PGI®

SUCCESS STORY

# Researchers at North Carolina State Use OpenACC to Run a Fully Implicit 3D CFD Solver on a GPU

OpenACC
Directives for Accelerators

## CHALLENGE

Researchers from the Aerospace Engineering Computational Fluid Dynamics Laboratory in NCSU wanted to successfully port a 3D incompressible Navier-Stokes solver to a GPGPU.

## SOLUTION

Using OpenACC, the team successfully ported the implicit solver on the GPGPU. OpenACC enabled the translation of mathematical formulations with minimal coding.

## RESULTS

- Coding a simplified 2D version of INCOMP3D using OpenACC took just five person-days
- Speedups of 12 times were achieved with OpenACC for the full 3D version of INCOMP3D.

## Background

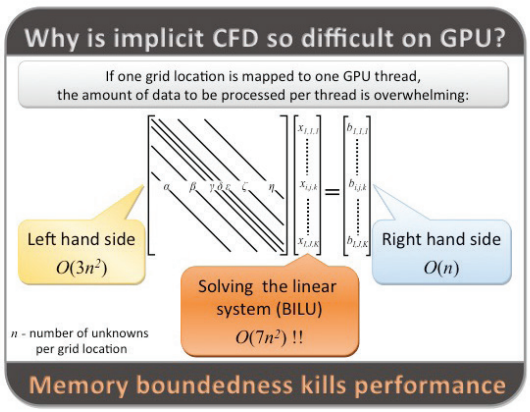
The Aerospace Engineering Computational Fluid Dynamics Laboratory (AECFDL) at North Carolina State University (NCSU), led by Dr. Jack R. Edwards, conducts research in computational fluid dynamics, with an emphasis on the development of advanced physical models and algorithms suitable for conducting large-scale simulations of diverse fluid phenomena, such as modeling transonic flow over a Boeing 747.

Recently, it conducted a study in GPGPU-based computation technology by attempting to port a 3D incompressible Navier-Stokes solver called INCOMP3D using OpenACC, with an aim of high maintainability and portability. The original CPU code is based on an existing solver validated for a range of model problems.

The porting of INCOMP3D is a part of a larger multidisciplinary co-design research effort led by Dr. Feng at Virginia Tech and seven other professors, including Dr. Edwards. Four Computational Fluid Dynamics (CFD) codes (SENSEI and GenIDLEST at Virginia Tech and INCOMP3D and RDGFLO at NCSU) were studied as models for porting legacy codes to GPGPU. The research on GPGPU computation seeks to overcome fundamental restrictions, including the difficulty of creating a good linear solver on the GPGPU, and the fact that naïve algorithms for Jacobian matrix filling are very inefficient on GPGPU.

## Challenge

Lixiang Luo, a researcher at the Aerospace Engineering Computational Fluid Dynamics Laboratory in NCSU, was trained as a mechanical engineer with a focus on numerical methods for fluid dynamics and control systems, and had a sound background on MPI and OpenMP before starting the project, but his knowledge of GPU programming was very limited. "Given our skillset, OpenACC was a good choice for the job," he said. "However, the team was unsure that the time and effort invested on developing an OpenACC version of the code would enable further space for optimization."



## Solution

Luo said OpenACC played a critical role in the development of the linear solver employed in INCOMP3D as well in the full INCOMP3D verison, by enabling him to translate mathematical formulations with a few simple statements and without worrying about GPU implementation details. "It was extremely helpful not having to worry about the GPU implementation details in the algorithm design stage, when a lot of mathematical experiments must be carried out," he said. "Additionally, OpenACC forces me to write algorithms in well-defined steps, which helps to avoid parallel programming pitfalls often found in CUDA codes."

Using OpenACC, Luo's team was able to port the program one loop at a time, which proved to be a huge time-saver. It also reduces the chance of hard-to-locate bugs, because every step could be verified individually. Additionally, using OpenACC enabled better collaboration throughout the project. Because OpenACC allows the original code to retain most of its modular structures, the computation kernels remain in their context, greatly improving readability. Engineers without any GPU programming experience can modify the ported code as needed, without much intervention from the programmer.

"Over the course of conducting INCOMP3D project, I quickly picked up OpenACC," he said. Luo also learned CUDA Fortran and CUDA C, which he used during the second phase of his research (see Sidebar on the next page).

"OpenACC is a highly effective tool for programming fully implicit CFD solvers on GPU to achieve true 12X speedup."

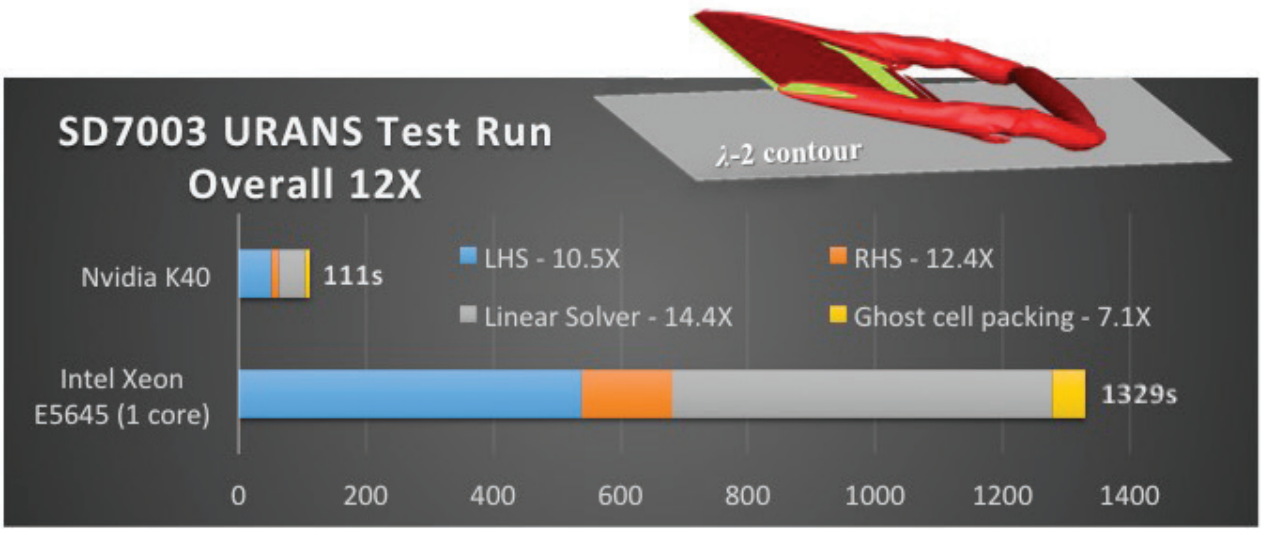
Lixiang Luo, Researcher at the Aerospace Engineering Computational Fluid Dynamics Laboratory in North Carolina State University (NCSU)

## Results

Using OpenACC, Luo and his team created a simplified 2D version of INCOMP3D in just five person-days, greatly reducing the research effort during the development of the full 3D version.They compared run times on an Intel Xeon E5645 CPU core with run times on the NVIDIA M2050 GPU. Speedup numbers were obtained from Virginia Tech's Hokiespeed cluster, which consisted of the following:

- 2x Intel Xeon 5645@2.4G
- 24GB main memory
- 2x NVIDIAM2050
- PGI v13.8
- CUDA-aware OpenMPI v1.8.3 and MVAPICH2 v1.8

Overall speedups achieved on the GPU were 12X versus one CPU core. Because all relevant data remains on GPU throughout the main computation loop, data transfer times are negligible.



"OpenACC shines in scientific computing," said Luo. "If the same attempts were made with CUDA or OpenCL, coding efforts would be multiplied by several times, not to mention that the development can be much more error-prone and further delayed."

Equipped with a highly-optimized BILU-based linear solver, INCOMP3D is arguably the first fully implicit 3D CFD solver to achieve such significant speedups on a GPU.

"Our research proves that, contrary to what many in the CFD community believe, fully implicit CFD solvers are indeed able to run very well on the GPGPU, if and only if the proper fine-grained algorithms are implemented," said Luo. "We plan to generalize some of these algorithms for a fully implicit scheme, in the hope that they can serve as the building blocks for porting more production-level CFD codes on to the promising GPGPU architecture."

Luo added, "OpenACC is a highly effective tool for programming fully implicit CFD solvers on GPU to achieve 12X speedup. I believe GPU and OpenACC will have a much brighter future when people learn that GPU can do implicit schemes very efficiently."

## Straightforward Translation from OpenACC to CUDA Further Optimizes Performance

Once an algorithm is written in OpenACC, it is straightforward to translate to CUDA. Luo's team wrote the fine-grained ILU-based linear solver in OpenACC, then developed a CUDA version to optimize performance, based on the OpenACC algorithm.

"The advanced synchronization features of CUDA enable me to merge several kernels together, and because data management is compatible, each kernel can be individually translated, greatly reducing errors," said Luo.

The CUDA Fortran version of the code with advanced optimization achieved speedups of up to 35 times. This version was tested on the same configuration as was the OpenACC version.



PGI®