Blue Waters: Cray reveal webinar

Reveal: modules and compilation setup

```
arnoldg@h2ologin1:~/openacc-workshop/solutions/001-laplace2D-kernels> module list | grep perf
33) perftools-base/7.0.2
35) perftools-lite-loops
  (from history)
1046 ftn -c perftools-lite-loops laplace.f90
1047 ftn -o laplace_f90_reveal laplace.o
arnoldg@h2ologin1:~/openacc-workshop/solutions/001-laplace2D-kernels> ftn -o laplace_f90_reveal laplace.o
INFO: creating the CrayPat-instrumented executable 'laplace_f90_reveal'
(loop_profile)
...OK
```

Get a batch job.

in a job: aprun with perftools-lite-loops

```
arnoldg@nid27559:~/openacc-workshop/solutions/001-laplace2D-kernels> aprun -n 1 ./laplace_f90_reveal
[PE_0]: MPI rank order: Using default aprun rank ordering.
[PE_0]: rank 0 is on nid26410
CrayPat/X: Version 7.0.2 Revision a975333 05/16/18 15:45:29
Jacobi relaxation Calculation: 4096 x 4096 mesh  0 0.250000  100 0.002397
200 0.001204...
900 0.000269 completed in 102.654 seconds

# # # CrayPat-lite Performance Statistics # # #

CrayPat/X: Version 7.0.2 Revision a975333 05/16/18 15:45:29
Experiment: lite lite/loop_profile
Number of PEs (MPI ranks): 1
Numbers of PEs per Node: 1
Numbers of Threads per PE: 1
Number of Cores per Socket: 8
Execution start time: Tue Aug 14 10:53:12 2018
System name and speed: nid26410 2.300 GHz (nominal)
AMD Interlagos CPU Family: 21 Model: 1 Stepping: 2
Core Performance Boost: 1 PE has CPB capability
DRAM: 64 GiB DDR0-#10 on 2.3 GHz nodes
```
<table>
<thead>
<tr>
<th>Loop</th>
<th>Loop Incl</th>
<th>Time</th>
<th>Loop Hit</th>
<th>Loop</th>
<th>Loop</th>
<th>Loop</th>
<th>Function =/.LOOP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time%</td>
<td>Adj.)</td>
<td>Avg</td>
<td>Min</td>
<td>Max</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>------</td>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| 99.5%| 102.653319| 0.000607| 1   | 1,000.0 | 1,000 | 1,000 | laplace
| _..LOOP.1.li.41 |       |      |      |      |      |      |                  |
| 60.4%| 62.263172 | 0.088223| 1,000 | 4,094.0 | 4,094 | 4,094 | laplace
| _..LOOP.2.li.44 |       |      |      |      |      |      |                  |
| 60.3%| 62.174949 | 62.174949| 4,094,000 | 4,094.0 | 4,094 | 4,094 | laplace
| _..LOOP.3.li.45 |       |      |      |      |      |      |                  |
| 39.2%| 40.389539 | 0.102904| 1,000 | 4,094.0 | 4,094 | 4,094 | laplace
| _..LOOP.4.li.55 |       |      |      |      |      |      |                  |
| 39.1%| 40.286635 | 40.286635| 4,094,000 | 4,094.0 | 4,094 | 4,094 | laplace
| _..LOOP.5.li.56 |       |      |      |      |      |      |                  |

Program invocation: ./laplace_f90_reveal
For a complete report with expanded tables and notes, run:
pat_report /mnt/a/u/staff/arnoldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace_f90_reveal+23727-26410t
For help identifying callers of particular functions:
pat_report -O callers+src /mnt/a/u/staff/arnoldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace_f90_reveal+23727-26410t
To see the entire call tree:
pat_report -O calltree+src /mnt/a/u/staff/arnoldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace_f90_reveal+23727-26410t
For interactive, graphical performance analysis, run:
app2 /mnt/a/u/staff/arnoldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace_f90_reveal+23727-26410t

Here are a couple Apprentice 2 (app2) views of the perftools-lite-loops data as suggested by the output above (interactive, gui performance analysis).
Notice the loop counts in Table 1, above or Table 2 from the app2 text report--those will also appear in reveal if you provide the performance_data/ directory path as the 2nd argument.
create a program library for reveal

```bash
arnoldg@h2ologin1:~/openacc-workshop/solutions/001-laplace2D-kernels> ftn -O3 -hpl=my_program.pl -c laplace.f90
```

Analyze with Reveal--the performance_data/ is the directory generated from the loop analysis job's aprun.

launch reveal

```bash
arnoldg@h2ologin1:~/openacc-workshop/solutions/001-laplace2D-kernels> reveal my_program.pl/ laplace_f90_reveal+23727-26410t/
```
If you let Reveal insert OpenMP directives, they will be maintained in your Reveal sessions for the current program library. To save them to the filesystem, filesave in Reveal.

Rebuild the code and run it again with OpenMP threads:

```
arnoldg@h2ologin3:~/openacc-workshop/solutions/001-laplace2D-kernels> ftn
laplace.f90 -h msgs
```
A = 0.0_fp_kind

ftn-6066 crayftn: SCALAR LAPLACE, File = laplace.f90, Line = 28 A loop
nest at line 28 collapsed to a single loop.

ftn-6230 crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 28 A loop
starting at line 28 was replaced with multiple library calls.

Anew = 0.0_fp_kind

ftn-6004 crayftn: SCALAR LAPLACE, File = laplace.f90, Line = 29 A loop
starting at line 29 was fused with the loop starting at line 28.

\[ \text{A(0,:) = 1.0_fp_kind} \]

ftn-6004 crayftn: SCALAR LAPLACE, File = laplace.f90, Line = 32 A loop
starting at line 32 was fused with the loop starting at line 32.

\[ \text{call cpu_time(start_time)} \]

ftn-3021 crayftn: IPA LAPLACE, File = laplace.f90, Line = 37, Column = 8
"_CPU_TIME_8" (called from "laplace") was not inlined because the compiler
was unable to locate the routine.

do while ( error.gt. tol .and. iter.lt. iter_max )

ftn-6286 crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 41 A loop
starting at line 41 was not vectorized because it contains input/output
operations at line 57.

!$OMP parallel do default(none)

ftn-6823 crayftn: THREAD LAPLACE, File = laplace.f90, Line = 45 A region
starting at line 45 and ending at line 55 was multi-threaded.

\[ \text{do j=1,m-2} \]

ftn-6294 crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 49 A loop
starting at line 49 was not vectorized because a better candidate was
found at line 50.

ftn-6817 crayftn: THREAD LAPLACE, File = laplace.f90, Line = 49 A loop
starting at line 49 was partitioned.

\[ \text{do i=1,n-2} \]

ftn-6005 crayftn: SCALAR LAPLACE, File = laplace.f90, Line = 50 A loop
starting at line 50 was unrolled 6 times. ftn-6204 crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 50 A loop
starting at line 50 was vectorized.

!$OMP parallel do default(none)

ftn-6823 crayftn: THREAD LAPLACE, File = laplace.f90, Line = 61 A region
starting at line 61 and ending at line 68 was multi-threaded.

\[ \text{do j=1,m-2} \]

ftn-6294 crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 64 A loop
starting at line 64 was not vectorized because a better candidate was
found at line 65.

ftn-6817 crayftn: THREAD LAPLACE, File = laplace.f90, Line = 64 A loop
starting at line 64 was partitioned.

\[ \text{do i=1,n-2} \]

ftn-6202 crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 65 A loop
starting at line 65 was replaced by a library call.

\[ \text{call cpu_time(stop_time)} \]

ftn-3021 crayftn: IPA LAPLACE, File = laplace.f90, Line = 72, Column = 8
"_CPU_TIME_8" (called from "laplace") was not inlined because the compiler
was unable to locate the routine.

Cray Fortran : Version 8.4.6
(20160328193024_838fea6bbef776e483380a4ecf458b3dff3 03cd0)
Cray Fortran : Wed Aug 15, 2018 09:44:24
Cray Fortran : Compile time: 0.3120 seconds
Cray Fortran : 76 source lines
Cray Fortran : 0 errors, 0 warnings, 18 other messages, 0 ansi

arnoldg@nid25355:/mnt/a/u/staff/arnoldg/openacc-workshop/solutions/001-laplace2D-kernels> time OMP_NUM_THREADS=8 aprun -n 1 -d 16 ./a.out
[PE_0]: MPI rank order: Using default aprun rank ordering.
[PE_0]: rank 0 is on nid26361
Jacobi relaxation Calculation: 4096 x 4096 mesh
 0 0.250000 100 0.002397 200 0.001204 300 0.000804 400
 0.000603 500 0.000483 600 0.000403 700 0.000345 800 0.000302
 900 0.000269 completed in 358.486 seconds
Application 69060514 resources:
  utime ~359s, stime ~1s, Rss ~264432, inblocks ~10932, outblocks ~27567
real 0m48.949s user 0m1.828s sys 0m0.360s

SPEEDUP (not linear) ! Success.

**time command**

Using the time command in conjunction with other performance tools is "free". You should get into the habit of timing your code with time so that you have an accurate accounting of the wall time used for your code.

See "man reveal" to review the steps and for further information.
The reveal quickstart help screen has detailed information about how to use the GUI.
### CCE only

Your code needs to build with PrgEnv-cr ay and CCE (the Cray Compiler Environment) in order to use reveal.

### References:

https://bluewaters.ncsa.illinois.edu/reveal-and-openmp