VirginiaTech
College of Engineering

Aerospace & Ocean Engineering at Virginia Tech

GPU Acceleration of the SENSEI CFD Code Suite

Chris Roy, Brent Pickering, Chip Jackson, Joe Derlaga, Xiao Xu
Aerospace and Ocean Engineering

Primary Collaborators:
Tom Scogland, Wu Feng (Computer Science)
Kasia Swirydowicz, Arielle Grim-McNally, Eric de Sturler (Math)
Ross Glandon, Paul Tranquilli, Adrian Sandu (Computer Science)

Virginia Tech, Blacksburg, VA, 24061

SENSEI CFD Code

SENSEI: Structured Euler/Navier-Stokes Explicit Implicit

- SENSEI: solves Euler and Navier-Stokes equations
  - 2D/3D multi-block structured grids
  - Finite volume discretization
  - Currently for steady-state problems
    - Explicit: Euler explicit and multi-stage Runge-Kutta
    - Implicit: Euler implicit with linear solves via basic GMRES
  - Written in *modern* Fortran 03/08 and employs object-oriented features such as derived types and procedure pointers
  - Currently uses OpenMP and MPI for parallelism
  - Employs Array-of-Struct data structures (problems for GPUs…)

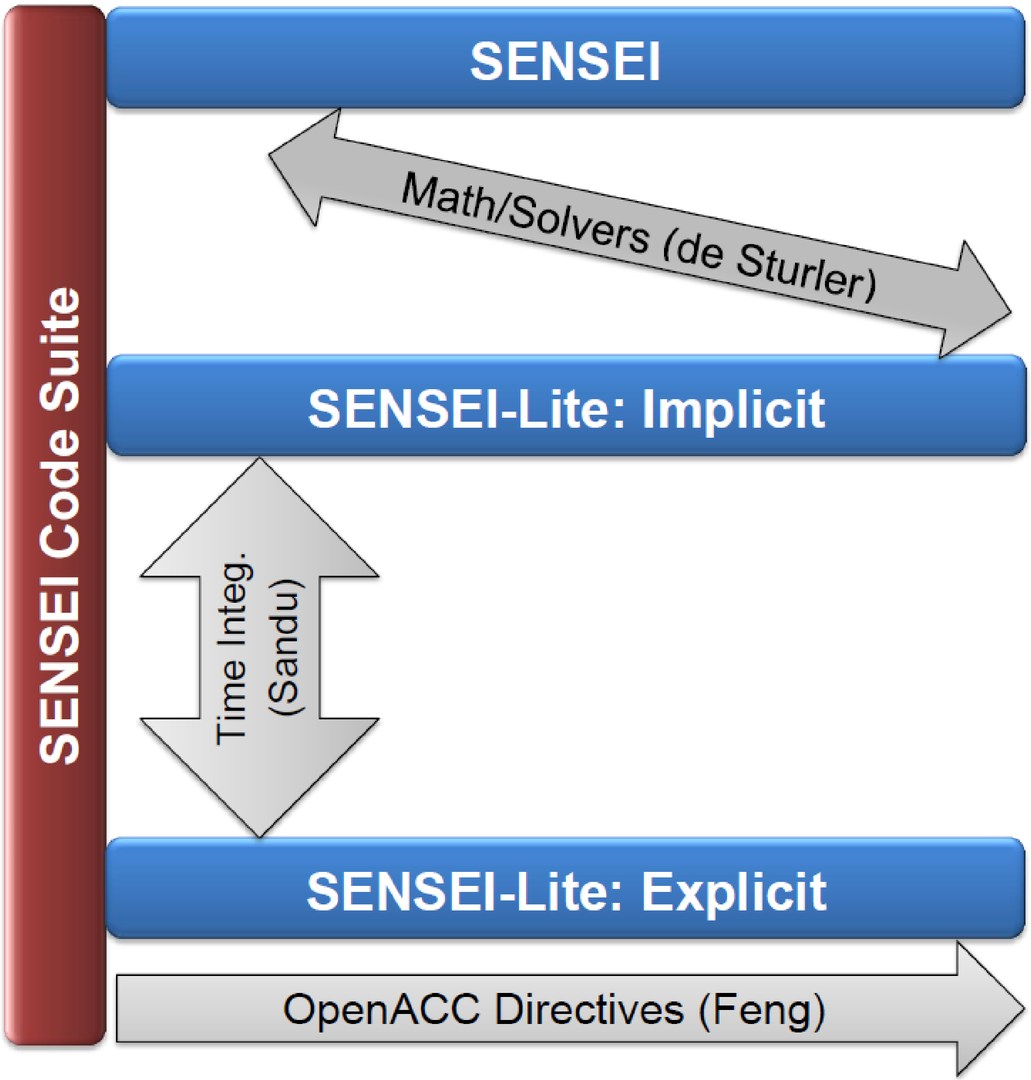
- SENSEI leverages another grant on error estimation and control for CFD (in AFOSR Computational Math)

SENSEI-Lite CFD Code

Preliminary testing of directive-based parallelism employs simplified version: SENSEI-Lite

- 2D Navier-Stokes equations
- Finite difference discretization on Cartesian grids
- Employs same artificial compressibility approach as SENSEI for extending compressible CFD code to incompressible flows
- Avoids Array-of-Struct data structures

Collaborations with SENSEI Code Suite



Joe Derlaga
(PhD Student)

Xiao Xu
(AOE/Math Postdoc)

Brent Pickering
(PhD Student)

Virginia Tech College of Engineering

AOE Aerospace & Ocean Engineering at Virginia Tech

Collaborations w/ Math

Collaborations w/ Eric de Sturler (Math) and his group for expertise in parallel implicit solvers

- SENSEI only has basic GMRES with very limited preconditioners
- Work has begun to incorporate advanced parallel solvers and preconditioners into SENSEI
- Work is near completion for serial and parallel solvers and preconditioners in SENSEI-Lite

Collaborations w/ CS (#1)

Collaborations w/ Adrian Sandu (CS) and his group for expertise in implicit/explicit time integrators for unsteady flows

- MAV application requires large variations in cell size from freestream to near-wall boundary layer
- Explicit schemes tend to be more naturally parallel, but suffer from severe stability limits for small cells
- Initial collaborations begun using SENSEI-Lite

Collaborations w/ CS (#2)

Collaborations w/ Wu Feng (CS) and his group for expertise in OpenACC directives and hardware / software optimization

- Initial work has focused on explicit version of SENSEI-Lite
- Tight collaborations with Wu Feng and Tom Scogland resulted in AIAA Paper at SciTech (journal submission soon)
- Our collaborations have establish potential paths forward for GPU parallelization of full SENSEI

Introduction

Directive Based Programming provides an alternative to platform specific languages such as CUDA C.

- Directives or pragmas (in Fortran or C/C++) permit re-use of existing source codes—*ideally with no modifications*.
  - Maintain a single cross-platform code base.
  - Easily incorporate legacy code

- Shifts re-factoring burden to compiler.
  - Compiler vendors can extend functionality to new architectures.
  - Source code expresses algorithm, not implementation.

- Directive based APIs are already familiar to many computational scientists (OpenMP).

Efficient GPU Data Storage

- SOA preferred over AOS on GPU.
- Permits contiguous access on GPUs and SIMD hardware.
- Avoids need for *scatter-gather*.

Array of Struct (AOS)

Pressure Node 1	U-velocity Node 1	V-velocity Node 1	Pressure Node 2	U-velocity Node 2	V-velocity Node 2	Pressure Node 3	U-velocity Node 3	V-velocity Node 3

Struct of Array (SOA)

Pressure Node 1	Pressure Node 2	Pressure Node 3	U-velocity Node 1	U-velocity Node 2	U-velocity Node 3	V-velocity Node 1	V-velocity Node 2	V-velocity Node 3

Layouts in linear memory for a sequence of 3 grid nodes (3-DOF per node). Red nodes represent memory access for a three-point stencil.

Virginia Tech College of Engineering

CFD Code: SENSEI-Lite

Aerospace & Ocean Engineering at Virginia Tech

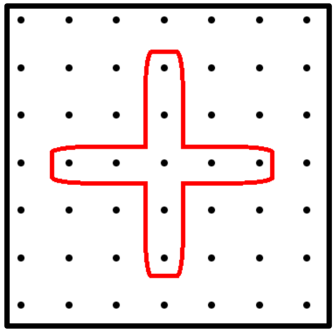
- Solves 2D incompressible Navier Stokes using the artificial compressibility method (Chorin).
  - Nonlinear system with 3 equations (cons. of mass, x+y momentum).
  - Fourth derivative damping (artificial viscosity).
  - Steady state formulation.

$$\frac{1}{\beta^2}\frac{\partial p}{\partial t}+\frac{\partial u_j}{\partial x_j}-\lambda_j \Delta x_j C_j \frac{\partial^4 p}{\partial x_j^4}=0 \qquad \text{(Cons. of mass)}$$

$$\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}+\frac{1}{\rho}\frac{\partial p}{\partial x_i}-\nu\frac{\partial^2 u_i}{\partial x_j^2}=0 \qquad \text{(Momentum)}$$

- Compressibility term β is calculated from local flow characteristics.

CFD Code: SENSEI-Lite

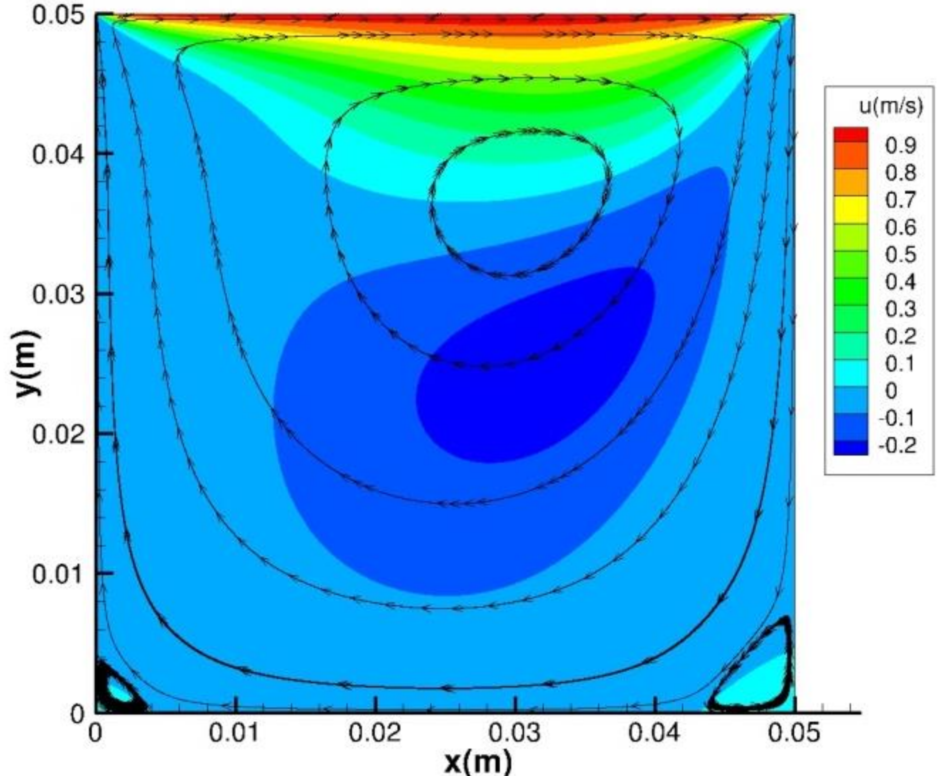


- Structured grid FDM with second order accurate spatial discretization.
  - 3 D.O.F. per node.
  - 5-point and 9-point stencils require 19 data loads per node (152B double precision, 76B with single precision)
  - 130 FLOPS per node, including 3 FP divide and 2 full-precision square roots.
- *Explicit* time integration (forward Euler).
  - Keeps the performance focus on the stencil operations—not linear solvers.
  - Data parallel algorithm stores two solution copies (time step n and n+1).
- Verified code using Method of Manufactured Solutions.
- Relatively simple code → Easy to modify and experiment with.

CFD Code: SENSEI-Lite

Benchmark case: 2D lid driven cavity flow.

- All cases used fixed-size square domain, with lid velocity $1\ m/s$, Re 100, and density constant $1\ kg/m^3$.
- Uniform Cartesian computational grids were used, ranging in size from 128×128 to 8192×8192 nodes. (Max size: 200 million DOF)

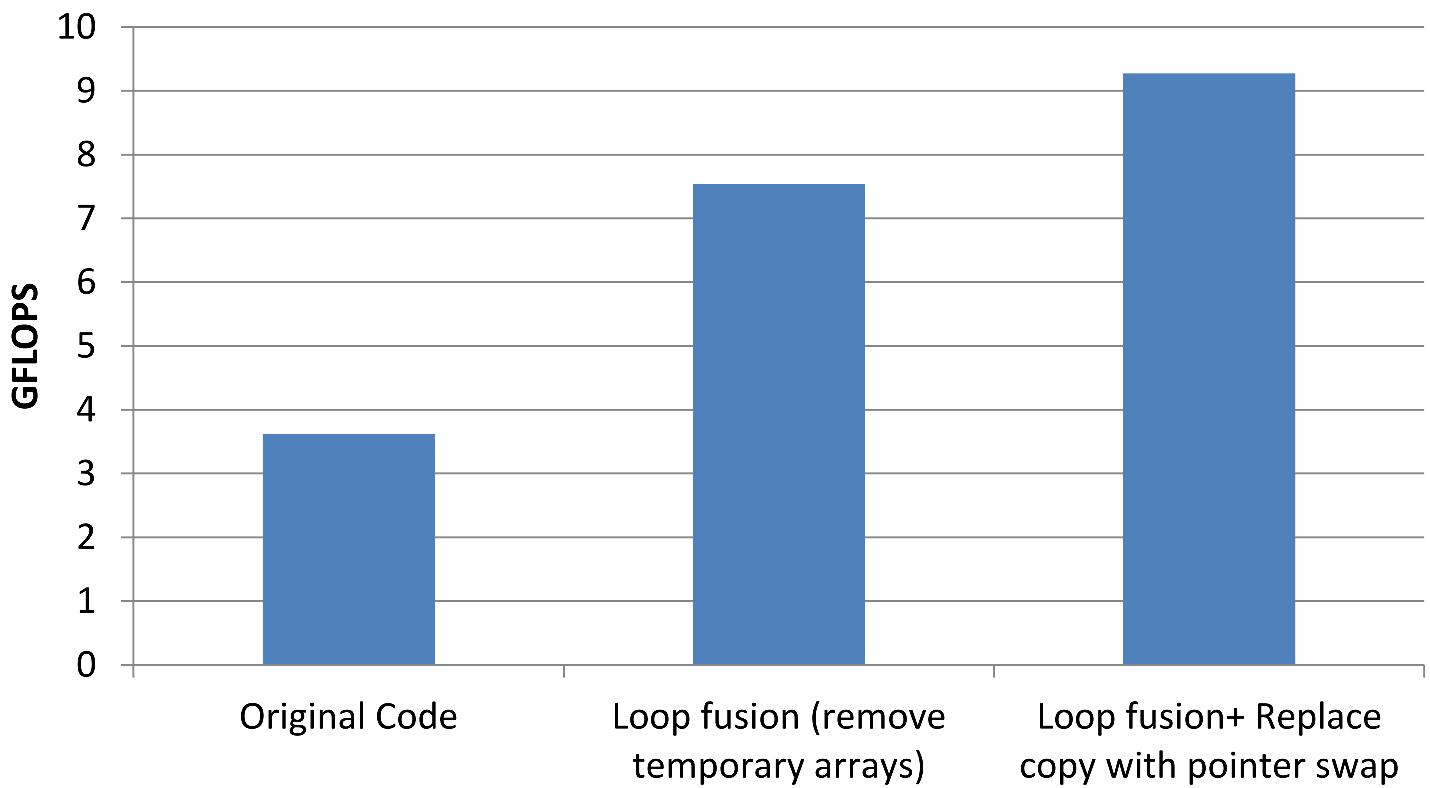


VirginiaTech College of Engineering

Preliminary Code Modifications

Aerospace & Ocean Engineering at Virginia Tech

- Reduce memory traffic and eliminate *temporary arrays*.
- Fuse loops having only *local dependencies* (e.g., artificial viscosity and residual).
- Eliminate unnecessary mem-copy.



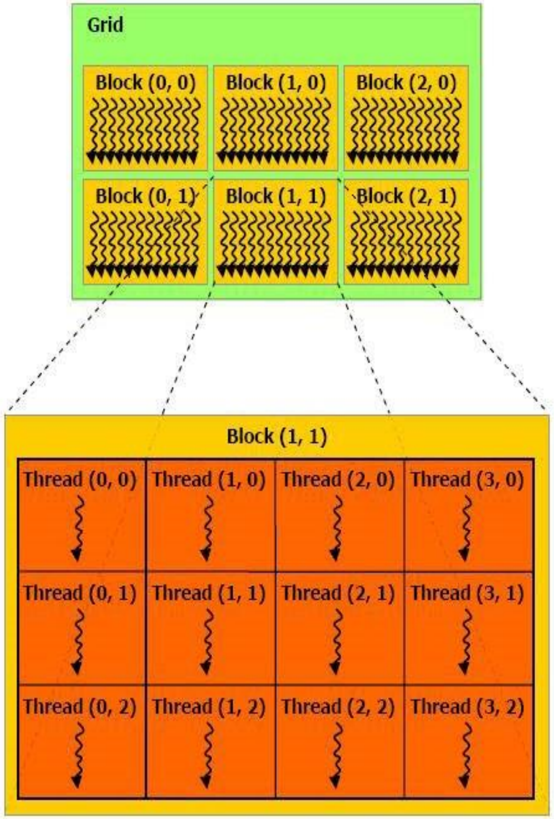
OpenACC Performance

- Evaluated on three models of NVIDIA GPU representing two microarchitectures (Fermi, Kepler).
- Compiler = PGI 13.6 and 13.10.
- For NVIDIA case, compiler automatically generated binaries for *Compute Capability 1.x, 2.x, and 3.x* devices.
- Experimented with manual optimization of the OpenACC code.

OpenACC Performance

Manual Performance Tuning

- Compiler defaulted to 64 × 4 2D thread-block configuration, but this could be overridden using the *vector* clause.



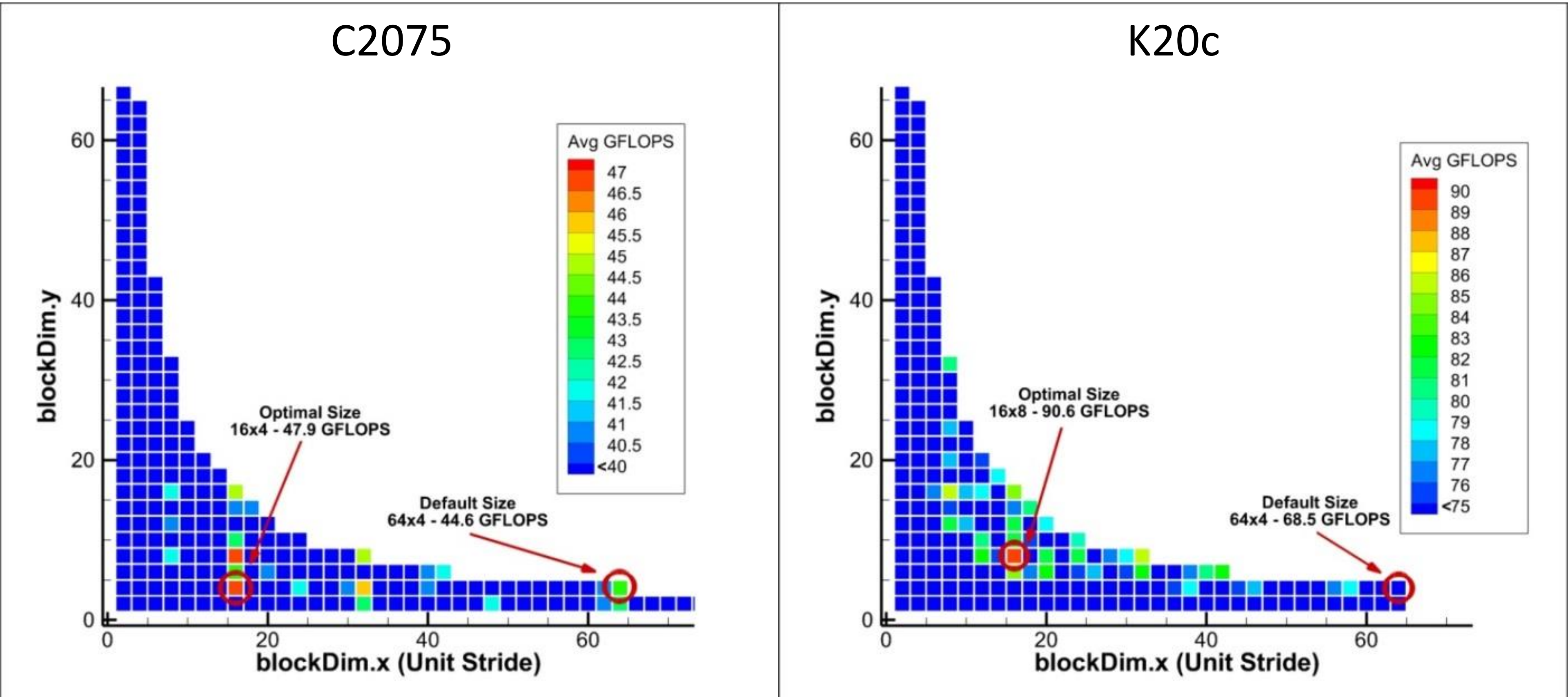
- On CUDA devices, launch configuration can have a significant effect on performance.
  - Utilization of shared memory and registers.
  - Can lead to variations in occupancy.
  - *Shape* of the thread blocks can affect the layout of the shared data structures → possible bank conflicts.

- Used a fixed size problem of 4097x4097 nodes (50 million DOF), and explored full parameter space of thread-block dimensions.

OpenACC Performance

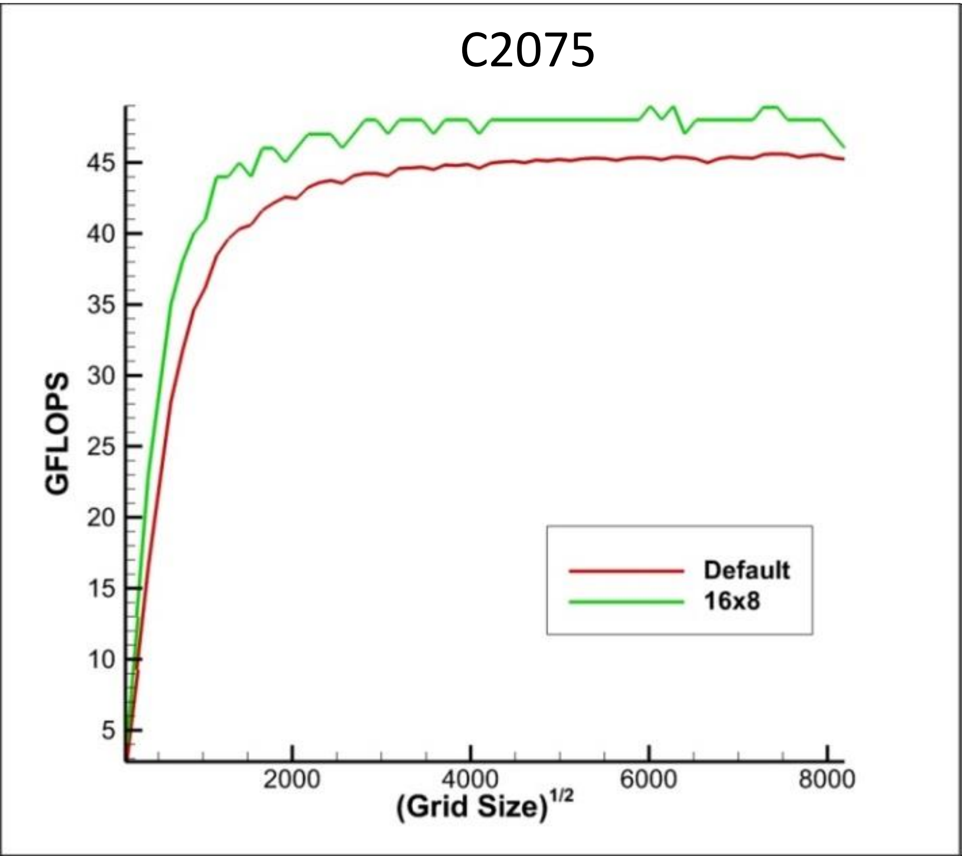
Manual Performance Tuning

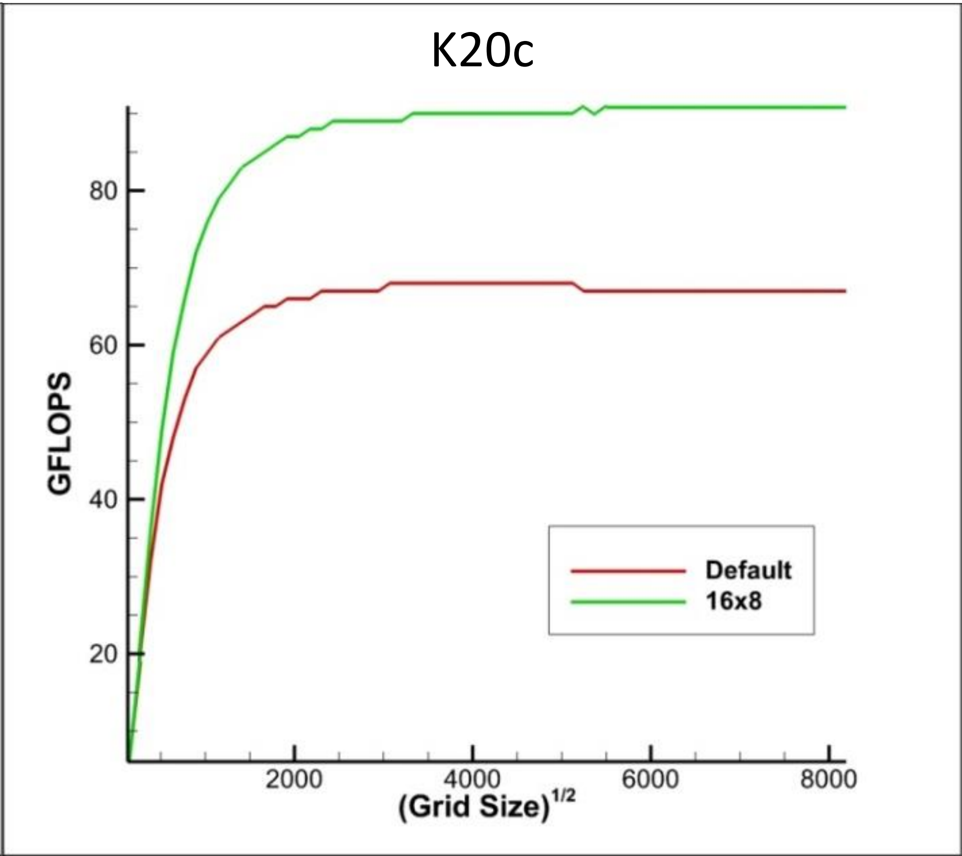
(Double Precision, 4097x4097 cavity)



OpenACC Performance

- Default and optimized thread-block dimensions—PGI 13.6
- Double Precision





OpenACC Performance

Single vs Double Precision

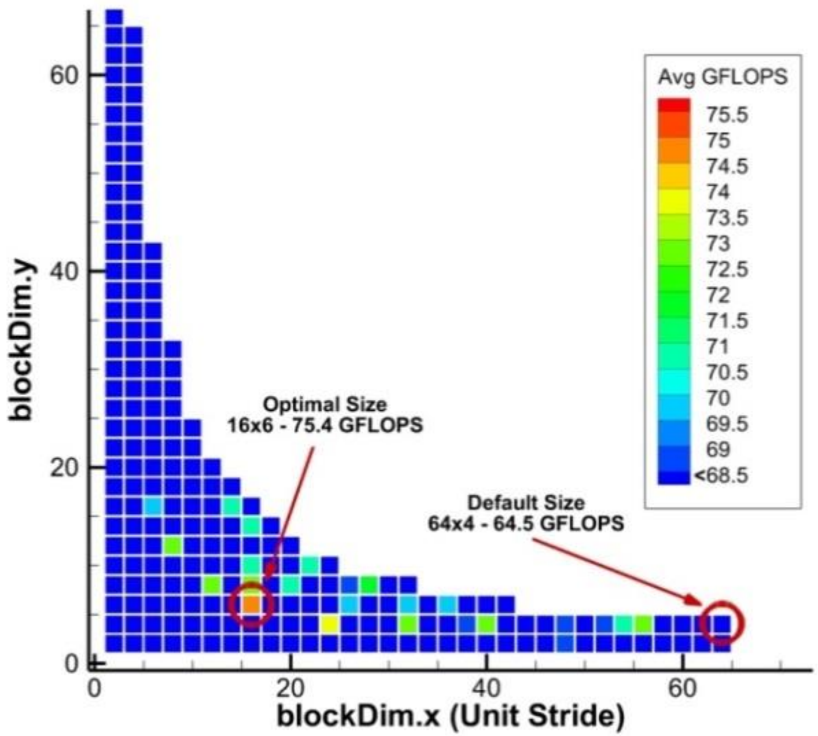
- Single prec. maximum throughput 2-3x greater than double prec.
- 32-bit data types require half the memory bandwidth, half the register file and half the shared memory space of 64-bit types.
- Trivial to switch in Fortran by editing the *precision* parameter used for the *real* data type.
  - Like SENSEI, INS code uses ISO C binding module→ precision was always *c_double or c_float.*
- For the *explicit* INS code, single precision was adequate for converged solution.
  - Lost about 7 significant digits, equivalent to round-off error.
  - May be more of a problem with implicit algorithms.

OpenACC Performance

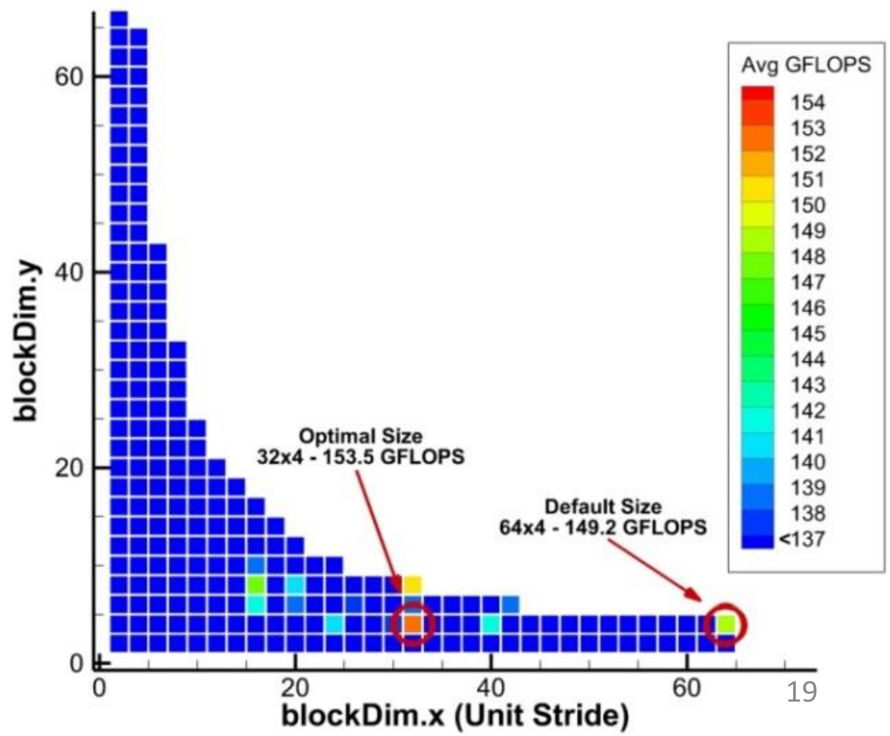
Manual Performance Tuning

(Single Precision, 4097x4097 cavity)

C2075



K20c



OpenACC Performance

Summary

Tesla C2075 (Fermi)	Default Block Size (GFLOPS)	Optimal Block Size (GFLOPS)	Speedup (%)
Double Precision (GFLOPS)	44.6	47.9	7.4%
Single Precision (GFLOPS)	64.5	75.4	16.9%
Speedup (%)	44.6%	57.4%	

Tesla K20c (Kepler)	Default Block Size (GFLOPS)	Optimal Block Size (GFLOPS)	Speedup (%)
Double Precision (GFLOPS)	68.5	90.6	32.3%
Single Precision (GFLOPS)	149.2	153.5	2.9%
Speedup (%)	117.8%	69.4%	

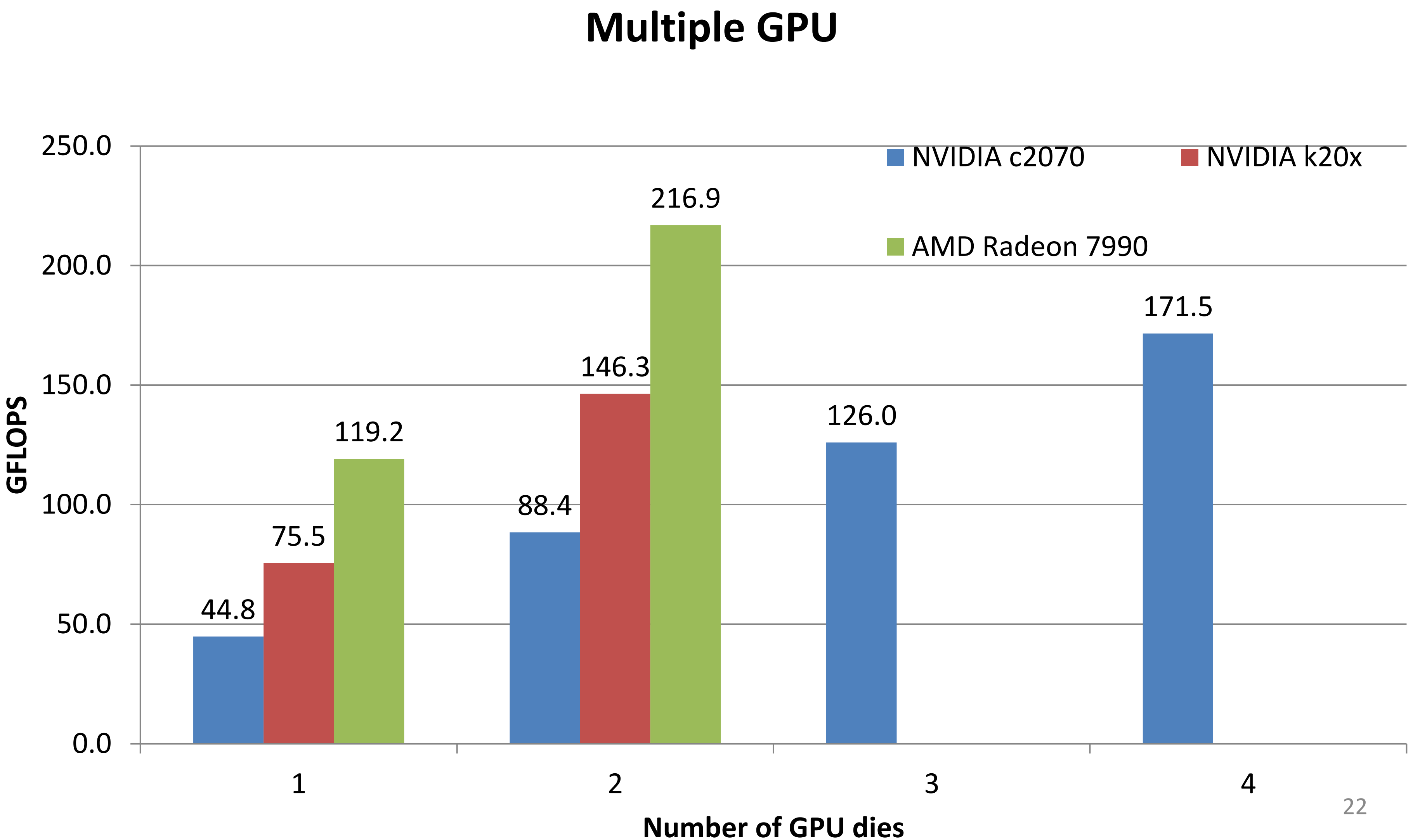
Optimal Thread-block Dimension	Double Precision	Single Precision
C2075 (Fermi)	16x4 (16x8)	16x6
K20c (Kepler)	16x8	32x4

OpenACC Performance

Multiple GPU

- PGI 13.10 *beta*.
- Used domain decomposition. OpenMP parallel region with one GPU per thread.
- Boundary values are transferred between regions on each iteration.
  - Incurs thread-synch overhead and requires data transfer across PCIe.
  - OpenACC does not expose mechanism for direct GPU to GPU copy.
- 13.10 permits use of AMD GPUs, such as the HD 7990.
  - 7990 is equivalent to two 7970 “South Islands” dies on a single card.
  - Requires multi-GPU code to make use of both simultaneously.

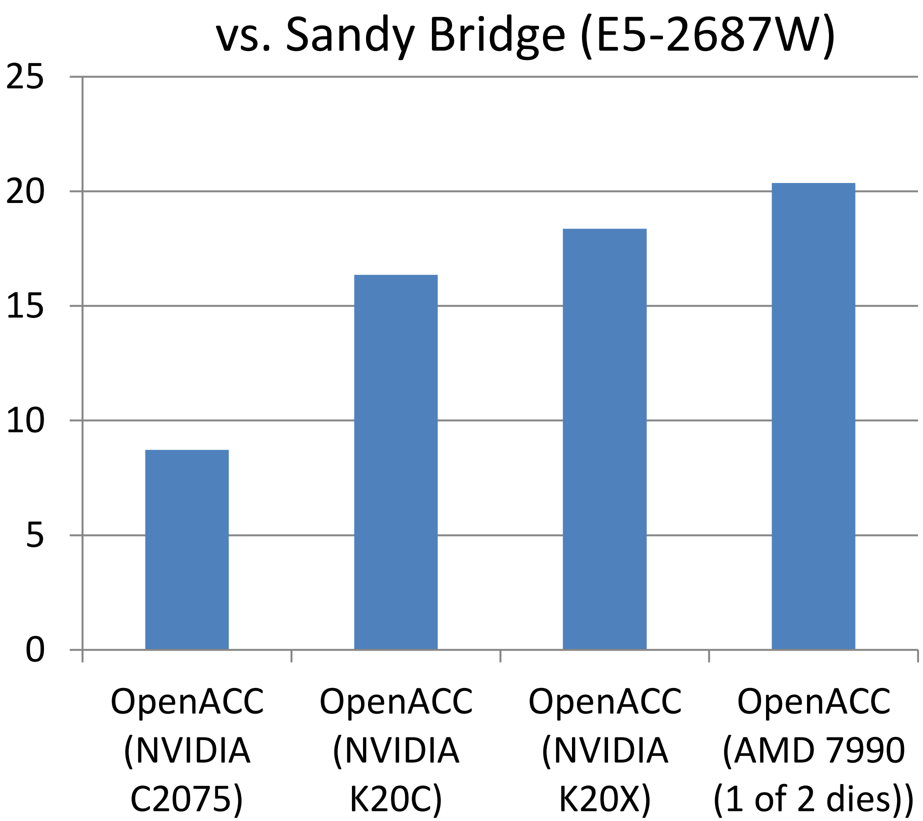
OpenACC Performance

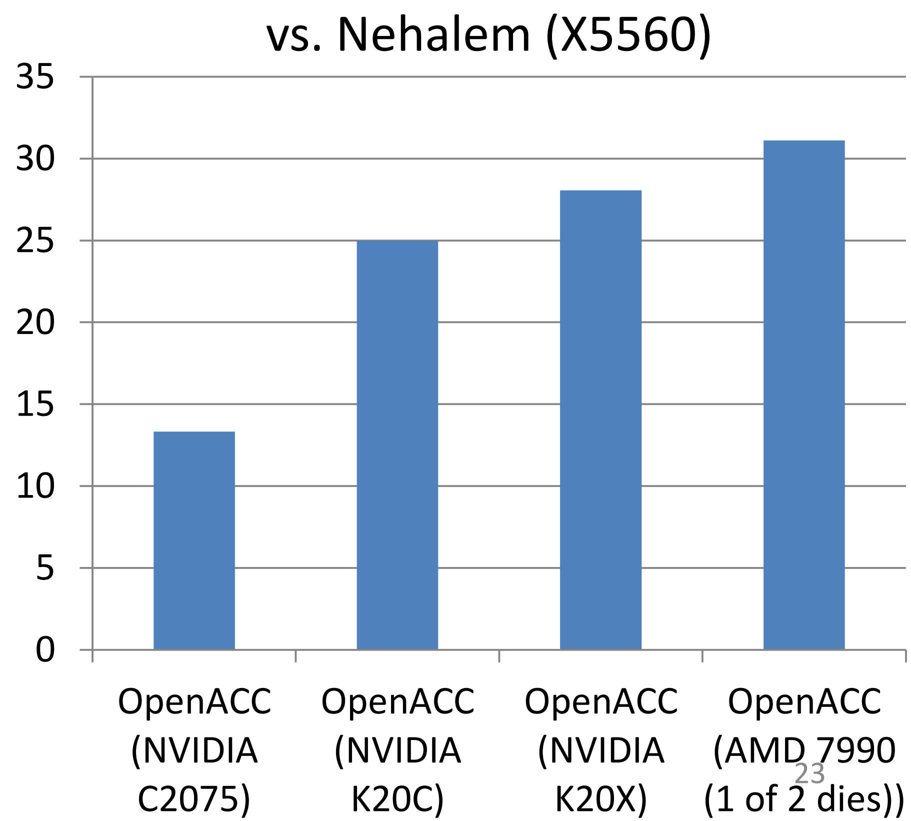


OpenACC Performance

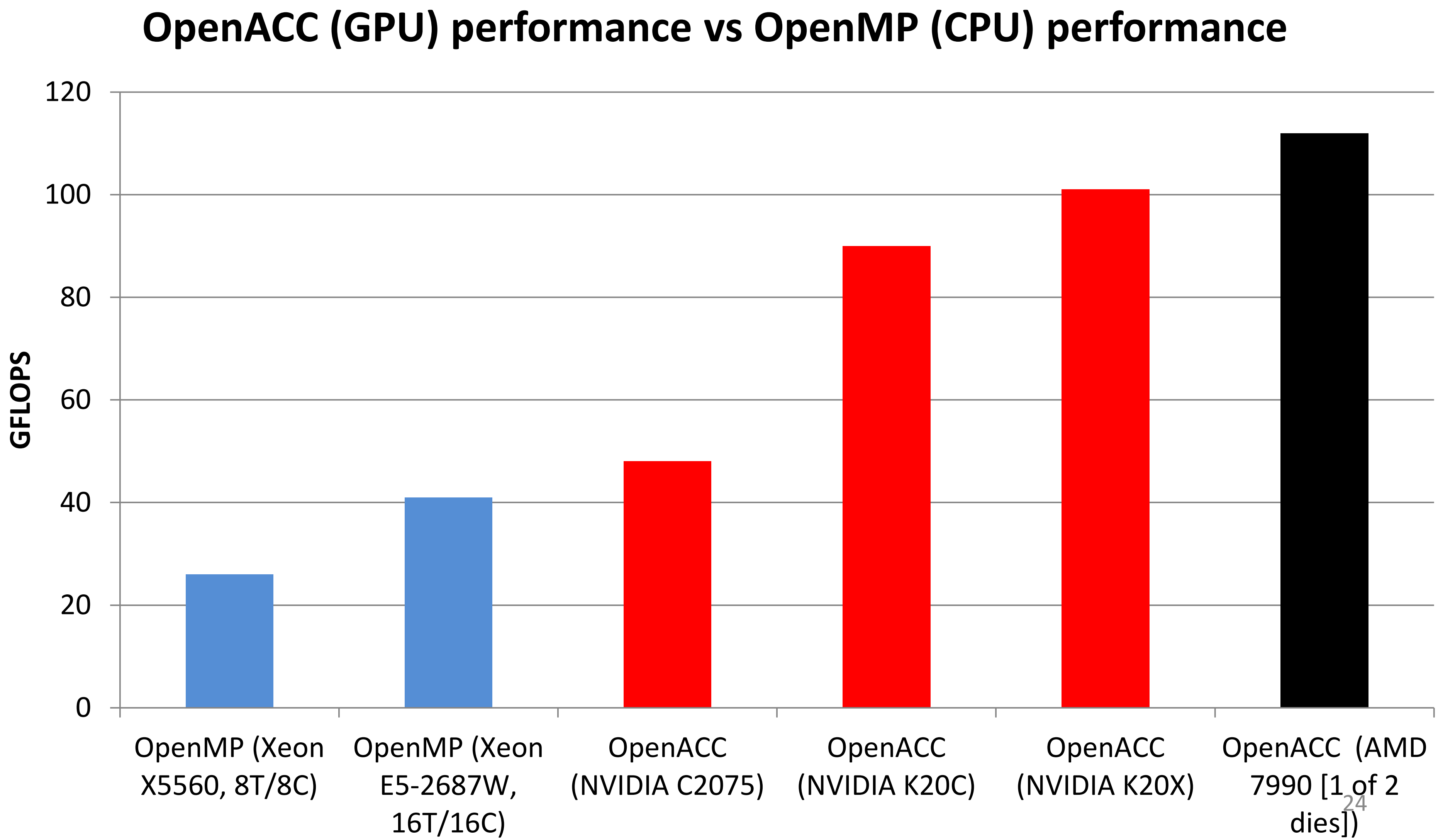
OpenACC speedup over single CPU thread

CPU version compiled from the *same source code* using PGI 13.6.



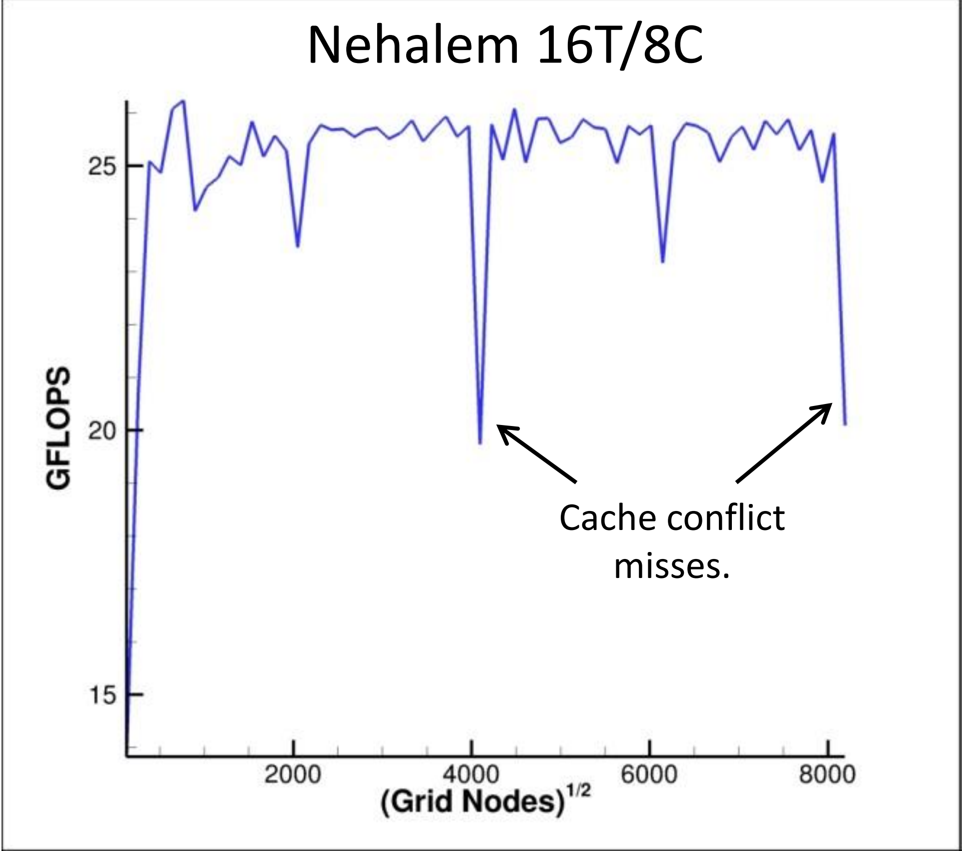


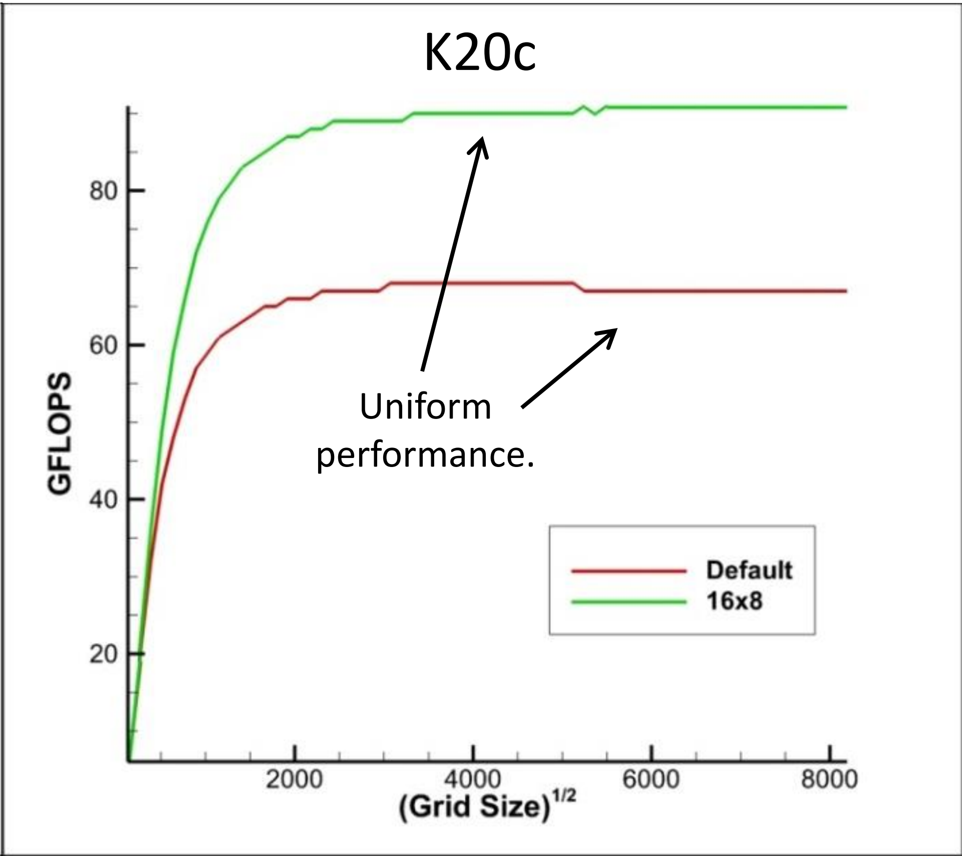
OpenACC Performance



OpenACC Performance

Advantage of *software managed* cache on GPU for computation on structured grids → avoids conflict misses.





Summary and Path Forward

- OpenACC provided good performance on GPU, and in conjunction with OpenMP directives permitted a *single code base* to serve CPU and GPU platforms → robust to rapid changes is hardware platforms. *However…*

- Decided **not** to apply the OpenACC 1.0 API to SENSEI.
  - Requires too many alterations to current code base.
  - Restricts use of object-oriented features in modern Fortran.

- Looking to other emerging standards that also support GPUs and other accelerators: OpenACC 2.0 and OpenMP 4.0.

VirginiaTech
College of Engineering

Publications

AEROSPACE & OCEAN ENGINEERING
AT VIRGINIA TECH

- J. M. Derlaga, T. S. Phillips, and C. J. Roy, "SENSEI Computational Fluid Dynamics Code: A Case Study in Modern Fortran Software Development," AIAA Paper 2013-2450, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 24-27, 2013.
- B. P. Pickering, C. W. Jackson, T. R. W. Scogland, W.-C. Feng, and C. J. Roy, "Directive-Based GPU Programming for Computational Fluid Dynamics," AIAA Paper 2014-1131, 52nd Aerospace Sciences Meeting, National Harbor, MD, January 13-17, 2014.

➢ Both are in preparation for journal submission