

CRAY®

**Reveal**

Heidi Poxon  
Sr. Principal Engineer  
Cray Programming Environment

# 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 publically 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.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. 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 in this document are the property of their respective owners.*

*Copyright 2017 Cray Inc.*



**Cray Compiler Optimization Feedback**

**OpenMP Assistance**

**MCDRAM Allocation Assistance**

# Reveal Overview



- **Reduce effort associated with adding OpenMP** to MPI programs when a pure MPI program no longer scales
- **Produce performance portable code** through OpenMP directives
- **Get insight into optimizations** performed by the Cray compiler
- Use 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

# When to Move to a Hybrid Programming Model



- **When code is network bound**
  - Increased MPI collective and point-to-point wait times
- **When MPI starts leveling off**
  - Too much memory used, even if on-node shared communication is available
  - As the number of MPI ranks increases, more off-node communication can result, creating a network injection issue
- **When contention of shared resources increases**



# Approach to Adding Parallelism

## 1. Identify key high-level loops

- Determine where to add additional levels of parallelism

## 1. Perform parallel analysis and scoping

- Split loop work among threads

## 2. Add OpenMP layer of parallelism

- Insert OpenMP directives

## 3. Analyze performance for further optimization, specifically vectorization of innermost loops

- We want a performance-portable application at the end

# 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)
  cx(n) = zdz(k)
  xa0(n) = zza(k)
  cx0(n) = zdz(k)
  e(n) = p(n) / (r(n)*gamma)+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,u1,flat)

call states(p1,u1,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 pmd umid)
call evolve(umid, pmd) ← contains more calls

call remap ← contains more calls

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

call remap ← contains more calls

return
end

```

COMPUTE

STORE

ANALYZE

# 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
  - `$ 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	Loop	Loop	Loop	Loop	Function=/.LOOP[.]
Incl	Hit	Trips	Trips	Trips	PE=HIDE
Time		Avg	Min	Max	
Total					
-----					
8.995914	100	25	0	25	sweepy_.LOOP.1.1i.33
8.995604	2500	25	0	25	sweepy_.LOOP.2.1i.34
8.894750	50	25	0	25	sweepz_.LOOP.05.1i.49
8.894637	1250	25	0	25	sweepz_.LOOP.06.1i.50
4.420629	50	25	0	25	sweepx2_.LOOP.1.1i.29
4.420536	1250	25	0	25	sweepx2_.LOOP.2.1i.30
4.387534	50	25	0	25	sweepx1_.LOOP.1.1i.29
4.387457	1250	25	0	25	sweepx1_.LOOP.2.1i.30
2.523214	187500	107	0	107	riemann_.LOOP.2.1i.63

# View Source and Optimization Information



Reveal

File Edit View Help

▼ vhone.pl ✖

**Navigation**

Loop Performance

- 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

Traceback

- PARABOLA@67
- PPMLR@51
- sweepx1\_LOOP.2.li.32@53
- sweepx1\_LOOP.1.li.31@32
- SWEEPX1@31
- VHONE@232

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

```
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
```

**Loopmark Legend**

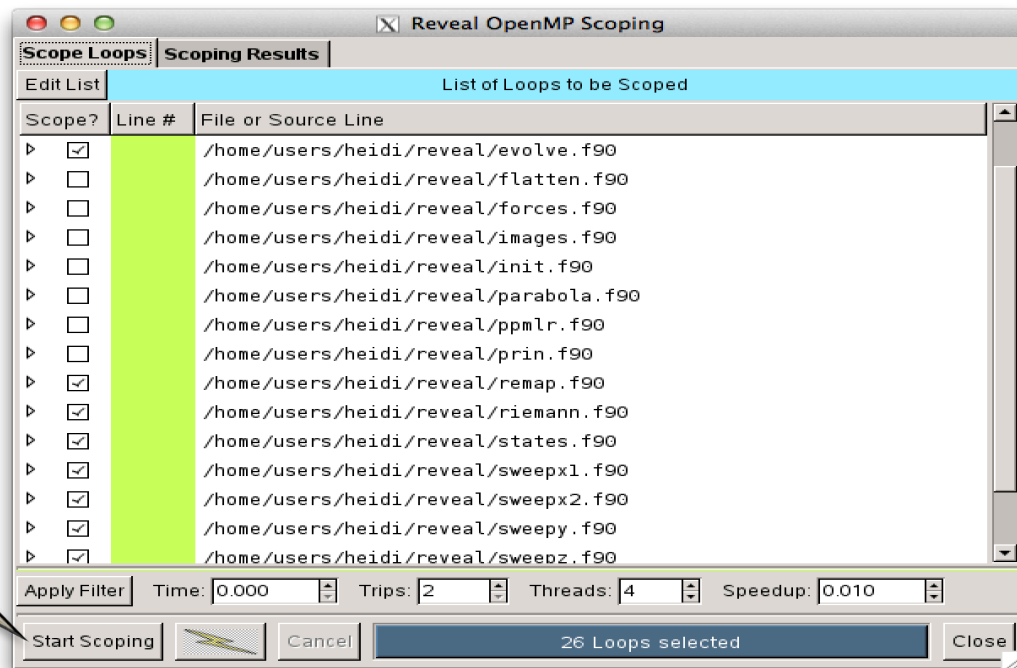
- ▶ A Pattern Matched
- ▼ C Collapsed  
A loop nest has been collapsed into one loop
- ▶ D Deleted
- ▶ E Cloned
- ▶ G Accelerated
- ▶ I Inlined
- ▶ II Not Inlined
- ▶ L Loop
- ▼ M Multithreaded  
A loop or block of code is multi-threaded
- ▶ R Region
- ▶ S Scoping Analysis
- ▶ V Vectorized
- ▶ a Atomic 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

**Info - Line 67**

■ A loop starting at line 67 was fused with the loop starting at line 53.

vhone.pl loaded. vhone\_loops.ap2 loaded.

# Scope Selected Loop(s)



- Trigger dependence analysis
- scope loops above given threshold

# Review Scoping Results



Loops with scoping information are flagged. Red needs user assistance

Navigation: Loop Performance

Source: /home/users/heidi/reveal/sweepz

```
40778 SWEEPZ@35
40779 SWEEPZ@36
40529 SWEEPX1@31
40526 SWEEPX1@32
40425 SWEEPX2@31
40423 SWEEPX2@32
38576 SWEEPZ@51
38573 SWEEPZ@52
22068 RIEMANN@63
12299 RIEMANN@64
08068 PARABOLA@67
05429 PARABOLA@44
05331 PARABOLA@75
04244 REMAP@83
03341 PARABOLA@30
02966 PARABOLA@84
02915 PARABOLA@53
02287 RIEMANN@44
02028 PARABOLA@36
02009 PARABOLA@117
01858 PARABOLA@24
01847 SWEEPZ@86
01771 STATES@64
01723 EVOLVE@70
01638 REMAP@111
01619 PARABOLA@129
01070 PARABOLA@139
00938 SWEEPZ@120
00936 SWEEPZ@121
00930 SWEEPZ@122
00925 SWEEPX1@59
00901 SWEEPZ@22
00898 SWEEPZ@23
00892 STATES@50
00880 SWEEPZ@105
```

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.

home/users/heidi/reveal/vphone\_loops.ap2 loaded.

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
wi@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 OpenMP Scoping

Scope Loops | Scoping Results

sweepy.f90: Loop@35

Call or I/O at line 62 of sweepy.f90  
4: /home/users/heidi/reveal/volume.f90:34  
3: /home/users/heidi/reveal/evolve.f90:21

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

First/Last Private  
☐ Enable FirstPrivate  
☐ Enable LastPrivate

Reduction:  
None

Find Name:

Insert Directive Show Directive Close

Reveal identifies calls that prevent parallelization

Reveal identifies shared reductions down the call chain

COMPUTE

STORE

ANALYZE

# Review Scoping Results (continued)



Reveal OpenMP Scoping

Scope Loops | Scoping Results | Footnote

m\_mat\_an.c: Loop@39

Name	Type	Scope	Info
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	<b>WARN:</b> Assuming no overlap with other objects. <b>INFO:</b> additional detail.
b	Scalar	Shared	<b>WARN:</b> Assuming no overlap with other objects. <b>INFO:</b> additional detail.
c	Scalar	Shared	<b>WARN:</b> Assuming no overlap with other objects. <b>INFO:</b> additional detail.

First/Last Private

☐ Enable FirstPrivate

☐ Enable LastPrivate

Reduction

None

Find Name:

Insert Directive Show Directive Close

Reveal OpenMP Scoping

Scope Loops | Scoping Results | Footnote

Scoping Footnote

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

Close

COMPUTE

STORE

ANALYZE

# 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
```

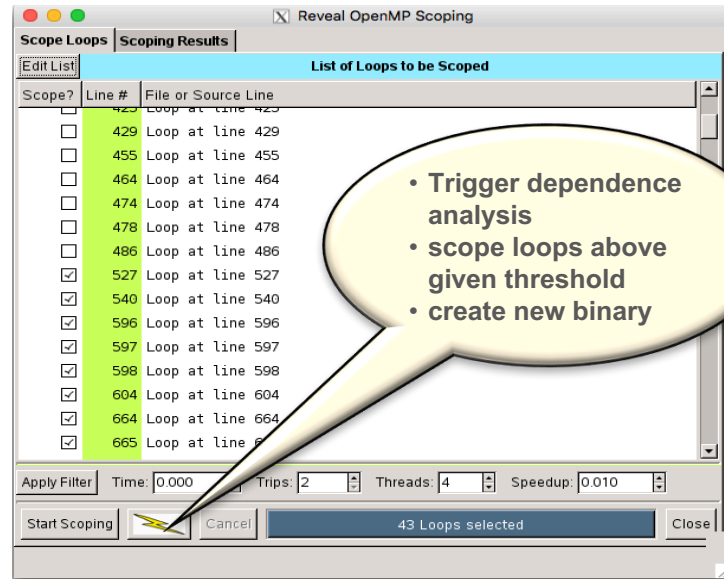
Reveal generates OpenMP directive with illegal clause marking variables that need addressing





# Reveal Auto-Parallelization

- Minimal user time investment includes time to set up and run optimization experiment
  - Collect loop work estimates
  - Build program library
  - Click button in Reveal
  - Run experimental binary,
  - Compare against original program
- Even if experiment does not yield a performance improvement, Reveal will provide insight into parallelization issues
- Use on X86 hardware or Armv8 hardware



# Validate User Inserted Directives



The screenshot shows the Cray Reveal IDE interface. The main window displays a Fortran source file with several OpenMP directives. A user has inserted a directive at line 69: `do l = lmin, lmax`. The IDE's "Scope Loops" panel shows the variable `n` as a scalar with a private scope, while the user's directive implies a shared scope, leading to a warning: "WARN: Scope does not agree with user OMP directive." A speech bubble points to this warning with the text: "User inserted directive with mis-scoped variable 'n'".

**Navigation**

- 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

```
64 !Directive inserted by Cray Reveal. May be incomplete.
65 !$OMP parallel do default(none)
66 !$OMP& private (l)
67 !$OMP& shared (lmin,lmax,prgh,urgh,vrgh,plft,ulft,vlft,pmid,clft,
68 !$OMP& crgh,gamfac1,gamfac2,plfti,pmold,prghi,umidl,umidr
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 found.
- A loop starting at line 69 was pa...

**Scope Loops**

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

**Reduction**

None

**Find Name:**

**Insert Directive** **Show Directive** **Close**

# Look For Vectorization Opportunities

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

The screenshot shows the Cray vhone.pl IDE interface. The 'Navigation' pane on the left lists various source files and line ranges. The 'Compiler Messages' view is selected, showing a list of messages. The 'Source' pane on the right displays the code for `riemann.f90`, with line 64 highlighted. The 'Info' pane at the bottom provides details about the selected line.

**Navigation**

- Compiler Messages
- Not Vectorized
- All
- line 123
- images.f90
- line 149
- init.f90
  - line 113 (0.000 sec)
  - line 114 (0.000 sec)
  - line 153 (0.000 sec)
  - line 154 (0.000 sec)
  - line 139
- prin.f90
  - line 125 (0.006 sec)
  - line 42 (0.000 sec)
  - line 43 (0.000 sec)
  - line 127 (0.000 sec)
  - line 128 (0.000 sec)
  - line 129 (0.000 sec)
  - line 104 (0.000 sec)
- riemann.f90
  - line 63 (0.387 sec)
  - line 64 (0.224 sec)
- sweepx1.f90
  - line 31 (4.053 sec)
  - line 32 (4.053 sec)
  - line 59 (0.093 sec)

**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.



COMPUTE



STORE



ANALYZE

The background of the slide is a complex, abstract digital composition. It features a series of glowing blue lines and dots that form a sense of depth and movement, resembling a data stream or a network. Large, semi-transparent binary digits (0s and 1s) are scattered throughout the scene, some appearing to float in the foreground and others receding into the background. The overall color palette is dominated by various shades of blue, from deep navy to bright, glowing cyan and white highlights.

**Thank You!**