SENSEI / SENSEI-Lite / SENEI-LDC
Updates

Chris Roy and Brent Pickering
Aerospace and Ocean Engineering Dept.
Virginia Tech

July 23, 2014
Collaborations with Math

Collaboration on the implicit SENSEI-LDC code

• Focus is on solvers and preconditioners
• Maximum efficiency is found when considering interactions between: matrix storage format, memory use, hardware, preconditioner, solver

Collaboration on SENSEI

• SENSEI uses modern Fortran, but includes ISO-C bindings so we can interface with existing solvers in C
• SENSEI uses a built in CPU solver library (Fortran), but has recently been extended towards GPU functionality using the CUDA ITSOL interface (C); this is the same interface used by de Sturler’s group
• The folks in Math should now have access to the SENSEI GIT repository
Collaborations with CS

Collaboration w/ Feng’s group: SENSEI-LDC and SENSEI

• Worked with Tom Scogland to get explicit SENSEI-LDC code running on multiple GPUs (AIAA Paper, journal submission in progress)
• Developed plan for GPU-parallelizing SENSEI using OpenACC

Collaboration w/ Sandu’s group: SENSEI-Lite

• Developed a MATLAB version of SENSEI: the “real” SENSEI-Lite
• Current code capabilities: structured grid, general geometry, finite volume method, single block, inviscid, and explicit solver
• Upcoming capabilities: viscous (Navier-Stokes) & implicit w/ full Jacobian
• Sandu’s group currently has access to code through github
• Sandu’s group will use the implicit code for studying their IMEX and ROK/EXPK schemes for time accurate simulations
Published:


To be submitted very soon:


Plans for Upcoming Year

SENSEI-LDC

- Publish article with de Sturler’s group on preconditioners/solvers for GPU
- Collaborate w/ Meuller’s group on MemTrace (explicit and implicit codes)

SENSEI-Lite

- Complete code development to include viscous terms & implicit Jacobian
- Collaborate with Sandu’s group on time accurate solutions
- Collaborate with de Sturler’s group on preconditioners/solvers

SENSEI

- Improve general preconditioners/solvers and implementation on GPU w/ de Sturler’s group
- Develop strategy for handling function pointers and allocatables within OpenACC w/ Feng’s group
- Implement OpenACC directives in SENSEI code base w/ Feng’s group
Additional work with incompressible Navier-Stokes (INS) finite-difference code.

**RECAP:**
- 2D Cartesian grid FDM.
- Solves incompressible Navier-Stokes using artificial compressibility (lid-driven cavity benchmark case).
- Ported from an existing Fortran code to run on GPUs using OpenACC + PGI compiler.

Recent work focused on performance optimization of the OpenACC code and running on multiple GPUs.

Newer versions of PGI compiler (14.x) support additional accelerator platforms, including AMD GPUs.

Above: Speedup of INS code on several GPU platforms relative to a single CPU thread (SSE vectorized) running on a Xeon X5560.
Can use OpenACC clauses to control the kernel launch configuration on NVIDIA devices.

- For CUDA accelerators, OpenACC vector parameter corresponds to the number of CUDA threads in a block.
- Enables application specific tuning.

Explored entire parameter space of possible 2D thread-block dimensions.

- Tested both Fermi (C2075) and Kepler (K20) GPUs, using double and single precision arithmetic.
- On all platforms, the default block size (when no vector clause was used) was observed to be 64x4.
- Manual tuning showed performance increases ranging from 6-33% on the different GPUs. The compiler default was never found to be optimal.

Above: Optimization results for a K20c GPU using double precision.

**Default:** 64x4 threads/block → 68.5 GFLOPS  
**Optimal:** 16x8 threads/block → 90.6 GFLOPS
Multi-GPU Scaling

- Near linear performance scaling using multiple-GPUs.
  - Used domain decomposition, with each domain partition residing on one GPU for the duration of the simulation (only ghost-cells had to be exchanged on each iteration).
  - One control CPU thread per GPU.
  - PGI 14.1 compiler can generate code for AMD GPUs in addition to NVIDIA.
Matlab framework for studying high-order time-integration methods

- Code is based directly on the SENSEI CFD code (same numerical methods).
- 2D, single-block structured grid FVM.
- Written with “mex” files (C++) for better performance.
- Splits the RHS flux contributions to facilitate working with IMEX schemes.
  - E.g., split equations into convective + diffusive components.

Goal: Full LHS(Jacobian) + RHS equation splitting to enable testing of general time-accurate methods.

Above: Solution from Matlab CFD code for NACA 0012 at Mach 0.8, 5\(^\circ\) AOA.