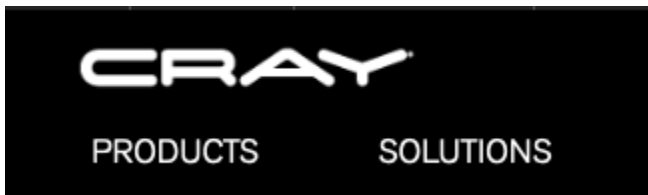


Blue Waters: Cray reveal webinar



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
```

```

Avg Process Time: 103.23 secs
High Memory: 275.6 MiBytes 275.6 MiBytes per PE
Table 1: Inclusive and Exclusive Time in Loops (from -hprofile_generate)
Loop | Loop Incl | Time | Loop Hit | Loop | Loop | Loop | Function =/.LOOP
[.]
Incl | Time | (Loop | | Trips | Trips | Trips |
Time% | | Adj.) | | Avg | Min | Max |
|-----|
| 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
===== End of CrayPat-lite output =====
Application 69044344 resources: utime ~104s, stime ~1s, Rss ~282284,
inblocks ~1 7785, outblocks ~37075
arnoldg@nid27559:~/openacc-workshop/solutions/001-laplace2D-kernels>

```

Here are a couple Apprentice 2 (app2) views of the perftools-lite-loops data as suggested by the output above (interactive, gui performance analysis).

laplace_f90_reveal+23727-26410t

File Compare View Help

About Apprentice2 laplace_f90_reveal+23727-26410t

Overview Profile Text Report Activity Call Tree

This table shows only lines with:
 LU > 0.0095
 Loop Hit > 0
 (To set thresholds to zero, specify: -T)

Loop stats can be used in the loop_info compiler directives:
 !DIR\$ LOOP_INFO est_trips(Avg) min_trips(Min) max_trips(Max)
 #pragma _CRI loop_info est_trips(Avg) min_trips(Min) max_trips(Max)
 The compiler interprets Avg as an estimate, but Min and Max as guarantees.

Times in this table include instrumentation overhead and so may be larger than times in Profile or other tables.

Table 2: Inclusive and Exclusive Time in Loops (from -hprofile_generate)

| Loop | Loop Incl | Time | Loop Hit | Loop | Loop | Loop | Function=/LOOP[] |
|-------|------------|-----------|-----------|-------|-------|-------|----------------------|
| Incl | Time | (Loop | Trips | Trips | Trips | | |
| Time% | Adj.) | Avg | Min | Max | | | |
| 99.5% | 102.653319 | 0.000607 | 1 | 1,000 | 1,000 | 1,000 | laplace_LOOP.1.li.41 |
| 60.4% | 62.263172 | 0.088223 | 1,000 | 4,094 | 4,094 | 4,094 | laplace_LOOP.2.li.44 |
| 60.3% | 62.174949 | 62.174949 | 4,094,000 | 4,094 | 4,094 | 4,094 | laplace_LOOP.3.li.45 |
| 39.2% | 40.389539 | 0.102904 | 1,000 | 4,094 | 4,094 | 4,094 | laplace_LOOP.4.li.55 |
| 39.1% | 40.286635 | 40.286635 | 4,094,000 | 4,094 | 4,094 | 4,094 | laplace_LOOP.5.li.56 |

Notes for table 3:
 Table option:
 -O himem

Wallclock time: 103.000000s


Done (0.0005s)

laplace_f90_reveal+23727-26410t

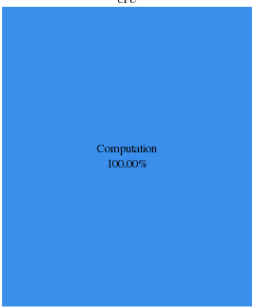
File Compare View Help

About Apprentice2 laplace_f90_reveal+23727-26410t


Overview Profile Text Report Activity Call Tree




Function/Region Profile
100.0% = laplace_




Profile
CPU
Computation
100.00%



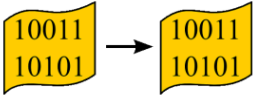
Memory Utilization
Process HiMem (MiBytes) 275.6



Load Imbalance



Energy Usage
No Energy Data Collected.



Data Movement
No data collected.

Wallclock time: 103.000000s

Done (0.0005s)

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.

my_program.pl

File Edit View Help

Navigation

Loop Performance

- 102,0000 LAPLACE@41
- 102,0000 Instance #
- 62,0000 LAPLACE@45
- 62,0000 LAPLACE@44
- 62,0000 Instance #
- 40,0000 LAPLACE@56
- 40,0000 Instance #
- 40,0000 LAPLACE@55
- 40,0000

Source - ... ldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace.f90

cce/8.4.6 Up Down Save

```

52  if(mod(iter,100).eq.0 ) write(*,'(i5,f10.6)')  iter, error
53  iter = iter + 1
54
! Directive inserted by Cray Reveal.  May be incomplete.
!$OMP parallel do default(none)
!$OMP&  private (i,j)
!$OMP&  shared (a,anew)
55  do j=1,m-2
56  do i=1,n-2
57  A(i,j) = Anew(i,j)
end do
end do
end do
call cpu_time(stop_time)
write(*,'(a,f10.3,a)')  ' completed in ', stop_time-start_time, ' sec
65

```

Info

my_program.pl/ loaded. laplace_f90_reveal+23727-26410t/ loaded.

create a program library for reveal

```
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

```
arnoldg@h2ologin1:~/openacc-workshop/solutions/001-laplace2D-kernels>
reveal my_program.pl/ laplace_f90_reveal+23727-26410t/
```

my_program.pl

File Edit View Help

Navigation

Loop Performance

- 102,0000 LAPLACE#41
- 62,0000 LAPLACE#45
- 62,0000 LAPLACE#44
- 40,0000 LAPLACE#56
- 40,0000 LAPLACE#55

40,0000 Instance #1

Source - ... ldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace.f90

```

49     end do
50     end do
51
52     if(mod(iter,100).eq.0 ) write(*,'(i
53     iter = iter + 1
54
55     do j=1,m-2
56     do i=1,n-2
57         A(i,j) = Anew(i,j)
58     end do
59     end do
60
61     end do
62
63     call cpu_time(stop_time)
64     write(*,'(a,f10.3,a)' ) ' completed i
65
66     deallocate (A,Anew)
67     end program laplace

```

Info - Line 55

- A loop starting at line 55 was scoped without errors.
- A loop starting at line 55 was not vectorized because it contains a call to a subr
- A loop starting at line 56 was replaced by a library call.

Reveal OpenMP Scoping

Scope Loops Scoping Results

Edit List

List of Loops to be Scoped

| Scope? | Line # | File or Source Line |
|-------------------------------------|--------|--------------------------|
| <input checked="" type="checkbox"/> | 41 | Loop in function LAPLACE |
| <input checked="" type="checkbox"/> | 44 | Loop in function LAPLACE |
| <input checked="" type="checkbox"/> | 45 | Loop in function LAPLACE |
| <input checked="" type="checkbox"/> | 55 | Loop in function LAPLACE |
| <input checked="" type="checkbox"/> | 56 | Loop in function LAPLACE |

Apply Filter Time: 0.000 Trips: 2 Threads: 4 Speedup: 0.010

Start Scoping Cancel 5 Loops selected Close

my_program.pl loaded. laplace_f90_reveal#23727-264101/ loaded.

my_program.pl

File Edit View Help

Navigation

Loop Performance

- 102,0000 LAPLACE#41
- 62,0000 LAPLACE#45
- 62,0000 LAPLACE#44
- 40,0000 LAPLACE#56
- 40,0000 LAPLACE#55

40,0000 Instance #1

Source - ... ldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace.f90

```

49     end do
50     end do
51
52     if(mod(iter,100).eq.0 ) write(*,'(i
53     iter = iter + 1
54
55     do j=1,m-2
56     do i=1,n-2
57         A(i,j) = Anew(i,j)
58     end do
59     end do
60
61     end do
62
63     call cpu_time(stop_time)
64     write(*,'(a,f10.3,a)' ) ' completed i
65
66     deallocate (A,Anew)
67     end program laplace

```

Info - Line 55

- A loop starting at line 55 was scoped without errors.
- A loop starting at line 55 was not vectorized because it contains a call to a subr
- A loop starting at line 56 was replaced by a library call.

Reveal OpenMP Scoping

Scope Loops Scoping Results

laplace.f90: Loop@55

| Name | Type | Scope | Info |
|------|--------|---------|------|
| i | Scalar | Private | |
| j | Scalar | Private | |
| a | Array | Shared | |
| anew | Array | Shared | |

OpenMP Directive

```

! Directive inserted by Cray Reveal. May be incomplete.
! $OMP parallel do default(none)
! $OMP private (i,j)
! $OMP shared (a,anew)

```

Find Name: Copy Directive Cancel

Insert Directive Show Directive Close

my_program.pl loaded. laplace_f90_reveal#23727-264101/ loaded.

File Edit View Help

Navigation

Loop Performance

- ▶ 102,0000 LAPLACE@41
- ▶ 62,0000 LAPLACE@45
- ▶ 62,0000 LAPLACE@44
- ▶ 40,0000 LAPLACE@56
- ▼ 40,0000 LAPLACE@55
- 40,0000 Instance

Source - ... ldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace.f90

cce/8.4.6 Up Down Save

```

52  if(mod(iter,100).eq.0 ) write(*,'(i5,f10.6)') iter, error
53  iter = iter + 1
54
! Directive inserted by Cray Reveal.  May be incomplete.
!$OMP parallel do default(none)
!$OMP& private (i,j)
!$OMP& shared (a,anew)
S 55  do j=1,m-2
AS 56  do i=1,n-2
57      A(i,j) = Anew(i,j)
58  end do
59  end do
60
61  end do
62
63  call cpu_time(stop_time)
64  write(*,'(a,f10.3,a)') ' completed in ', stop_time-start_time, '

```

my_program.pl

File Edit View Help

Navigation

Loop Performance

- ▼ 102,0000 LAPLACE@41
- 102,0000 Instance #1
- ▼ 62,0000 LAPLACE@45
- 62,0000 Instance #1
- ▼ 62,0000 LAPLACE@44
- 62,0000 Instance #1
- ▶ 40,0000 LAPLACE@56
- ▼ 40,0000 LAPLACE@55
- 40,0000 Instance #1

Source - ... ldg/openacc-workshop/solutions/001-laplace2D-kernels/laplace.f90

cce/8.4.6 Up Down Save

```

52  if(mod(iter,100).eq.0 ) write(*,'(i5,f10.6)') iter, error
53  iter = iter + 1
54
! Directive inserted by Cray Reveal.  May be incomplete.
!$OMP parallel do default(none)
!$OMP& private (i,j)
!$OMP& shared (a,anew)
S 55  do j=1,m-2
AS 56  do i=1,n-2
57      A(i,j) = Anew(i,j)
58  end do
59  end do
60
61  end do
62
63  call cpu_time(stop_time)
64  write(*,'(a,f10.3,a)') ' completed in ', stop_time-start_time, ' second
65
66  deallocate (A,Anew)
67  end program laplace

```

Info - Line 55

- A loop starting at line 55 was scoped without errors.
- A loop starting at line 55 was not vectorized because it contains a call to a subroutine or function on line 56.
- A loop starting at line 56 was replaced by a library call.

my_program.pl loaded. laplace_f90_reveal+23727-26410t loaded.

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:

Test the OpenMP version from reveal, rebuilt with "ftn"

```

arnoldg@h2ologin3:~/openacc-workshop/solutions/001-laplace2D-kernels> ftn
laplace.f90 -h msgsg

```

```

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.
  A(0,:) = 1.0_fp_kind
ftn-6332 crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 32      A loop
starting at line 32 was not vectorized because it does not map well onto
the target architecture.
ftn-6005 crayftn: SCALAR LAPLACE, File = laplace.f90, Line = 32      A loop
starting at line 32 was unrolled 8 times.
  Anew(0,:) = 1.0_fp_kind
ftn-6004 crayftn: SCALAR LAPLACE, File = laplace.f90, Line = 33      A loop
starting at line 33 was fused with the loop starting at line 32.
  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.
!$OMP parallel do default(none)      &
ftn-6823 crayftn: THREAD LAPLACE, File = laplace.f90, Line = 45      A region
starting at line 45 and ending at line 55 was multi-threaded.
  do j=1,m-2
ftn-6294 crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 49      A loop
starting at line 49 was not vectorized because a better candidate was
found at line 50.
ftn-6817 crayftn: THREAD LAPLACE, File = laplace.f90, Line = 49      A loop
starting at line 49 was partitioned.
  do i=1,n-2 ftn-6005 crayftn: SCALAR LAPLACE, File = laplace.f90,
Line = 50      A loop starting at line 50 was unrolled 6 times. ftn-6204
crayftn: VECTOR LAPLACE, File = laplace.f90, Line = 50      A loop starting
at line 50 was vectorized.
!$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.
  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
f      ound at line 65.
ftn-6817 crayftn: THREAD LAPLACE, File = laplace.f90, Line = 64      A loop
starting at line 64 was partitioned.
  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.
  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.


```
arnoldg@h2ologin3:~$ cd ~/laplace2D-kernels
```

Generating a Program Library

To generate a `program library.pl` file, make sure the `perftools-lite-loops` module is unloaded, make sure the Cray (CCE) programming environment module is loaded, and then use the CCE `-h pl` option to generate the `program library` in your current working directory. The `perftools-base` module should remain loaded in order to provide access to Reveal.

```
$ module load PrgEnv-cray
$ module unload perftools-lite-loops

$ ftn -O3 -hpl=my_program.pl -c my_program_file1.f90
$ ftn -O3 -hpl=my_program.pl -c my_program_file2.f90
$ ftn -O3 -hpl=my_program.pl -c my_program_file3.f90
$ ...
```

Note: The `-h profile_generate` option disables most automatic compiler optimizations, which is why Cray recommends generating the `program library` file separately from the loop work estimate. The `program library` is most useful when generated from fully optimized code.

Exploring the Results

After you have collected performance data from program execution and generated a `program library` file, launch Reveal and use it to integrate the results and explore opportunities for code optimization. The `perftools-base` module provides access to Reveal.

```
$ module load PrgEnv-cray
$ reveal my_program.pl experiment_data_directory
```

Note: The `PrgEnv-cray` module must be loaded in order to perform automatic OMP scoping of loops.

Manual page reveal(1) line 133

```
arnoldg@h2ologin3:~$ cd ~/laplace2D-kernels
```

Generating Loop Work Estimates

Loop work estimates are generated using the `perftools-lite-loops` instrumentation module. This instrumentation module invokes the Cray compiler with the CCE `-h profile_generate` option and then instruments the program for tracing.

To generate a loop work estimate, follow these steps. Make sure the following modules are loaded.

```
$ module load PrgEnv-cray
$ module load perftools-base
$ module load perftools-lite-loops
```

The `perftools-base` module does not affected program behavior and can be left loaded when not collecting performance data. Compile and link the program with CCE.

```
$ ftn -c my_program.f
$ ftn -o my_program my_program.o
```

Note: This option disables most automatic compiler optimizations, which is why Cray recommends generating this data separately from generating the `program library` file. The `program library` is most useful when generated from fully optimized code.

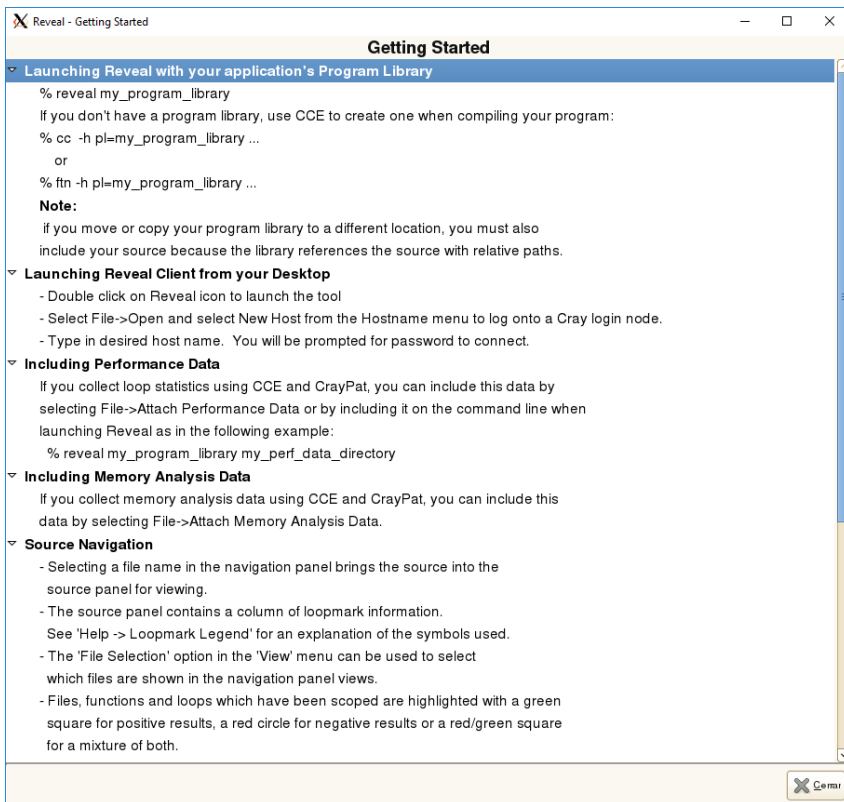
The resulting binary, `my_program`, is instrumented to collect work estimates. Execute the instrumented executable.

```
$ aprun -n pes ./my_program
```

This generates an experiment data directory and a loops report to `stdout`. The experiment data directory can be fed as input to Reveal.

Manual page reveal(1) line 97

The reveal quickstart help screen has detailed information about how to use the GUI.



CCE only

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

References:

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