Benchmark Performance of Different Compilers on a Cray XE6

Mike Stewart and Helen He
NERSC User Services Group
May 23-26, CUG 2011
Outline

• Introduction
• Available Compilers on Hopper
• Recommended Compiler Options
• Benchmarks Used in the study
• Performance Results from Each Compiler
• Summary and Recommendations
• Cray XE6, 6,384 nodes, 153,126 cores.
• Each node has 2 twelve-core AMD MagnyCours 2.1 GHz procs.
• 1.28 Pflops/peak, 212 TB memory.
Available Compilers on Hopper

- **Portland Group Compilers**
  - This is the default compiler on Hopper

- **Pathscale Compilers**
  - `% module swap PrgEnv-pgi PrgEnv-pathscale`

- **Cray Compilers**
  - `% module swap PrgEnv-pgi PrgEnv-cracy`

- **GNU Compilers**
  - `% module swap PrgEnv-pgi PrgEnv-gnu`
Compile Codes on Hopper

• Cross compilation from login nodes to build executables to run on the compute nodes.
• To use a particular compiler, first swap to the corresponding PrgEnv.
• Then use compiler wrappers:
  – ftn for Fortran codes
  – cc for C codes
  – CC for C++ codes
• The wrappers can find the proper system and MPI libraries.
## Compiler Flags Comparison

<table>
<thead>
<tr>
<th>PGI</th>
<th>Pathscale</th>
<th>Cray</th>
<th>GNU</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-fast</td>
<td>-Ofast</td>
<td>-O3</td>
<td>-O3</td>
<td>High level optimization</td>
</tr>
<tr>
<td>-mp=nonuma</td>
<td>-mp</td>
<td>-h omp</td>
<td>-fopenmp</td>
<td>Enable OpenMP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(default)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-byteswapio</td>
<td>-byteswapio</td>
<td>-h byteswapio</td>
<td>-fconvert=swap</td>
<td>Read files in big-endian</td>
</tr>
<tr>
<td>-Mfixed</td>
<td>-fixedform</td>
<td>-f fixed</td>
<td>-ffixed-form</td>
<td>Fixed form source</td>
</tr>
<tr>
<td>-Mfree</td>
<td>-freeform</td>
<td>-f free</td>
<td>-ffree-form</td>
<td>Free form source</td>
</tr>
<tr>
<td>-V</td>
<td>-dumpversion</td>
<td>-V</td>
<td>--version</td>
<td>Show version info</td>
</tr>
<tr>
<td>not implemented</td>
<td>-zerouv</td>
<td>-e 0</td>
<td>-finit-local-zero</td>
<td>Zero fill uninitialized values</td>
</tr>
</tbody>
</table>
Recommended Options: PGI Compiler

• NERSC recommends:
  ➢ -fast or –fastsse

• PGI User Documentation:
  ➢ “-fast –Mipa=fast” is a good set of options.

• Cray recommends:
  ➢ -fast –Mipa=fast
  ➢ If can be flexible with precision, also try –Mfpreleaxed.
Recommended Options: Pathscale Compiler

- **NERSC recommends:**
  - -Ofast

- **Pathscale User Documentation:**
  - Start with –O2, then –O3,
  - then –O3 –OPT:Ofast, then -Ofast.

- **Cray recommends:**
  - -Ofast
Recommended Options: Cray Compiler

• NERSC recommends:
  ➢ -O3

• Cray recommends:
  ➢ Use default –O2, which is equivalent to –O3 or –fast in other compilers.
  ➢ Use –O3,fp3 (or –O3 –hfp3)
    ➢ -O3 only slightly better than –O2
    ➢ -hfp3 gives maximum freedom in floating point optimization, may not conform to IEEE standard.
Recommended Options: GNU Compiler

• NERSC recommends:
  ➢ -O3

• Cray recommends:
  ➢ -O3 –ffast-math –funroll-loops
  ➢ -ffast-math: may not conform IEEE standard
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Science</th>
<th>Algorithm</th>
<th>Concurrency</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTC</td>
<td>Fusion</td>
<td>PIC, finite difference</td>
<td>2048 (waeking scaling)</td>
<td>F90</td>
</tr>
<tr>
<td>IMPACT-T</td>
<td>Accelerator Physics</td>
<td>PIC, FFT</td>
<td>1024 (strong scaling)</td>
<td>F90</td>
</tr>
<tr>
<td>MAESTRO</td>
<td>Astrophysics</td>
<td>Block structured-grid multiphysics</td>
<td>2048 (weak scaling)</td>
<td>F90</td>
</tr>
<tr>
<td>MILC</td>
<td>Lattice Gauge Physics (QCD)</td>
<td>Conjugate gradient, sparse matrix, FFT</td>
<td>1024 (weak scaling)</td>
<td>C, Assembly</td>
</tr>
<tr>
<td>PARATEC</td>
<td>Material Science</td>
<td>DFT, FFT, BLAS</td>
<td>1024 (string scaling)</td>
<td>F90</td>
</tr>
</tbody>
</table>
## NPB 3.3 Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Full Name</th>
<th>Level</th>
<th>Concurrency</th>
</tr>
</thead>
<tbody>
<tr>
<td>BT</td>
<td>Block Tridiagonal</td>
<td>D</td>
<td>256</td>
</tr>
<tr>
<td>CG</td>
<td>Conjugate Gradient</td>
<td>E</td>
<td>256</td>
</tr>
<tr>
<td>EP</td>
<td>Embarassingly Parallel</td>
<td>E</td>
<td>256</td>
</tr>
<tr>
<td>FT</td>
<td>Fast Fourier Transform</td>
<td>D</td>
<td>256</td>
</tr>
<tr>
<td>LU</td>
<td>Lower-Upper Symmetric Gauss-Siedel</td>
<td>E</td>
<td>256</td>
</tr>
<tr>
<td>MG</td>
<td>MultiGrid</td>
<td>E</td>
<td>256</td>
</tr>
<tr>
<td>SP</td>
<td>Scalar Pentadiagonal</td>
<td>D</td>
<td>256</td>
</tr>
</tbody>
</table>
• Other 3 options do not significantly improve performance over “-fast”.
• The NPB FT case D is an exception.
Pathscale Compiler Results

Pathscale Compiler Performance

- -O2 performs worse than other 3 options.
- -O3 optimizes almost all benchmarks well.
- Extra options on top of –O3 do not improve significantly.
Cray Compiler Results

Only one benchmark with –Ofp3 shows significant improvement over default –O2.
-O3 generally gives a good level of optimization.

Worth to try –ffast-math option. Improves performance significantly in some cases.
Overall Compilers Comparison

- Pathscale fastest: 6 out of 12.
- Cray fastest: 3 out of 12.
- PGI fastest: 2 out of 12.
- GNU fastest: 1 out of 12.
- Mean against PGI: Cray 0.96, Pathscale 0.94, GNU 0.99
Summary and Recommendations

- Users should experiment with different compilers and compiler options to tune their application performance on Hopper.
- On the average the Pathscale and Cray compilers produce somewhat faster code on Hopper (or another Cray system), since they are specifically designed for these processors. In addition the Cray compilers make use of the Cray math libraries at compile time to further optimize codes.
- PGI compilers are available on a wide variety of platforms other than Cray machines. Many existing codes have PGI targeted Makefiles, could generate very good performance.
- Using the gnu compilers allows you to compile on virtually every Unix and Linux system. Although the performance on Hopper for some codes with GNU compilers is quite good, there is no guarantee for optimal performance on other platforms.