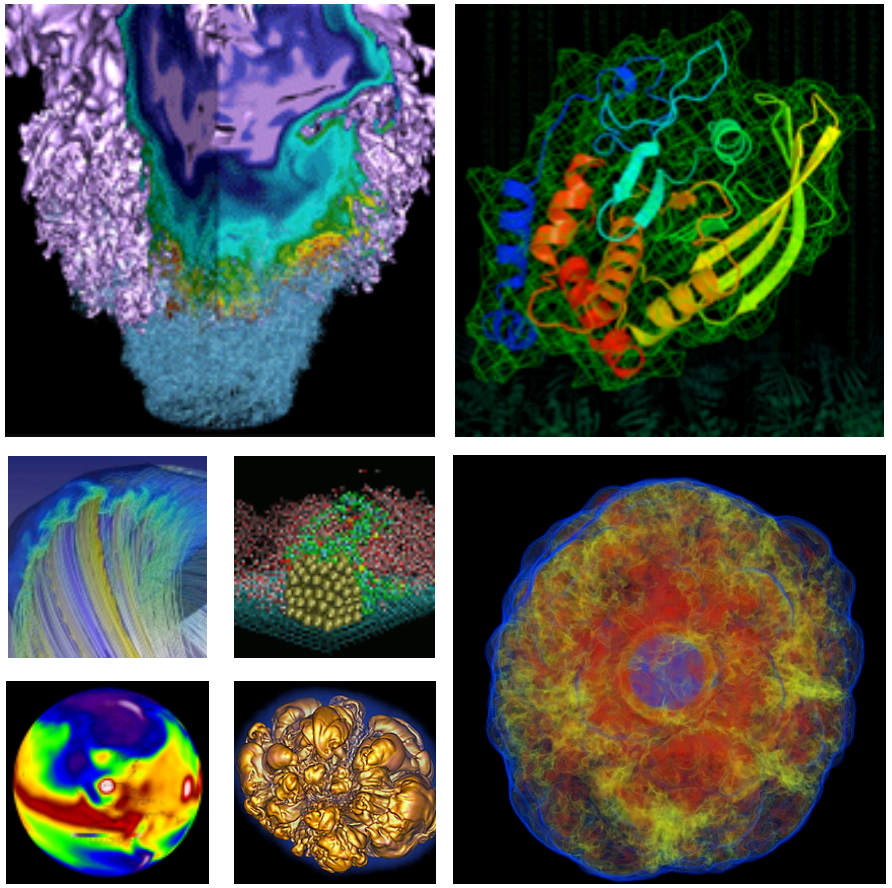
Adding OpenMP to Your Code Using Cray Reveal



Helen He
NERSC User Services Group

October 10, 2013

Current Architecture Trend

- **Multi-socket nodes with rapidly increasing core counts**
- **Memory per core decreases**
- **Memory bandwidth per core decreases**
- **Network bandwidth per core decreases**
- **Need a hybrid programming model with three levels of parallelism**
  - MPI between nodes or sockets
  - Shared memory (such as OpenMP) on the nodes/sockets
  - Increase vectorization for lower level loop structures

Advantages of hybrid MPI/OpenMP

- Reduce number of MPI ranks per node
- Minimize network injection contention
- Avoids the extra communication overhead with MPI within node
- Reduce memory footprint
- Chance of overlapping MPI communication with OpenMP thread computation

What is Reveal

- A tool developed by Cray to help developing the hybrid programming model
- Part of the Cray Perftools software package
- Only works under PrgEnv-cray
- Utilizes the Cray CCE program library for loopmark and source code analysis, combined with performance data collected from CrayPat
- Helps to identify top time consuming loops, with compiler feedback on dependency and vectorization
- Loop scope analysis provides variable scope and compiler directive suggestions for inserting OpenMP parallelism to a serial or pure MPI code

Steps to Use Reveal on Edison (1)

- **Load the user environment**
  - % module swap PrgEnv-intel PrgEnv-cray
  - % module unload darshan
  - % module load perftools (current default is version 6.1.2)
- **Generate loop work estimates**
  - % ftn –c –h profile_generate myprogram.f90
  - % ftn –o poisson_serial –h profile_generate myprogram.o
    - Good to separate compile and link to keep object files
    - Optimization flags disabled with –h profile-generate
  - % pat_build –w myprogram (-w enables tracing)
    - It will generate executable "myprogram+pat"
  - Run the program "myprogram+pat"
    - It will generate one or more myprogram+pat+…xf files
  - % pat_report myprogram+pat…xf > myprogram.rpt
    - It will generate myprogram+pat….ap2 file

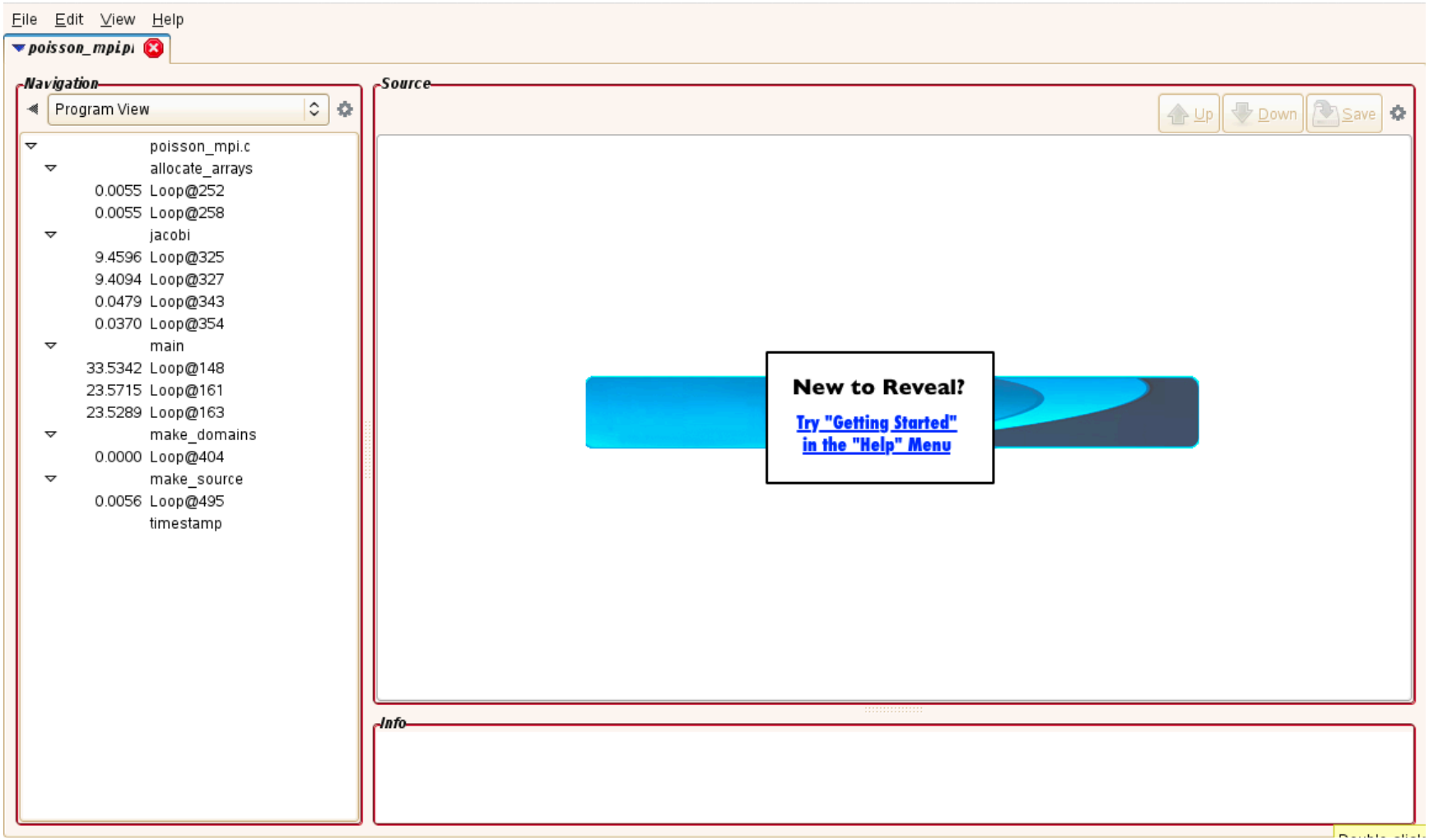
Steps to Use Reveal on Edison (2)

- **Generate a program library**
  - % ftn –O3 –hpl=myprogram.pl –c myprogram.f90
  - Optimization flags can be used
  - Build one source code at a time, with "-c" flag
  - Use absolute path for program library if sources are in multiple directories
  - User needs to clean up program library from time to time
- **Launch Reveal**
  - % reveal myprogram.pl myprogram+pat…ap2

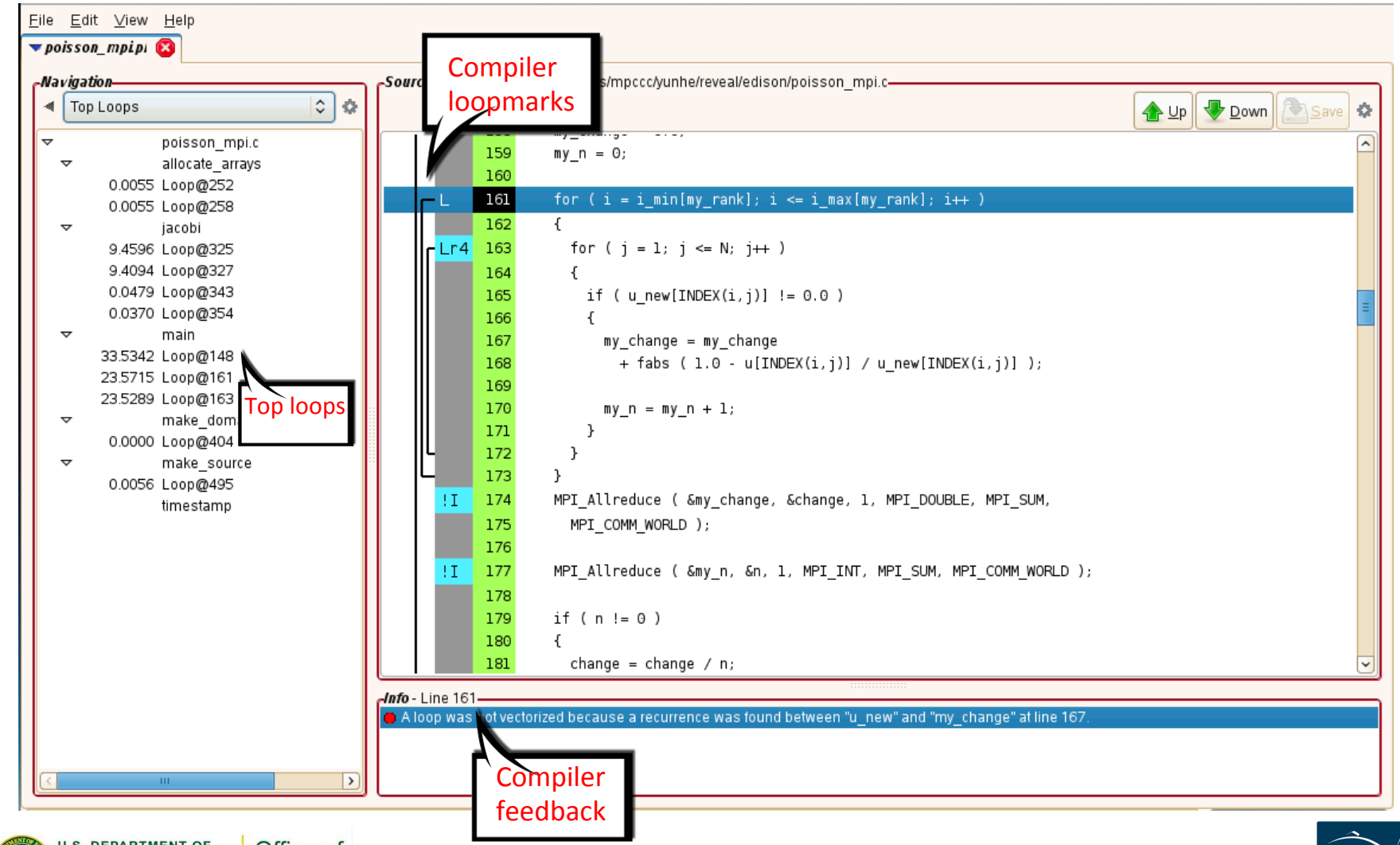
Steps to Use Reveal on Hopper

- **To use the newest version perftools/6.1.2, which is built upon cray-mpich/6.x.x**
  - % module unload cray-libsci cray-mpich2
  - % module load cray-libsci/12.1.01
  - % module load cray-mpich/6.1.0
  - % module unload darshan
  - % module load perftools-lite/6.1.2
- **To use perftools-lite/6.1.1 or older**
  - % module unload darshan
  - % module load perftools-lite/6.1.2
- **Follow the rest of steps for Edison**

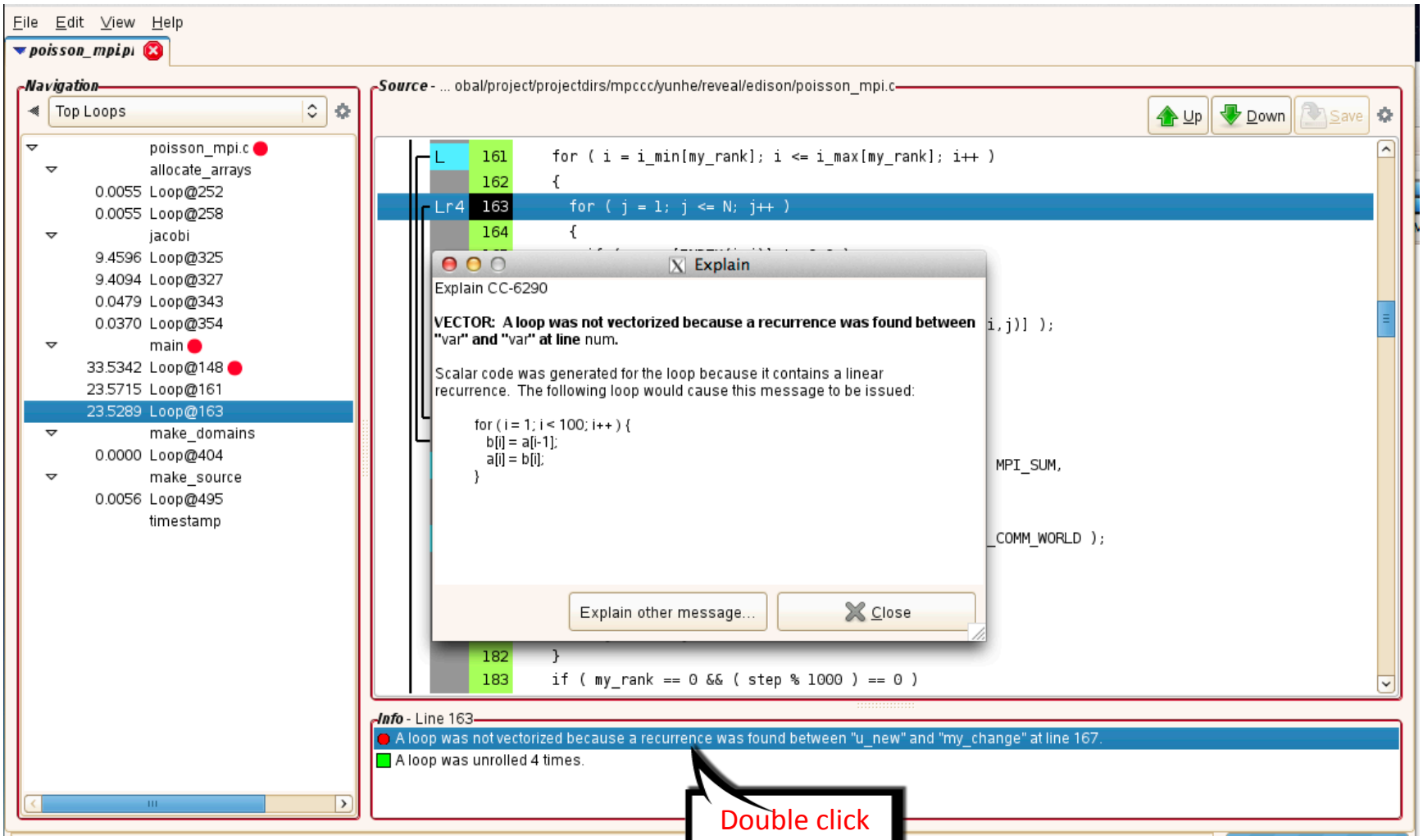
Cray Reveal GUI



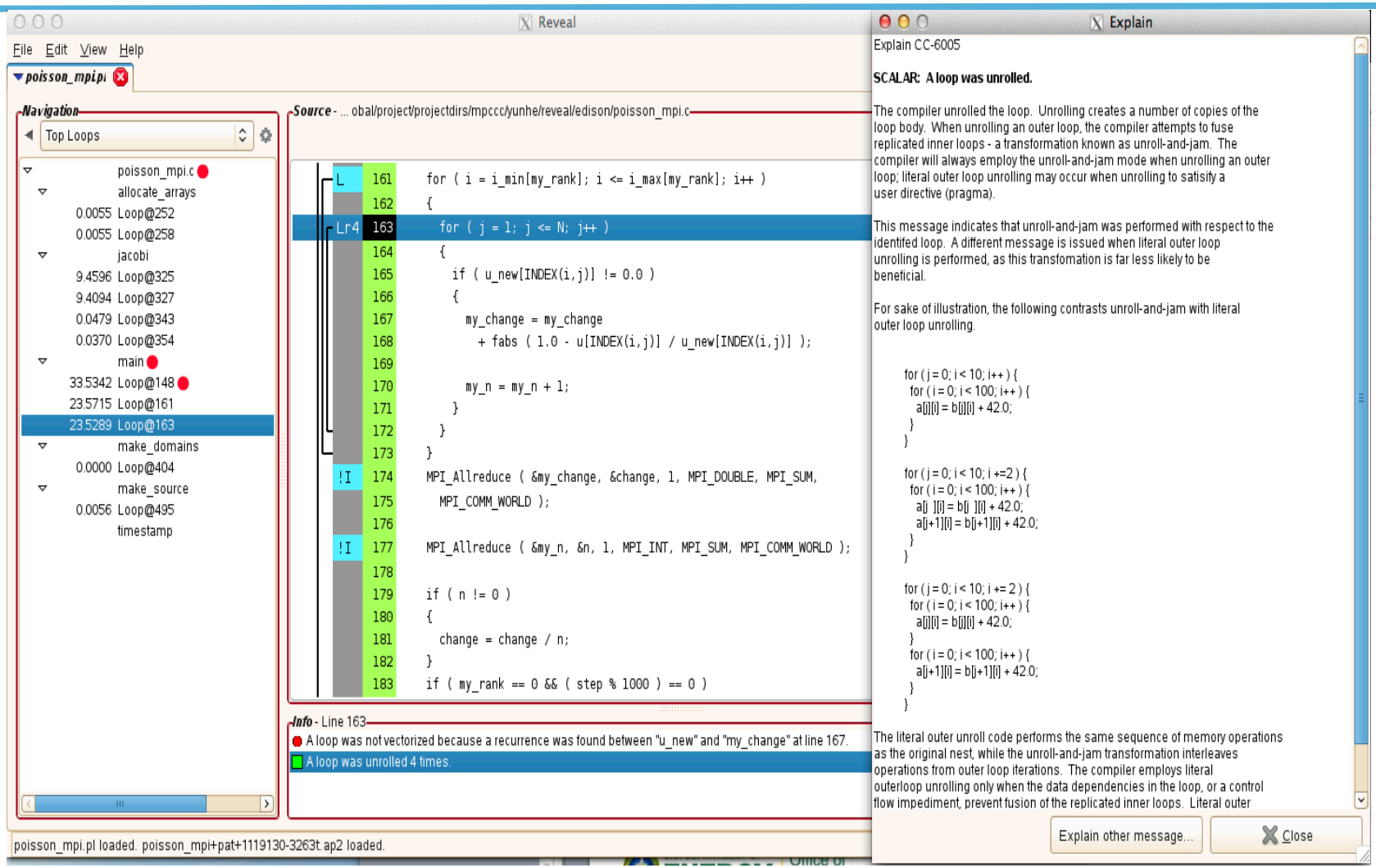
Top loops with compiler loopmarks and feedback



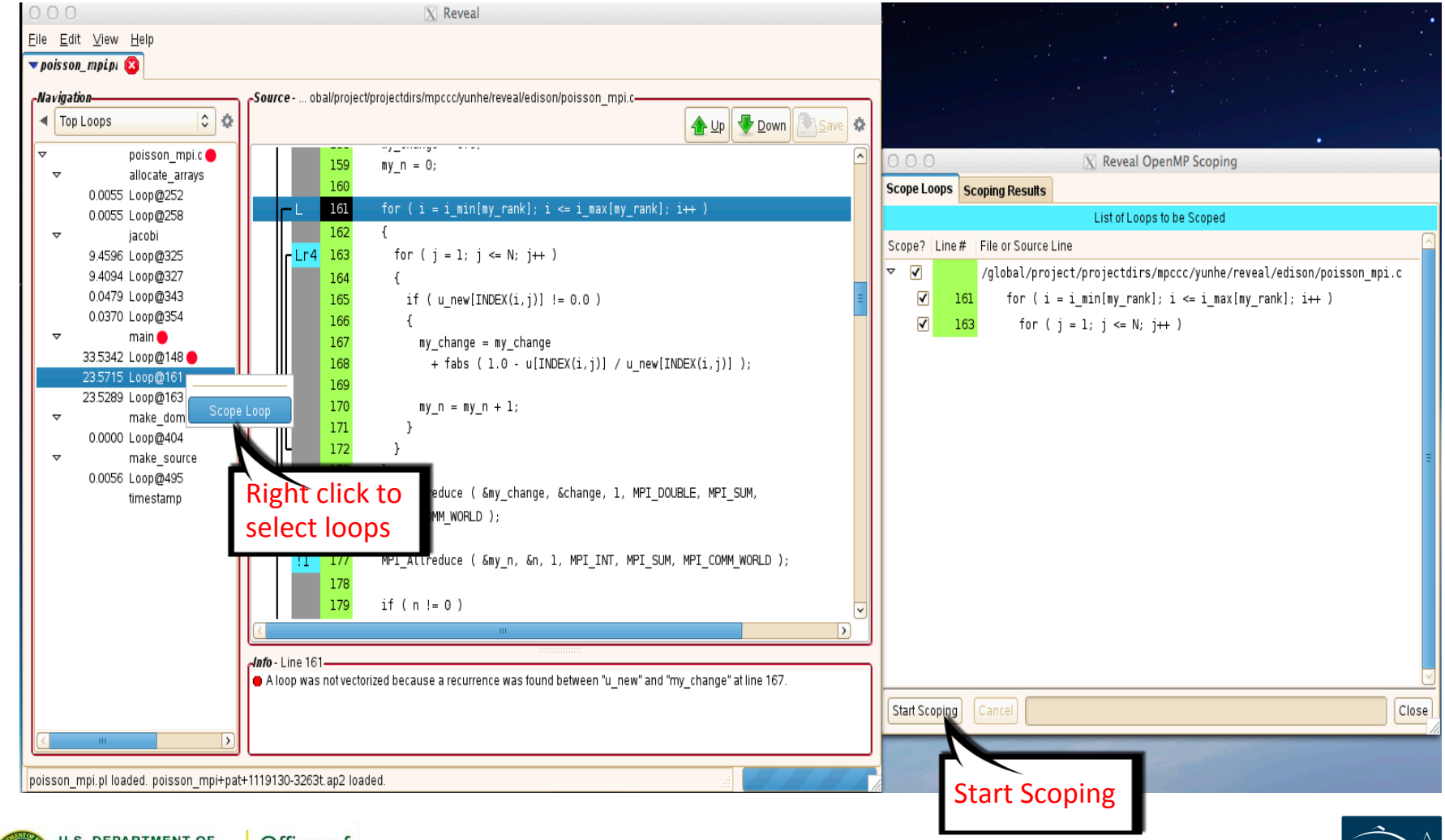
Compiler feedback explanation



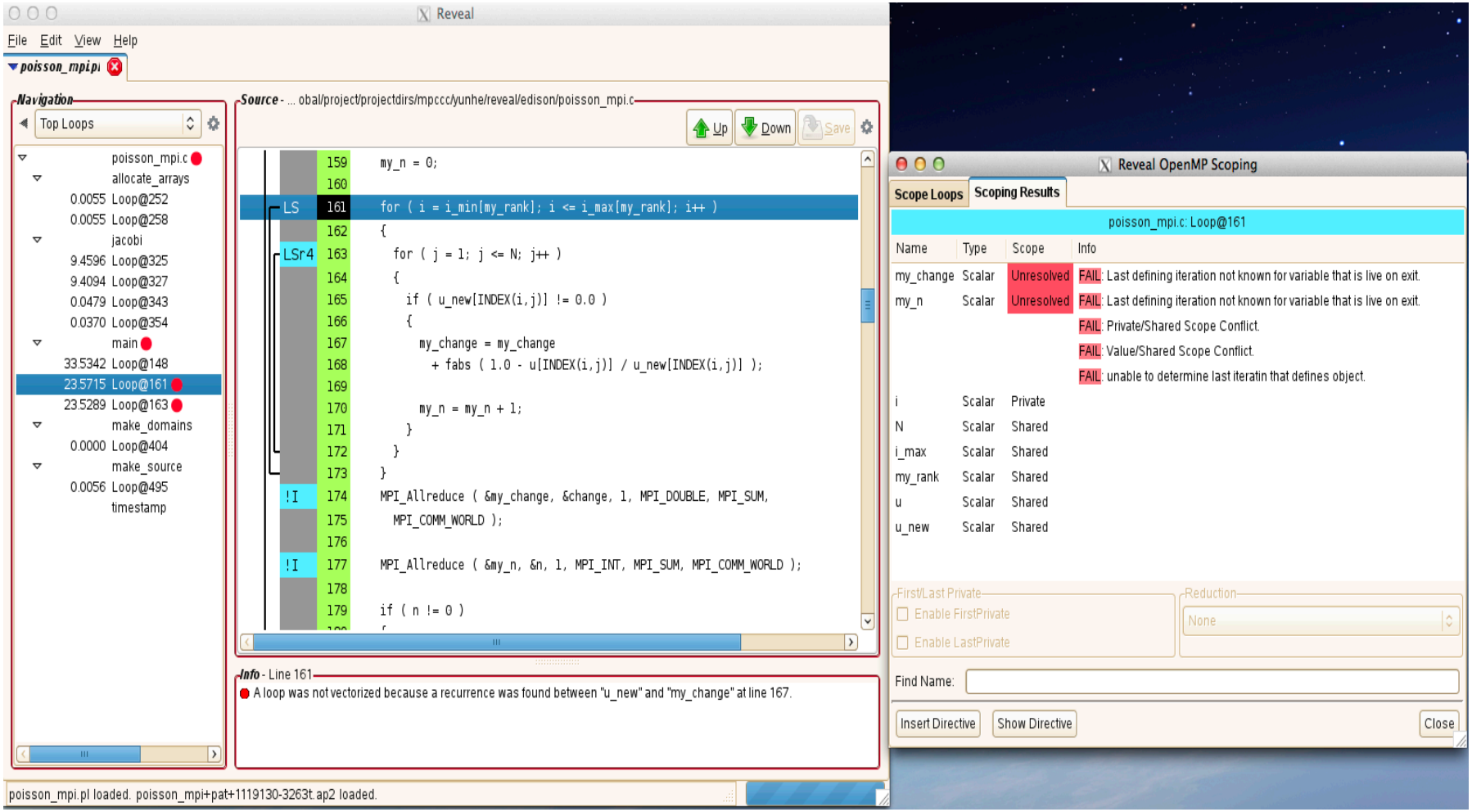
Compiler feedback explanation (2)



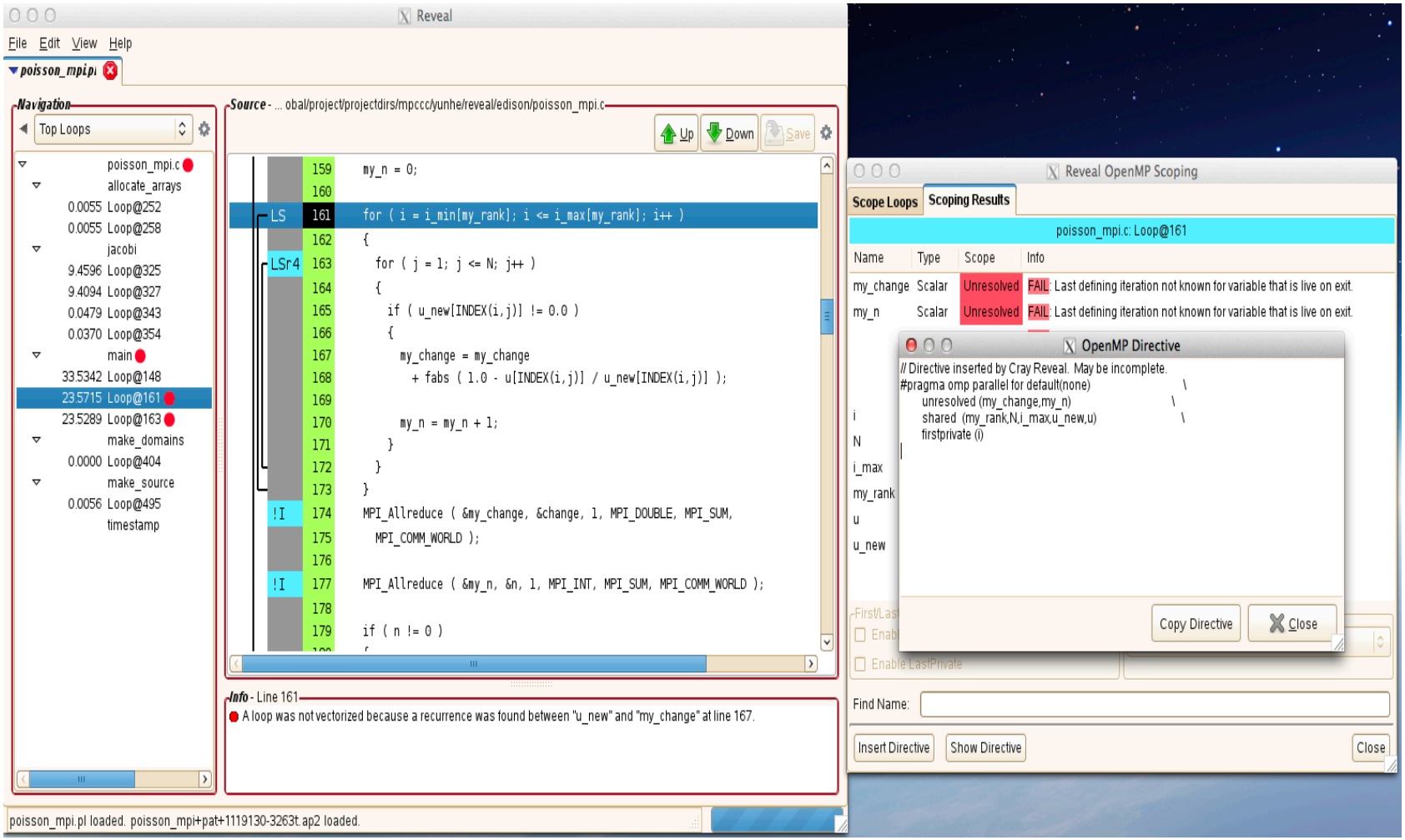
Reveal scoping assistance



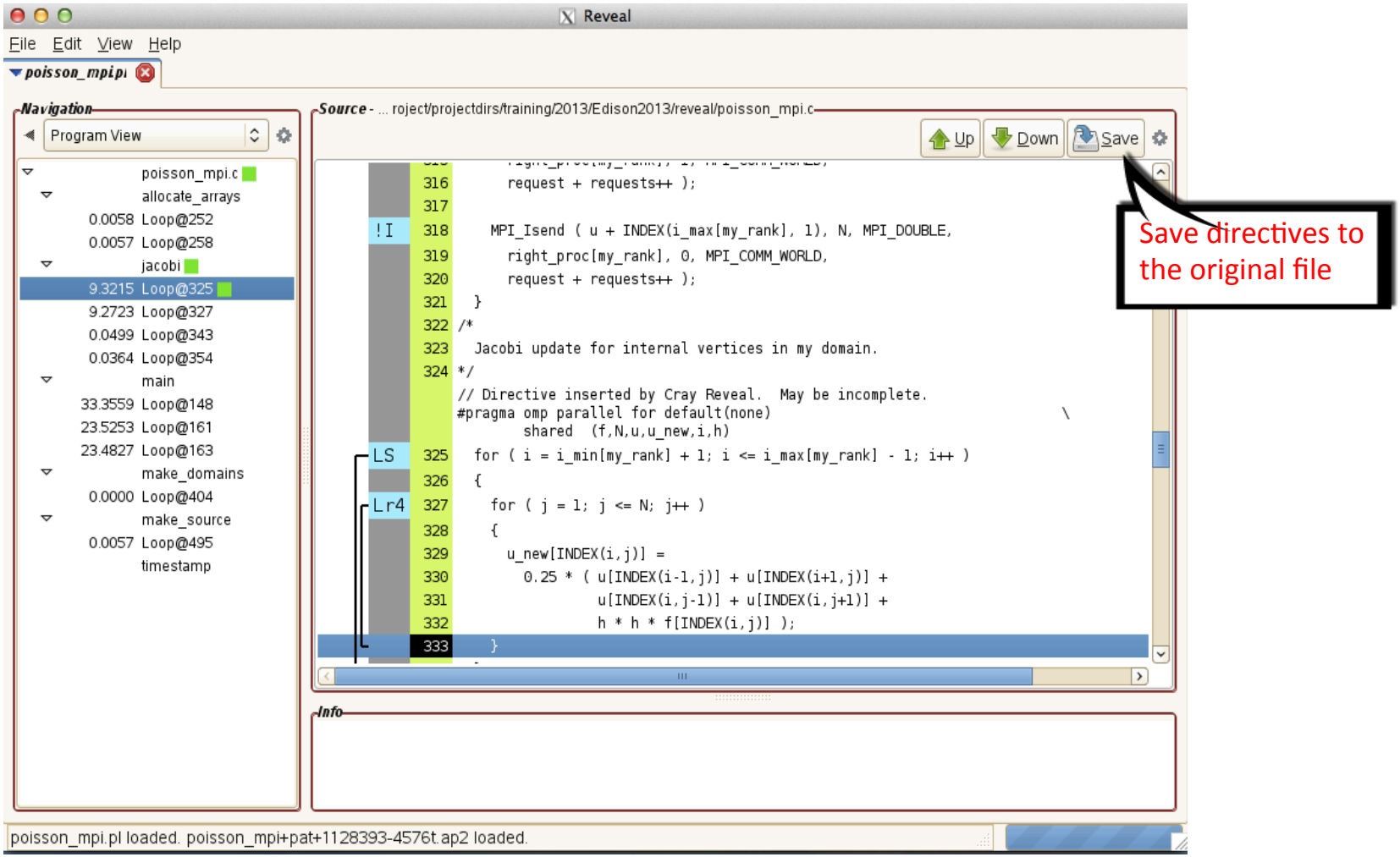
Scoping Results



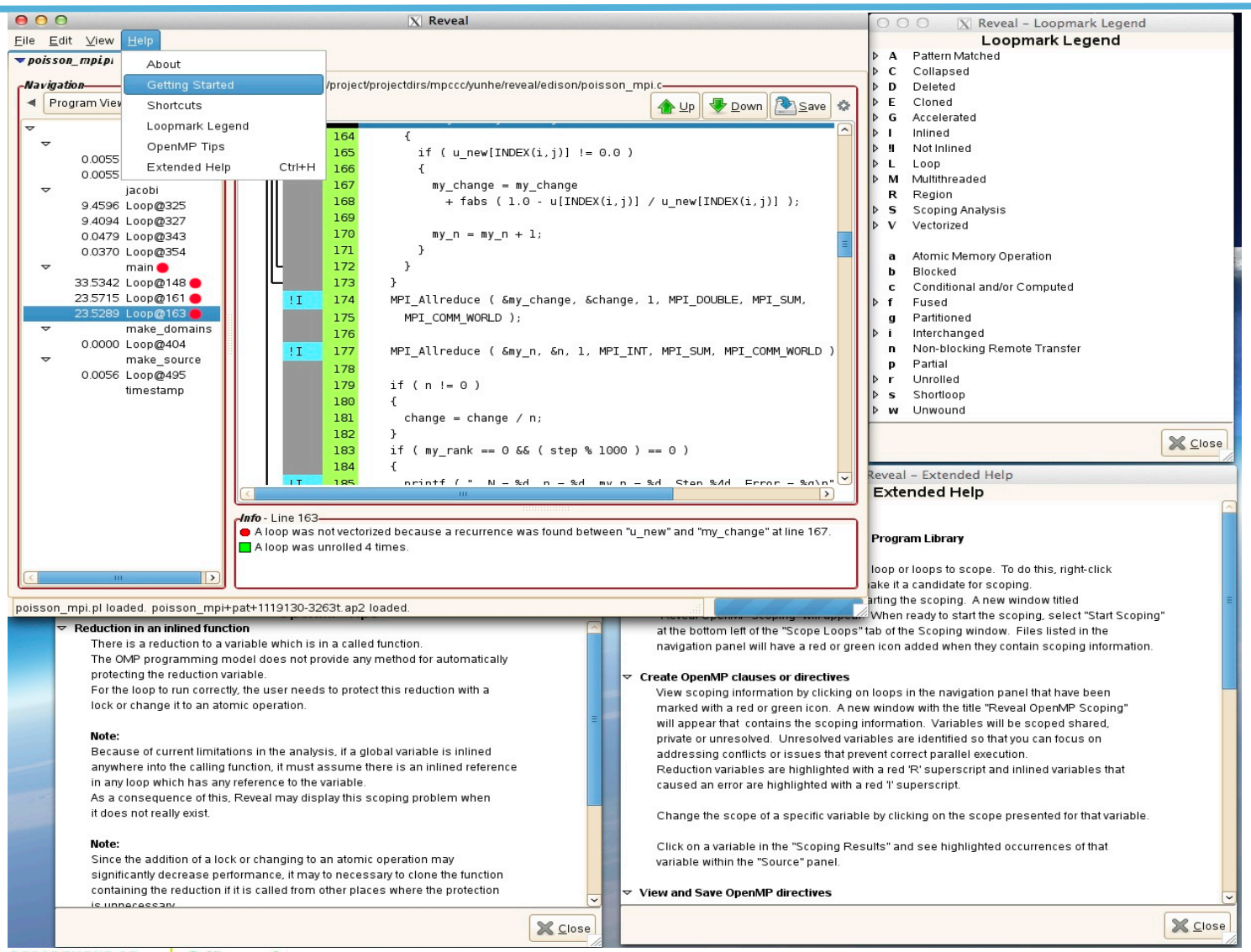
Suggested OpenMP directives



Save the directives



Extensive "Help" topics in Reveal



Reveal helps to start adding OpenMP

- **Only under PrgEnv-cray, with CCE compiler**
- **Start from most time consuming loops first**
- **Insert OpenMP directives**
  - Make sure to save a copy of the original code first, since the saved new file will overwrite the original code
- **There will be unresolved and incomplete variable scopes**
- **There maybe more incomplete and incorrect variables identified when compiling the resulted OpenMP codes**
- **User still needs to understand OpenMP, and resolves the issues.**
- **Verify correctness and performance**
- **Repeat as necessary**
- **No OpenMP tasks, barrier, critical, atomic regions, etc**

More work after reveal (1)

Reveal suggests:

```
#pragma omp parallel for default(none)                \
      unresolved (my_change,my_n)                     \
      shared  (my_rank,N,i_max,u_new,u)               \
      firstprivate (i)

  for ( i = i_min[my_rank]; i <= i_max[my_rank]; i++ )
  {
   for ( j = 1; j <= N; j++ )
   {
    if ( u_new[INDEX(i,j)] != 0.0 )
    {
     my_change = my_change
      + fabs ( 1.0 - u[INDEX(i,j)] / u_new[INDEX(i,j)] );

     my_n = my_n + 1;
    }
   }
  }
```

Final code:
(Lots of changes from Reveal suggestions, but will still make the code slower than without OpenMP directives, so will not use any directives)

```
#pragma omp parallel for default(none)                \
      private (my_change,my_n)                        \
      shared  (my_rank,N,i_min,i_max,u_new,u)         \
      private (j)                                     \
      private (i)

  for ( i = i_min[my_rank]; i <= i_max[my_rank]; i++ )
  {
   for ( j = 1; j <= N; j++ )
   {
    if ( u_new[INDEX(i,j)] != 0.0 )
#pragma omp critical
    {
     my_change = my_change
      + fabs ( 1.0 - u[INDEX(i,j)] / u_new[INDEX(i,j)] );

     my_n = my_n + 1;
    }
   }
  }
```

More work after Reveal (2)

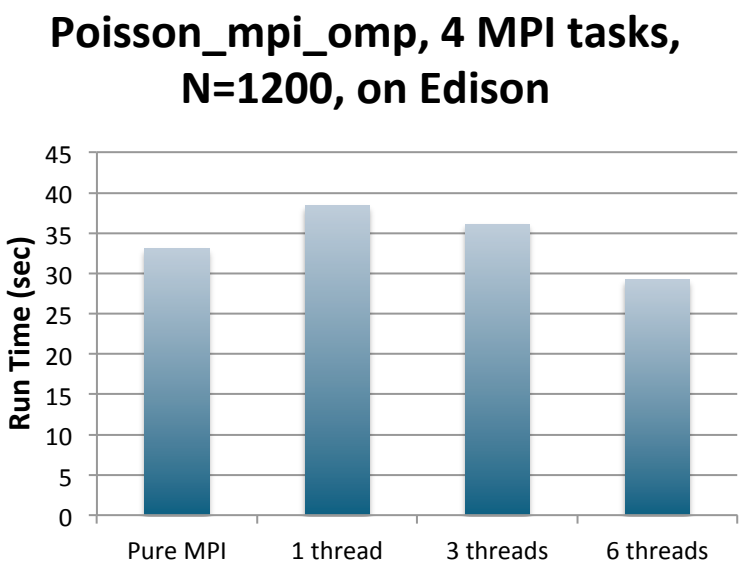
Reveal suggests:

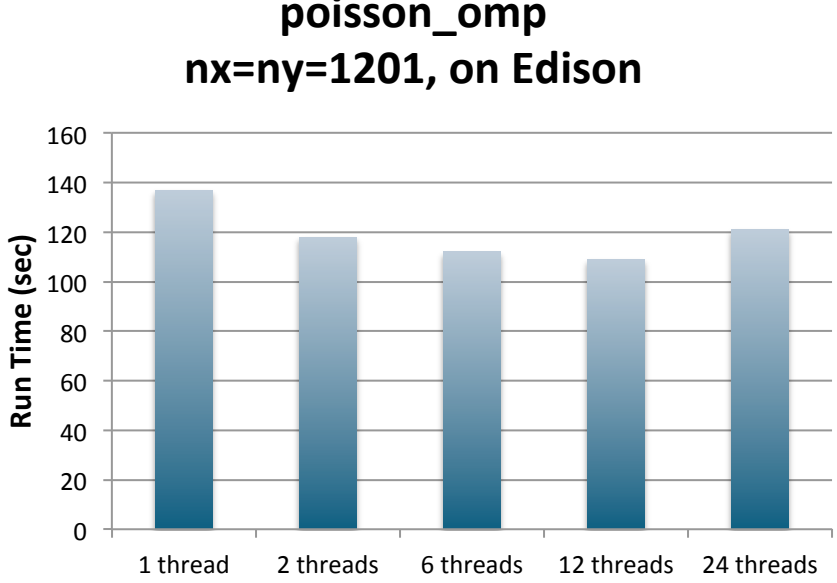
```
#pragma omp parallel for default(none)                \
      shared  (f,N,u,u_new,i,h)

 for ( i = i_min[my_rank] + 1; i <= i_max[my_rank] - 1; i++ )
 {
  for ( j = 1; j <= N; j++ )
  {
   u_new[INDEX(i,j)] =
    0.25 * ( u[INDEX(i-1,j)] + u[INDEX(i+1,j)] +
           u[INDEX(i,j-1)] + u[INDEX(i,j+1)] +
           h * h * f[INDEX(i,j)] );
  }
 }
```

Final code:

```
#pragma omp parallel for default(none)        \
      private (my_rank,j,i)                   \
      shared  (f,N,u,u_new,h,i_min,i_max)

 for ( i = i_min[my_rank] + 1; i <= i_max[my_rank] - 1; i++ )
 {
  for ( j = 1; j <= N; j++ )
  {
   u_new[INDEX(i,j)] =
    0.25 * ( u[INDEX(i-1,j)] + u[INDEX(i+1,j)] +
           u[INDEX(i,j-1)] + u[INDEX(i,j+1)] +
           h * h * f[INDEX(i,j)] );
  }
 }
```

Performance with OpenMP added





More information

- **% module load training**
- **See example codes, reports, detailed steps in README at:**
  - $EXAMPLES/Edison2013/reveal
- **Documentations:**
  - % man reveal (when the “perftools” module is loaded)
  - Using Cray Performance Measurement and Analysis Tools http://docs.cray.com/books/S-2376-612/S-2376-612.pdf

National Energy Research Scientific Computing Center