Single Node Optimization on Hopper

Michael Stewart, NERSC
Introduction

- Why are there so many compilers available on Hopper?
- Strengths and weaknesses of each compiler.
- Advice on choosing the most appropriate compiler for your work.
- Comparative benchmark results.
- How to compile and run with OpenMP for each compiler.
- Recommendations for running hybrid MPI/OpenMP codes on a node.
Why So Many Compilers on Hopper?

- Franklin was delivered with the only commercially available compiler for Cray Opteron systems, PGI.
- GNU compilers were on Franklin, but at that time GNU Fortran optimization was poor.
- Next came Pathscale because of superior optimization.
- Cray was finally legally allowed to port their compiler to the Opteron so it was added next.
- Intel was popular on Carver, and it produced highly optimized codes on Hopper.
- PGI is still the default, but this is not a NERSC recommendation. Cray's current default is the Cray compiler, but we kept PGI to avoid disruption.
PGI

● Strengths
  ○ Available on a wide variety of platforms making codes very portable.
  ○ Because of its wide usage, it is likely to compile almost any valid code cleanly.

● Weaknesses
  ○ Does not optimize as well as compilers more narrowly targeted to AMD architectures.

● Optimization recommendation:
  ○ -fast
Cray

● Strengths
  ○ Fortran is well optimized for the Hopper architecture.
  ○ Uses Cray math libraries for optimization.
  ○ Well supported.

● Weaknesses
  ○ Compilations can take much longer than with other compilers.
  ○ Not very good optimization of C++ codes.

● Optimization recommendations:
  ○ Compile with no explicit optimization arguments. The default level of optimization is very high.
Intel

- **Strengths**
  - Optimizes C++ and Fortran codes very well.
  - Supports C++ very well.

- **Weaknesses**
  - Occasional problems in porting codes to this compiler.

- **Optimization recommendations:**
  - Compile with no explicit optimization arguments. The default level of optimization is very high.
GNU/GCC

● Strengths
  ○ Available on a wide variety of platforms for free.
  ○ Exposure to a wide variety of codes, so any given code is likely to compile cleanly.
  ○ Very good at C++ optimization.
  ○ Optimizes Fortran codes as well as PGI on the average.

● Weaknesses
  ○ Not a commercial product, so no guarantee of bug fixes.

● Optimization recommendation:
  ○ -O3 -ffast-math
Pathscale

● Strengths
  ○ Good optimization.

● Weaknesses
  ○ Support level and future of the product are questionable.
  ○ Cray is withdrawing library support for this compiler.

● Optimization recommendation:
  ○ -O3
Which Compiler to Use?

- Porting a code to Hopper.
  - Use the existing compiler if it is on Hopper, since relatively minor changes should be necessary to the Makefile or configure script.

- Developing a code on Hopper.
  - For C++ use Intel or GNU.
  - Will it only run on Hopper? The Cray Fortran and Intel compilers are likely to produce the fastest code.
  - Will it be ported to other systems? GNU and PGI will produce relatively fast code and can be ported more easily to other architectures.
Hopper Benchmark Performance

![Hopper Benchmark Performance Chart]
Compiling for OpenMP on Hopper

- Cray compiler:  -Oomp (on by default)
- PGI:  -mp=nonuma
- Intel: -openmp
- GNU:  -fopenmp
- Pathscale:  -mp
Running with OpenMP on Hopper

● Run time all compilers:
  ○ - set OMP_NUM_THREADS to number of threads
  ○ aprun -d numthreads ...
● Pathscale run time - set PSC_OMP_AFFINITY to FALSE.
● Intel run time - use "-cc none" or "-cc numa_node" arguments to aprun.
OpenMP/Hybrid Run Time Optimization

- Each 24 core Hopper compute node consists of 4 6 core "numa nodes"
- Best hybrid code performance when you allocate 1 MPI process with 6 threads to each of these nodes and use their local memory
- Single node parameters:
  - `export OMP_NUM_THREADS=6`
  - `aprun -d 6 -N 4 -S 1 -ss`
Questions?