Three Fortran Compilers on Atlas

Grace Foo

(SVU/Academic Computing, Computer Centre, NUS)

Atlas is the new 16-node high performance Linux cluster at SVU. Each node has a dual Intel Xeon 3.06 GHz CPU and 2 GB of memory. Connecting the nodes is a high-speed, low-latency Myrinet switch. As with all SVU systems, batch queues have been set up to facilitate job/load management on the cluster. Both serial and parallel jobs are supported. Access to the cluster is via the e-SVU portal or by directly logging into the head nodes atlas0 or atlas1. More information about the cluster, such as software, libraries and tools, is found at the following link: http://e-svu.nus.edu.sg/linuxdocs

Atlas is installed with three Fortran compilers:

- g77, a GNU Open Source software,
- pgf77 from Portland Group (PG), and
- ifc from Intel.

All are available on atlas0 and atlas1. Which one gives the best performance? To help address this question, we tested these compilers on three sample programs (all are serial codes).

First, we compiled the codes (using no compiler optimization options) and collected the CPU times they took to run (average of 3 runs), for example,

g77 –o c1g c1.f
pgf77 –o c1p c1.f
ifc –o c1i c1.f

The timings (in seconds) are shown in the following table:

<table>
<thead>
<tr>
<th>Code name</th>
<th>Memory</th>
<th>g77</th>
<th>pgf77</th>
<th>ifc</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>490-730 B</td>
<td>429.6</td>
<td>320.6</td>
<td>259.9</td>
</tr>
<tr>
<td>C2</td>
<td>21 MB</td>
<td>214.5</td>
<td>72.8</td>
<td>56.3</td>
</tr>
<tr>
<td>C3</td>
<td>90 MB</td>
<td>1968.4</td>
<td>213.7</td>
<td>112.3</td>
</tr>
</tbody>
</table>

The results show that for each code, the Intel compiler gives the fastest results, followed by PG, then GNU. The codes compiled using ifc ran 23-90% faster than the PG compiled ones, and 65% to an incredible 1650% faster than the GNU compiled.

Compiling for Optimization

When the same codes were compiled with optimization flags, for example,

g77 –O5 –o c1g_opt c1.f
pgf77 –fast –o c1p_opt c1.f
ifc –O3 –o c1i_opt c1.f
we obtained the results in the table below. You can find out what optimization flags each compiler has from the man page (on atlas0 or atlas1):

```
man g77
man pgf77
man ifc
```

<table>
<thead>
<tr>
<th>Code name</th>
<th>Memory</th>
<th>g77 (-O5)</th>
<th>pgf77 (-fast)</th>
<th>ifc (-O3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>490-730 B</td>
<td>430.9</td>
<td>310.2</td>
<td>259.9</td>
</tr>
<tr>
<td>C2</td>
<td>21 MB</td>
<td>75.2</td>
<td>62.5</td>
<td>55.5</td>
</tr>
<tr>
<td>C3</td>
<td>90 MB</td>
<td>1187.8</td>
<td>130.1</td>
<td>112.4</td>
</tr>
</tbody>
</table>

It is interesting to note that when compiled with optimization flags, the CPU times did not improve for the Intel compiler on all three codes. But CPU times were improved for the PG optimized codes, which ran 3-39% faster, and for the GNU optimized, even more, 39-65% faster.

A quick check of the ifc man page, shows that even when compiled with no optimization flag, a default O1 optimization is used. There is a flag to disable optimizations (O0) which we try on the codes to give CPU run times of 271.9, 130.7, and 650.8 seconds, for C1, C2, C3, respectively. Indeed, these are all longer compared with the default optimized.

With optimization flags on, the Intel compiler still gives the fastest results, although the PG and GNU have improved significantly. Intel ifc is some 12-19% faster than PG, and 35% to 950% faster than GNU.

Do check the correctness/accuracy of your code results when using more aggressive optimization.

**Summary**

Compiler optimizations play a significant role in reducing the run times of your code. With appropriate optimization flags, there seems to be only a small difference in CPU run times between the PG and Intel compiled codes, but still quite a significant difference with GNU compiled codes. Your code’s performance under the compilers is likely to vary, as it did for the sample codes. So if you have codes that you run many times, it is worth spending some time testing out the compilers and optimization flags on the code.

While Intel ifc seems to give the best results, there are other considerations beside CPU run times when choosing a compiler to use, such as accuracy of results, availability/ease of use of the debugger, and availability of libraries.

Please send your queries/problems on compilers to the SVU helpdesk at ccesvuhelp@nus.edu.sg.