

Adding OpenMP

Heidi Poxon

Sr. Principal Engineer

Cray Programming Environment



CRAY®

Legal Disclaimer



Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publicly announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: CHAPEL, CLUSTER CONNECT, CLUSTERSTOR, CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE, REVEAL. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used on this website are the property of their respective owners.

Reveal



The screenshot displays the Cray Reveal application interface. The main window shows a source code file named 'vhone.pl' with a list of loops on the left and the source code in the center. The 'Source' pane shows a loop starting at line 51, which was not vectorized because it contains a loop. The 'Scope Loops' pane shows a list of loops with their types and scopes. The 'Scoping Results' pane shows a table of variables and their scopes, with warnings for unresolved variables and incompatible scopes. An 'OpenMP Directive' dialog box is open, showing the directive inserted by Cray Reveal. The directive is:

```
! Directive inserted by Cray Reveal. May be incomplete.
!$OMP parallel do default(none) &
!$OMP & unresolved (dvol,dx,dx0,e,flat,p,para,q,r,radius,stheta,svel, &
!$OMP & theta,u,v,xa,xa0) &
!$OMP & private (l,j,k,m,n,delp2,delp1,shock,temp2,old_flat,onemf,hd, &
!$OMP & sinx0,gamfac1,gamfac2,dtheta,deltb,fracn,ekin) &
!$OMP & shared (gamma,isz,j,ks,mypex,ngeomz,nleitz,npez,nrightz, &
!$OMP & recv3,send4,zdz,zc,zc,zza)
```

- Reduce effort associated with adding OpenMP to MPI programs
- Get insight into optimizations performed by the Cray compiler
- Add OpenMP as a first step to parallelize loops that will target GPUs
- Track requests to memory and evaluate the bandwidth contribution of objects within a program for loop tuning

Approach to Adding Parallelism

1. Identify key high-level loops
2. Perform parallel analysis and scoping
3. Add OpenMP directive layer of parallelism
4. Analyze performance for further optimization, specifically vectorization of innermost loops
5. Port parallel loops to GPU with OpenMP target directives

The Problem – How Do I Parallelize This Loop?

- How do I know this is a good loop to parallelize?
- What prevents me from parallelizing this loop?
- Can I get help building a directive?

```
subroutine sweepz
...
do j = 1, js
  do i = 1, isz
    radius = zxc(i+mypez*isz)
    theta = zyc(j+mypey*js)
    do m = 1, npez
      do k = 1, ks
        n = k + ks*(m-1) + 6
        r(n) = recv3(1,j,k,i,m)
        p(n) = recv3(2,j,k,i,m)
        u(n) = recv3(5,j,k,i,m)
        v(n) = recv3(3,j,k,i,m)
        w(n) = recv3(4,j,k,i,m)
        f(n) = recv3(6,j,k,i,m)
      enddo
    enddo
    ...
    call ppmlr
    do k = 1, kmax
      n = k + 6
      xa(n) = zza(k)
      dx(n) = zdz(k)
      xa0(n) = zza(k)
      dx0(n) = zdz(k)
      e(n) = p(n)/(r(n)*gamm)+0.5 &
        *(u(n)**2+v(n)**2+w(n)**2)
    enddo
    call ppmlr
  ...
enddo
enddo
```

```
subroutine ppmlr

call boundary
call flatten
call paraset(nmin-4, nmax+5, para, dx, xa)

call parabola(nmin-4,nmax+4,para,p,dp,p6,p1,flat)
call parabola(nmin-4,nmax+4, para,r,dr,r6,r1,flat)
call parabola(nmin-4,nmax+4,para,u,du,u6,ul,flat)

call states(pl,ul,r1,p6,u6,r6,dp,du,dr,plft,ulft,&
  rlft,prgh,urgh,rrgh)
call riemann(nmin-3,nmax+4,gam,prgh,urgh,rrgh,&
  plft,ulft,rlft pmid umid)
call evolve(umid, pmid) ← contains more calls

call remap ← contains more calls

call volume(nmin,nmax,ngeom,radius,xa,dx,dvol)

call remap ← contains more calls

return
end
```

Loop Work Estimates



*Gather loop statistics using the **Cray performance tools** and the **Cray Compiling Environment (CCE)** to determine which loops have the most work*

- Helps identify high-level serial loops to parallelize
 - Based on runtime analysis, approximates how much work exists within a loop

Collect Loop Work Estimates



- Set up loop work estimates experiment with Cray compiler and Cray performance tools
 - `user@login> module load PrgEnv-cray perftools-lite-loops`
- Build program with Cray program library
 - `-h pl=/full_path/program.pl`
- Run program to get loop work estimates

Example Loop Statistics

Table 2: Loop Stats by Function

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

View Source and Optimization Information

The screenshot displays the Reveal IDE interface. The main window shows the source code of a file named `parabola.f90` located at `/home/users/heidi/reveal/parabola.f90`. The code is as follows:

```
66  
67 do n = nmin, nmax  
68   deltaa(n) = ar(n) - al(n)  
69   a6(n)      = 6. * (a(n) - .5 * (al(n) + ar(n)))  
70   scrch1(n) = (ar(n) - a(n)) * (a(n) - al(n))  
71   scrch2(n) = deltaa(n) * deltaa(n)  
72   scrch3(n) = deltaa(n) * a6(n)  
73 enddo  
74  
75 do n = nmin, nmax  
76   if(scrch1(n) <= 0.0) then  
77     ar(n) = a(n)  
78     al(n) = a(n)  
79   endif
```

The code is annotated with optimization markers. A bracket labeled `f` spans from line 67 to line 79, indicating a fused loop. Another bracket labeled `Vr2` spans from line 75 to line 79, indicating a vectorized region.

On the left, the **Navigation** pane shows a list of loops and their performance metrics:

Time	Loop Name	Stars
4.0423	SWEEPX2@32	★
3.8576	SWEEPZ@51	★
3.8573	SWEEPZ@52	★
2.2068	RIEMANN@63	★
1.2299	RIEMANN@64	★
0.8068	PARABOLA@67	
0.0146	Instance #1	
0.0156	Instance #2	
0.0156	Instance #3	
0.0163	Instance #4	
0.0163	Instance #5	
0.0174	Instance #6	
0.0167	Instance #7	

Below the navigation pane, the **Traceback** pane shows the following stack:

- PARABOLA@67
- PPMLR@51
- sweepx1_.LOOP.2.li.32@53
- sweepx1_.LOOP.1.li.31@32
- SWEEPX1@31
- VHONE@232

On the right, the **Reveal - Loopmark Legend** pane provides a key for the optimization markers:

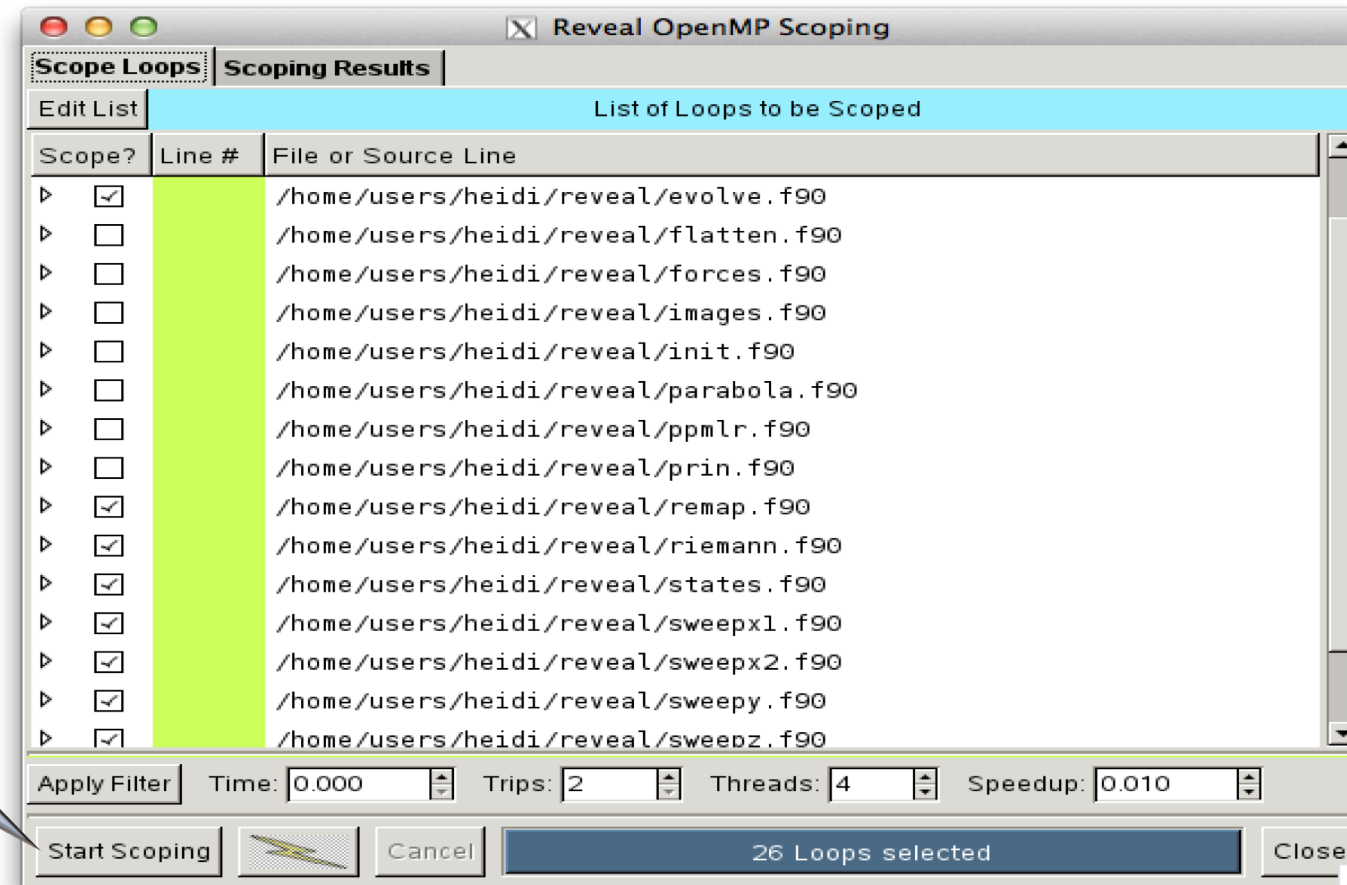
- A** Pattern Matched
- C** Collapsed: A loop nest has been collapsed into one loop
- D** Deleted
- E** Cloned
- G** Accelerated
- I** Inlined
- II** Not Inlined
- L** Loop
- M** Multithreaded: A loop or block of code is multi-threaded
- R** Region
- S** Scoping Analysis
- V** Vectorized
- a** Atomic Memory Operation
- b** Blocked
- c** Conditional and/or Computed
- f** Fused
- g** Partitioned
- i** Interchanged
- n** Non-blocking Remote Transfer
- p** Partial
- r** Unrolled
- s** Shortloop: A loop was converted to a single vector iteration
- w** Unwound

At the bottom, the **Info - Line 67** pane displays the message: "A loop starting at line 67 was fused with the loop starting at line 53."

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

Scope Selected Loop(s)

- Trigger dependence analysis
- scope loops above given threshold



Review Scoping Results

vhone.pl

File Edit View Help

Navigation

Loop Performance

- 4.0778 SWEEPY@35 ★
- 4.0773 SWEEPY@36 ★
- 4.0529 SWEEPX1@31 ★
- 4.0526 SWEEPX1@32 ★
- 4.0425 SWEEPX2@31 ★
- 4.0423 SWEEPX2@32 ★
- 3.8576 SWEEPZ@51 ★
- 3.8573 SWEEPZ@52 ★
- 2.2068 RIEMANN@63 ★
- 1.2299 RIEMANN@64 ★
- 0.8068 PARABOLA@67 ★
- 0.5429 PARABOLA@44 ★
- 0.5331 PARABOLA@75 ★
- 0.4244 REMAP@83 ★
- 0.3341 PARABOLA@30 ★
- 0.2966 PARABOLA@84 ★
- 0.2915 PARABOLA@53 ★
- 0.2287 RIEMANN@44 ★
- 0.2028 PARABOLA@36 ★
- 0.2009 PARABOLA@117 ★
- 0.1858 PARABOLA@24 ★
- 0.1847 SWEEPY@86 ★
- 0.1771 STATES@64 ★
- 0.1723 EVOLVE@70 ★
- 0.1638 REMAP@111 ★
- 0.1619 PARABOLA@129 ★
- 0.1070 PARABOLA@139 ★
- 0.0938 SWEEPZ@120 ★
- 0.0936 SWEEPZ@121 ★
- 0.0930 SWEEPZ@122 ★
- 0.0925 SWEEPX1@59 ★
- 0.0901 SWEEPZ@22 ★
- 0.0898 SWEEPZ@23 ★
- 0.0892 STATES@50 ★
- 0.0880 SWEEPZ@105 ★

Source - /home/users/heidi/reveal/sweepz.f90

```
52 do i = 1, isz
53   radius = zxc(i+mypey*isz)
54   theta = zyc(j+mypey*js)
55   stheta = sin(theta)
56   radius = radius * stheta
57
58 ! Put state variables into 1D arrays, padding with 6 ghost zones
59 do m = 1, npez
60   do k = 1, ks
61     n = k + ks*(m-1) + 6
62     r(n) = recv3(1,j,k,i,m)
63     p(n) = recv3(2,j,k,i,m)
64     u(n) = recv3(5,j,k,i,m)
65     v(n) = recv3(3,j,k,i,m)
66     w(n) = recv3(4,i,k,i,m)
```

Info - Line 51

- A loop starting at line 51 was scoped with errors. See Scoping Tool for more information. "ppmlr" (called from "sweepz") was not inlined because I/O was detected in "volume".
- "ppmlr" (called from "sweepz") was not inlined because the enclosing loop body did not completely flatten.
- A loop starting at line 105 is flat (contains no external calls).
- A loop starting at line 105 was not vectorized because it does not map well onto the target architecture.
- A loop starting at line 105 was unrolled 8 times.
- A loop starting at line 51 was not vectorized because it contains a call to subroutine "ppmlr" on line 81.
- A loop starting at line 52 was not vectorized because it contains a call to subroutine "ppmlr" on line 81.
- A loop starting at line 59 is flat (contains no external calls).
- A loop starting at line 59 was not vectorized because a better candidate was found at line 60.
- A loop starting at line 60 is flat (contains no external calls).
- A loop starting at line 60 was not vectorized because it does not map well onto the target architecture.
- A loop starting at line 60 was unrolled 8 times.
- A loop starting at line 71 is flat (contains no external calls).
- A loop starting at line 71 was vectorized.

Info - Line 51

home/users/heidi/reveal/vhone_loops.ap2 loaded.

Loops with scoping information are flagged. Red needs user assistance

Reveal OpenMP Scoping

Scope Loops Scoping Results

sweepz.f90: Loop@51

Call or I/O at line 81 of sweepz.f90
4: /home/users/heidi/reveal/volume.f90:34
3: /home/users/heidi/reveal/evolve.f90:21
2: /home/users/heidi/reveal/ppmlr.f90:73
1: /home/users/heidi/reveal/sweepz.f90:81
Call or I/O at line 81 of sweepz.f90
4: /home/users/heidi/reveal/volume.f90:34

Name	Type	Scope	Info
wl@remap_1	Scalar	Unresolved	FAIL: Possible recurrence involving this object. FAIL: Possible resolvable recurrence involving this object.
xa	Array	Unresolved	FAIL: Possible recurrence involving this object. FAIL: Possible resolvable recurrence involving this object. WARN: LastPrivate of array may be very expensive.
xa0	Array	Unresolved	FAIL: Possible recurrence involving this object. FAIL: Possible resolvable recurrence involving this object. WARN: LastPrivate of array may be very expensive.
i	Scalar	Private	
j	Scalar	Private	
k	Scalar	Private	
m	Scalar	Private	
n	Scalar	Private	
stheta	Scalar	Private	
theta	Scalar	Private	
gamm	Scalar	Shared	
isz	Scalar	Shared	
js	Scalar	Shared	
ks	Scalar	Shared	
mypey	Scalar	Shared	

First/Last Private

☐ Enable FirstPrivate

☐ Enable LastPrivate

Find Name:

Insert Directive Show Directive Close

Parallelization inhibitor messages are provided to assist user with analysis

Review Scoping Results (continued)

Reveal identifies calls that prevent parallelization

Name	Type	Scope	Info
ks	Scalar	Shared	
mypey	Scalar	Shared	
ndim	Scalar	Shared	
npay	Scalar	Shared	
recv1	Array	Shared	
send2	Array	Shared	
svel RI	Scalar	Shared	WARN: atomic reduction operator required unless reduction fully
zdy	Array	Shared	
zxc	Array	Shared	
zya	Array	Shared	

Reveal identifies shared reductions down the call chain

Review Scoping Results (continued)

Name	Type	Scope	Info
m_mat_an.c: Loop@39			
a0i	Scalar	Private	
a0r	Scalar	Private	
a1i	Scalar	Private	
a1r	Scalar	Private	
a2i	Scalar	Private	
a2r	Scalar	Private	
b0i	Scalar	Private	
b0r	Scalar	Private	
b1i	Scalar	Private	
b1r	Scalar	Private	
b2i	Scalar	Private	
b2r	Scalar	Private	
j	Scalar	Private	
a	Scalar	Shared	WARN: Assuming no overlap with other objects. INFO: additional detail.
b	Scalar	Shared	WARN: Assuming no overlap with other objects. INFO: additional detail.
c	Scalar	Shared	WARN: Assuming no overlap with other objects. INFO: additional detail.

First/Last Private: ☐ Enable FirstPrivate ☐ Enable LastPrivate

Reduction: None

Find Name:

Assume no overlap between lattice[*].mom[*] and tempmom[*][*]

Review Scoping Results (continued)

Reveal OpenMP Scoping

Scope Loops

Scoping Results

fluxk.f: Loop@28

Name	Type	Scope	Info
fsk	Array	Private	WARN: LastPrivate of array may be very expensive. FAIL: FirstPrivate/Shared Scope Conflict.
fsk I	Array	Private	FAIL: FirstPrivate/Shared Scope Conflict.
i	Scalar	Private	
j	Scalar	Private	
k	Scalar	Private	
l	Scalar	Private	FAIL: Ambiguous store conflict.
qs	Array	Private	WARN: LastPrivate of array may be very expensive. FAIL: Last defining iteration not known for variable that may be live on exit.
qsp	Scalar	Private	
qspk	Scalar	Private	
dq	Array	Shared	
dtv	Array	Shared	
ind	Scalar	Shared	FAIL: conflicting requirements, unable to scope.
jcmx	Scalar	Shared	
kadd	Scalar	Shared	

First/Last Private

☐ Enable FirstPrivate
☐ Enable LastPrivate

Reduction

None

Find Name:

Insert Directive

Show Directive

Close

View Loops through Call Chain

The screenshot displays the 'Reveal' application window. The top menu bar includes 'File', 'Edit', 'View', and 'Help'. The main window is divided into several panes:

- Navigation Pane (Left):** Shows a list of performance metrics and code locations. The 'Loop Performance' section is expanded, showing a list of loops with their execution times and locations. The 'Loop instances' section is also visible, showing a list of loop instances with their locations. A callout bubble points to the 'Loop instances' section, labeled 'Loop instances'.
- Source Pane (Right):** Displays the source code of the selected loop. The code is for a function named 'do l = lmin, lmax' and contains several nested loops and calculations. A callout bubble points to the source code, labeled 'Loop traceback'.
- Info Pane (Bottom Right):** Displays information about the selected loop. It shows 'Line 63' and a message: 'A loop starting at line 63 was not vectorized for an unspecified reason.'

The status bar at the bottom of the window shows: 'vhone.pl loaded. vhone_loops.ap2 loaded.'

Generate OpenMP Directives

```
! Directive inserted by Cray Reveal. May be incomplete.
!$OMP parallel do default(none)
!$OMP& unresolved (dvol,dx,dx0,e,f,flat,p,para,q,r,radius,svel,u,v,w, &
!$OMP& xa,xa0) &
!$OMP& private (i,j,k,m,n,$$ _n,delp2,delp1,shock,temp2,old_flat, &
!$OMP& onemfl,hdt,sinxf0,gamfac1,gamfac2,dtheta,deltx,fractn, &
!$OMP& ekin) &
!$OMP& shared (gamm,isy,js,ks,mypey,ndim,ngeomy,nlefty,npey,nrighty, &
!$OMP& recv1,send2,zdy,zxc,zya)
do k = 1, ks
do i = 1, isy
radius = zxc(i+mypey*isy)

! Put state variables into 1D arrays, padding with 6 ghost zones
do m = 1, npey
do j = 1, js
n = j + js*(m-1) + 6
r(n) = recv1(1,k,j,i,m)
p(n) = recv1(2,k,j,i,m)
u(n) = recv1(4,k,j,i,m)
v(n) = recv1(5,k,j,i,m)
w(n) = recv1(3,k,j,i,m)
f(n) = recv1(6,k,j,i,m)
enddo
enddo

do j = 1, jmax
n = j + 6
```

Reveal generates OpenMP directive with illegal clause marking variables that need addressing

Validate User Inserted Directives

Navigation

- Program View
- images.f90
- init.f90
- parabola.f90
- ppmlr.f90
- prin.f90
- remap.f90
- riemann.f90
- RIEMANN
- Loop@44
- Loop@69
- Loop@70
- Loop@89
- states.f90
- sweepx1.f90
- sweepx2.f90
- sweepy.f90
- sweepz.f90
- vh1.mods.f90
- vhone.f90
- volume.f90
- zonemod.f90

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

```
63 : Directive inserted by Cray Reveal. May be incomplete.  
64 !$OMP parallel do default(none)  
65 !$OMP& private (l)  
66 !$OMP& shared (lmin,lmax,prgh,urgh,vrgh,plft,ulft,vlft,pmid,clft,  
67 !$OMP& crgh,gamfac1,gamfac2,plfti,pmold,prghi,umidl,umidr  
68 !$OMP& wlft,wrgh,zlft,zrgh,n)  
69 do l = lmin, lmax  
70 do n = 1, 12  
71 pmold(l  
72 wlft (l  
73 wrgh (l  
74 wlft (l  
75 wrgh (l  
76 zlft (l
```

Info - Line 69

- A loop starting at line 69 was not
- A loop starting at line 69 was pa

Scope Loops

riemann.f90: Loop@69

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

First/Last Private

- ☐ Enable FirstPrivate
- ☐ Enable LastPrivate

Reduction

None

Find Name:

vhone.pl loaded

User inserted directive with mis-scoped variable 'n'

Look For Vectorization Opportunities

Choose "Compiler Messages" view to access message filtering, then select desired type of message

The screenshot shows the vhone.pl IDE interface. The 'Navigation' pane on the left is set to 'Compiler Messages' and 'Not Vectorized'. It lists various source files and line numbers, with 'riemann.f90' line 64 (0.224 sec) selected. The 'Source' pane on the right displays the Fortran code for 'riemann.f90', starting with a loop at line 64. The 'Info' pane at the bottom provides details about the selected line: a green square indicates a flat loop, and a red circle indicates a non-vectorized loop due to a recurrence on 'pmid' at line 77.

Navigation

Compiler Messages

Not Vectorized

- images.f90
- init.f90
- prin.f90
- riemann.f90
- sweepx1.f90

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

```
62
63 do l = lmin, lmax
64 do n = 1, 12
65   pmold(l) = pmid(l)
66   wlft(l) = 1.0 + gamfac1*(pmid(l) - plft(l)) * plfti(l)
67   wrgh(l) = 1.0 + gamfac1*(pmid(l) - prgh(l)) * prghi(l)
68   wlft(l) = clft(l) * sqrt(wlft(l))
69   wrgh(l) = crgh(l) * sqrt(wrgh(l))
70   zlft(l) = 4.0 * vlft(l) * wlft(l) * wlft(l)
71   zrgh(l) = 4.0 * vrgh(l) * wrgh(l) * wrgh(l)
72   zlft(l) = -zlft(l) * wlft(l)/(zlft(l) - gamfac2*(pmid(l) - plft(l)
73   zrgh(l) = zrgh(l) * wrgh(l)/(zrgh(l) - gamfac2*(pmid(l) - prgh(l)
74   umidl(l) = ulft(l) - (pmid(l) - plft(l)) / wlft(l)
75   umidr(l) = urgh(l) + (pmid(l) - prgh(l)) / wrgh(l)
76   pmid(l) = pmid(l) + (umidr(l) - umidl(l))*(zlft(l) * zrgh(l)) / (
77   pmid(l) = max(smallp,pmid(l))
78   if (abs(pmid(l)-pmold(l))/pmid(l) < tol ) exit
79 enddo
```

Info - Line 64

- A loop starting at line 64 is flat (contains no external calls).
- A loop starting at line 64 was not vectorized because a recurrence was found on "pmid" at line 77.

vhone.pl loaded. vhone_loops.ap2 loaded.

QUESTIONS?

