

Performance Measurement and Analysis Tools for the Cray XK System

Heidi Poxon
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

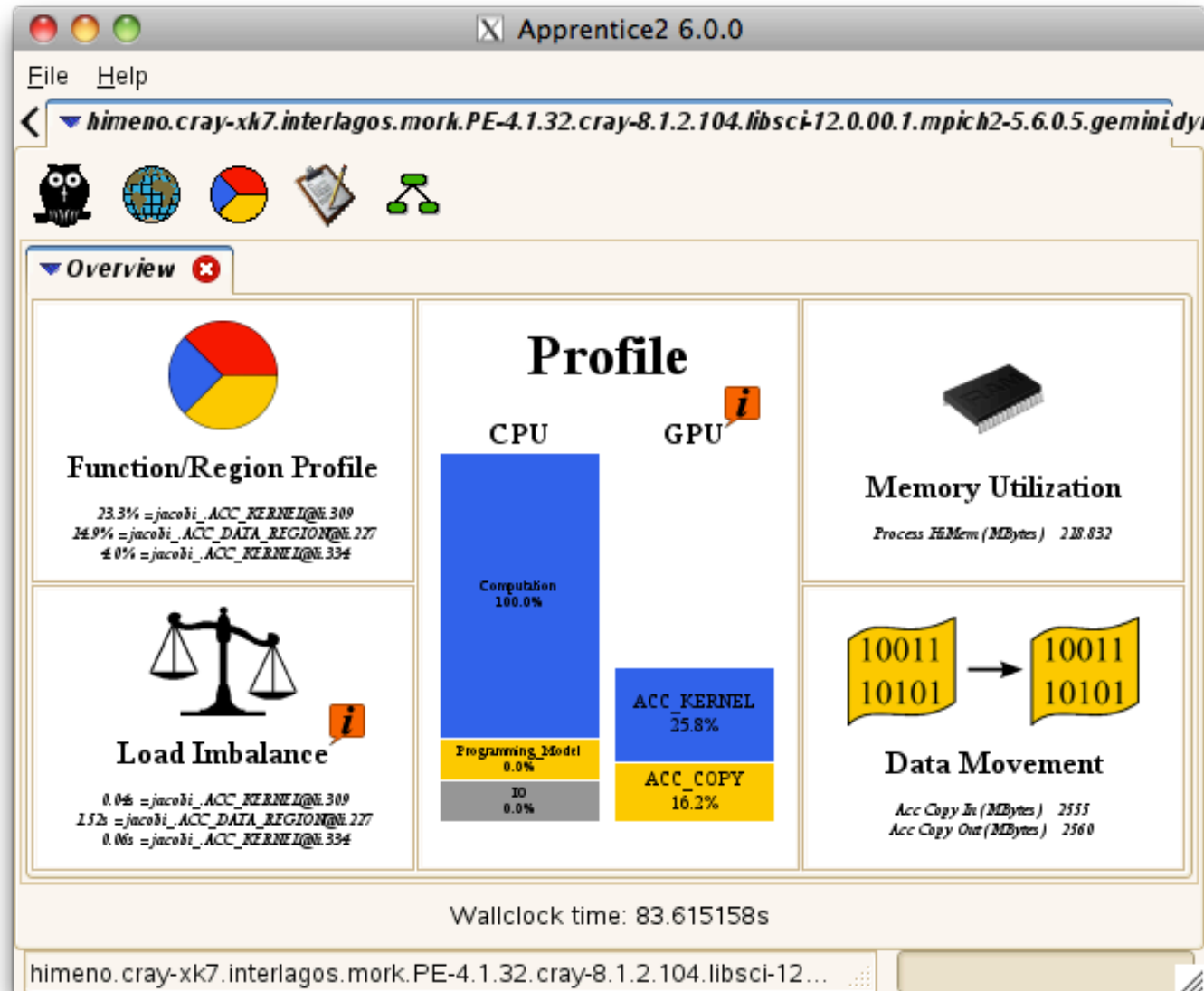
- **Supports traditional post-mortem performance analysis**
 - Automatic identification of performance problems
 - Indication of causes of problems
 - Suggestions of modifications for performance improvement
 - `pat_build`: provides automatic instrumentation
 - `CrayPat` run-time library collects measurements (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



Where to Run Instrumented Application

- By default, data files are written to the execution directory
- Default behavior requires file system that supports record locking, such as Lustre (/mnt/snx3/... , /lus/..., /scratch/...,etc.)
 - Can use `PAT_RT_EXPFIL` to point to existing directory that resides on a high-performance file system if not execution directory
- Number of files used to store raw data
 - 1 file created for program with 1 – 256 processes
 - \sqrt{n} files created for program with 257 – n processes
 - Ability to customize with `PAT_RT_EXPFIL_MAX`
- See [intro_craypat\(1\)](#) man page

Apprentice2 Overview





Sampling with Line Number information

heidi@limited: /h/heidi — ssh — 81×26

Table 2: Profile by Group, Function, and Line

Samp%	Samp	Imb.	Imb.	Group
		Samp	Samp%	Function
				Source
				Line
				PE=HIDE
100.0%	8376.9	--	--	Total
93.2%	7804.0	--	--	USER
51.7%	4328.7	--	--	calc3_
31				heidi/DARPA/cache_util/calc3.do300-ijswap.F
4	15.7%	1314.4	93.6	6.8% line.78
4	13.9%	1167.7	98.3	7.9% line.79
4	14.5%	1211.6	97.4	7.6% line.80
4	1.2%	103.1	26.9	21.2% line.93
4	1.1%	88.4	22.6	20.8% line.94
4	1.0%	84.5	17.5	17.6% line.95
4	1.0%	86.8	33.2	28.2% line.96
4	1.3%	105.0	23.0	18.4% line.97
4	1.4%	116.5	24.5	17.7% line.98
	=====			
				144,1 38%

MPI Messages By Caller

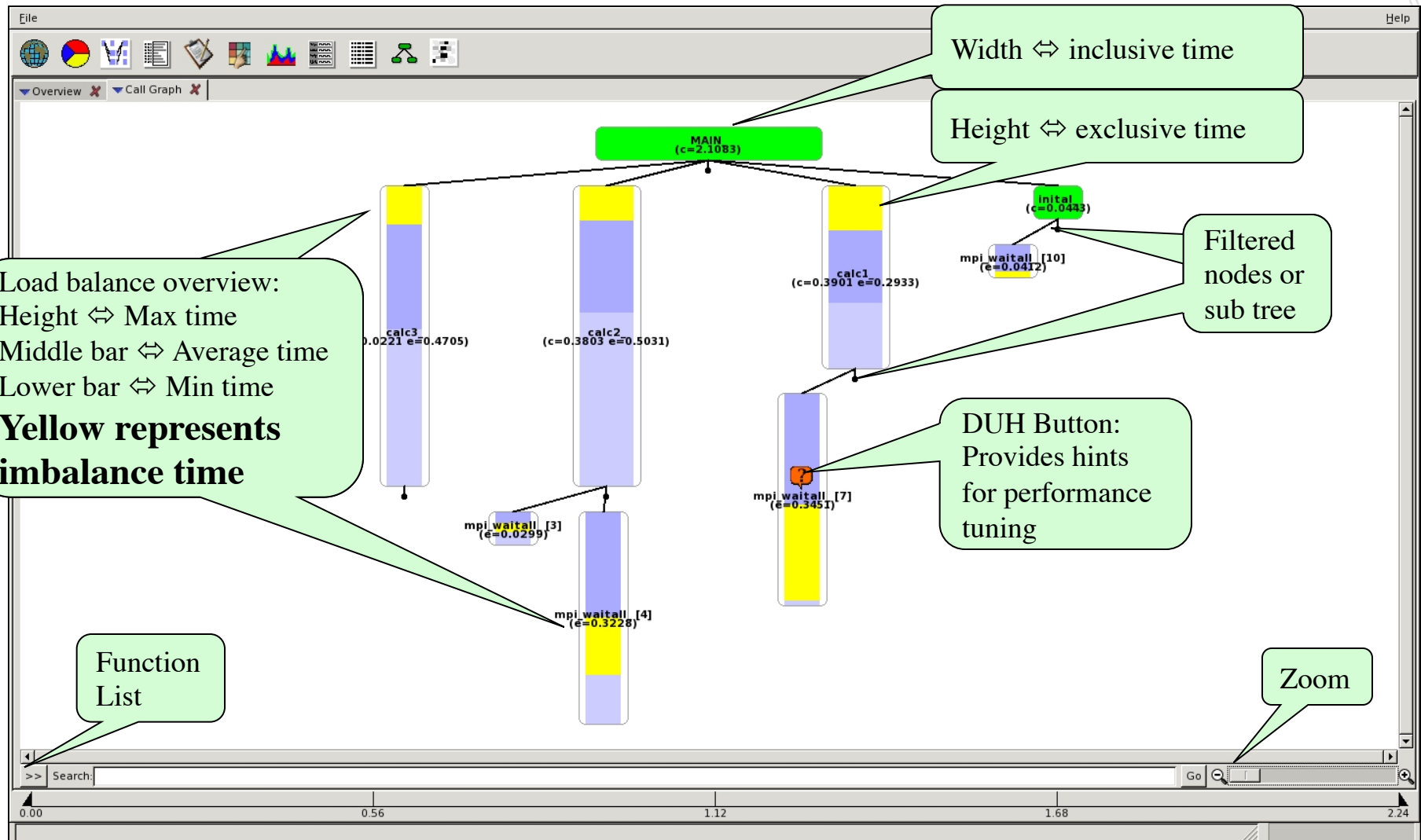
heidi@limited: /h/heidi — ssh — 81×26

Table 4: MPI Message Stats by Caller

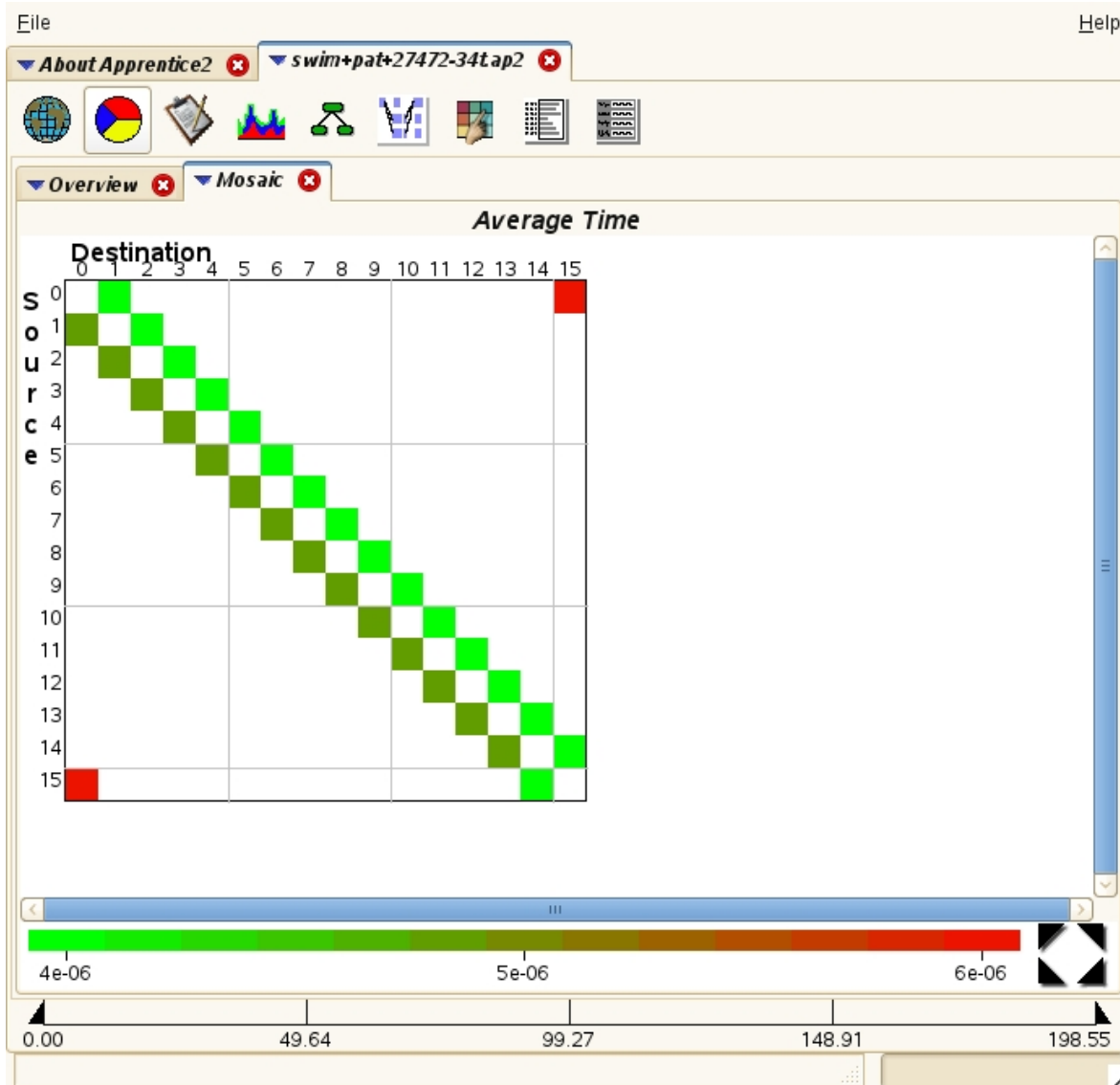
MPI Msg Bytes	MPI Msg Count	MsgSz <16B Count	4KB<= MsgSz <64KB Count	Function Caller PE=[mmm]
140166953.8	8890.6	339.8	8550.8	Total
140166833.8	8875.6	324.8	8550.8	MPI_ISEND
78272400.0	4850.0	75.0	4775.0	calc2_ shallow_
78700800.0	7200.0	2400.0	4800.0	lpe.0
78681600.0	4800.0	0.0	4800.0	lpe.1
59020800.0	4800.0	1200.0	3600.0	lpe.47
59421800.0	3725.0	100.0	3625.0	calc1_ shallow_
78700800.0	7200.0	2400.0	4800.0	lpe.0
59011200.0	3600.0	0.0	3600.0	lpe.1
59011200.0	3600.0	0.0	3600.0	lpe.24

624,3 79%

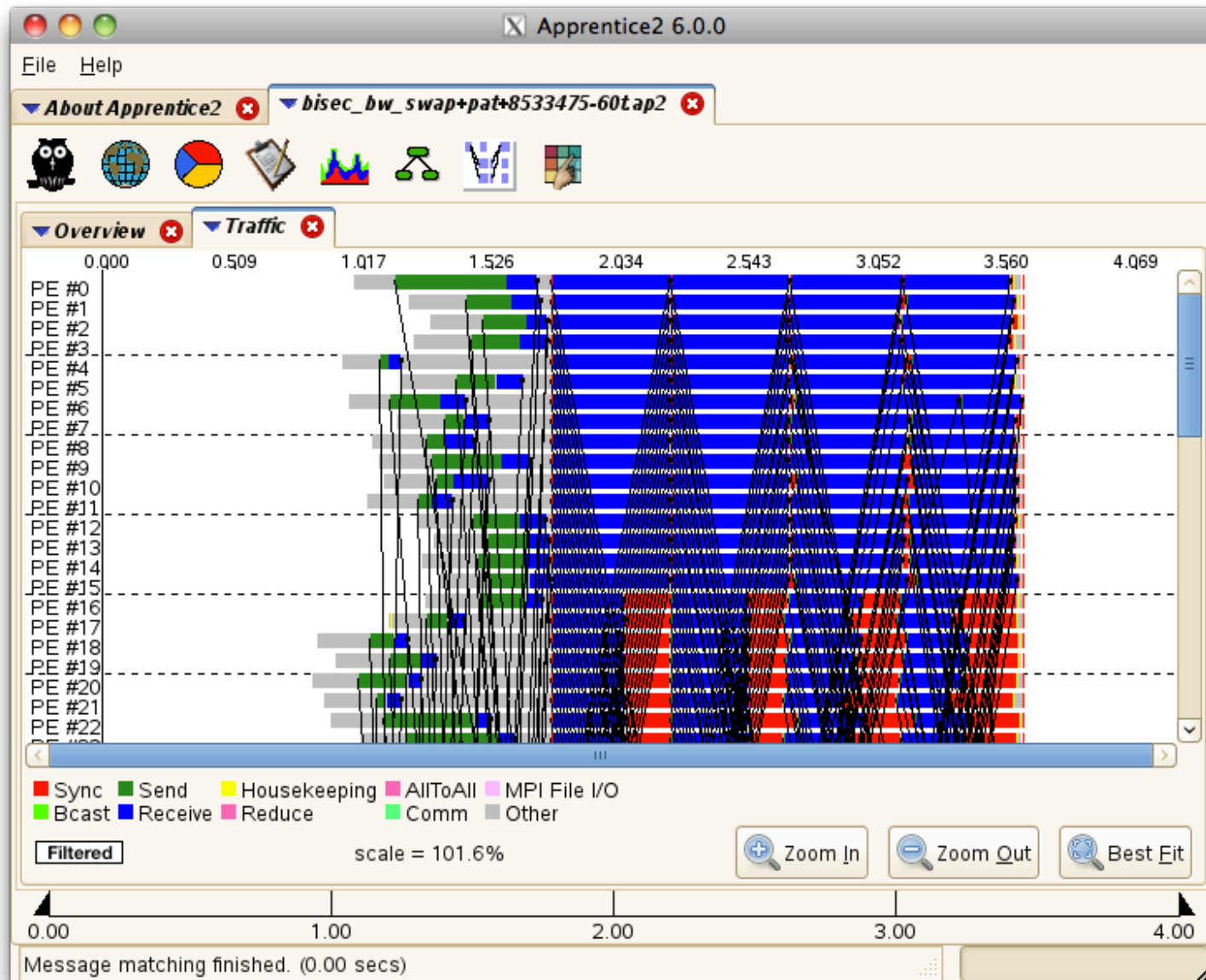
Call Tree View



Mosaic View – Shows Communication Pattern



Traffic Report – MPI Communication Timeline



Porting to a Hybrid or Many-core System

A Porting and Optimization Strategy for Hybrid and Many-core Systems

- Maximize on-node communication between MPI ranks
- Relieve on-node shared resource contention by pairing threads or processes that perform different work (for example computation with off-node communication) on the same node
- Add parallelism to MPI ranks to take advantage of cores within a node while minimizing network injection contention
- Accelerate work intensive parallel loops

Grid Detection and Rank Reordering Suggestions



Automatic Communication Grid Detection

- **Analyze runtime performance data to identify grids in a program to maximize on-node communication**
 - Example: nearest neighbor exchange in 2 dimensions
 - Sweep3d uses a 2-D grid for communication
- **Determine whether or not a custom MPI rank order will produce a significant performance benefit**
- **Grid detection is helpful for programs with significant point-to-point communication**
- **Doesn't interfere with MPI collective communication optimizations**



Automatic Communication Grid Detection (2)

- Tools produce a custom rank order if it's beneficial based on grid size, grid order and cost metric
- **Heuristics available for:**
 - MPI sent message statistics
 - User time (time spent in user functions) – can be used for PGAS codes
 - Hybrid of sent message and user time)
- **Summarized findings in report**
- **Available with sampling or tracing**
- **Describe how to re-run with custom rank order**

MPI Rank Order Observations

Table 1: Profile by Function Group and Function

Time%	Time	Imb. Time	Imb. Time%	Calls	Group Function PE=HIDE
100.0%	463.147240	--	--	21621.0	Total
52.0%	240.974379	--	--	21523.0	MPI
47.7%	221.142266	36.214468	14.1%	10740.0	mpi_recv
4.3%	19.829001	25.849906	56.7%	10740.0	MPI_SEND
43.3%	200.474690	--	--	32.0	USER
41.0%	189.897060	58.716197	23.6%	12.0	sweep_
1.6%	7.579876	1.899097	20.1%	12.0	source_
4.7%	21.698147	--	--	39.0	MPI_SYNC
4.3%	20.091165	20.005424	99.6%	32.0	mpi_allreduce_(sync)
0.0%	0.000024	--	--	27.0	SYSCALL



MPI Rank Order Observations (2)

MPI Grid Detection:

There appears to be point-to-point MPI communication in a **96 X 8 grid pattern**. The **52% of the total 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 Custom rank order from the following table.

Rank Order	On-Node Bytes/PE	On-Node Bytes/PE% of Total Bytes/PE	MPICH_RANK_REORDER_METHOD
Custom	2.385e+09	95.55%	3
SMP	1.880e+09	75.30%	1
Fold	1.373e+06	0.06%	2
RoundRobin	0.000e+00	0.00%	0



MPICH_RANK_ORDER File

```
# The 'Custom' rank order in this file targets nodes with multi-core
# processors, based on Sent Msg Total Bytes collected for:
#
# Program:    /lus/nid00030/heidi/sweep3d/mod/sweep3d.mpi
# Ap2 File:   sweep3d.mpi+pat+27054-89t.ap2
# Number PEs: 48
# Max PEs/Node: 4
#
# To use this file, make a copy named MPICH_RANK_ORDER, and set the
# environment variable MPICH_RANK_REORDER_METHOD to 3 prior to
# executing the program.
#
# The following table lists rank order alternatives and the grid_order
# command-line options that can be used to generate a new order.
...
```

Auto-Generated MPI Rank Order File

```
# The
'USER_Time_hybrid'
rank_order in this
file targets nodes
with multi-core
# processors, based on
Sent Msg Total Bytes
collected for:
#
# Program: /lus/ 86,396,30,428,62,460,5 18,514,74,586,58,626,8 252,505,140,425,212,45 352,301,320,325,288,35
nid00023/malice/ 4,492,118,420,22,452,9 2,546,106,634,90,578,1 7,156,385,172,417,180, 7,328,304,360,312,376,
craypat/WORKSHOP/bh2o- 4,388,126,484 14,618,122,610 449,148,489,220,481 293,296,368,336,344
demo/Rank/sweep3d/src/ 129,563,193,531,161,57 135,315,167,339,199,34 131,534,195,542,163,56 258,338,266,346,282,31
sweep3d 1,225,539,241,595,233, 7,259,307,231,371,239, 6,227,526,235,574,203, 4,274,370,766,306,710,
# Ap2 File: 523,249,603,185,555 379,191,331,247,299 598,243,558,187,606 378,742,330,678,362
sweep3d.gmpi-u.ap2 153,587,169,627,137,63 175,363,159,323,143,35 251,590,211,630,179,63 646,298,750,322,718,35
# Number PEs: 768 5,201,619,177,515,145, 5,255,291,207,275,183, 8,139,622,155,550,171, 4,758,290,734,662,686,
# Max PEs/Node: 16 579,209,547,217,611 283,151,267,215,223 518,219,582,147,614 670,726,702,694,654
#
# To use this file,
make a copy named
MPICH_RANK_ORDER, and
set the
# environment variable
MPICH_RANK_REORDER_MET
HOD to 3 prior to
# executing the
program.
#
0,532,64,564,32,572,96 128,533,192,541,160,56 4,535,36,543,68,567,10 762,659,738,651,706,66
,540,8,596,72,524,40,6 5,232,525,224,573,240, 0,527,12,599,44,575,28 7,746,643,714,691,674,
04,24,588 597,184,557,248,605 ,559,76,607 699,754,683,730,723
104,556,16,628,80,636, 168,589,200,517,152,62 52,591,20,631,60,639,8 722,731,763,658,642,75
56,620,48,516,112,580, 9,136,549,176,637,144, 4,519,108,623,92,551,1 5,739,675,707,650,682,
88,548,120,612 621,208,581,216,613 16,583,124,615 715,698,666,690,747
```



Approach to Adding Parallelism

1. Identify possible accelerator kernels

- Determine where to add additional levels of parallelism
 - Assumes MPI application is functioning correctly on X86
 - Find top serial work-intensive loops (perftools + CCE loop work estimates)

2. Perform parallel analysis, scoping and vectorization

- Split loop work among threads
 - Do parallel analysis and restructuring on targeted high level loops
 - Use CCE loopmark feedback, Reveal loopmark and source browsing

3. Move to OpenMP and then to OpenACC

- Add parallel directives and acceleration extensions
 - Insert OpenMP directives (Reveal scoping assistance)
 - Run on X86 to verify application and check for performance improvements
 - Convert desired OpenMP directives to OpenACC

4. Analyze performance from optimizations

Step 1 - Identify possible accelerator kernels

Loop Work Estimates

- **Helps identify high-level serial loops to parallelize**
 - Based on runtime analysis, approximates how much work exists within a loop
 - Provides min, max and average trip counts that can be used to approximate work and help carve up loop on GPU

Collecting Loop Statistics

- Load PrgEnv-cray module
- Load perftools module
- Compile **AND** link with `-h profile_generate`
- Instrument binary for tracing
 - `pat_build -w my_program`
- Run application
- Create report with loop statistics
 - `pat_report my_program.xf > loops_report`

Example Report – Inclusive Loop Time

Table 2: Loop Stats by Function (from -hprofile_generate)

Loop Incl Time Total	Loop Hit	Loop Trips Avg	Loop Trips Min	Loop Trips Max	Function=/.LOOP[.] PE=HIDE
8.995914	100	25	0	25	sweepy_.LOOP.1.li.33
8.995604	2500	25	0	25	sweepy_.LOOP.2.li.34
8.894750	50	25	0	25	sweepz_.LOOP.05.li.49
8.894637	1250	25	0	25	sweepz_.LOOP.06.li.50
4.420629	50	25	0	25	sweepx2_.LOOP.1.li.29
4.420536	1250	25	0	25	sweepx2_.LOOP.2.li.30
4.387534	50	25	0	25	sweepx1_.LOOP.1.li.29
4.387457	1250	25	0	25	sweepx1_.LOOP.2.li.30
2.523214	187500	107	0	107	riemann_.LOOP.2.li.63
1.541299	20062500	12	0	12	riemann_.LOOP.3.li.64
0.863656	1687500	104	0	108	parabola_.LOOP.6.li.67



**Step 2 - Perform parallel
analysis, scoping and
vectorization**

&

**Step 3 - Move to
OpenMP and then to
OpenACC**



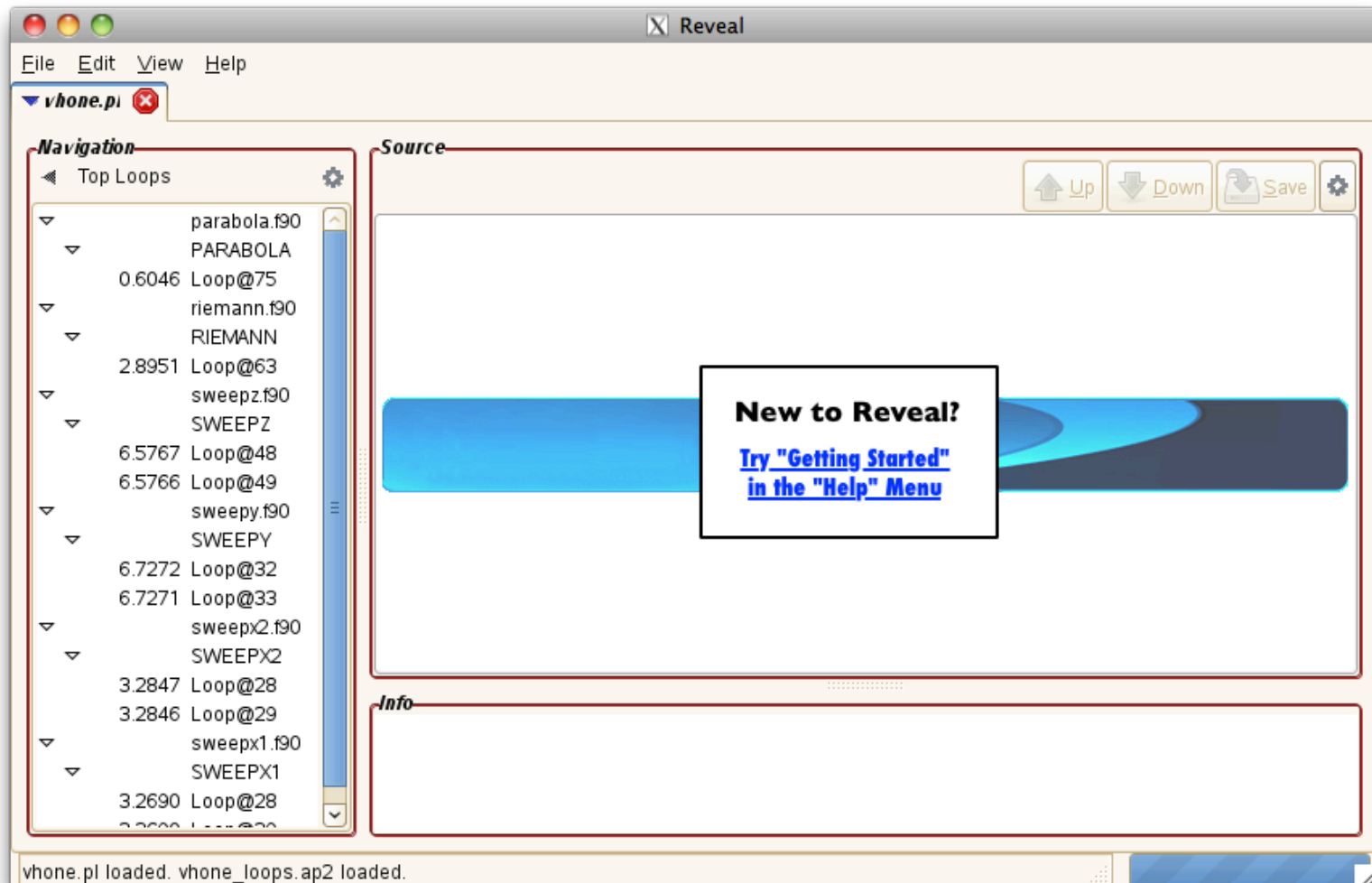


Reveal

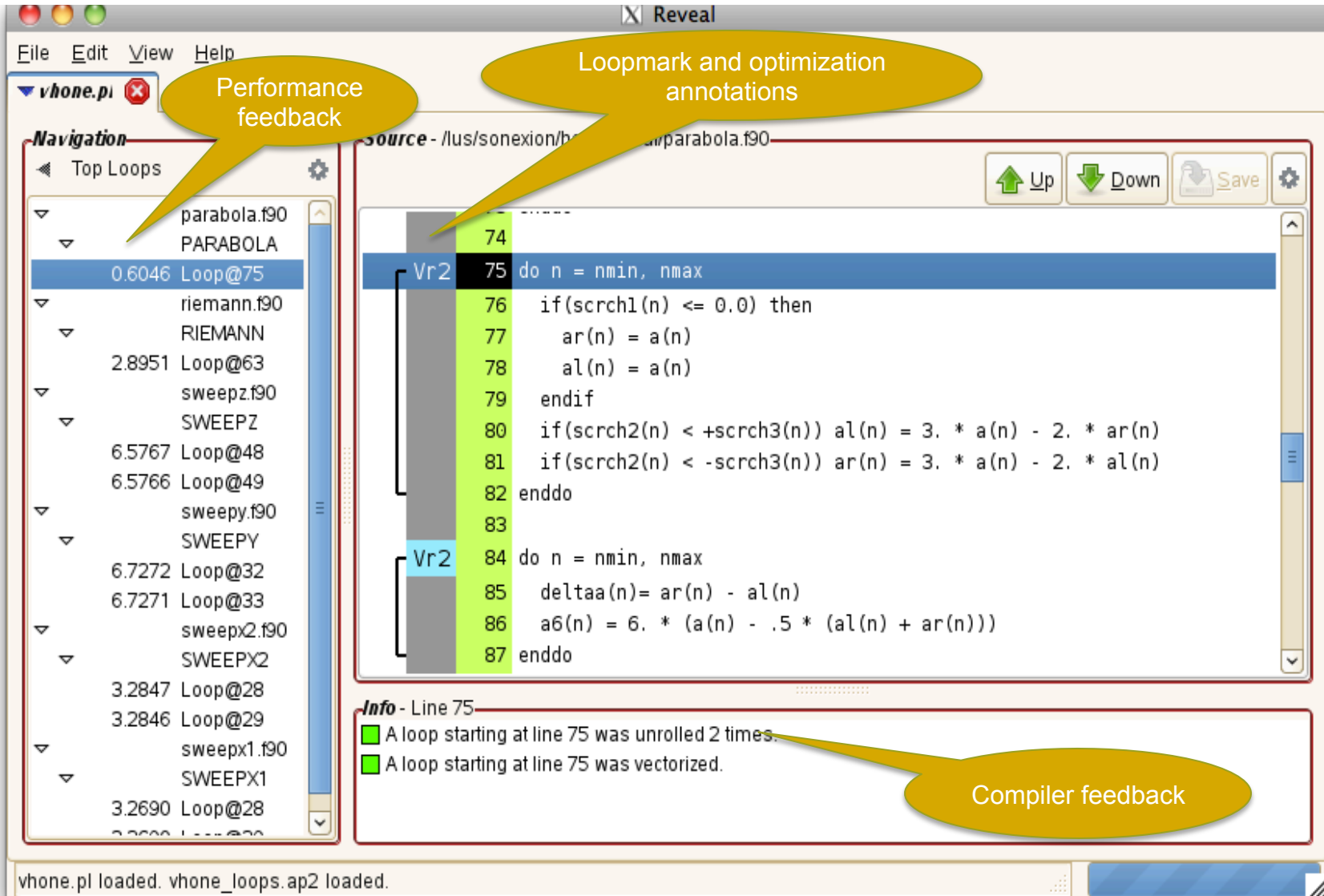
New code analysis and restructuring assistant...

- Uses both the performance toolset and CCE's program library functionality to provide static and runtime analysis information
- **Key Features**
 - **Annotated source code** with compiler optimization information
 - Feedback on critical dependencies that prevent optimizations
 - **Scoping analysis**
 - Identify, shared, private and ambiguous arrays
 - Allow user to privatize ambiguous arrays
 - Allow user to override dependency analysis
 - **Source code navigation** based on performance data collected through CrayPat

Reveal with Loop Work Estimates



Visualize Loopmark with Performance Information



The screenshot shows the 'Reveal' tool interface with the following components:

- Navigation Panel (Left):** A tree view of loops. The '0.6046 Loop@75' is selected, which corresponds to the 'PARABOLA' loop in the 'parabola.f90' file.
- Source Code (Center):** The source code for 'parabola.f90' is displayed. Line 75 is highlighted in blue, indicating the start of the selected loop. The code includes a 'do' loop (lines 75-82) and another 'do' loop (lines 84-87). A bracket on the left side of the code indicates the loop structure.
- Performance Feedback (Top Left):** A yellow callout bubble points to the '0.6046 Loop@75' in the navigation panel, indicating the performance feedback for this loop.
- Loopmark and Optimization Annotations (Top Center):** A yellow callout bubble points to the 'Vr2' label next to line 75, indicating loopmark and optimization annotations.
- Compiler Feedback (Bottom Right):** A yellow callout bubble points to the 'Info' section, which contains the following feedback:
 - A loop starting at line 75 was unrolled 2 times.
 - A loop starting at line 75 was vectorized.

The status bar at the bottom indicates: 'vhone.pl loaded. vhone_loops.ap2 loaded.'

Visualize CCE's Loopmark with Performance Profile (2)



The screenshot shows the 'Reveal' window with the 'vhone.pl' file loaded. The 'Navigation' pane on the left shows a tree view of the file, with '0.53% SWEEPX2' expanded. The 'Source' pane shows the code for 'sweep2.f90', with lines 32-45 highlighted. The 'Info' pane at the bottom shows a message: 'A loop starting at line 33 was not vectorized because it does not have a constant stride'. A yellow callout bubble points to this message with the text 'Integrated message 'explain support''. The 'Explain' window on the right shows the 'OPT_INFO' message: 'A loop starting at line %s was unrolled.' It explains that the compiler unrolled the loop, creating a number of copies of the loop body. It also contrasts unroll-and-jam with literal outer loop unrolling, showing code examples for both. The 'Explain' window has buttons for 'Explain other message...' and 'Close'.

Navigation

- Program View
- riemann.f90
- remap.f90
- evolve.f90
- volume.f90
- forces.f90
- ppmlr.f90
- states.f90
- flatten.f90
- sweepz.f90
- sweepy.f90
- boundary.f90
- prin.f90
- sweepx2.f90
- 0.53% SWEEPX2
- Loop@28
- Loop@29
- Loop@32
- Loop@33
- Loop@44
- Loop@58
- sweepx1.f90

Source - /lus/sonexion/heidi/reveal/sweep2.f90

```
32 do m = 1, npey
33 do i = 1, isy
34   n = i + isy*(m-1) + 6
35   r(n) = recv2(1,k,i,j,m)
36   p(n) = recv2(2,k,i,j,m)
37   u(n) = recv2(3,k,i,j,m)
38   v(n) = recv2(4,k,i,j,m)
39   w(n) = recv2(5,k,i,j,m)
40   f(n) = recv2(6,k,i,j,m)
41 enddo
42 enddo
43
44 do i = 1,imax
45   n = i + 6
```

Info - Line 33

- A loop starting at line 33 was not vectorized because it does not have a constant stride
- A loop starting at line 33 was unrolled 8 times

OPT_INFO: A loop starting at line %s was unrolled.

The compiler unrolled the loop. Unrolling creates a number of copies of the loop body. When unrolling an outer loop, the compiler attempts to fuse replicated inner loops - a transformation known as unroll-and-jam. The compiler will always employ the unroll-and-jam mode when unrolling an outer loop; literal outer loop unrolling may occur when unrolling to satisfy a user directive (pragma).

This message indicates that unroll-and-jam was performed with respect to the identified loop. A different message is issued when literal outer loop unrolling is performed, as this transformation is far less likely to be beneficial.

For sake of illustration, the following contrasts unroll-and-jam with literal outer loop unrolling.

```
# 426 "/tmp/ulib/buildslaves/pdgc8-81-edition-build/tbs/build/release/pdgc8/pdgc8_ftn.msg.c"
DO J = 1,10
DO I = 1,100
A(I,J) = B(I,J) + 42.0
ENDDO
ENDDO

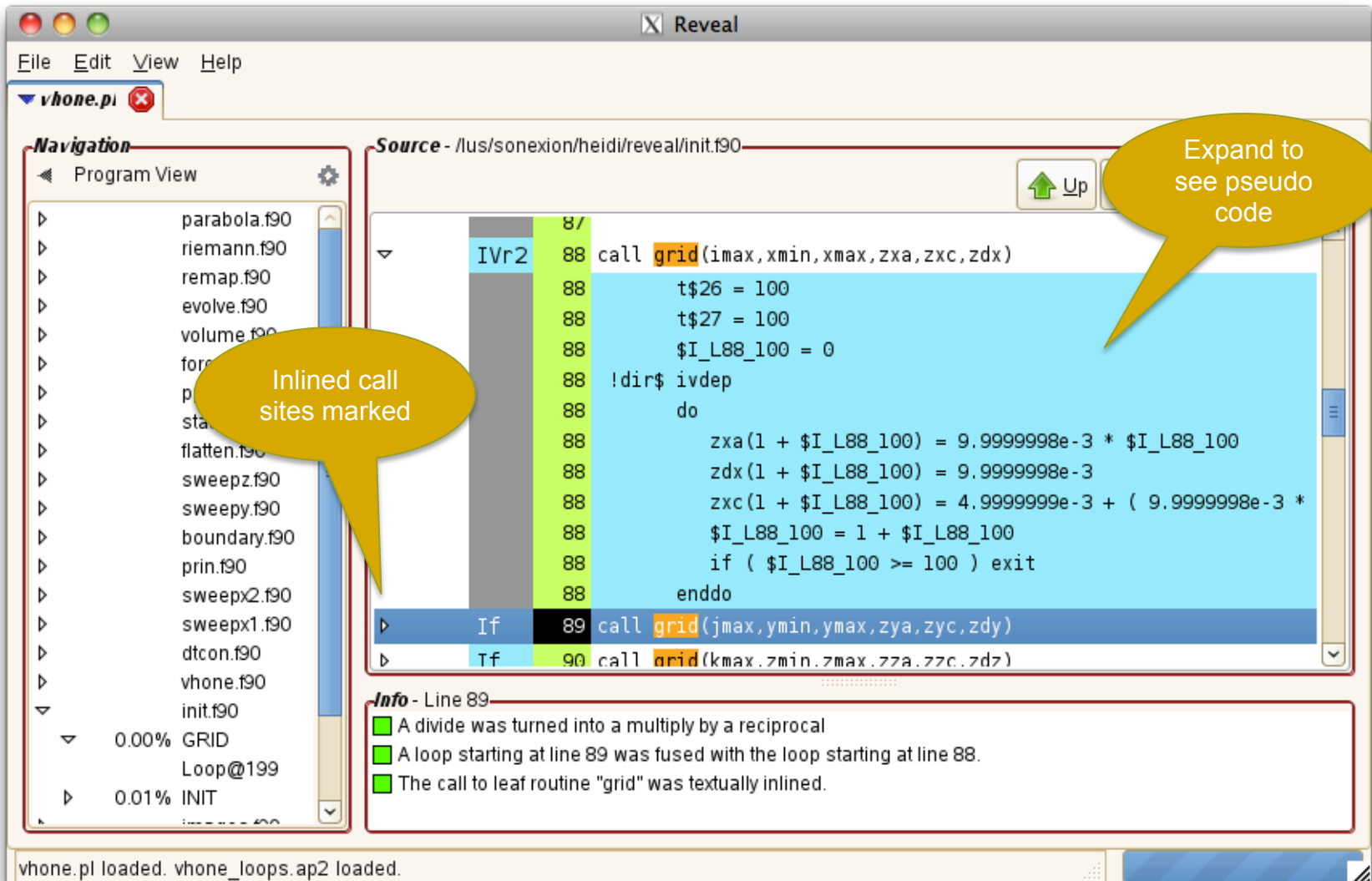
DO J = 1,10,2
DO I = 1,100
A(I,J) = B(I,J) + 42.0 ! unroll-and-jam
A(I,J+1) = B(I,J+1) + 42.0
ENDDO
ENDDO

DO J = 1,10,2
DO I = 1,100
A(I,J) = B(I,J) + 42.0 ! literal outer unroll
ENDDO
DO I = 1,100
A(I,J+1) = B(I,J+1) + 42.0
ENDDO
ENDDO
```

The literal outer unroll code performs the same sequence of memory operations as the original nest, while the unroll-and-jam transformation interleaves operations from outer loop iterations. The compiler employs literal outerloop unrolling only when the data dependencies in the loop, or a control flow impediment, prevent fusion of the replicated inner loops. Literal outer loop unrolling is generally not desirable. It is provided to ensure expected behavior and for those rare instances where the user has determined that it is beneficial.

Explain other message... Close

View Pseudo Code for Inlined Functions



The screenshot shows the Cray Reveal IDE interface. The top menu bar includes File, Edit, View, and Help. The main window is titled "Source - /lus/sonexion/heidi/reveal/init.f90". The left sidebar shows a "Navigation" pane with a "Program View" tree listing various files like parabola.f90, riemann.f90, remap.f90, evolve.f90, volume.f90, for, p, sta, flatten.f90, sweepz.f90, sweepy.f90, boundary.f90, prin.f90, sweepx2.f90, sweepx1.f90, dtcon.f90, vhone.f90, and init.f90. The main editor displays the source code for init.f90, showing a loop starting at line 88. The code includes calls to the grid function, variable assignments, and a conditional exit. A yellow callout bubble points to the "grid" function calls, stating "Inlined call sites marked". Another yellow callout bubble points to the "grid" function call, stating "Expand to see pseudo code". The bottom status bar shows "vhone.pl loaded. vhone_loops.ap2 loaded."

Navigation

Program View

- parabola.f90
- riemann.f90
- remap.f90
- evolve.f90
- volume.f90
- for
- p
- sta
- flatten.f90
- sweepz.f90
- sweepy.f90
- boundary.f90
- prin.f90
- sweepx2.f90
- sweepx1.f90
- dtcon.f90
- vhone.f90
- init.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)
Tf    90 call grid(kmax,zmin,zmax,zza,zzc,zdz)

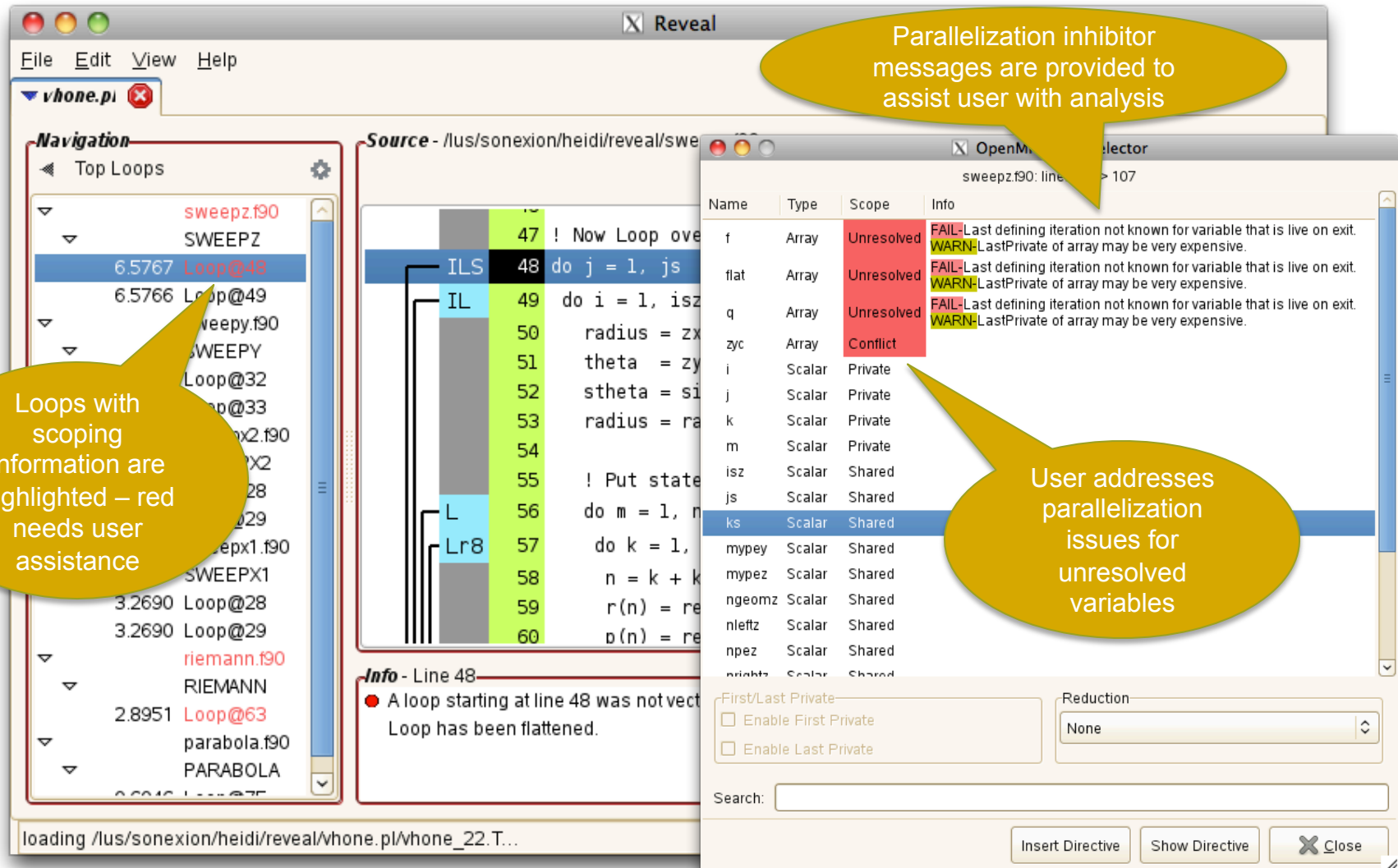
```

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.

vhone.pl loaded. vhone_loops.ap2 loaded.

Scoping Assistance – Review Scoping Results



Navigation Pane: Lists loops and code blocks. Loops with scoping information are highlighted in red, indicating they need user assistance.

- sweepz.f90
 - SWEEPZ
 - 6.5767 Loop@48 (highlighted in red)
 - 6.5766 Loop@49
 - sweepy.f90
 - SWEEPY
 - Loop@32
 - Loop@33
 - Loop@28
 - Loop@29
 - riemann.f90
 - RIEMANN
 - 2.8951 Loop@63 (highlighted in red)
 - parabola.f90
 - PARABOLA

Source Editor: Displays the Fortran code. Line 48 is highlighted in blue, indicating a loop that was not vectorized and has been flattened.

```

47 ! Now Loop over
48 do j = 1, js
49 do i = 1, isz
50   radius = zxc
51   theta = zyc
52   stheta = sin(theta)
53   radius = radius * stheta
54
55 ! Put state
56 do m = 1, n
57   do k = 1, m
58     n = k + k
59     r(n) = radius
60     p(n) = radius
  
```

Open Message Selector: A dialog box showing a list of messages related to the selected loop. The messages are categorized by type (Array, Scalar) and scope (Private, Shared).

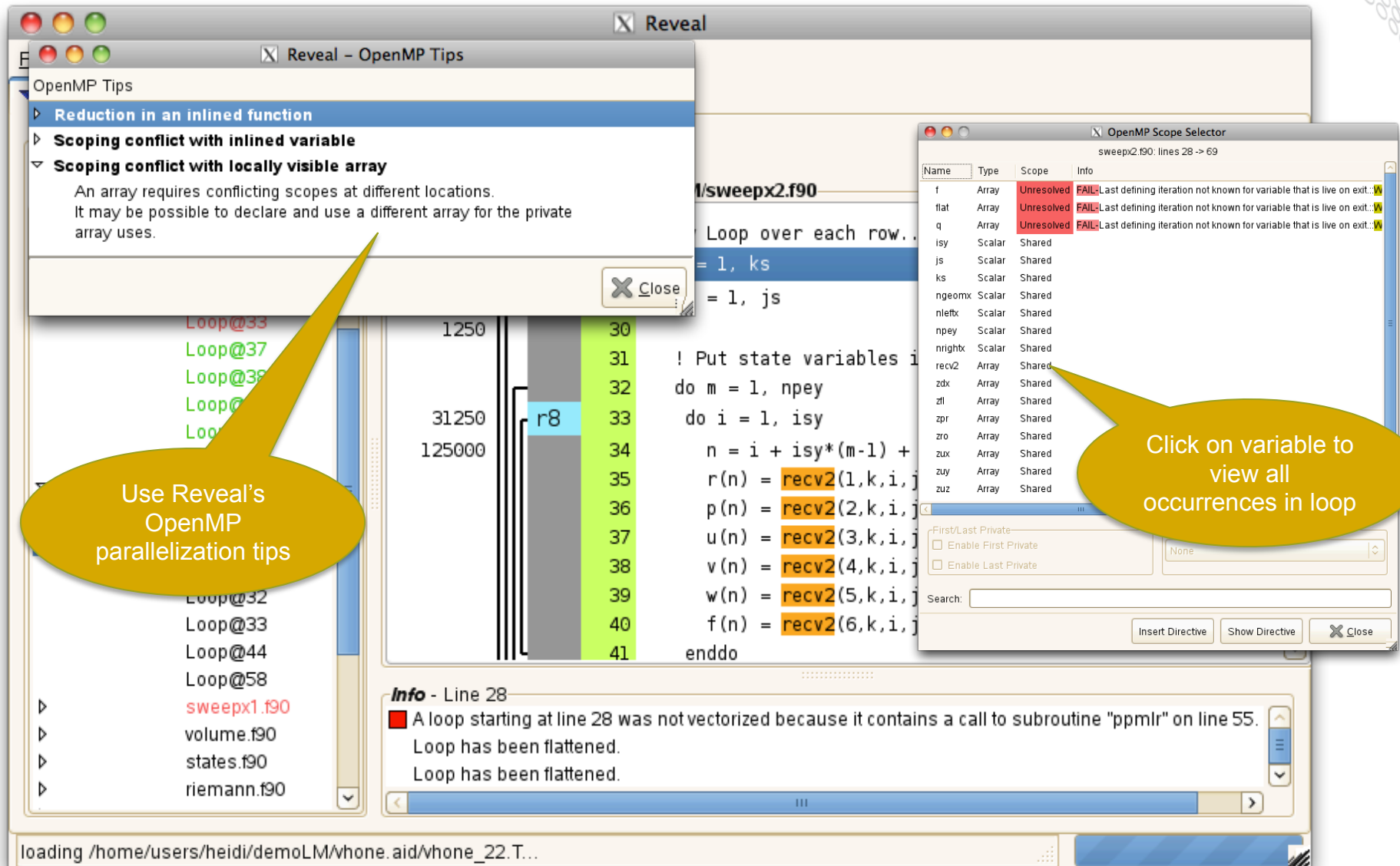
Name	Type	Scope	Info
f	Array	Unresolved	FAIL-Last defining iteration not known for variable that is live on exit. WARN-LastPrivate of array may be very expensive.
flat	Array	Unresolved	FAIL-Last defining iteration not known for variable that is live on exit. WARN-LastPrivate of array may be very expensive.
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	
nrightz	Scalar	Shared	

Info - Line 48: A loop starting at line 48 was not vectorized. Loop has been flattened.

Open Message Selector Controls:

- First/Last Private:
 - ☐ Enable First Private
 - ☐ Enable Last Private
- Reduction: None
- Search:
- Buttons: Insert Directive, Show Directive, Close

Scoping Assistance – User Resolves Issues



OpenMP Tips

- Reduction in an inlined function
- Scoping conflict with inlined variable
- Scoping conflict with locally visible array
 - An array requires conflicting scopes at different locations. It may be possible to declare and use a different array for the private array uses.

OpenMP Scope Selector
sweepx2.f90: lines 28 -> 69

Name	Type	Scope	Info
f	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit.
flat	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit.
q	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit.
isy	Scalar	Shared	
js	Scalar	Shared	
ks	Scalar	Shared	
ngeomx	Scalar	Shared	
nletx	Scalar	Shared	
npey	Scalar	Shared	
nrightx	Scalar	Shared	
recv2	Array	Shared	
zdx	Array	Shared	
zfi	Array	Shared	
zpr	Array	Shared	
zro	Array	Shared	
zux	Array	Shared	
zuy	Array	Shared	
zuz	Array	Shared	

Code Editor:

```

! Put state variables i
do m = 1, npey
  do i = 1, isy
    n = i + isy*(m-1) +
    r(n) = recv2(1,k,i,j)
    p(n) = recv2(2,k,i,j)
    u(n) = recv2(3,k,i,j)
    v(n) = recv2(4,k,i,j)
    w(n) = recv2(5,k,i,j)
    f(n) = recv2(6,k,i,j)
  enddo

```

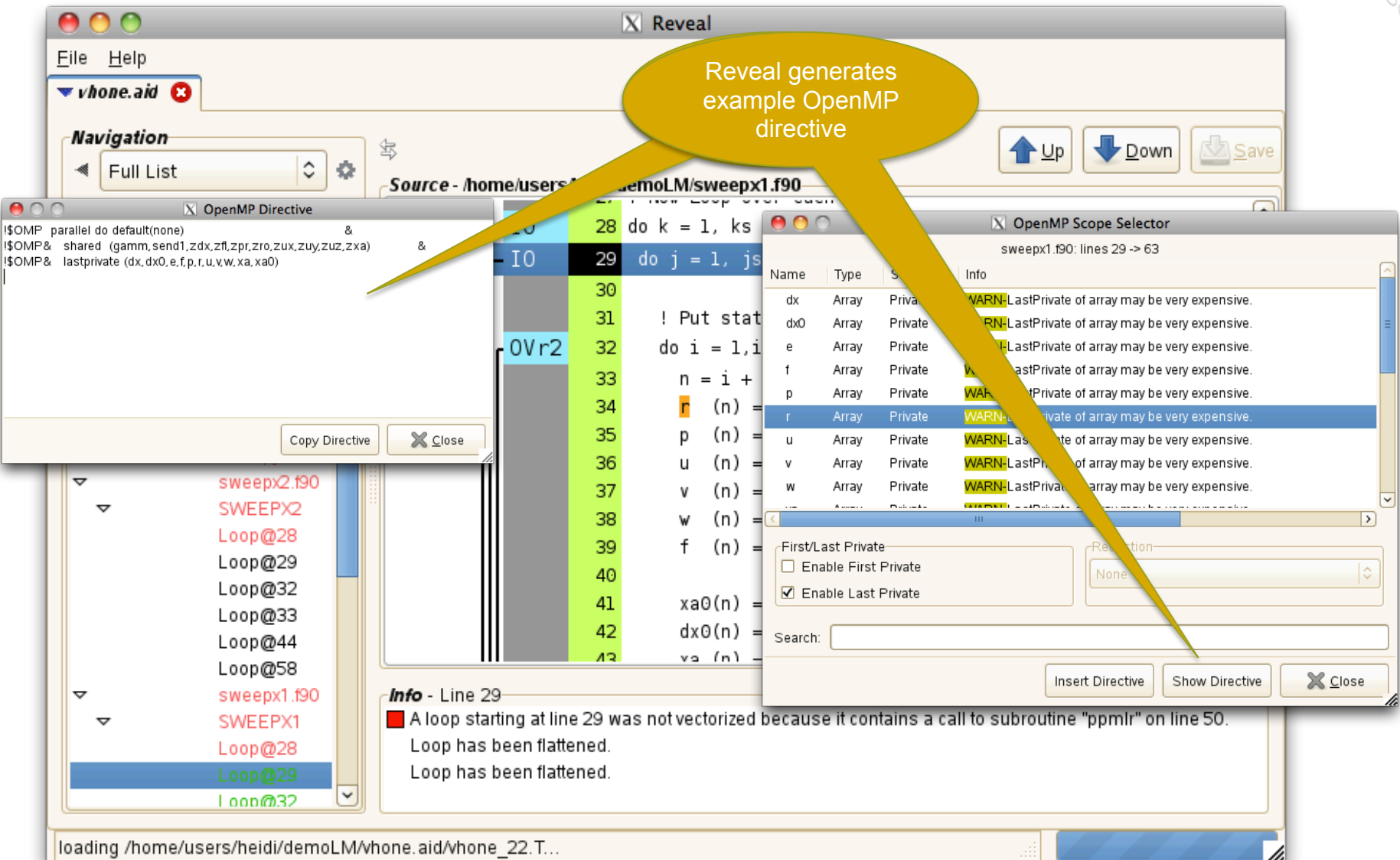
Info - Line 28

A loop starting at line 28 was not vectorized because it contains a call to subroutine "ppmlr" on line 55.
 Loop has been flattened.
 Loop has been flattened.

Annotations:

- Use Reveal's OpenMP parallelization tips
- Click on variable to view all occurrences in loop

Scoping Assistance – Generate Directive



Reveal generates example OpenMP directive

Source - /home/users/heidi/demoLM/sweepx1.f90

```

28 do k = 1, ks
29 do j = 1, js
30
31 ! Put stat
32 do i = 1, i
33 n = i +
34 r (n) =
35 p (n) =
36 u (n) =
37 v (n) =
38 w (n) =
39 f (n) =
40
41 xa0(n)
42 dx0(n)
43 va (n)

```

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)

```

Copy Directive Close

OpenMP Scope Selector

sweepx1.f90: lines 29 -> 63

Name	Type	Scope	Info
dx	Array	Private	WARN: LastPrivate of array may be very expensive.
dx0	Array	Private	WARN: LastPrivate of array may be very expensive.
e	Array	Private	WARN: LastPrivate of array may be very expensive.
f	Array	Private	WARN: LastPrivate of array may be very expensive.
p	Array	Private	WARN: LastPrivate of array may be very expensive.
r	Array	Private	WARN: LastPrivate of array may be very expensive.
u	Array	Private	WARN: LastPrivate of array may be very expensive.
v	Array	Private	WARN: LastPrivate of array may be very expensive.
w	Array	Private	WARN: LastPrivate of array may be very expensive.

First/Last Private

☐ Enable First Private

☒ Enable Last Private

Resolution: None

Search:

Insert Directive Show Directive Close

Info - Line 29

A loop starting at line 29 was not vectorized because it contains a call to subroutine "ppmlr" on line 50. Loop has been flattened. Loop has been flattened.

loading /home/users/heidi/demoLM/vhone.aid/vhone_22.T...

Step 4 – Analyze performance of accelerator regions



Programming Models Supported for the GPU

- Goal is to provide **whole program analysis** for programs written for x86 or hybrid x86 + GPUs
- Development focus is on support of CCE with OpenACC directives
- **Cray XK programming models supported**
 - OpenACC, CUDA, PGI acc (or OpenACC) directives

Collecting GPU Statistics for OpenACC

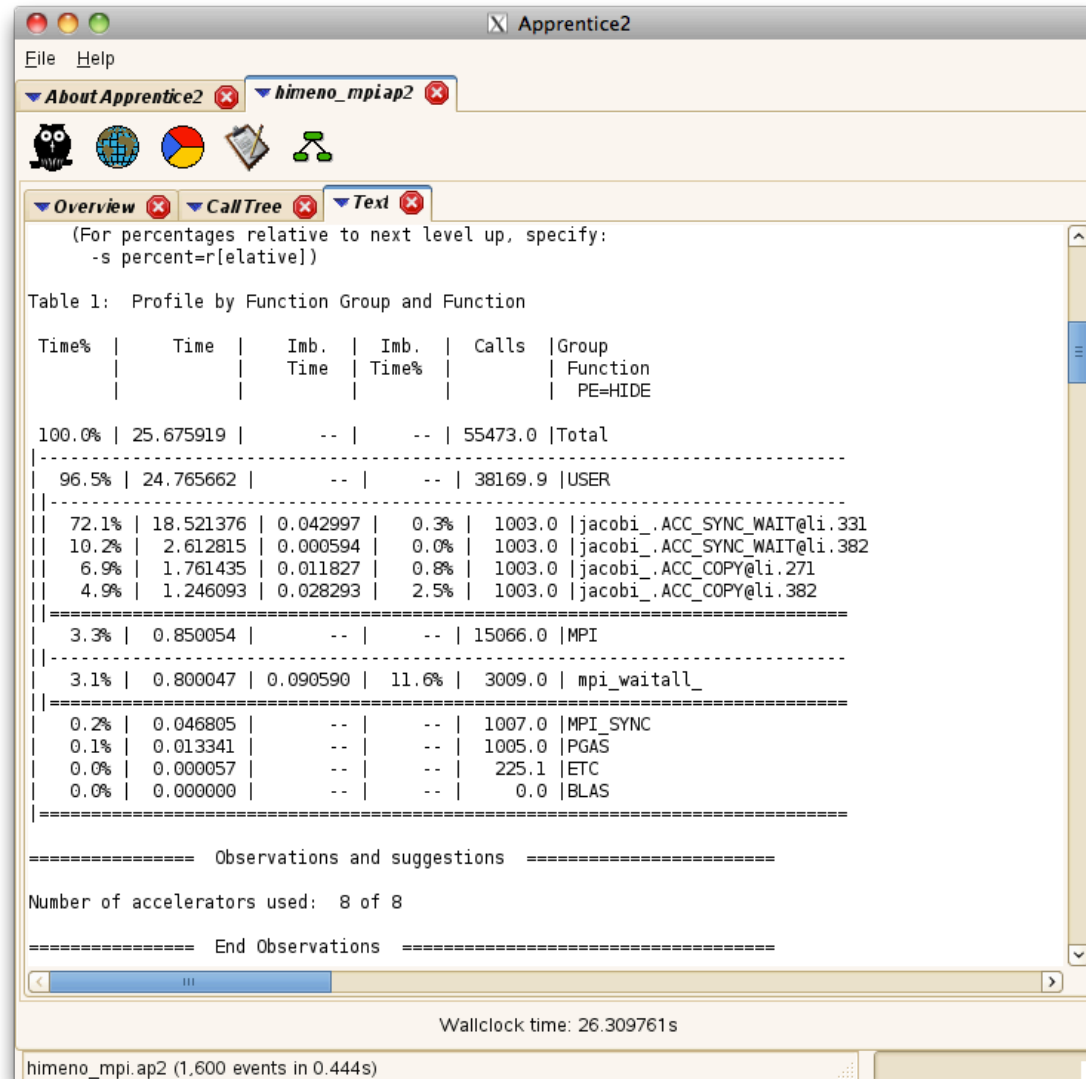
- Load PrgEnv-cray module
- Load perftools module
- Instrument binary for tracing and collecting GPU statistics
 - `pat_build -u -g mpi,blas my_program`
- Run application
- Create report with GPU statistics
 - `pat_report my_program.xf > GPU_stats_report`



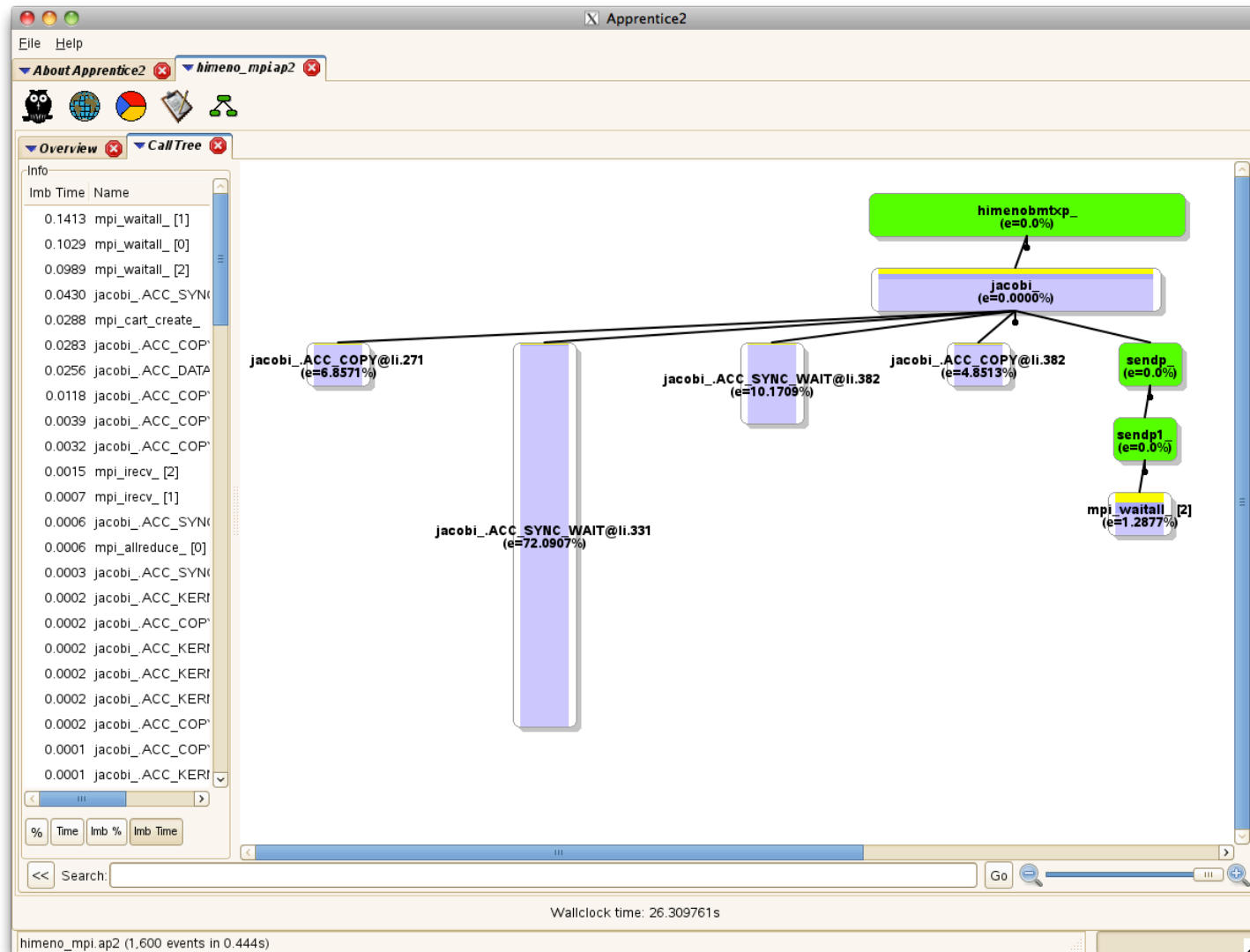
Analyze Performance of Accelerated Program

- **Statistics collected for programs with OpenACC directives**
 - Number of GPUs used in the job
 - Host time for kernel launches, data copies and synchronization with the accelerator
 - Accelerator time for kernel execution and data copies
 - Data copy size to and from the accelerator
 - Kernel grid size
 - Block size
 - Amount of shared memory dynamically allocated for kernel
 - GPU performance counters
 - Derived metrics based on performance counters

Profile with GPU



Call Tree with GPU regions



Example Accelerator Statistics

Table 1: Time and Bytes Transferred for Accelerator Regions

Host	Host	Acc	Acc Copy	Acc Copy	Calls	Calltree
Time%	Time	Time	In	Out		PE=HIDE
			(MBytes)	(MBytes)		
100.0%	2.750	2.015	2812.760	13.568	103	Total

100.0%	2.750	2.015	2812.760	13.568	103	lbm3d2p_d_
						lbm3d2p_d_.ACC_DATA_REGION@li.104

3 63.5%	1.747	1.747	2799.192	--	1	lbm3d2p_d_.ACC_COPY@li.104
3 22.1%	0.609	0.088	12.304	12.304	36	streaming_

4 20.6%	0.566	0.046	12.304	12.304	27	streaming_exchange_
5						streaming_exchange_.ACC_DATA_REGION@li.526
6 18.8%	0.517	--	--	--	1	streaming_exchange_.ACC_DATA_REGION@li.526(exclusive)
4 1.6%	0.043	0.042	--	--	9	streaming_.ACC_DATA_REGION@li.907
5 1.1%	0.031	0.031	--	--	4	streaming_.ACC_REGION@li.909
6 1.1%	0.031	--	--	--	1	streaming_.ACC_REGION@li.909(exclusive)
=====						

...

Example Kernel Statistics

Table 2: Kernel Stats for Accelerator Regions

Avg	Avg	Avg	Avg	Avg	Avg	Function
Grid	Grid	Grid	Block	Block	Block	
X	Y	Z	X Dim	Y Dim	Z Dim	
Dim	Dim	Dim				

62163	1	1	1024	1	1	streaming_.ACC_KERNEL@li.909
402	1	1	128	1	1	grad_exchange_.ACC_KERNEL@li.443
402	1	1	128	1	1	grad_exchange_.ACC_KERNEL@li.467
402	1	1	128	1	1	grad_exchange_.ACC_KERNEL@li.476
402	1	1	128	1	1	grad_exchange_.ACC_KERNEL@li.500
400	1	1	512	1	1	cal_velocity_.ACC_KERNEL@li.1126
400	1	1	512	1	1	collisiona_.ACC_KERNEL@li.474
400	1	1	128	1	1	collisionb_.ACC_KERNEL@li.597
400	1	1	128	1	1	wall_boundary_.ACC_KERNEL@li.973
400	1	1	128	1	1	collisionb_.ACC_KERNEL@li.629
400	1	1	512	1	1	recolor_.ACC_KERNEL@li.823
128	1	1	64	1	1	injection_.ACC_KERNEL@li.1281
128	1	1	128	1	1	streaming_exchange_.ACC_KERNEL@li.829
128	1	1	128	1	1	streaming_exchange_.ACC_KERNEL@li.729
128	1	1	128	1	1	streaming_exchange_.ACC_KERNEL@li.641
128	1	1	128	1	1	streaming_exchange_.ACC_KERNEL@li.538
101	1	1	128	1	1	collisionb_.ACC_KERNEL@li.612
101	1	1	128	1	1	set_boundary_micro_press_.ACC_KERNEL@li.299
101	1	1	128	1	1	set_boundary_macro_press2_.ACC_KERNEL@li.259
14	1	1	256	1	1	streaming_.ACC_KERNEL@li.919
=====						

Questions ?

Files Generated and the Naming Convention

File Suffix	Description
a.out+pat	Program instrumented for data collection
a.out...s.xf	Raw data for sampling experiment, available after application execution
a.out...t.xf	Raw data for trace (summarized or full) experiment, available after application execution
a.out...st.ap2	Processed data, generated by pat_report, contains application symbol information
a.out...s.apa	Automatic profiling analysis template, generated by pat_report (based on pat_build -O apa experiment)
a.out+apa	Program instrumented using .apa file
MPICH_RANK_ORDER.Custom	Rank reorder file generated by pat_report from automatic grid detection and reorder suggestions

GPU HW Performance Counters



Accelerator Hardware Performance Counters

- **Enable collection similarly to CPU counter collection:**
 - CPU: `PAT_RT_HWPC=group or events`
 - GPU: `PAT_RT_ACCPC=group or events`
- **Enabling GPU counters causes change in behavior of application:**
 - Host needs to synchronize with the accelerator at each event (since accelerator executes asynchronously with the host)
 - Can be seen through accelerator table
 - No counters: time spent waiting for kernel to complete is shown with `ACC_SYNC_WAIT` (a synchronization created by the compiler)
 - Counters: `perftools` syncs with accelerator with each event so Host Time is exclusive time for the containing region (since waiting occurs within the event's trace point instead of in the compiler sync)



Accelerator HW Counter Groups

- **A predefined set of groups has been created for ease of use**
 - Combines events that can be counted together
- **ACCPC groups start at 1000, and will be incremented by 100 as new families of accelerators are supported**
- **Specify group by number or name**
 - `PAT_RT_ACCPC=1000` *OR*
 - `PAT_RT_ACCPC=inst_exec_gst`
- **See `accpc(5)` and `accpc_k20(5)` man pages for list of groups and their descriptions**