Reveal

Heidi Poxon

Purpose

Help users find and create additional levels of parallelism within an application

- Reduce effort associated with adding OpenMP to MPI programs
- Produce performance portable code
- Get insight into optimizations performed by the Cray compiler
- Use as a first step to parallelize loops that will target GPUs

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
2. **Perform parallel analysis and scoping**
   - Split loop work among threads
3. **Add OpenMP layer of parallelism**
   - Insert OpenMP directives
4. **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)
     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,pl,flat)
call parabola(nmin-4,nmax+4, para,r,dr,r6,rl,flat)
call parabola(nmin-4,nmax+4,para,u,du,u6,ul,flat)

call states(pl,ul,rl,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
```

COMPUTE | STORE | ANALYZE

Loop Work Estimates

Gather loop statistics using the Cray performance tools and 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

- **Provides the following statistics**
  - Min, max and average trip counts
  - Inclusive time spent in loops
  - Number of times a loop was executed

Reveal Usage Recipe

- **Access Cray compiler**
  - $ module load PrgEnv-cray
- **Set up perftools loop work estimates experiment**
  - $ module load perftools-base, perftools-lite-loops
- **Build program (make)**
- **Run program to get loop work estimates in file with .ap2 suffix**

Example Loop Work Estimates

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

Reveal Usage Recipe (2)

- **Disable loop work estimates program instrumentation so we can get fully optimized program now**
  - $ module unload perftools-lite-loops
- **Create program library with CCE:**
  - Add –h pl=/*full_path*/my_program.pl to program's Makefile
- **Rebuild application with full optimization**
  - $ make clean
  - $ make
- **Launch Reveal**
  - $ reveal /*full_path*/my_program.pl loop_work_estimates.ap2

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

Up

```
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
```

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.

Reveal - Loopmark Legend (on g

Loopmark Legend

- A Pattern Matched
- C Collapsed
  A loop nest has been collapsed into one loop
- D Deleted
- E Cloned
- G Accelerated
- I Inlined
- !! 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

Close

Access Cray Compiler Message Information

Reveal

File Edit View Help

vhone.pl

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/sweepx2.f90

```
L    32    do m = 1, npey
Lr8  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
V    44    do i = 1,imax
     45      n = i + 6
```

Info - Line 33

A loop starting at line 33 was not vecto because it doe

A loop starting at line 33 was unro mes.

vhone.pl loaded. vhone_loops.ap2

Double click on optimization message for more detailed information

Explain

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.

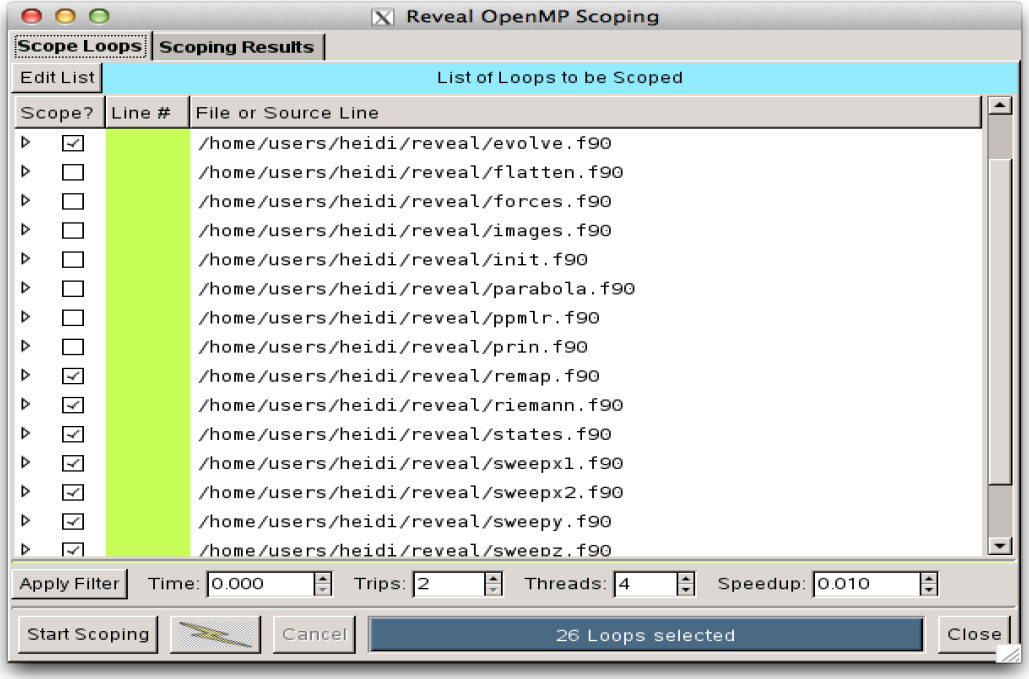
For sake of illustration, the following contrasts unroll-and-jam with literal outer loop unrolling.

```
# 426 "/ptmp/ulib/buildslaves/pdgcs-81-edition-build/tbs/build/release/pdgcs/pdgcs_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

Scope Selected Loop(s)



Review Scoping Results

Loops with scoping information are flagged. Red needs user assistance

Parallelization inhibitor messages are provided to assist user with analysis

Review Scoping Results (3)

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 RI	Scalar	Shared	WARN: atomic reduction operator required unless reduction fully
zdy	Array	Shared	
zxc	Array	Shared	
zya	Array	Shared	

Reveal identifies calls that prevent parallelization

Reveal identifies shared reductions down the call chain

First/Last Private
Enable FirstPriv
Enable La

Reduction
None

Find Name:

Insert Directive | Show Directive | Close

Review Scoping Results (2)

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	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:

Insert Directive | Show Directive | Close

Reveal OpenMP Scoping

Scope Loops | Scoping Results | Footnote

Scoping Footnote

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

Close

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

Reveal

File Edit View Help

vhone.pl

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
vh1mods.f90
vhone.f90
volume.f90
zonemod.f90

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

Up Down Save

```
R   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)
LSg 69 do l = lmin, lmax
L   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 no

A loop starting at line 69 was pa

vhone.pl loaded

Reveal Open

Scope Loops | Scoping Results

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:

Insert Directive | Show Directive | Close

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

vhone.pl

File Edit View Help

Navigation

Compiler Messages

Not Vectorized | All

- 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

Up Down Save

```
FS  63  do l = lmin, lmax
F   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.

Summary

- Reveal can be used to simplify the task of adding OpenMP to MPI programs
- Reveal can be used to validate existing user-inserted OpenMP directives
- The result is performance portable code: OpenMP directives (programs can be built with any compiler that supports OpenMP)

Q&A

Heidi Poxon
heidi@cray.com

Legal Disclaimer