Enhancements to the Cray Performance Measurement and Analysis Tools

Heidi Poxon
Technical Lead & Manager, Performance Tools
Cray Inc.

Strengths

Provide a complete solution from instrumentation to measurement to analysis to visualization of data

- **Performance measurement and analysis on large systems**
  - Automatic Profiling Analysis
  - Load Imbalance
  - HW counter derived metrics
  - Predefined trace groups provide performance statistics for libraries called by program (blas, lapack, pgas runtime, netcdf, hdf5, etc.)
  - Observations of inefficient performance
  - Data collection and presentation filtering
  - Data correlates to user source (line number info, etc.)
  - Support MPI, SHMEM, OpenMP, UPC, CAF, OpenACC
  - Access to network counters
  - Minimal program perturbation

The Cray Performance Analysis Framework

- **Supports traditional post-mortem performance analysis**
  - Automatic identification of performance problems
    - Indication of causes of problems
    - Suggestions of modifications for performance improvement

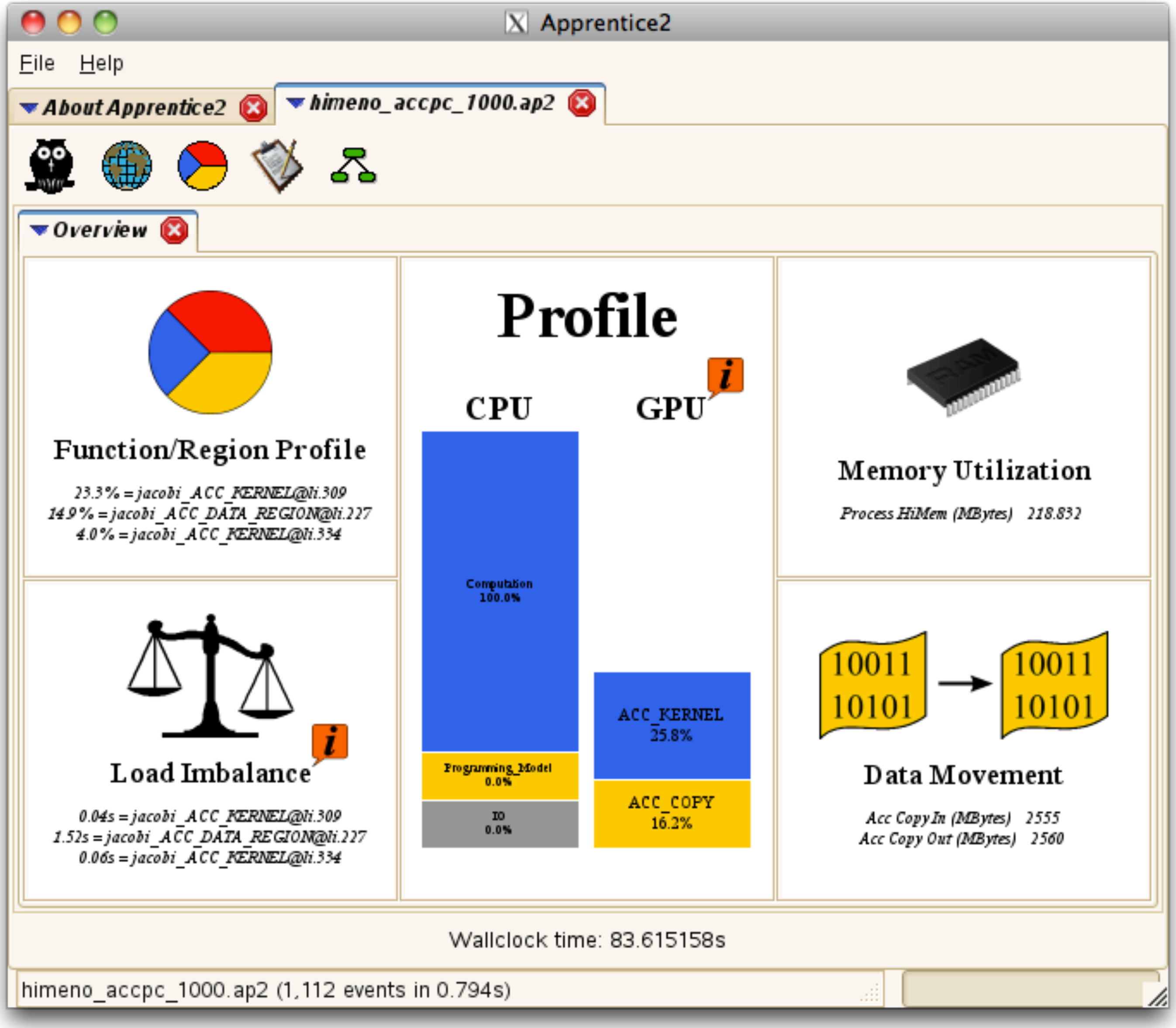
  - pat_build: provides automatic instrumentation
  - CrayPat run-time library collects measurements (transparent to the user)
  - pat_report performs analysis and generates text reports
  - pat_help: online help utility
  - Cray Apprentice2: graphical visualization tool

- **To access software:**
  - module load perftools

Recent Enhancements

- Apprentice2 for the Mac
- Aries™ network counters
- PAPI Cray network component
- Apprentice2 application performance summary
- Reveal 1.0 released
- CrayPat-lite

Application Performance Summary with GPUs



Function/Region Profile

23.3% = jacobi_ACC_KERNEL@li.309
14.9% = jacobi_ACC_DATA_REGION@li.227
4.0% = jacobi_ACC_KERNEL@li.334

Profile

CPU	GPU
Computation 100.0%	ACC_KERNEL 25.8%
Programming_Model 0.0%	ACC_COPY 16.2%
IO 0.0%	

Memory Utilization

Process HiMem (MBytes) 218.832

Load Imbalance

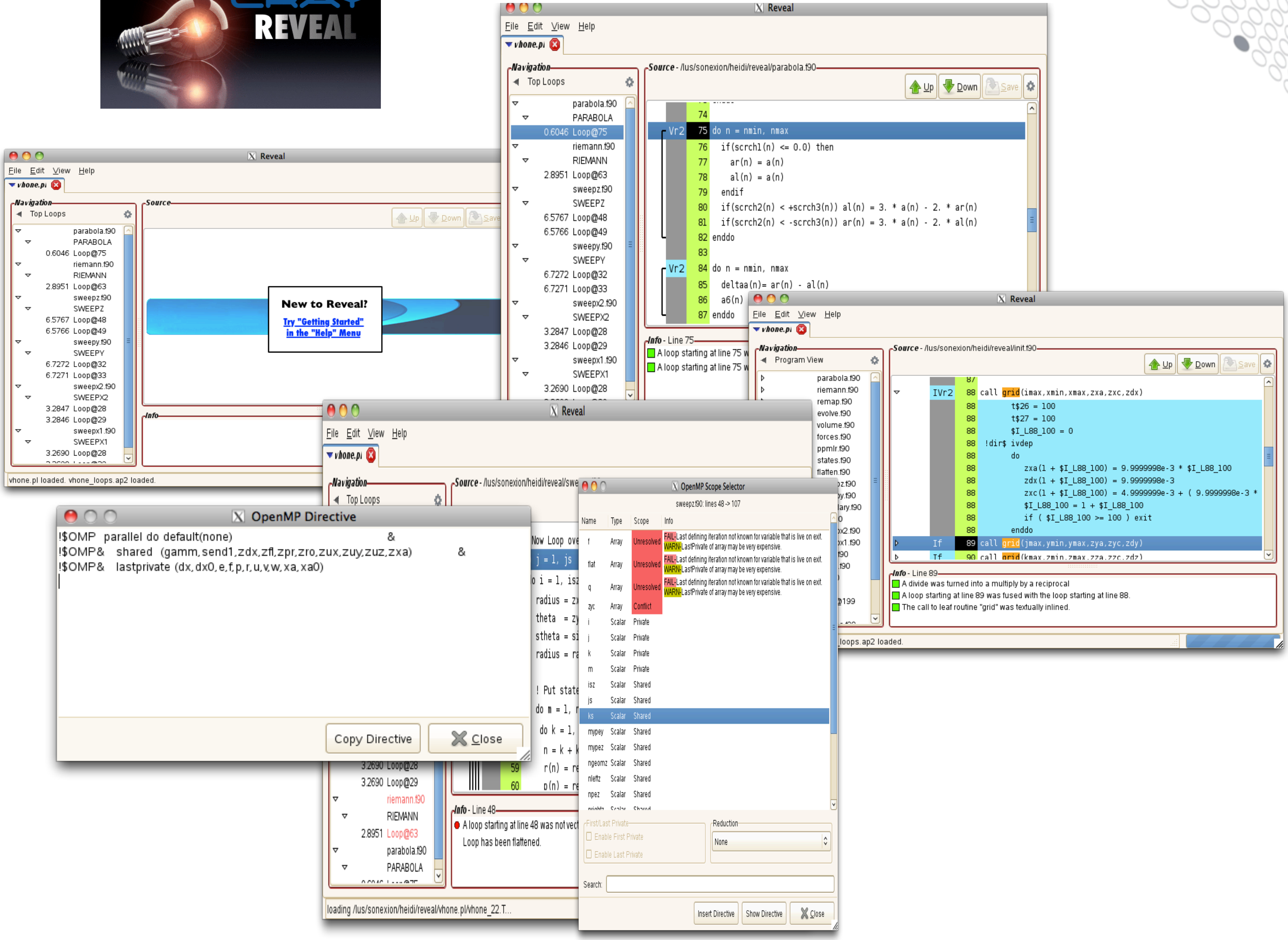
0.04s = jacobi_ACC_KERNEL@li.309
1.52s = jacobi_ACC_DATA_REGION@li.227
0.06s = jacobi_ACC_KERNEL@li.334

Data Movement

Acc Copy In (MBytes) 2555
Acc Copy Out (MBytes) 2560

Wallclock time: 83.615158s

himeno_accpc_1000.ap2 (1,112 events in 0.794s)



```
parabola.f90
  PARABOLA
    0.6046 Loop@75
riemann.f90
  RIEMANN
    2.8951 Loop@63
sweepz.f90
  SWEEPZ
    6.5767 Loop@48
    6.5766 Loop@49
sweepy.f90
  SWEEPY
    6.7272 Loop@32
    6.7271 Loop@33
sweepx2.f90
  SWEEPX2
    3.2847 Loop@28
    3.2846 Loop@29
sweepx1.f90
  SWEEPX1
    3.2690 Loop@28
```

vhone.pl loaded. vhone_loops.ap2 loaded.

Source - /lus/sonexion/heidi/reveal/parabola.f90

```
      74
Vr2   75 do n = nmin, nmax
      76   if(scrch1(n) <= 0.0) then
      77     ar(n) = a(n)
      78     al(n) = a(n)
      79   endif
      80   if(scrch2(n) < +scrch3(n)) al(n) = 3. * a(n) - 2. * ar(n)
      81   if(scrch2(n) < -scrch3(n)) ar(n) = 3. * a(n) - 2. * al(n)
      82 enddo
      83
Vr2   84 do n = nmin, nmax
      85   deltaa(n)= ar(n) - al(n)
      86   a6(n)
      87 enddo
```

Info - Line 75

🟩 A loop starting at line 75 w

🟩 A loop starting at line 75 w

OpenMP Directive

```
!$OMP  parallel do default(none)                        &
!$OMP&   shared  (gamm,send1,zdx,zfl,zpr,zro,zux,zuy,zuz,zxa)        &
!$OMP&   lastprivate (dx,dx0,e,f,p,r,u,v,w,xa,xa0)
```

Source - /lus/sonexion/heidi/reveal/swe

```
Now Loop ove
 j = 1, js
o i = 1, isz
 radius = zx
 theta  = zy
 stheta = si
 radius = ra

! Put state
do m = 1, n
 do k = 1,
  n = k + k
  r(n) = re
  p(n) = re
```

```
3.2690 Loop@28
3.2690 Loop@29
riemann.f90
  RIEMANN
    2.8951 Loop@63
parabola.f90
  PARABOLA
```

Info - Line 48

● A loop starting at line 48 was not vect

Loop has been flattened.

loading /lus/sonexion/heidi/reveal/vhone.pl/vhone_22.T...

OpenMP Scope Selector

sweepz.f90: lines 48 -> 107

Name	Type	Scope	Info
f	Array	Unresolved	FAIL-Last defining iteration not known for variable that is live on exit. WARN-LastPrivate of array may be very expensive.
flat	Array	Unresolved	FAIL-Last defining iteration not known for variable that is live on exit. WARN-LastPrivate of array may be very expensive.
q	Array	Unresolved	FAIL-Last defining iteration not known for variable that is live on exit. WARN-LastPrivate of array may be very expensive.
zyc	Array	Conflict	
i	Scalar	Private	
j	Scalar	Private	
k	Scalar	Private	
m	Scalar	Private	
isz	Scalar	Shared	
js	Scalar	Shared	
ks	Scalar	Shared	
mypey	Scalar	Shared	
mypez	Scalar	Shared	
ngeomz	Scalar	Shared	
nleftz	Scalar	Shared	
npez	Scalar	Shared	

First/Last Private

☐ Enable First Private

☐ Enable Last Private

Reduction: None

Search:

```
parabola.f90
riemann.f90
remap.f90
evolve.f90
volume.f90
forces.f90
ppmlr.f90
states.f90
flatten.f90
```

Source - /lus/sonexion/heidi/reveal/init.f90

```
       87
IVr2   88 call grid(imax,xmin,xmax,zxa,zxc,zdx)
       88       t$26 = 100
       88       t$27 = 100
       88       $I_L88_100 = 0
       88  !dir$ ivdep
       88       do
       88          zxa(1 + $I_L88_100) = 9.9999998e-3 * $I_L88_100
       88          zdx(1 + $I_L88_100) = 9.9999998e-3
       88          zxc(1 + $I_L88_100) = 4.9999999e-3 + ( 9.9999998e-3 *
       88          $I_L88_100 = 1 + $I_L88_100
       88          if ( $I_L88_100 >= 100 ) exit
       88       enddo
If     89 call grid(jmax,ymin,ymax,zya,zyc,zdy)
If     90 call grid(kmax,zmin,zmax,zza,zzc,zdz)
```

Info - Line 89

🟩 A divide was turned into a multiply by a reciprocal

🟩 A loop starting at line 89 was fused with the loop starting at line 88.

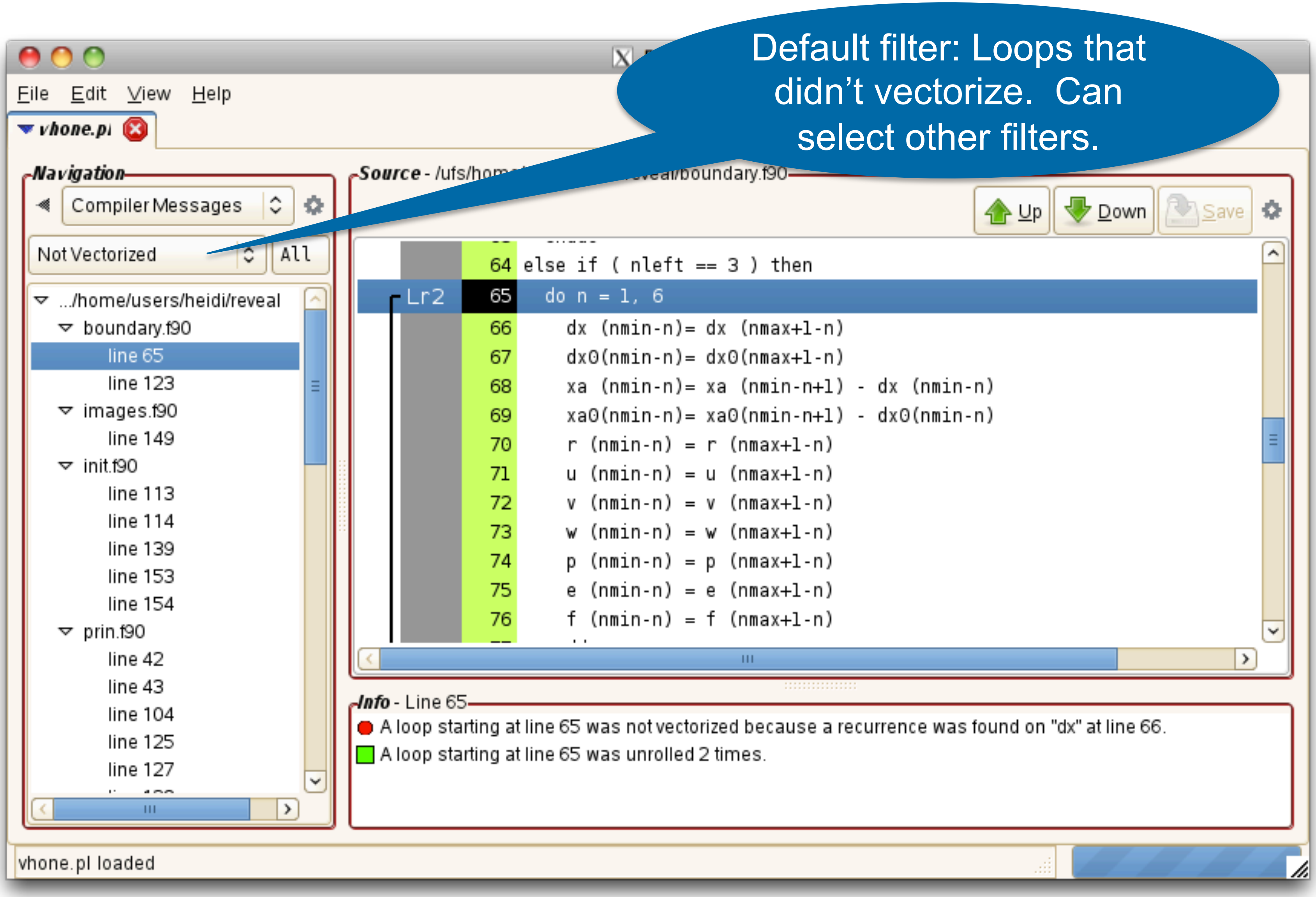
🟩 The call to leaf routine "grid" was textually inlined.

loops.ap2 loaded.

Reveal 1.1 Functionality – March 2013

- Message filtering (allows you to identify all loops that didn’t vectorize or which functions were not inlined, etc.)
- New “Insert All Valid Directives” menu option (which inserts OpenMP directives for all “green” loops scoped successfully)
- Usability enhancements (Example: scoping loop selection and results windows combined into one tabbed window (to reduce the number of additional windows that “pop-up”)

Message Filtering



Default filter: Loops that didn’t vectorize. Can select other filters.

- …/home/users/heidi/reveal
  - boundary.f90
    - line 65
    - line 123
  - images.f90
    - line 149
  - init.f90
    - line 113
    - line 114
    - line 139
    - line 153
    - line 154
  - prin.f90
    - line 42
    - line 43
    - line 104
    - line 125
    - line 127

```
      64 else if ( nleft == 3 ) then
Lr2   65   do n = 1, 6
      66     dx (nmin-n)= dx (nmax+1-n)
      67     dx0(nmin-n)= dx0(nmax+1-n)
      68     xa (nmin-n)= xa (nmin-n+1) - dx (nmin-n)
      69     xa0(nmin-n)= xa0(nmin-n+1) - dx0(nmin-n)
      70     r (nmin-n) = r (nmax+1-n)
      71     u (nmin-n) = u (nmax+1-n)
      72     v (nmin-n) = v (nmax+1-n)
      73     w (nmin-n) = w (nmax+1-n)
      74     p (nmin-n) = p (nmax+1-n)
      75     e (nmin-n) = e (nmax+1-n)
      76     f (nmin-n) = f (nmax+1-n)
```

🟥 A loop starting at line 65 was not vectorized because a recurrence was found on "dx" at line 66.

🟩 A loop starting at line 65 was unrolled 2 times.

vhone.pl loaded

CrayPat-lite

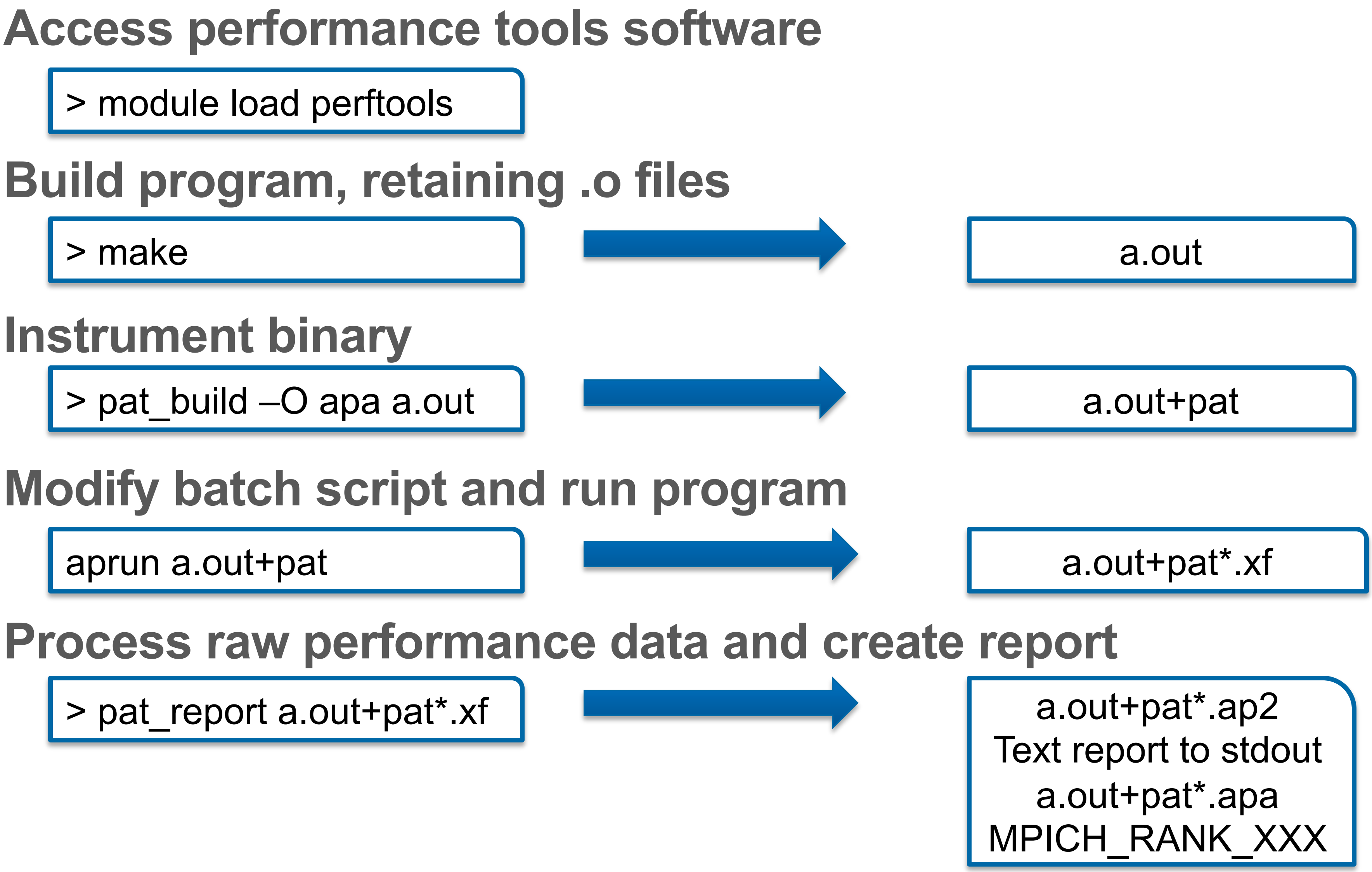
CrayPat-lite Goals

- **Provide automatic application performance statistics at the end of a job**
  - Focus is to offer a simplified interface to basic application performance information for users not familiar with the Cray performance tools and perhaps new to application performance analysis
  - Provides a simple performance summary mechanism for Cray performance tools users before they move on to more detailed analysis with classic perftools
  - Gives sites the option to enable/disable application performance data collection for all users for a period of time

- **Keep traditional or “classic” perftools working the same as before**

- **Provide a simple way to transition from perftools-lite to perftools to encourage further tool use for performance analysis**

Steps to Using CrayPat “classic”



Access performance tools software

```
> module load perftools
```

Build program, retaining .o files

```
> make
```

→ a.out

Instrument binary

```
> pat_build –O apa a.out
```

→ a.out+pat

Modify batch script and run program

```
aprun a.out+pat
```

→ a.out+pat*.xf

Process raw performance data and create report

```
> pat_report a.out+pat*.xf
```

→ a.out+pat*.ap2
Text report to stdout
a.out+pat*.apa
MPICH_RANK_XXX

Steps to Using CrayPat-lite

Access light version of performance tools software

```
> module load perftools-lite
```

Build program

```
> make
```

→ a.out (instrumented program)

Run program (no modification to batch script)

```
aprun a.out
```

→

Condensed report to stdout
a.out*.rpt (same as stdout)
a.out*.ap2
MPICH_RANK_XXX files

Benefits of CrayPat-lite

- Program is automatically relinked to add instrumentation in a.out (pat_build step done for the user)
- .o files are automatically preserved
- No modifications are needed to a batch script to run instrumented binary, since original binary is replaced with instrumented version
- pat_report is automatically run before job exits
- Performance statistics are issued to stdout
- User can use "classic" CrayPat for more in-depth performance investigation

Performance Statistics Available

- **Job information**
  - Number of MPI ranks
  - Number of PEs per node
  - Number of threads
  - Number of cores per socket
  - Execution start time
  - System name and speed

  - Wallclock
  - High memory water mark
  - Aggregate MFLOPS (CPU only)

Performance Statistics Available (2)

- **Profile of top time consuming routines with load balance information by group (user functions, MPI, etc.)**
- **Observations**
  - Currently reporting MPI rank reorder suggestions if applicable
- **Instructions on how to access additional information that is available**

Predefined Set of Performance Experiments

- **Set of predefined experiments, enabled with the CRAYPAT_LITE environment variable**
  - sample_profile
  - event_profile
  - GPU

What do the predefined events mean to someone familiar with the Cray performance tools?

CRAYPAT_LITE=sample_profile

- **Default experiment**
- **Equivalent to "`pat_build -O apa a.out`"**
- **Provides profile based on sampling**
  - Includes collection of summary CPU performance counters around MAIN (for MFLOPS)
  - Includes Imbalance information
- **More information available in .ap2 file**
  - Can get classic report by running pat_report

CRAYPAT_LITE=event_profile

- **Provides profile based on summarization of events**
- **Includes OpenMP and OpenACC information if these models are used within program**
- **Equivalent to "`pat_build -u -gmpi a.out`" +**
  - Collection of summary CPU performance counters
  - Filter to only trace functions above 1200 bytes
    - In most cases, omits tiny repetitive functions that can perturb results (like ranf())
    - Can give coarser granularity results over classic perftools
- **More information available in .ap2 file**

CRAYPAT_LITE=GPU

- **Provides more detailed OpenACC GPU statistics**

- **Equivalent to "`pat_build –w a.out`" (coarsest granularity tracing, around MAIN)**

- **Output similar to classic perftools accelerator table**
  - Includes host and device time
  - Bytes transferred between host and device
  - Time to transfer data between host and device

- **More information available in .ap2 file**

Default Output – Job Summary Info

```
#################################################################
#                                                               #
#            CrayPat-lite Performance Statistics                #
#                                                               #
#################################################################

CrayPat/X:  Version 6.1.0.10929 Revision 10929… 03/04/13 23:51:00
Experiment:                      lite  sample_profile
Number of PEs (MPI ranks):         64
Numbers of PEs per Node:           32  PEs on each of  2  Nodes
Numbers of Threads per PE:          1
Number of Cores per Socket:        16
Execution start time:  Tue Mar  5 18:17:03 2013
System name and speed:  mork 2100 MHz


Wall Clock Time:    75.432429 secs
High Memory:            43.96 MBytes
MFLOPS (aggregate):  25718.61 M/sec
```

Default Output – Condensed Profile

```
Table 1:  Profile by Function Group and Function (top 7 functions shown)

 Samp%   |  Samp   | Imb.   |  Imb.   |Group
         |         | Samp   | Samp%   | Function
         |         |        |         |  PE=HIDE

 100.0% | 7422.6 |     -- |     -- |Total
|-----------------------------------------------------------------------
|  88.7% | 6585.5 |     -- |     -- |USER
||----------------------------------------------------------------------
||  71.7% | 5325.1 | 111.9 |   2.1% |LAMMPS_NS::PairLJCut::compute
||   8.9% |  659.2 |  17.8 |   2.7% |LAMMPS_NS::Neighbor::half_bin_newton
||   3.1% |  227.7 |  67.3 |  23.2% |LAMMPS_NS::FixNVE::initial_integrate
||   1.7% |  124.9 |  27.1 |  18.1% |LAMMPS_NS::FixNVE::final_integrate
||   1.5% |  112.0 |  16.0 |  12.7% |LAMMPS_NS::Verlet::run
||======================================================================
|  11.1% |  825.0 |     -- |     -- |MPI
||----------------------------------------------------------------------
||   7.2% |  534.0 | 199.0 |  27.6% |MPI_Send
||   3.0% |  226.1 | 135.9 |  38.1% |MPI_Wait|
=======================================================================
```

Default Output – For More Information…

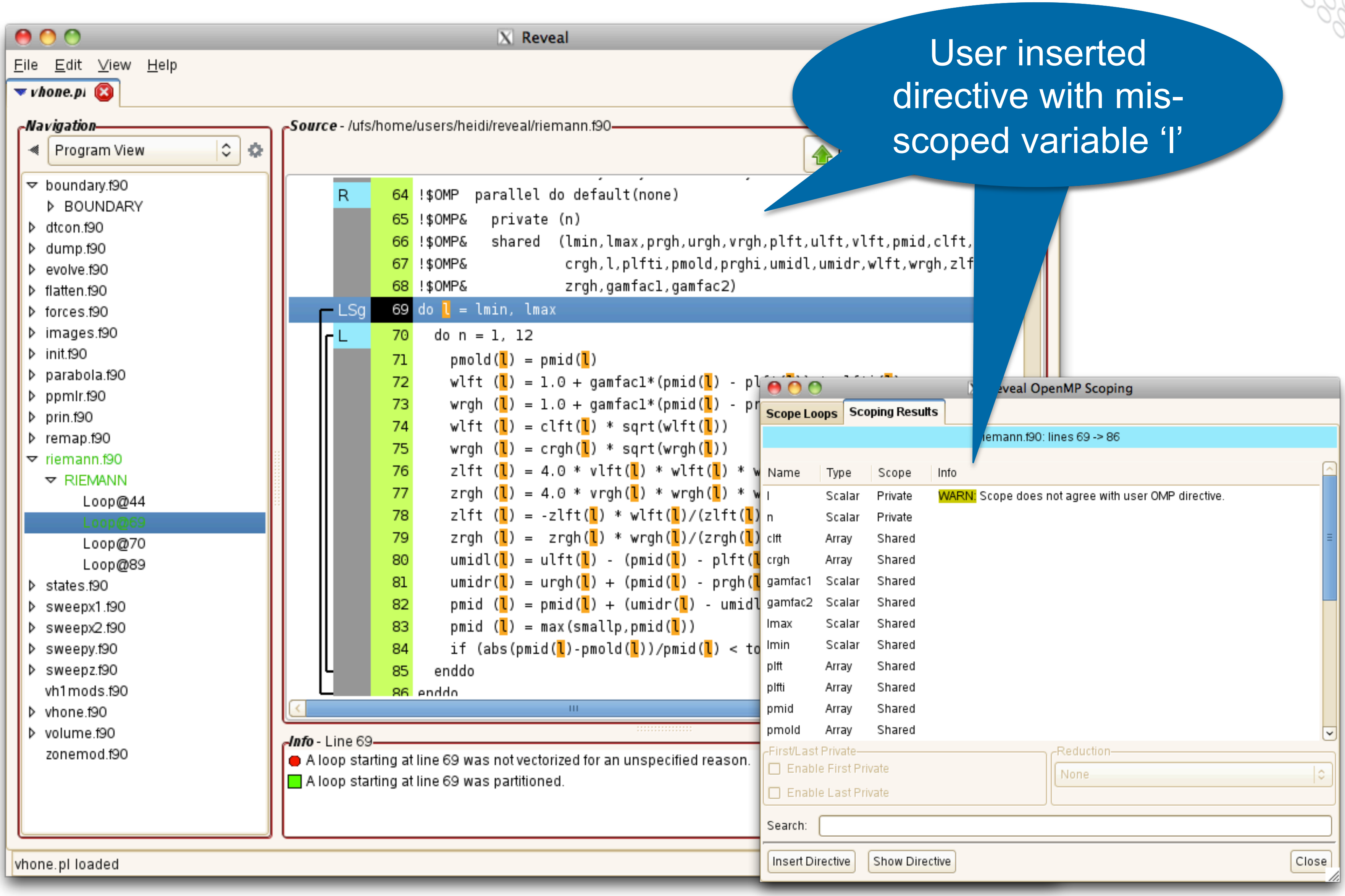
```
Program invocation:
  lammps.x -var x 4 -var y 2 -var z 8

For more detailed performance reports, run:
  pat_report /lus/scratch/test/lammps.x.lj.64pe.32ppn.ap2

For interactive performance analysis, run:
  app2 /lus/scratch/test/lammps.x.lj.64pe.32ppn.ap2

End of CrayPat output.
```

Event Profile Output - Observations

```
================  Observations and suggestions  ========================

MPI Grid Detection:

There appears to be point-to-point MPI communication in a 4 X 2 X 8 grid
    pattern. The execution time spent in MPI functions might be reduced
    with a rank order that maximizes communication between ranks on the
    same node. The effect of several rank orders is estimated below.

    A file named MPICH_RANK_ORDER.Grid was generated along with this
    report and contains usage instructions and the Hilbert rank order
    from the following table.

         Rank     On-Node      On-Node   MPICH_RANK_REORDER_METHOD
        Order    Bytes/PE    Bytes/PE%
                              of Total
                              Bytes/PE

       Hilbert  5.533e+10       90.66%   3
          Fold  4.907e+10       80.42%   2
           SMP  4.883e+10       80.02%   1
    RoundRobin  3.740e+10       61.28%   0
```

What’s Next…

Reveal OpenMP Directive Validation



Source - /ufs/home/users/heidi/reveal/riemann.f90

```
64 !$OMP  parallel do default(none)
65 !$OMP&   private (n)
66 !$OMP&   shared  (lmin,lmax,prgh,urgh,vrgh,plft,ulft,vlft,pmid,clft,
67 !$OMP&             crgh,l,plfti,pmold,prghi,umidl,umidr,wlft,wrgh,zlf
68 !$OMP&             zrgh,gamfacl,gamfac2)
69 do l = lmin, lmax
70   do n = 1, 12
71     pmold(l) = pmid(l)
72     wlft  (l) = 1.0 + gamfacl*(pmid(l) - pl
73     wrgh  (l) = 1.0 + gamfacl*(pmid(l) - pr
74     wlft  (l) = clft(l) * sqrt(wlft(l))
75     wrgh  (l) = crgh(l) * sqrt(wrgh(l))
76     zlft  (l) = 4.0 * vlft(l) * wlft(l) * w
77     zrgh  (l) = 4.0 * vrgh(l) * wrgh(l) * w
78     zlft  (l) = -zlft(l) * wlft(l)/(zlft(l)
79     zrgh  (l) =  zrgh(l) * wrgh(l)/(zrgh(l)
80     umidl(l) = ulft(l) - (pmid(l) - plft(l
81     umidr(l) = urgh(l) + (pmid(l) - prgh(l
82     pmid  (l) = pmid(l) + (umidr(l) - umidl
83     pmid  (l) = max(smallp,pmid(l))
84     if (abs(pmid(l)-pmold(l))/pmid(l) < to
85   enddo
86 enddo
```

Info - Line 69

🟥 A loop starting at line 69 was not vectorized for an unspecified reason.

🟩 A loop starting at line 69 was partitioned.

vhone.pl loaded

iemann.f90: lines 69 -> 86

Name	Type	Scope	Info
l	Scalar	Private	WARN: Scope does not agree with user OMP directive.
n	Scalar	Private	
clft	Array	Shared	
crgh	Array	Shared	
gamfac1	Scalar	Shared	
gamfac2	Scalar	Shared	
lmax	Scalar	Shared	
lmin	Scalar	Shared	
plft	Array	Shared	
plfti	Array	Shared	
pmid	Array	Shared	
pmold	Array	Shared	

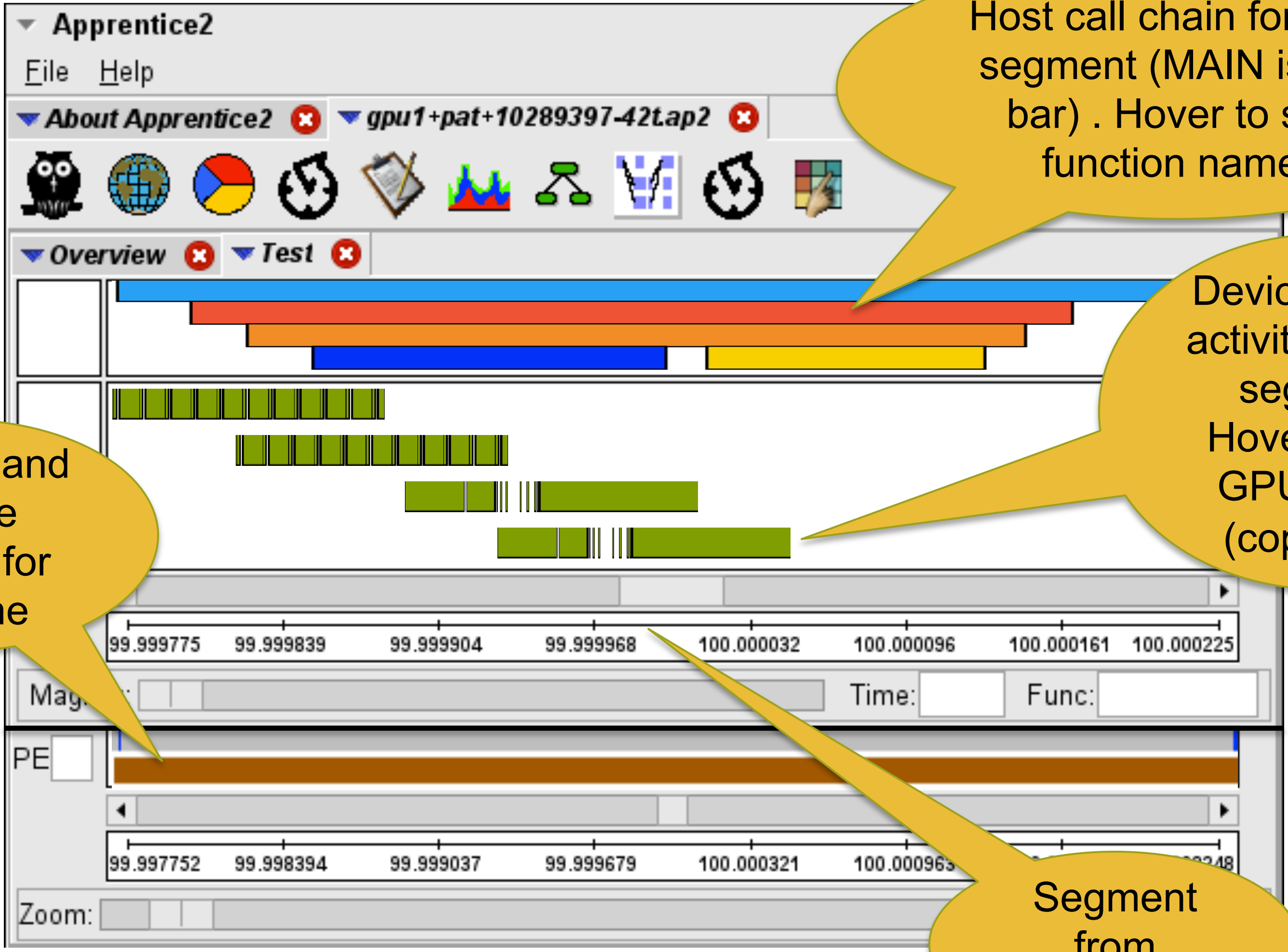
First/Last Private
□ Enable First Private
□ Enable Last Private

Reduction: None

Search:

User inserted directive with mis-scoped variable ‘l’

GPU Timeline



Host call chain for time segment (MAIN is top bar) . Hover to see function name.

Device stream activity for time segment. Hover to see GPU event (copy, etc.)

PE host and device activity for timeline

Segment from timeline

Questions ?