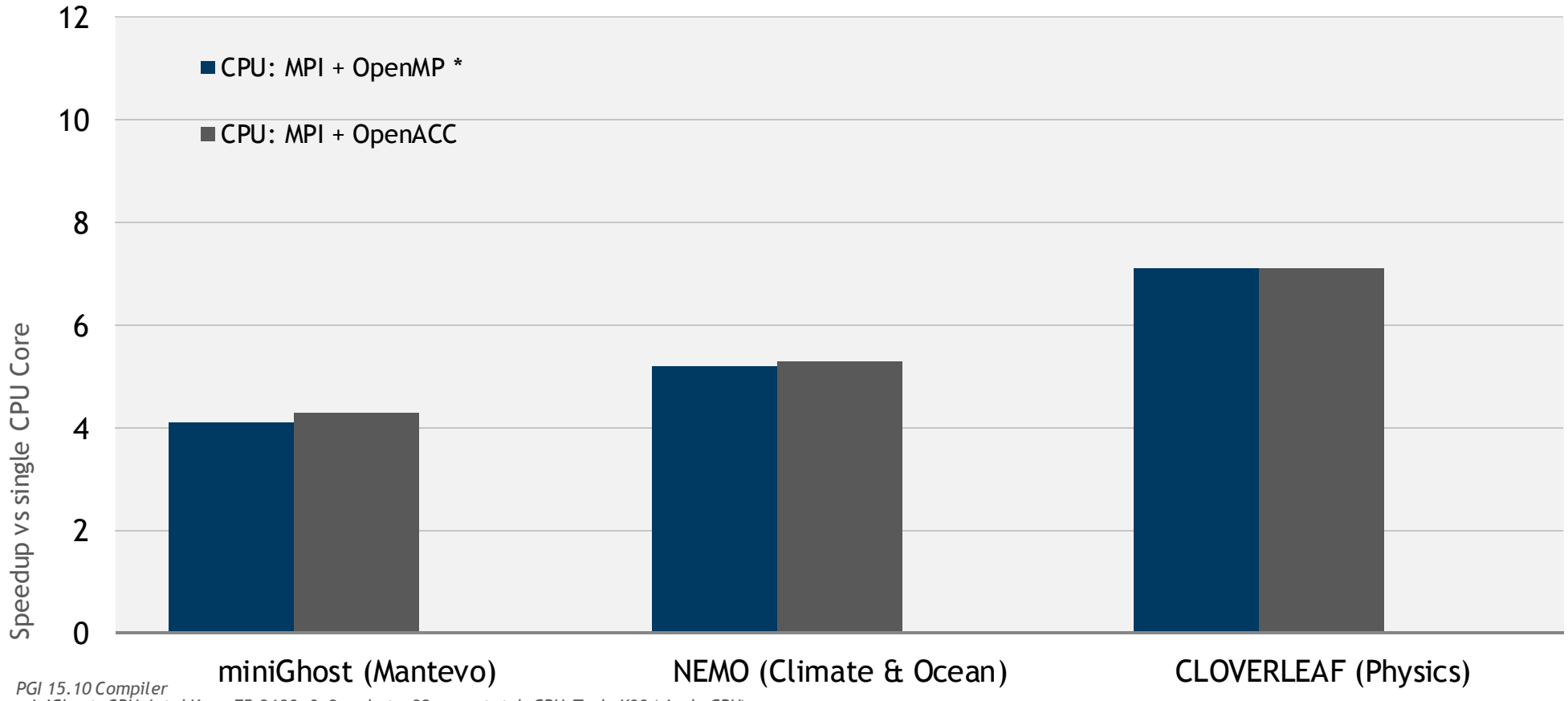
# OPENACC DELIVERS PERFORMANCE PORTABILITY




*PGI 15.10 Compiler*
*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 CPUE5-2690 v2, 20 cores total, GPU: Tesla K80 both GPUs*

** NEMO run used all-MPI*

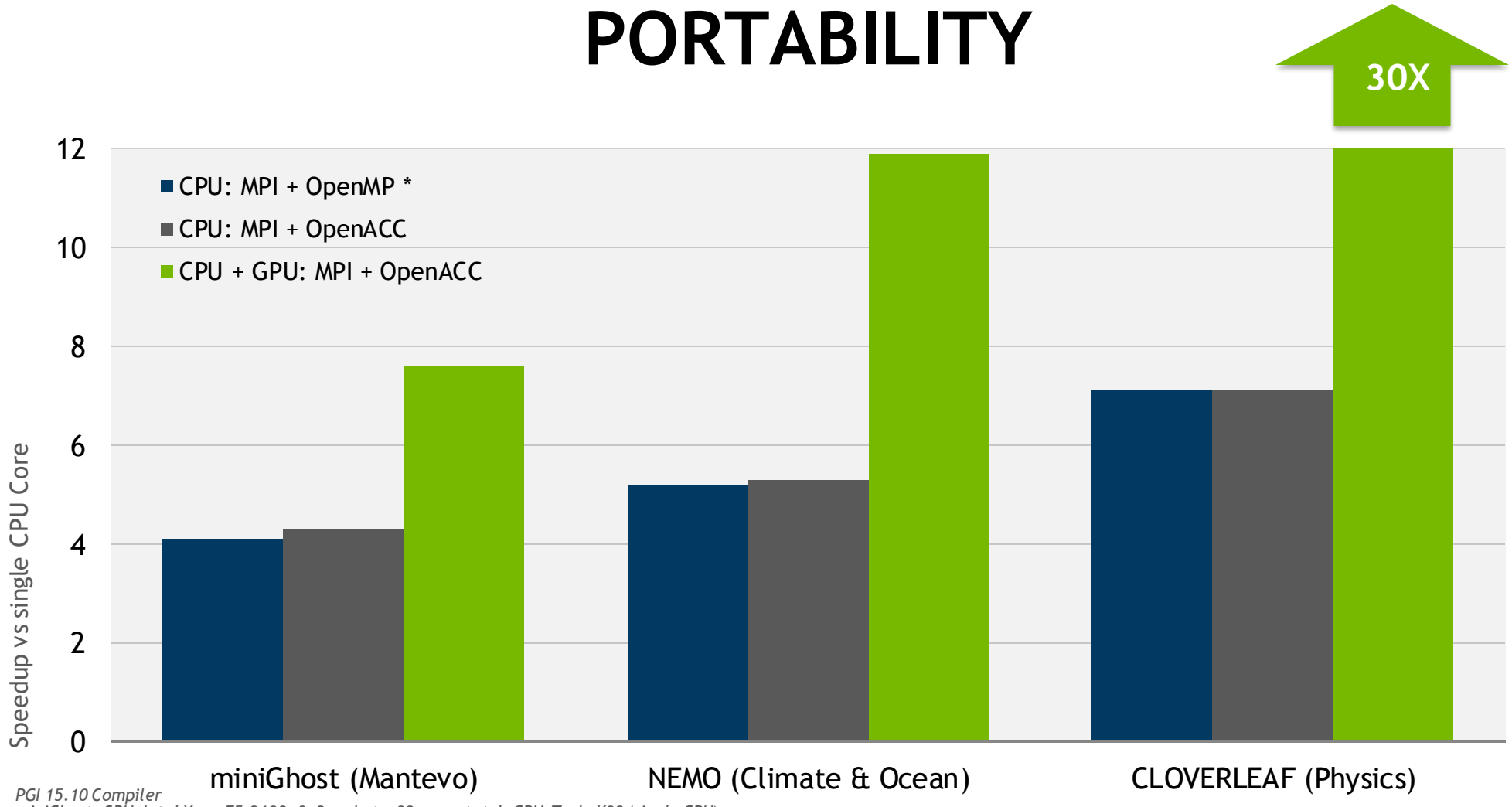
# OPENACC DELIVERS PERFORMANCE PORTABILITY




*PGI 15.10 Compiler*
*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 CPUE5-2690 v2, 20 cores total, GPU: Tesla K80 both GPUs*

** NEMO run used all-MPI*

# OPENACC DELIVERS PERFORMANCE PORTABILITY




*PGI 15.10 Compiler*
*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 CPUE5-2690 v2, 20 cores total, GPU: Tesla K80 both GPUs*

** NEMO run used all-MPI*

# WHY OPENACC?

The right programming model for performance portability.

## DESCRIPTIVE

Directives inform and enable the compiler to effectively parallelize on *any* architecture.

## INCREMENTAL

Programmer can tune the code when needed without ifdefs or loss of portability.

## MODERN

Removes scalability blockers by design to support modern processors.

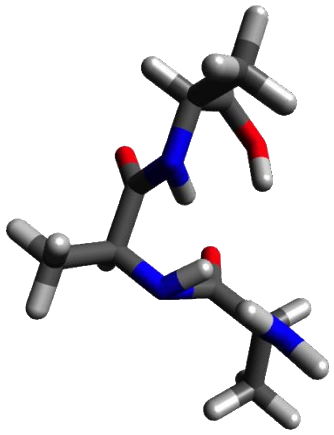
## MATURE

Multiple mature implementations on multiple platforms.

# OpenACC is enabling Scientific discoveries

# LS-DALTON

Large-scale application for calculating high-accuracy molecular energies



"*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.*"

*Janus Juul Eriksen, PhD Fellow*
*qLEAP Center for Theoretical Chemistry, Aarhus University*

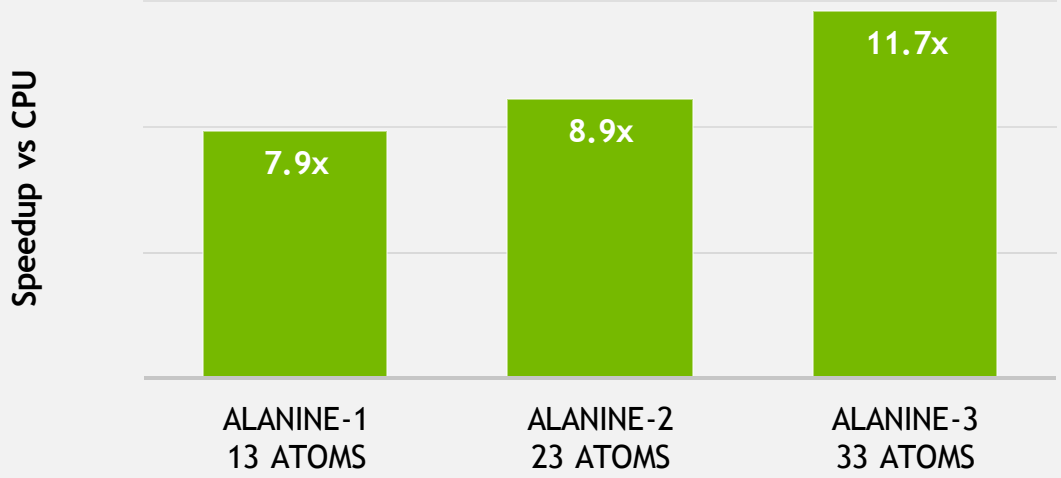
## Minimal Effort

| Lines of Code Modified | # of Weeks Required | # of Codes to Maintain |
|---|---|---|
| <100 Lines | 1 Week | 1 Source |

## Big Performance

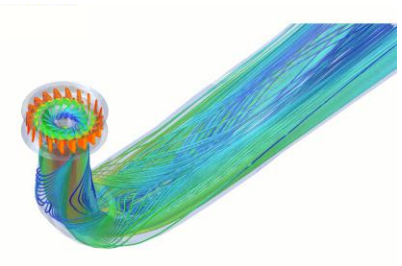
LS-DALTON CCSD(T) Module

*Benchmarked on Titan Supercomputer (AMD CPU vs Tesla K20X)*



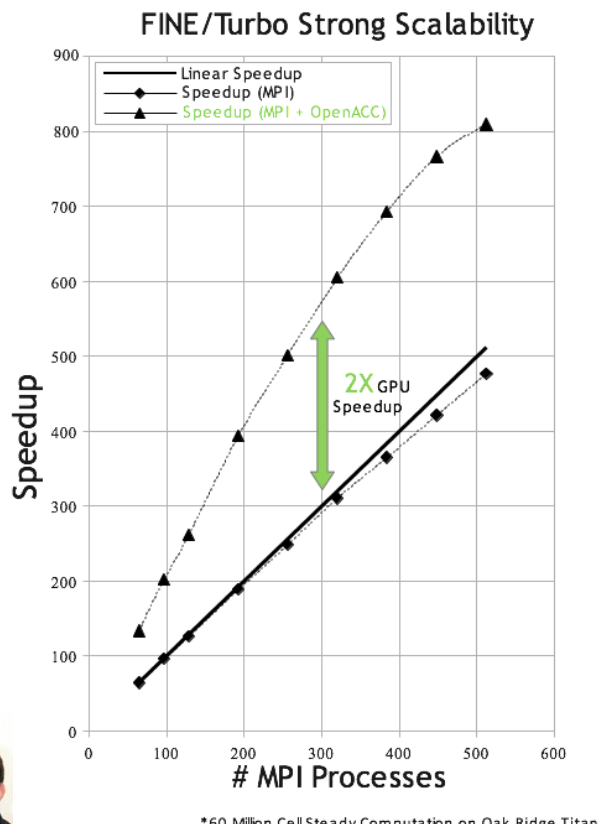
# NUMECA

## OpenACC Accelerates Commercial CFD Application



*"OpenACC enabled us to target routines for GPU acceleration without rewriting code, allowing us to maintain portability on a code that is 20-year old."*

*David Gutzwiller, Head of HPC*
*NUMECA International*




### CHALLENGE

- Accelerate 20 year old highly optimized code on GPUs

### SOLUTION

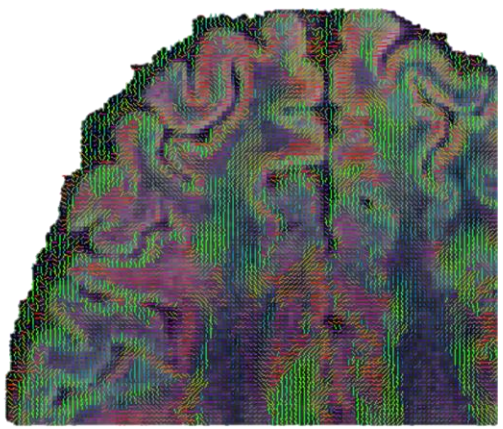
- Accelerated computationally intensive routines with OpenACC

### RESULTS

- Achieved 10x or higher speed-ups on key routines
- Full app speedup of up to 2x on the Oak Ridge Titan supercomputer
- Total time spent on optimizing various routines with OpenACC was just five person-months

# MRI RECONSTRUCTION

## OpenACC Accelerates Advanced MRI Reconstruction Model



*"Now that we've seen how easy it is to program the GPU using OpenACC and the PGI compiler, we're looking forward to translating more of our projects"*

*Brad Sutton, Associate Professor of Bioengineering and Technical Director of the Biomedical Imaging Center University of Illinois at Urbana-Champaign*

### CHALLENGE

- Produce detailed and accurate brain images by applying computationally intensive algorithms to MRI data
- Within short reconstruction time for diagnostic use

### SOLUTION

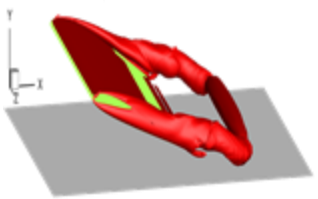
- Accelerated MRI reconstruction application with OpenACC using NVIDIA GPUs

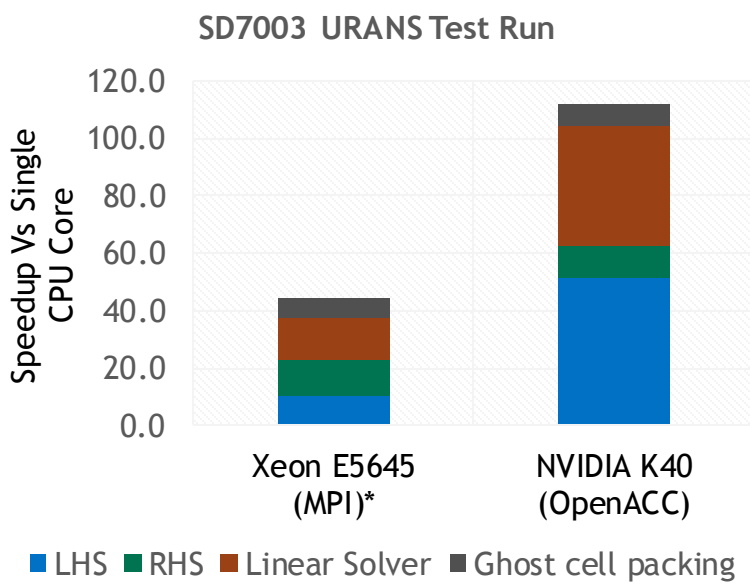
### RESULTS

- Reduced reconstruction time for a single high-resolution MRI scan from 40 days to a couple of hours
- Scaled on Blue Waters at NCSA to reconstruct 3000 images in under 24 hours

# INCOMP3D

## 3D incompressible Navier-Stokes fully implicit CFD solver






## CHALLENGE

- Accelerate a complex memory bounded implicit CFD solver on GPU

## SOLUTION

- Accelerated mathematical formulations one loop at a time with OpenACC

## RESULTS

- Achieved average speedups of 3X on Tesla GPU over parallel MPI implementation on x86 multicore processor
- Structured approach to parallelism provided by OpenACC allowed better algorithm design without worrying about GPU internals

*"OpenACC is a highly effective tool for programming fully implicit CFD solvers on GPU to achieve 3X speedup"*

*Lixiang Luo, Researcher*
*Aerospace Engineering Computational Fluid Dynamics Laboratory*
*North Carolina State University (NCSU)*

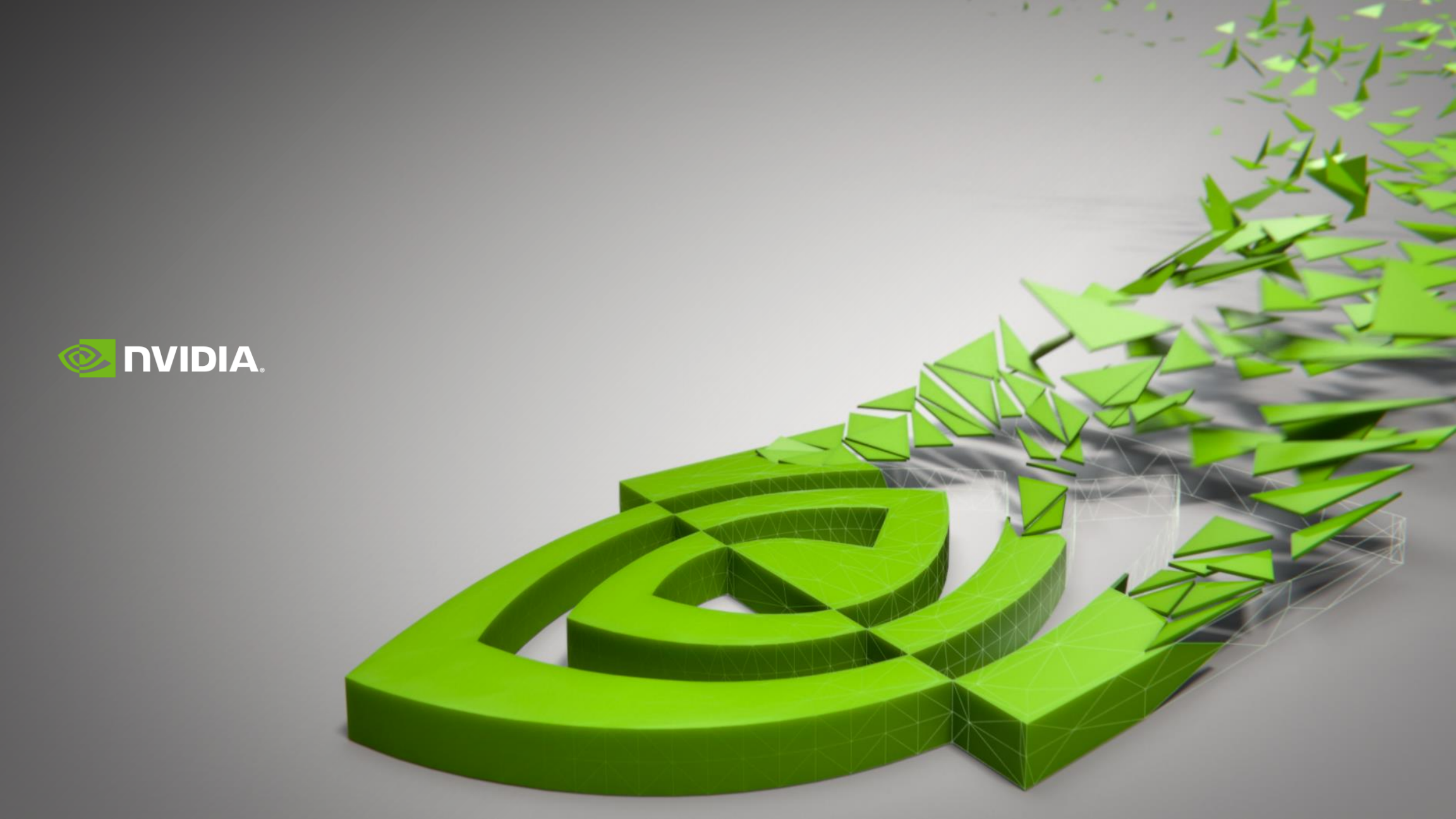
* CPU Speedup on 6 cores of Xeon E5645, with additional cores performance reduces due to partitioning and MPI overheads

Can a single programming model provide both functional and performance portability?

# TWO TYPES OF PORTABILITY

| FUNCTIONAL PORTABILITY | PERFORMANCE PORTABILITY |
| --- | --- |
| The ability for a single code to run anywhere. | The ability for a single code to run well anywhere. |

Many programming models provide functional portability, *but OpenACC can provide performance portability too!*


NVIDIA