Profiling and Analyzing Program Performance Using Cray Tools

Heidi Poxon

CUG.2017.CAFFEINATED COMPUTING

Redmond, Washington May 7-11, 2017

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, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, REVEAL, 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.

Agenda

- Recent Cray performance tools enhancements
- What’s coming next?
- Managing MCDRAM usage on Intel® Xeon Phi™ 7250 (hereafter referred to as "KNL”)

Cray Performance Tools

- Cray tools offer functionality that reduces the time investment associated with porting and tuning applications on Cray systems

- Whole program performance analysis across many nodes to help you find critical performance bottlenecks within a program

- Novice and advanced user interfaces for ease of use

Highlights Since Last CUG

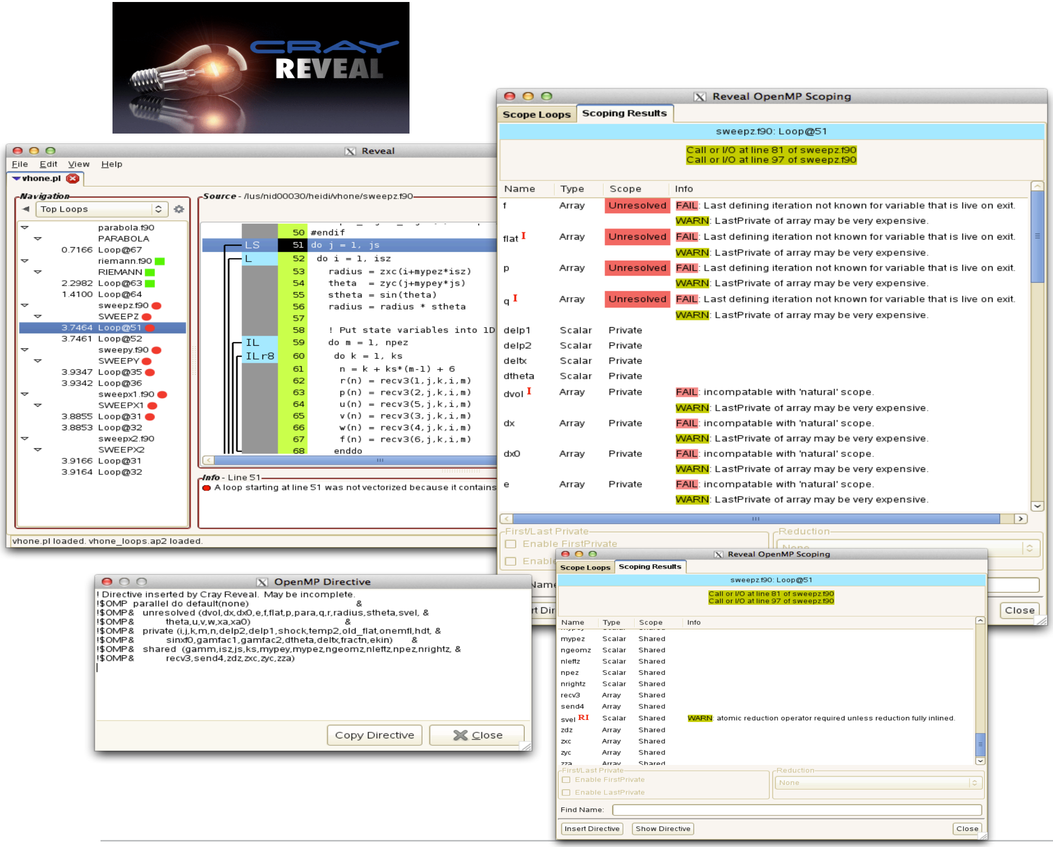
- **Switch to perftools-base + instrumentation modules (6.4.0)**
  - perftools-base provides access Reveal, Apprentice2, pat_report and man pages without modification to applications
  - **perftools-base** loaded by default starting with **cdt-prgenv 6.0.4** (May 2017)
  - Load an instrumentation module to collect performance data

```
Examples:

$ module load perftools-lite
$ module load perftools
```

Highlights Since Last CUG (continued)

- **Memory high water mark per NUMA domain (6.4.4)**
- **Charm++ support (6.4.4)**
  - Build and Run Charm++ and NAMD on the Cray XC with CrayPat
    - http://docs.cray.com/books/S-2802-10//S-2802-10.pdf
- **Address job termination issues for OpenMP programs built with Intel compiler (6.4.4)**
- **MCDRAM configuration statistics included in job summary (6.4.4)**
- **HBM memory analysis tool (6.4.4)**

Reveal



```
parabola.f90
  PARABOLA
0.7166 Loop@67
riemann.f90
  RIEMANN
2.2982 Loop@63
1.4100 Loop@64
sweepz.f90
  SWEEPZ
3.7464 Loop@51
3.7461 Loop@52
sweepy.f90
  SWEEPY
3.9347 Loop@35
3.9342 Loop@36
sweepx1.f90
  SWEEPX1
3.8855 Loop@31
3.8853 Loop@32
sweepx2.f90
  SWEEPX2
3.9166 Loop@31
3.9164 Loop@32
```

Source - /lus/nid00030/heidi/vhone/sweepz.f90

```
         50 #endif
LS       51 do j = 1, js
L        52  do i = 1, isz
         53    radius = zxc(i+mypez*isz)
         54    theta  = zyc(j+mypey*js)
         55    stheta = sin(theta)
         56    radius = radius * stheta
         57
         58    ! Put state variables into 1D
IL       59    do m = 1, npez
ILr8     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,j,k,i,m)
         67      f(n) = recv3(6,j,k,i,m)
         68     enddo
```

Info - Line 51

A loop starting at line 51 was not vectorized because it contains

vhone.pl loaded. vhone_loops.ap2 loaded.

sweepz.f90: Loop@51

Call or I/O at line 81 of sweepz.f90
Call or I/O at line 97 of sweepz.f90

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 I	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit. WARN: LastPrivate of array may be very expensive.
p	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit. WARN: LastPrivate of array may be very expensive.
q I	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit. WARN: LastPrivate of array may be very expensive.
delp1	Scalar	Private	
delp2	Scalar	Private	
deltx	Scalar	Private	
dtheta	Scalar	Private	
dvol I	Array	Private	FAIL: incompatable with 'natural' scope. WARN: LastPrivate of array may be very expensive.
dx	Array	Private	FAIL: incompatable with 'natural' scope. WARN: LastPrivate of array may be very expensive.
dx0	Array	Private	FAIL: incompatable with 'natural' scope. WARN: LastPrivate of array may be very expensive.
e	Array	Private	FAIL: incompatable with 'natural' scope. WARN: LastPrivate of array may be very expensive.

```
! 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,stheta,svel, &
!$OMP&          theta,u,v,w,xa,xa0)                          &
!$OMP&   private (i,j,k,m,n,delp2,delp1,shock,temp2,old_flat,onemfl,hdt, &
!$OMP&          sinxf0,gamfac1,gamfac2,dtheta,deltx,fractn,ekin)      &
!$OMP&   shared  (gamm,isz,js,ks,mypey,mypez,ngeomz,nleftz,npez,nrightz, &
!$OMP&          recv3,send4,zdz,zxc,zyc,zza)
```

sweepz.f90: Loop@51

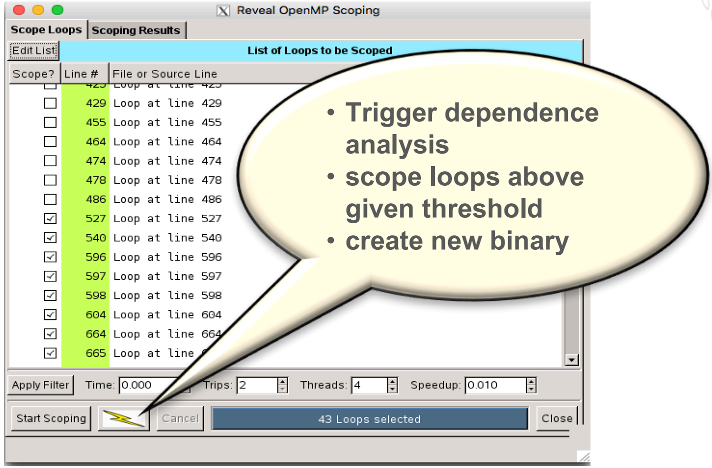
Call or I/O at line 81 of sweepz.f90
Call or I/O at line 97 of sweepz.f90

Name	Type	Scope	Info
mypez	Scalar	Shared	
ngeomz	Scalar	Shared	
nleftz	Scalar	Shared	
npez	Scalar	Shared	
nrightz	Scalar	Shared	
recv3	Array	Shared	
send4	Array	Shared	
svel RI	Scalar	Shared	WARN: atomic reduction operator required unless reduction fully inlined.
zdz	Array	Shared	
zxc	Array	Shared	
zyc	Array	Shared	
zza	Array	Shared	

- Reduce effort associated with adding OpenMP to MPI programs
- Get insight into optimizations performed by the Cray compiler
- Use to add OpenMP or 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

Reveal Enhancements

- **Reveal auto-parallelization (6.4.0)**
  - With one-click, build experimental binary that includes automatic runtime-assisted parallelization
- **Reveal client for Mac OS X (6.4.0)**
  - Improved tool response time with client that executes locally on laptop



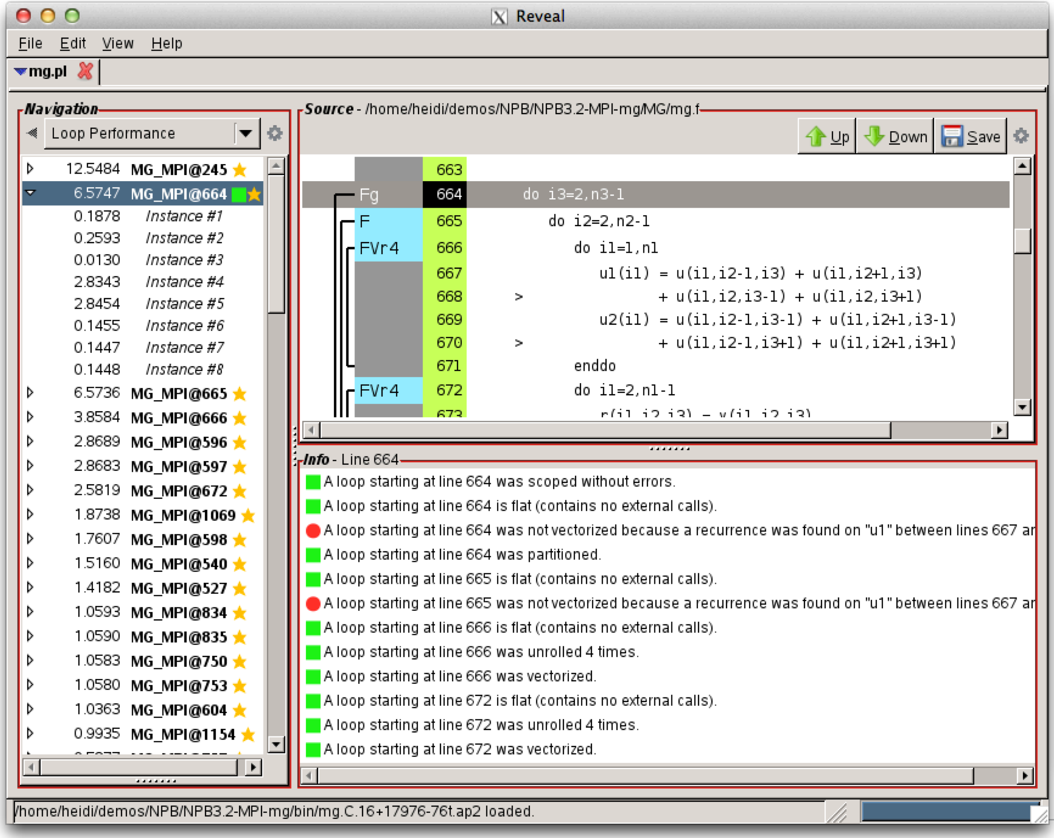
Scope?	Line #	File or Source Line
□	425	Loop at line 425
□	429	Loop at line 429
□	455	Loop at line 455
□	464	Loop at line 464
□	474	Loop at line 474
□	478	Loop at line 478
□	486	Loop at line 486
☑	527	Loop at line 527
☑	540	Loop at line 540
☑	596	Loop at line 596
☑	597	Loop at line 597
☑	598	Loop at line 598
☑	604	Loop at line 604
☑	664	Loop at line 664
☑	665	Loop at line 6

Time: 0.000 Trips: 2 Threads: 4 Speedup: 0.010

43 Loops selected

- Trigger dependence analysis
- scope loops above given threshold
- create new binary

Reveal Auto-Parallelization Feedback



```
▷  12.5484  MG_MPI@245  ★
▽   6.5747  MG_MPI@664  🟩★
     0.1878   Instance #1
     0.2593   Instance #2
     0.0130   Instance #3
     2.8343   Instance #4
     2.8454   Instance #5
     0.1455   Instance #6
     0.1447   Instance #7
     0.1448   Instance #8
▷   6.5736  MG_MPI@665  ★
▷   3.8584  MG_MPI@666  ★
▷   2.8689  MG_MPI@596  ★
▷   2.8683  MG_MPI@597  ★
▷   2.5819  MG_MPI@672  ★
▷   1.8738  MG_MPI@1069 ★
▷   1.7607  MG_MPI@598  ★
▷   1.5160  MG_MPI@540  ★
▷   1.4182  MG_MPI@527  ★
▷   1.0593  MG_MPI@834  ★
▷   1.0590  MG_MPI@835  ★
▷   1.0583  MG_MPI@750  ★
▷   1.0580  MG_MPI@753  ★
▷   1.0363  MG_MPI@604  ★
▷   0.9935  MG_MPI@1154 ★
```

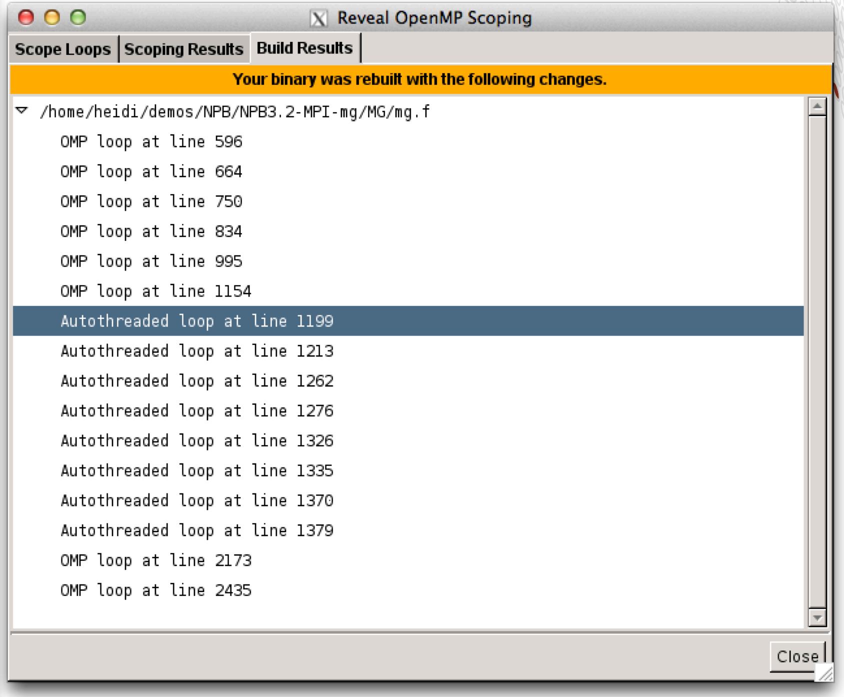
Source - /home/heidi/demos/NPB/NPB3.2-MPI-mg/MG/mg.f

```
        663
 Fg     664          do i3=2,n3-1
 F      665             do i2=2,n2-1
 FVr4   666                do i1=1,n1
        667                   u1(i1) = u(i1,i2-1,i3) + u(i1,i2+1,i3)
        668      >                   + u(i1,i2,i3-1) + u(i1,i2,i3+1)
        669                   u2(i1) = u(i1,i2-1,i3-1) + u(i1,i2+1,i3-1)
        670      >                   + u(i1,i2-1,i3+1) + u(i1,i2+1,i3+1)
        671                enddo
 FVr4   672                do i1=2,n1-1
        673                   r(i1,i2,i3) = v(i1,i2,i3)
```

Info - Line 664

- 🟩 A loop starting at line 664 was scoped without errors.
- 🟩 A loop starting at line 664 is flat (contains no external calls).
- ● A loop starting at line 664 was not vectorized because a recurrence was found on "u1" between lines 667 ar
- 🟩 A loop starting at line 664 was partitioned.
- 🟩 A loop starting at line 665 is flat (contains no external calls).
- ● A loop starting at line 665 was not vectorized because a recurrence was found on "u1" between lines 667 ar
- 🟩 A loop starting at line 666 is flat (contains no external calls).
- 🟩 A loop starting at line 666 was unrolled 4 times.
- 🟩 A loop starting at line 666 was vectorized.
- 🟩 A loop starting at line 672 is flat (contains no external calls).
- 🟩 A loop starting at line 672 was unrolled 4 times.
- 🟩 A loop starting at line 672 was vectorized.

/home/heidi/demos/NPB/NPB3.2-MPI-mg/bin/mg.C.16+17976-76t.ap2 loaded.



Your binary was rebuilt with the following changes.

```
▽ /home/heidi/demos/NPB/NPB3.2-MPI-mg/MG/mg.f
    OMP loop at line 596
    OMP loop at line 664
    OMP loop at line 750
    OMP loop at line 834
    OMP loop at line 995
    OMP loop at line 1154
    Autothreaded loop at line 1199
    Autothreaded loop at line 1213
    Autothreaded loop at line 1262
    Autothreaded loop at line 1276
    Autothreaded loop at line 1326
    Autothreaded loop at line 1335
    Autothreaded loop at line 1370
    Autothreaded loop at line 1379
    OMP loop at line 2173
    OMP loop at line 2435
```

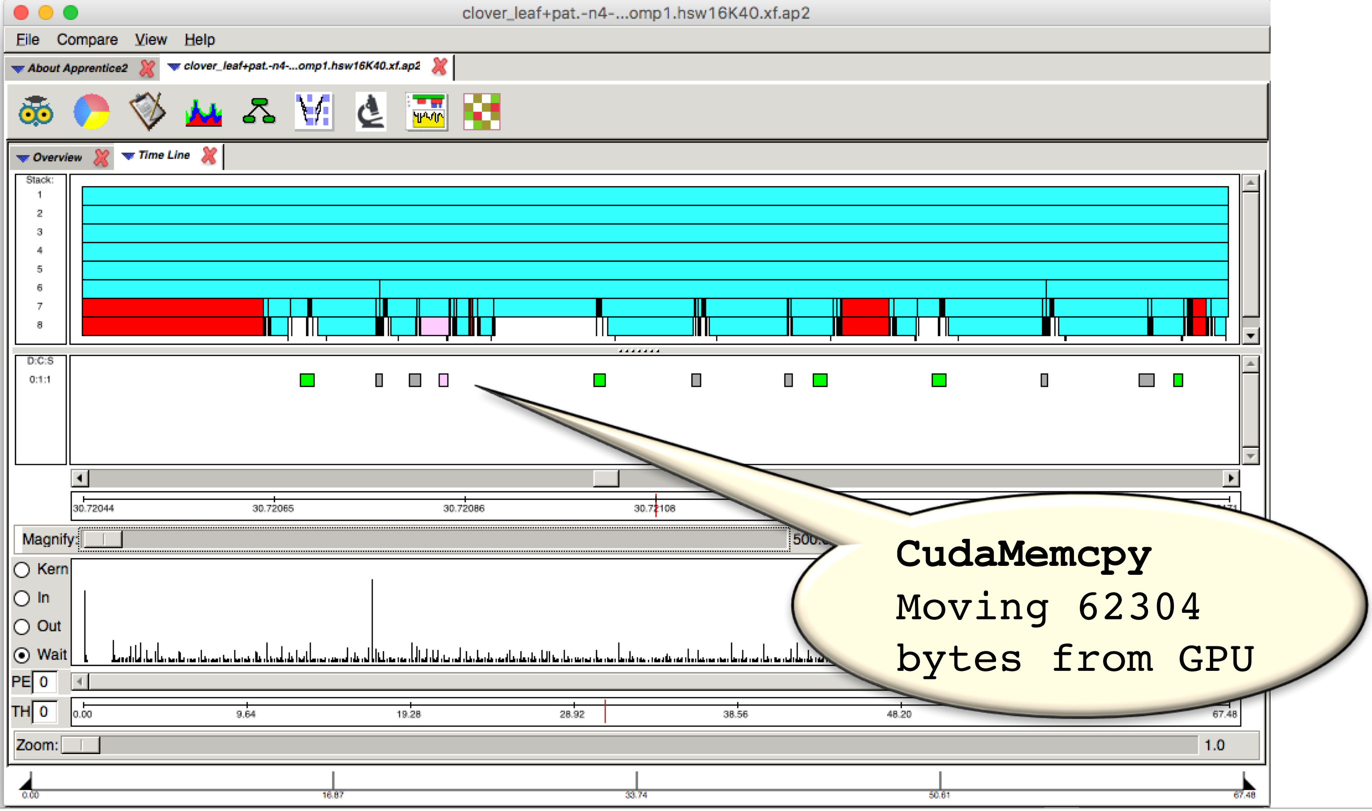
What’s Coming Next..

Functionality Coming Next

- CrayPat lite mode enhancement to disable instrumentation of test programs built through build systems (CMake/GNU Autotools) as part of application build
  - export CRAYPAT_LITE_BLACKLIST=test1,test2

- CUDA support in Apprentice2 (overview, GPU timeline)

CUDA Support in Apprentice2



CudaMemcpy Moving 62304 bytes from GPU

Functionality Coming Next (Continued)

- Consolidation of Cray performance tools results files into single directory
  - MPICH_RANK_ORDER.Grid,
  - MPICH_RANK_ORDER.USER_Time
  - MPICH_RANK_ORDER.USER_Time_hybrid
  - stencil_order+pat+49144-225t.xf
  - stencil_order+pat+49144-225t.ap2
  - stencil_order+pat+49144-225t.rpt
  - stencil_order+pat+49144-225t.apa

- Same prefix naming scheme as used with current xf files
  - stencil_order+pat+49144-225t/

HBM Memory Analysis Assistance

Managing Multi-tiered Memory

- **Future systems are likely to include a high performance tier (for bandwidth or latency) and a high capacity tier at a lower cost**

- **Our goals to assist users with using more than one type of explicitly addressable on-node memory:**
  - Provide easy-to-use interface for user to allocate data into HBM
  - Provide assistance with making best use of limited capacity

Identifying Arrays Is Difficult

```
subroutine ax_e()

do i=1,n
     wr = g(1,i)*ur(i) + g(2,i)*us(i) + g(3,i)*ut(i)
     ws = g(2,i)*ur(i) + g(4,i)*us(i) + g(5,i)*ut(i)
     wt = g(3,i)*ur(i) + g(5,i)*us(i) + g(6,i)*ut(i)
     ur(i) = wr
     us(i) = ws
     ut(i) = wt
enddo
```

```
subroutine glsc3()

do i=1,n
     tmp = tmp + a(i)*b(i)*mult(i)
continue
```

Arrays `a`, `b`, and `mult` have a higher bandwidth sensitivity than array `g`

MCDRAM Usage Assistance

- Combination of CCE, CrayPat and Reveal are used to identify arrays that contribute most to memory bandwidth

- First introduced in December 2016
  - cce/8.5.6
  - perftools-base/6.4.4

- See http://docs.cray.com/books/S-2803-10//S-2803-10.pdf

Perform Analysis on Xeon or Phi

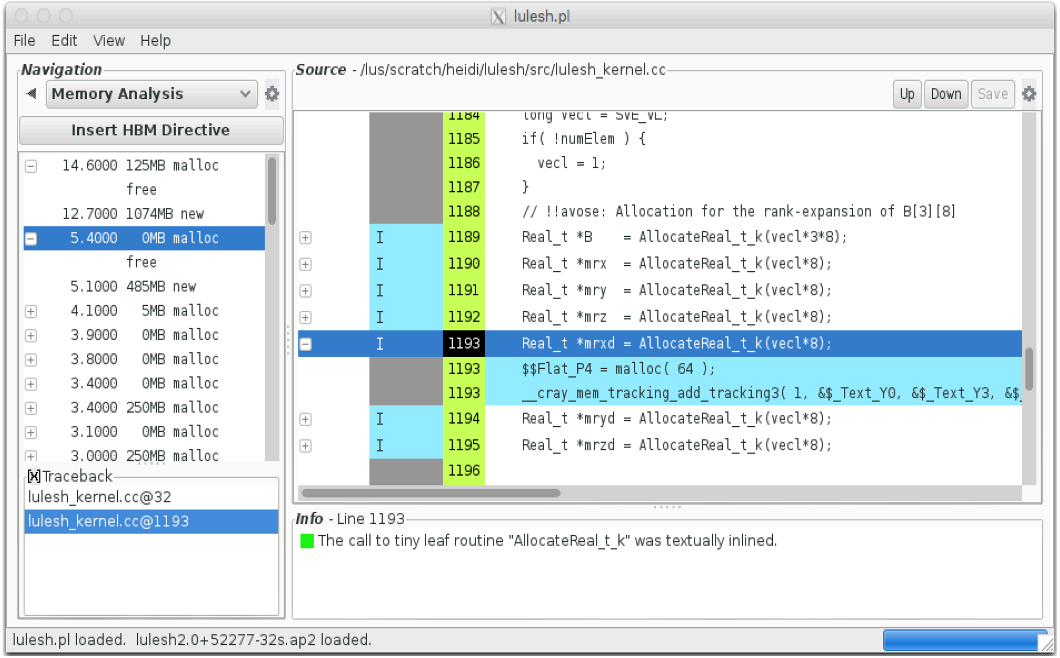
- **Haswell, Broadwell or KNL processors supported**
  - Memory traffic due to the hardware prefetcher on KNL is untracked
  - HSW is not KNL (differences with page table walks, L3 cache, etc.)
  - Prefer HSW for applications with streaming accesses

```
$ module load PrgEnv-cray, craype-haswell
$ module load perftools-lite-hbm

$ cc –h pl=/path/my_program.pl my_program.c
```

Run program (no batch script modifications required) to create `.ap2` file

```
$ reveal my_program.pl traffic_results.ap2
```

Ranked Arrays and Allocation Sites



```
 14.6000 125MB malloc
               free
 12.7000 1074MB new
  5.4000   0MB malloc
               free
  5.1000 485MB new
  4.1000   5MB malloc
  3.9000   0MB malloc
  3.8000   0MB malloc
  3.4000   0MB malloc
  3.4000 250MB malloc
  3.1000   0MB malloc
  3.0000 250MB malloc
```

lulesh_kernel.cc@32
lulesh_kernel.cc@1193

```
     1184  long vecl = SVE_VL;
     1185  if( !numElem ) {
     1186    vecl = 1;
     1187  }
     1188  // !!avose: Allocation for the rank-expansion of B[3][8]
  I  1189  Real_t *B    = AllocateReal_t_k(vecl*3*8);
  I  1190  Real_t *mrx  = AllocateReal_t_k(vecl*8);
  I  1191  Real_t *mry  = AllocateReal_t_k(vecl*8);
  I  1192  Real_t *mrz  = AllocateReal_t_k(vecl*8);
  I  1193  Real_t *mrxd = AllocateReal_t_k(vecl*8);
     1193  $$Flat_P4 = malloc( 64 );
     1193  __cray_mem_tracking_add_tracking3( 1, &$_Text_Y0, &$_Text_Y3, &$_
  I  1194  Real_t *mryd = AllocateReal_t_k(vecl*8);
  I  1195  Real_t *mrzd = AllocateReal_t_k(vecl*8);
     1196
```

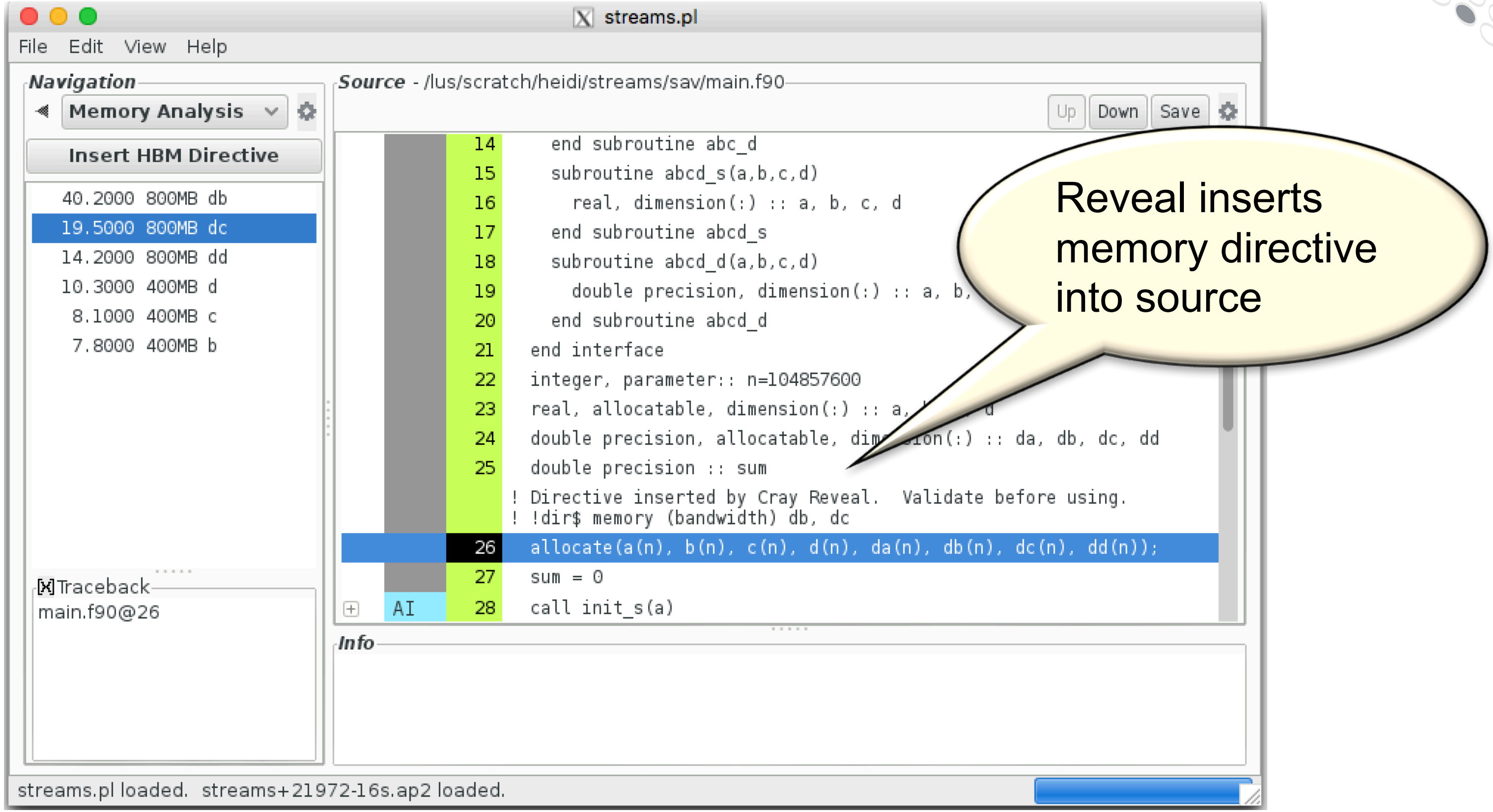
Info - Line 1193

🟩 The call to tiny leaf routine "AllocateReal_t_k" was textually inlined.

lulesh.pl loaded. lulesh2.0+52277-32s.ap2 loaded.

- **Objects sorted by memory bandwidth contribution**
- **Match free()s to mallocs()**
- **For C++, Reveal shows how STL objects rank with other arrays**
  - User must find declaration site, modify declaration to point to an hbw-aware allocator

Build CCE Memory Allocation Directive



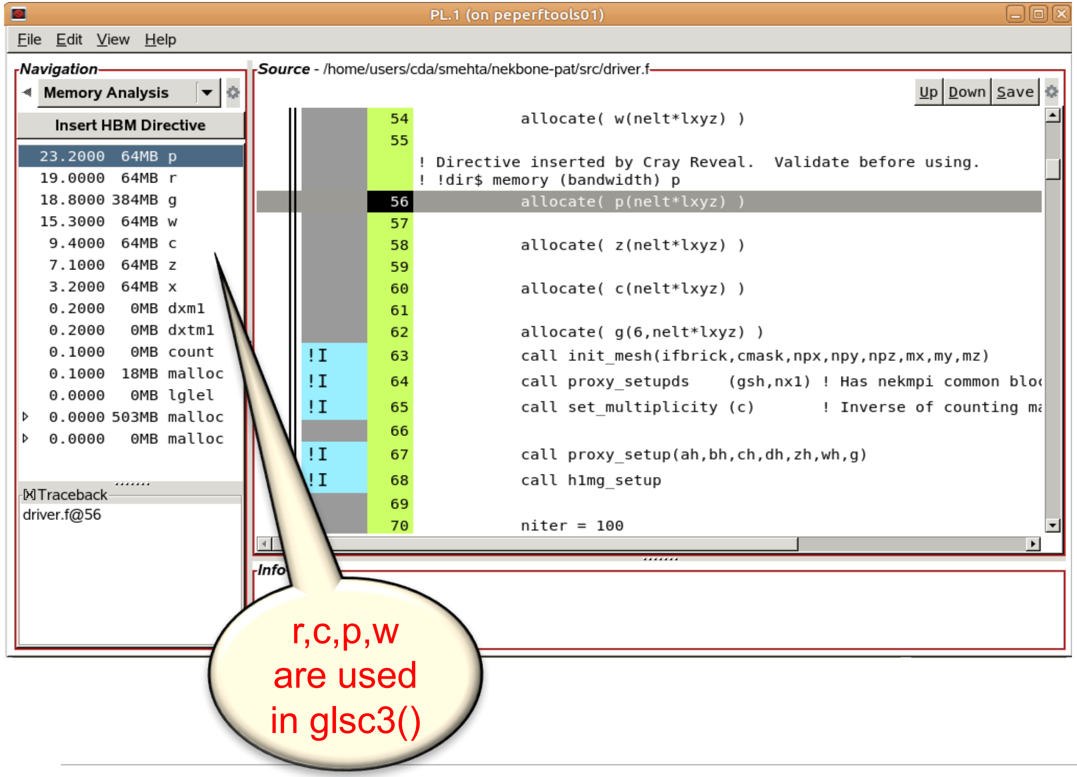
```
40.2000  800MB  db
19.5000  800MB  dc
14.2000  800MB  dd
10.3000  400MB  d
 8.1000  400MB  c
 7.8000  400MB  b
```

```
    end subroutine abc_d
    subroutine abcd_s(a,b,c,d)
      real, dimension(:) :: a, b, c, d
    end subroutine abcd_s
    subroutine abcd_d(a,b,c,d)
      double precision, dimension(:) :: a, b,
    end subroutine abcd_d
  end interface
  integer, parameter:: n=104857600
  real, allocatable, dimension(:) :: a,     d
  double precision, allocatable, dim   ion(:) :: da, db, dc, dd
  double precision :: sum
! Directive inserted by Cray Reveal.  Validate before using.
! !dir$ memory (bandwidth) db, dc
  allocate(a(n), b(n), c(n), d(n), da(n), db(n), dc(n), dd(n));
  sum = 0
  call init_s(a)
```

streams.pl loaded. streams+21972-16s.ap2 loaded.

Reveal inserts memory directive into source

Memory Analysis Results for Nekbone



```
23.2000  64MB p
19.0000  64MB r
18.8000 384MB g
15.3000  64MB w
 9.4000  64MB c
 7.1000  64MB z
 3.2000  64MB x
 0.2000   0MB dxm1
 0.2000   0MB dxtm1
 0.1000   0MB count
 0.1000  18MB malloc
 0.0000   0MB lglel
 0.0000 503MB malloc
 0.0000   0MB malloc
```

☒Traceback
driver.f@56

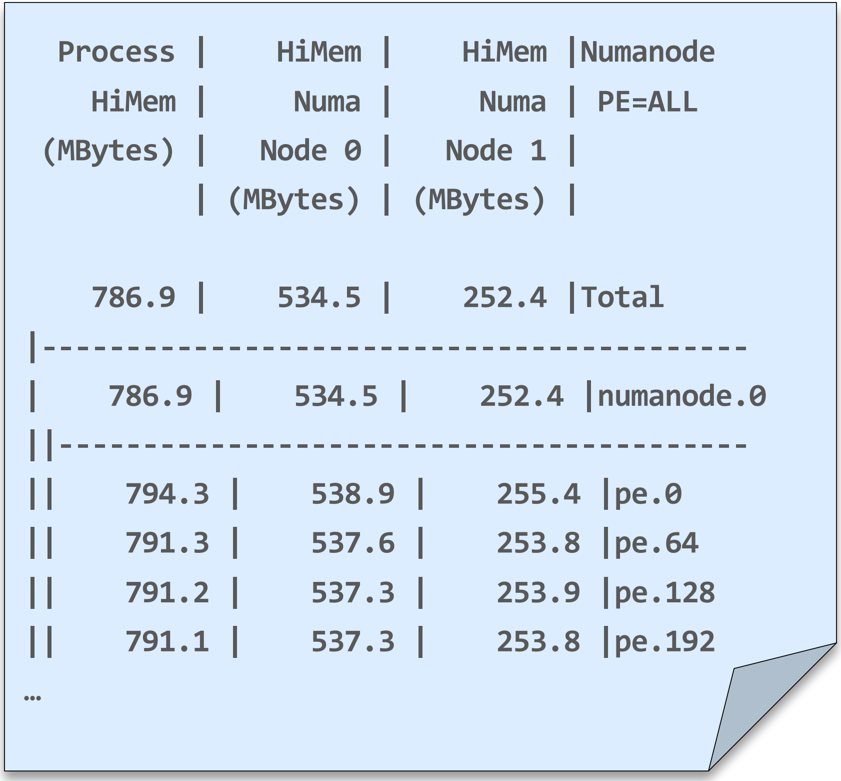
```
              allocate( w(nelt*lxyz) )

! Directive inserted by Cray Reveal.  Validate before using.
! !dir$ memory (bandwidth) p
              allocate( p(nelt*lxyz) )

              allocate( z(nelt*lxyz) )

              allocate( c(nelt*lxyz) )

              allocate( g(6,nelt*lxyz) )
!I            call init_mesh(ifbrick,cmask,npx,npy,npz,mx,my,mz)
!I            call proxy_setupds    (gsh,nx1) ! Has nekmpi common bloc
!I            call set_multiplicity (c)       ! Inverse of counting ma

!I            call proxy_setup(ah,bh,ch,dh,zh,wh,g)
!I            call h1mg_setup

              niter = 100
```

r,c,p,w
are used
in glsc3()

- **Low data collection overhead**
  - ~1% - few %
  - Can analyze large jobs
  - Functions that are entered and exited often and allocate arrays big enough to be tracked can increase runtime

Use Combination of Reveal and Text Report

- **Check program memory footprint**
  - Available at top of report with job summary information
  - If less than 16GB, use `numactl --membind=1` to force all allocations into MCDRAM
- **Get ranking of which objects are referenced the most**
- **Pinpoint memory activity**
- **Review memory high water mark per NUMA domain**



```
   Process |     HiMem |     HiMem |Numanode
     HiMem |      Numa |      Numa | PE=ALL
  (MBytes) |    Node 0 |    Node 1 |
           |  (MBytes) |  (MBytes) |

     786.9 |     534.5 |     252.4 |Total
|-----------------------------------------
|     786.9 |     534.5 |     252.4 |numanode.0
||----------------------------------------
||     794.3 |     538.9 |     255.4 |pe.0
||     791.3 |     537.6 |     253.8 |pe.64
||     791.2 |     537.3 |     253.9 |pe.128
||     791.1 |     537.3 |     253.8 |pe.192
…
```

Where to Find Memory Activity in Reports

- **High water mark by allocation site**
  - Number of all objects active at any time
- **Profile by Function table**
  - Shows where most of the memory traffic is happening within program
- **Profile by Group, Function, and Line table**
  - Identifies memory traffic hot spots within a function

MCDRAM Allocation Assistance Recap

- Cray Tools track requests to memory and evaluate the bandwidth contribution of objects within a program

- Helpful for memory-intensive programs that cannot fit within MCDRAM

- Reduces time investment associated with selectively allocating data into KNL's MCDRAM

- The result is performance portable code
  - CCE memory allocation directives are ignored on X86 processors