Using the Cray Compiler at NERSC
- Usability and Performance

Zhengji Zhao, Megan Bowling and Jack Deslippe
NERSC User Services
Cray Quarterly, July 25, 2012
• Provide feedback to Cray about how Cray compiler is used at NERSC, focusing on its usability and performance

• Report issues encountered with compilation, execution, validity check and performance, using a set of materials and chemistry application codes instead of using the standard N6 application benchmark codes
1. Automatic Library Tracking Database (ALTD, developed by NICS) tracks the library usage both at compile and run time by intercepting the “ld” and “aprun” commands, respectively.

2. 6/21/2012 in production on Hopper

Among the 224459 successful links, only 9% of them used the Cray compiler
Compiler usage on Hopper 
(2012-05-15 – 2012-07-21)--continued

9% of unique users who compiled codes on Hopper used Cray compiler.

4% of unique binaries were compiled with Cray compiler.
• Many application codes do not support the Cray compiler in their configure script – configure fails
• Cray compiler often fails to compile the codes that all other compilers, PGI, GNU, Intel, compile fine.
  – Pros: good for new code development, less buggy codes; can help finding bugs in the codes.
  – Cons: difficult to use with existing codes
• Atomics operation is not supported in Cray compiler
diff -r vasp.5.2/aedens.F ../orig/vasp.5.2/aedens.F
< TYPE (grid_3d),TARGET :: GRID_SOFT,GRIDC_,GRIDUS
---
> TYPE (grid_3d) GRID_SOFT,GRIDC_,GRIDUS

Failed with Cray compiler – works with all others on Hopper

Failed with Cray compiler – works with all others on Hopper

diff -r vasp.5.2/dfast.F ../orig/vasp.5.2/dfast.F
< USE dfast,only : NBLK
---
USE dfast
diff -r vasp.5.2/hamil.F ../orig/vasp.5.2/hamil.F
<  SUBROUTINE PW_CHARGE_TRACE(WDES1, CHARGE, CR1, CR2)
---
>  SUBROUTINE PW_CHARGE_TRACE(WDES1, CHARGE, NDIM, CR1, CR2)

diff -r vasp.5.2/subrot_lr.F ../orig/vasp.5.2/subrot_lr.F
<  W0%CW(:,:,NK,ISP), W0%CPROJ(:,:,NK,ISP), DEG_CLUSTEN(NK,ISP)
   %DEG_CLUSTEN, .FALSE., .FALSE.)
---
>  W0%CW(1,1,NK,ISP), W0%CPROJ(1,1,NK,ISP), DEG_CLUSTEN(NK,ISP)
   %DEG_CLUSTEN, .FALSE., .FALSE.)
Cray compiler takes longer time to compile

### VASP (Fortran) Compilation Time

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Compile options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray</td>
<td>Default; -O -ipa0</td>
</tr>
<tr>
<td>Intel</td>
<td>-O3, -fast</td>
</tr>
<tr>
<td>GNU</td>
<td>-O3, -ffast-math</td>
</tr>
<tr>
<td>PGI</td>
<td>-fattrs, -O3, -Mvect</td>
</tr>
</tbody>
</table>

### LAMMPS (C++) Compilation Time

Intel compiler takes longest time to compile
Cray compiler generates larger binaries in size

### VASP Executable Size

![Graph showing executable size for different compilers (Cray, Intel, GNU, PGI) for VASP. The graph illustrates that Cray generates significantly larger executables compared to the other compilers.]

### LAMMPS Executable Size

![Graph showing executable size for different compilers (Cray, Intel, GNU, PGI) for LAMMPS. The graph shows that Cray again generates the largest executables among the compilers.]

Run time and validity issues

• VASP failed validity check
  – Failed to run for 2 of the test cases (out of 3)
  – If remove all compiler optimizations, then code ran fine
  – “Randomly” lowered the optimization levels for the “relevant” routines, and then the code passed the other two test cases.

• Quantum Espresso generated wrong results similarly
  – Had to lower a specific routine’s compiler optimization levels.

• NWChem failed to run
  – Error

  MA internal error: MAi_inform_base: invalid datatype: 307307478419244017
  MA internal fatal error: MA_sizeof: unable to set sizes of FORTRAN datatypes
• Program Description
  – VASP is a Fortran code that performs atomic scale materials modeling.

• Options explored
  – Compilers and optimization flags used
    • PGI: -fastsse, -O3, -Mvect
    • Intel: -O3, -fast
    • GNU: -O3, -ffast-math
    • Cray: -O -ipa0

• Tested with 3 test cases
  – Algorithms: DIIS-RMM, Davidson, Hybrid
  – Concurrency: 48, 96, 144; 384,768; 48,72
Cray compiler outperforms other compilers with medium sized VASP runs

Test case 1:
- NERSC user provided test case:
- A 155 atom system
- The time to complete first 20 electronic steps were measured

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Performance gain relative to PGI compiler (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel</td>
<td>-12%</td>
</tr>
<tr>
<td>GNU</td>
<td>-6% ~ +1%</td>
</tr>
<tr>
<td>PGI</td>
<td>default</td>
</tr>
<tr>
<td>Cray</td>
<td>5.8%</td>
</tr>
</tbody>
</table>

VASP runs faster by 5.8% when switching to Cray compiler.
Cray compiler outperforms other compilers for larger test cases

Test case 2
- NERSC user provided
- A 660 atom system
- Time for first 4 electronic steps

VASP with Cray compiler runs faster by up to 11% for the larger test case.
Compiler performance varies depending on job types

Test case 3
- Provided by NERSC users
- Hybrid calculation for a 105 atom system

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Faster than PGI compiler by (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel</td>
<td>-6%</td>
</tr>
<tr>
<td>Cray, GNU</td>
<td>0%</td>
</tr>
<tr>
<td>PGI</td>
<td>default</td>
</tr>
</tbody>
</table>

VASP with Cray compiler runs at the same speed as PGI compiler for the hybrid jobs
### Performance increase compared to PGI

<table>
<thead>
<tr>
<th>Program</th>
<th>PGI</th>
<th>Intel</th>
<th>GNU</th>
<th>Cray</th>
<th>Best compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>VASP</td>
<td>0%</td>
<td>-12% ~ -5%</td>
<td>-6% ~ 4%</td>
<td>0% ~ 11%</td>
<td>Cray</td>
</tr>
<tr>
<td>QE</td>
<td>0%</td>
<td>2%</td>
<td>-1%</td>
<td>-7%</td>
<td>Intel</td>
</tr>
<tr>
<td>NAMD</td>
<td>0%</td>
<td>14%</td>
<td>18%</td>
<td>Failed</td>
<td>GNU</td>
</tr>
<tr>
<td>LAMMPS*</td>
<td>0%</td>
<td>5% ~ 17%</td>
<td>-5% ~ 9%</td>
<td>-6% ~ 4%</td>
<td>Intel</td>
</tr>
<tr>
<td>BerkeleyGW</td>
<td>0%</td>
<td>0%</td>
<td>-13%</td>
<td>-8%</td>
<td>PGI/Intel</td>
</tr>
<tr>
<td>NWChem</td>
<td>0%</td>
<td>12% ~ 34%</td>
<td>-9% ~ 28%</td>
<td>Failed</td>
<td>Intel</td>
</tr>
</tbody>
</table>

#### Compiler versions

- **PGI** 11.9.0
- **GNU** 4.6.2
- **Intel** 12.1.2.273
- **Cray** cce/8.0.1

Blue: max performance increase
Red: max performance decrease

*) LAMMPS data updated with newer versions of compilers, pgi/12.4.0, intel/12.1.4.319, cce/8.0.5, gcc/4.6.3
• Cray compiler is in low usage on Hopper
• Cray compiler is proven to be difficult to use for existing third party application codes.
• The performance varies, a good performance is observed with VASP (Fortran code), but not for other codes.
• We do not recommend changing the compiler default on Hopper to Cray compiler at any time soon until the usability issues are resolved or reduced to some extent.

zz217@hopper12:~> qsub -l -l mppwidth=24 -q debug -V
qsub: waiting for job 1975244.sdb to start
qsub: job 1975244.sdb ready

ModuleCmd_Switch.c(172):ERROR:152: Module 'PrgEnv-cray' is currently not loaded
zz217@nid04755:~>