HPE CRAY PROGRAMMING ENVIRONMENT

Isambard Hackathon

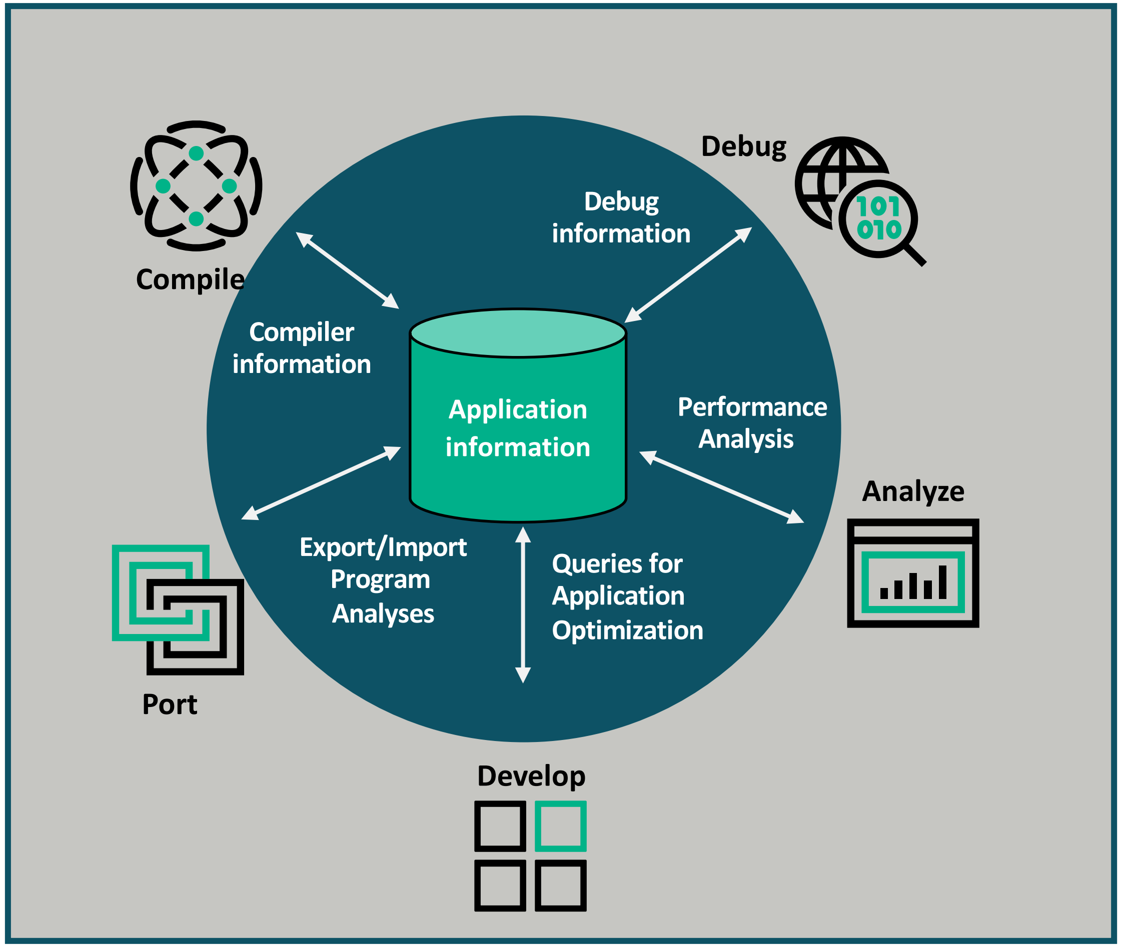
March 23, 2021

John M. Levesque
John.levesque@hpe.com
Technical Advisor for Cray Programming Environment
Distinguished Technologist
CTO Office

CRAY PROGRAMMING ENVIRONMENT

Why Cray PE?

- Full, production quality developer software stack that targets performance and programmability at scale
- Minimized time to productive use of system
- Focuses on real HPC applications, not just benchmarks
- Issues addressed in a timely fashion
- Traditional HPC capability coupled with advancing technology to embrace new problems
- Engagement with customers to understand goals and challenges



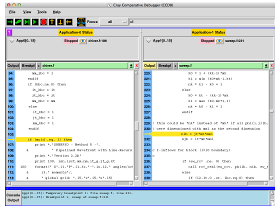
CRAY PE TECHNOLOGY APPLIED TO ARM SVE

- Cray technology **designed for real scientific applications**, not just for benchmarks

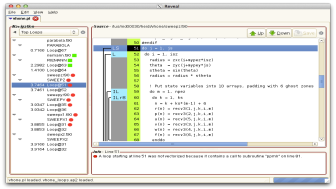
- **Modules simplify build environment**

- **Fortran, C, and C++ compilers**
  - Most complete vectorization capabilities that have evolved from custom Cray vectorizing processors, including unique outer loop vectorization technology
    - SVE instruction set has many features familiar to Cray's compiler such as wider vector widths, predication, gather/scatter, etc.

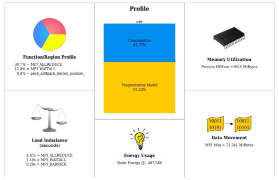
- Scalable debug tools like **gdb4hpc** and **CCDB** assist with porting to new processors or platforms



- **Scientific Libraries** extract maximum performance using standard interfaces for ease of use



- **Cray Reveal** combines compiler knowledge of the application with profiling information
  - Unique tool in industry that significantly reduces effort associated with adding OpenMP
    - Scoping tool to help users port and optimize applications



- **Cray Performance tools** deliver wealth of profiling capability including whole-program view of performance with simple-to-use interfaces

COMPILING ENVIRONMENT ADVANTAGES

We designed our compilers for real-life applications, not just benchmarks

Performance and programmability

- Our compiler exploits the scalar and vector hardware capabilities of the systems
- Compiler optimization feedback for application tuning

Fully integrated heterogeneous optimization capability

- Providing consistency across all HPE and HPE Cray systems, supporting:
  - X86-64 (both Intel & AMD) processors
  - Arm-based processors
  - GPUs

Integration with tools for performance optimization

- Integration with parallelization assistant through Program Library technology for adding OpenMP to your applications
- Integration with performance analysis tools for additional optimization insight

Focus on application portability and investment protection

Focus on compliance with latest standards and language support:

- Languages: Fortran with coarrays, C/C++ and UPC
- Programming models: OpenMP and OpenACC

CRAY COMPILER FOR BAYMAX

- Supports native compilation (no cross-compiling)
- Supports OpenMP, Cray Reveal
- Offers loopmark (compiler feedback)

- cce-sve
  - Fortran and C/C++ compiler generates ARM SVE code
  - C/C++ compiler based on Cray classic compiler (EDG front-end)
    – Not as strong C++ support

- cce
  - Released in Sept to provide stronger compiler for C++ code
  - Fortran and C/C++ compiler generates ARM Neon vector code
  - C/C++ compiler based on new Cray clang compiler (LLVM)
    – ARM SVE code generation coming when it’s available in LLVM (targeting spring of 2021)
    – Will move to SVE code generation for Fortran and C/C++

- Usage guidance
  - Choose compiler based on dominant/most important code (C++ strength vs Fortran SVE strength)
  - Cannot mix use between two compiling environments
  - Libsci targets Fujitsu A64FX processor and can be used with either cce-sve or cce compilers

MODULES FOR RUNNING WITH SVE AND PERFTOOLS

```
Currently Loaded Modulefiles:
 1) cpe-cray                     5) craype-network-infiniband
 2) cce-sve/10.0.2(default)      6) cray-libsci/20.10.1.2(default)
 3) craype/2.7.4(default)        7) slurm/slurm/no-version
 4) craype-arm-nsp1              8) perftools-base/21.02.0(default)
 9) perftools-lite               10) cray-mvapich2_nogpu_noregcache_sve/2.3.5(default)
```

MODULES FOR RUNNING WITH SVE WITHOUT PERFTOOLS

```
Currently Loaded Modulefiles:
 1) cpe-cray                   5) craype-network-infiniband
 2) cce-sve/10.0.2(default)    6) cray-libsci/20.10.1.2(default)
 3) craype/2.7.4(default)      7) slurm/slurm/no-version
 4) craype-arm-nsp1            8) cray-mvapich2_nogpu_sve/2.3.5(default)
```

LESLIE3D

- Three Dimension Computational Fluid Dynamics Code from Georgia Tech

SIMPLE PROFILE

Perftools-lite places profiles into standard out. - all exclusive times

```
Table 1:  Profile by Function
  Samp% |     Samp |  Imb. |  Imb. | Group
        |          |  Samp | Samp% |  Function=[MAX10]
        |          |       |       |   PE=HIDE
 100.0% | 3,586.7 |    -- |    -- | Total
|-------------------------------------------------------
|  68.7% | 2,465.6 |    -- |    -- | USER
||------------------------------------------------------
||  19.2% |   687.9 |  40.1 |  5.7% | fluxk_.LOOP@li.38
||  17.0% |   611.0 |  19.0 |  3.1% | fluxi_.LOOP@li.30
||  13.7% |   493.1 |  11.9 |  2.4% | fluxj_.LOOP@li.38
||   4.6% |   165.8 |  13.2 |  7.7% | extrapk_.LOOP@li.158
||   3.0% |   106.8 |  10.2 |  9.1% | update_.LOOP@li.18
||   2.8% |    99.5 |  15.5 | 13.9% | extrapj_.LOOP@li.159
||======================================================
|  28.6% | 1,024.1 |    -- |    -- | MPI
||------------------------------------------------------
||  11.2% |   401.4 |  74.6 | 16.3% | MPI_SEND
||   9.7% |   348.3 | 131.7 | 28.5% | MPI_ALLREDUCE
||   5.3% |   191.1 | 221.9 | 55.7% | MPI_REDUCE
||======================================================
|   2.6% |    94.8 |    -- |    -- | ETC
||------------------------------------------------------
||   2.2% |    78.7 |  61.3 | 45.4% | __ALLOCATE
|=======================================================
```

Sampling run 1/100 sec

Column

1 Percent

2 # samples

3 #imblanced samples

4 Percent of imbalance

5 Code segment

LINE LEVEL PROFILE

Perftools-lite places profiles into standard out

```
Table 2:  Profile by Group, Function, and Line

  Samp% |     Samp |  Imb. |  Imb. | Group
        |          |  Samp | Samp% |  Function=[MAX10]
        |          |       |       |   Source
        |          |       |       |    Line
        |          |       |       |     PE=HIDE

 100.0% | 3,586.7 |    -- |    -- | Total
|------------------------------------------------------------------------
|  68.7% | 2,465.6 |    -- |    -- | USER
||-----------------------------------------------------------------------
||  19.2% |   687.9 |    -- |    -- | fluxk_.LOOP@li.38
3|        |         |       |       |  levesque/leslie3d_mpi_orig/src/fluxk.f
||||---------------------------------------------------------------------
4|||   1.4% |    51.5 |  12.5 | 20.3% | line.41
4|||  13.8% |   495.5 |  31.5 |  6.2% | line.75
4|||   2.4% |    85.6 |  30.4 | 27.2% | line.84
||||=====================================================================
||  17.0% |   611.0 |    -- |    -- | fluxi_.LOOP@li.30
3|        |         |       |       |  levesque/leslie3d_mpi_orig/src/fluxi.f
||||---------------------------------------------------------------------
4|||  14.0% |   502.3 |  20.7 |  4.1% | line.65
4|||   1.0% |    37.0 |  12.0 | 25.4% | line.72
```

Sampling run 1/100 sec

Column

1 Percent

2 # samples

3 #imblanced samples

4 Percent of imbalance

5 Code segment

6 Line level

ALWAYS USE –HLIST=A TO GET COMPILER LISTING

```
+ F---------------<        DO J = 1, JCMAX
+ F F-------------<          DO K = 0, KCMAX
  F F V-----------<           DO I = 1, IND
  F F V                         QS(I) = UAV(I,J,K) * SKX(I,J,K) +
  F F V                >                  VAV(I,J,K) * SKY(I,J,K) +
  F F V                >                  WAV(I,J,K) * SKZ(I,J,K)
  F F V
  F F V                         IF ( NSCHEME .EQ. 2 ) THEN
  F F V                            L = K + 1 - KADD
  F F V                               QSP = U(I,J,L) * SKX(I,J,K) +
  F F V                >                  V(I,J,L) * SKY(I,J,K) +
  F F V                >                  W(I,J,L) * SKZ(I,J,K)
  F F V                               QSPK = (QSP - QS(I)) * DBLE(1 - 2 * KADD)
  F F V                               IF (QSPK .GT. 0.0D+00) QS(I) = 0.5D+00 * (QS(I) + QSP)
  F F V                         ENDIF
  F F V
  F F V                         FSK(I,K,1) =  QAV(I,J,K,1) * QS(I)
  F F V                         FSK(I,K,2) =  QAV(I,J,K,2) * QS(I) +
  F F V                >                            PAV(I,J,K) * SKX(I,J,K)
  F F V                         FSK(I,K,3) =  QAV(I,J,K,3) * QS(I) +
  F F V                >                            PAV(I,J,K) * SKY(I,J,K)
  F F V                         FSK(I,K,4) =  QAV(I,J,K,4) * QS(I) +
  F F V                >                            PAV(I,J,K) * SKZ(I,J,K)
  F F V                         FSK(I,K,5) = (QAV(I,J,K,5) + PAV(I,J,K)) *
  F F V                >                                                QS(I)
  F F V
  F F V                         IF (ISGSK .EQ. 1) THEN
  F F V                            FSK(I,K,7) = QAV(I,J,K,7) * QS(I)
  F F V                         ENDIF
  F F V
  F F V                         IF ( ICHEM .GT. 0 ) THEN
  F F V D---------<                DO L = 8, 7 + NSPECI
  F F V D                             FSK(I,K,L) = QAV(I,J,K,L) * QS(I)
  F F V D--------->                ENDDO
  F F V                         ENDIF
  F F V----------->           ENDDO
  F F
  F F V I--------<>             IF ( VISCOUS ) CALL VISCK ( 1, IND, J, K, FSK )
  F F------------->          ENDDO
```

NUMA INFORMATION

Perftools-lite places profiles into standard out

```
Table 3:  Memory Bandwidth by Numanode

  Memory |     Read |    Write |     Thread |   Memory |   Memory | Numanode
 Traffic |   Memory |   Memory |       Time |  Traffic |  Traffic |  Node Id=[max3,min3]
  GBytes |  Traffic |  Traffic |            |   GBytes |        / |   PE=HIDE
         |   GBytes |   GBytes |            |    / Sec |  Nominal |
         |          |          |            |          |     Peak |
|------------------------------------------------------------------------------
|   736.29 |   596.75 |   139.54 | 36.205149 |    20.34 |     7.9% | numanode.0
||-----------------------------------------------------------------------------
||   736.29 |   596.75 |   139.54 | 36.205149 |    20.34 |     7.9% | nid.19
||   731.44 |   592.71 |   138.73 | 36.156605 |    20.23 |     7.9% | nid.18
||   718.71 |   582.49 |   136.22 | 36.149776 |    19.88 |     7.8% | nid.17
||   703.97 |   570.05 |   133.92 | 36.156420 |    19.47 |     7.6% | nid.22
||   703.83 |   569.91 |   133.91 | 36.196569 |    19.44 |     7.6% | nid.20
||   689.34 |   558.02 |   131.32 | 36.157255 |    19.07 |     7.4% | nid.23
|==============================================================================
```

SAMPLING WITH CALL TREE

To get call-tree pat_report –Oct <experiment directory>

```
Table 1:  Function Calltree View
  Samp% |     Samp | Calltree
        |          |  PE=HIDE
 100.0% | 3,586.7 | Total
|-----------------------------------------------
|  79.9% | 2,867.5 | les3d_
||----------------------------------------------
||  23.8% |   855.0 | fluxk_
3|  23.8% |   854.9 |  fluxk_.REGION@li.33(ovhd)
||||--------------------------------------------
4|||  19.2% |   688.6 | fluxk_.REGION@li.33
5|||  19.2% |   688.0 |  fluxk_.LOOP@li.38
4|||   4.6% |   166.0 | extrapk_
5|||         |          |  extrapk_.REGION@li.156(ovhd)
6|||   4.6% |   165.8 |   extrapk_.REGION@li.156
7|||         |          |    extrapk_.LOOP@li.158
||||============================================
||  18.8% |   674.5 | fluxi_
3|  18.8% |   674.4 |  fluxi_.REGION@li.25(ovhd)
||||--------------------------------------------
4|||  17.0% |   611.5 | fluxi_.REGION@li.25
5|||  17.0% |   611.1 |  fluxi_.LOOP@li.30
4|||   1.7% |    62.7 | extrapi_
5|||         |          |  extrapi_.REGION@li.137(ovhd)
6|||   1.7% |    62.4 |   extrapi_.REGION@li.137
7|||   1.2% |    42.9 |    extrapi_.LOOP@li.141
||||============================================
```

NO MPI STATISTICS WITH PERFTOOLS-LITE, SO LETS TRY PERFTOOLS

Module swap perftools-lite perftools
Compile and link
Pat_build –u –g mpi <executable>
Srun run executable+pat

SIMPLE PROFILE

Pat_report <experiment file>

```
Table 1:  Profile by Function Group and Function


  Time% |      Time |     Imb. |  Imb. |    Calls | Group
        |           |     Time | Time% |          |  Function
        |           |          |       |          |   PE=HIDE


 100.0% | 35.764334 |       -- |    -- | 36,574.1 | Total
|-----------------------------------------------------------------------
|  71.4% | 25.550017 |       -- |    -- |  5,074.6 | USER
||----------------------------------------------------------------------
||  19.3% |  6.895061 | 0.140679 |  2.1% |    200.0 | fluxk_.LOOP@li.38
||  17.1% |  6.112560 | 0.137001 |  2.3% |    200.0 | fluxi_.LOOP@li.30
||  13.7% |  4.902728 | 0.074435 |  1.6% |    200.0 | fluxj_.LOOP@li.38
||   4.6% |  1.639311 | 0.087245 |  5.2% |    200.0 | extrapk_.LOOP@li.158
||   3.0% |  1.062270 | 0.598870 | 37.4% |  1,010.0 | mpicx_
||   2.8% |  1.004442 | 0.025128 |  2.5% |    200.0 | extrapj_.LOOP@li.159
||   1.6% |  0.578741 | 0.013319 |  2.3% |     21.0 | tmstep_
||   1.5% |  0.545074 | 0.011006 |  2.1% |    100.0 | update_
||   1.4% |  0.508350 | 0.037538 |  7.1% |    202.0 | parallel_
||   1.4% |  0.501630 | 0.011941 |  2.4% |    100.0 | update_.LOOP@li.18
||   1.3% |  0.457291 | 0.039064 |  8.2% |    476.1 | ghost_
||   1.1% |  0.409303 | 0.017826 |  4.3% |    200.0 | extrapi_.LOOP@li.141
||======================================================================
```

LOAD IMBALANCE WITH MPI

Pat_report <experiment file>

```
Table 2:  Profile of maximum function times

  Time% |      Time |      Imb. |  Imb. | Function
        |           |      Time | Time% |  PE=[max,min]
|-----------------------------------------------------------
| 100.0% | 7.035740 | 0.140679 |  2.1% | fluxk_.LOOP@li.38
||----------------------------------------------------------
|| 100.0% | 7.035740 |        -- |    -- | pe.8
||  95.9% | 6.745043 |        -- |    -- | pe.21
||==========================================================
|  88.8% | 6.249561 | 0.137001 |  2.3% | fluxi_.LOOP@li.30
||----------------------------------------------------------
||  88.8% | 6.249561 |        -- |    -- | pe.4
||  85.2% | 5.991107 |        -- |    -- | pe.25
||==========================================================
|  70.7% | 4.977162 | 0.074435 |  1.6% | fluxj_.LOOP@li.38
||----------------------------------------------------------
||  70.7% | 4.977162 |        -- |    -- | pe.16
||  68.4% | 4.812507 |        -- |    -- | pe.3
||==========================================================
|  64.6% | 4.542432 | 0.697049 | 15.9% | MPI_SEND
||----------------------------------------------------------
||  64.6% | 4.542432 |        -- |    -- | pe.0
||  48.5% | 3.413080 |        -- |    -- | pe.20
||==========================================================
```

LOAD IMBALANCE AND MESSAGE SIZES WITH MPI

Pat_report <experiment file>

```
Table 3:  Load Balance with MPI Message Stats

  Time% |      Time | MPI Msg |    MPI Msg Bytes |    Avg MPI | Group
        |           |   Count |                  |   Msg Size |  PE=[mmm]

 100.0% | 35.764426 | 9,968.9 | 1,765,373,643.4 | 177,088.87 | Total
|----------------------------------------------------------------------
|  71.4% | 25.550109 |     0.0 |              0.0 |         -- | USER
||---------------------------------------------------------------------
||  78.2% | 27.961406 |     0.0 |              0.0 |         -- | pe.0
||  71.2% | 25.479947 |     0.0 |              0.0 |         -- | pe.9
||  68.5% | 24.497551 |     0.0 |              0.0 |         -- | pe.25
||=====================================================================
|  28.5% | 10.194769 | 9,968.9 | 1,765,373,643.4 | 177,088.87 | MPI
||---------------------------------------------------------------------
||  31.5% | 11.247938 | 8,526.0 | 1,530,361,072.0 | 179,493.44 | pe.25
||  28.5% | 10.186792 | 8,526.0 | 1,540,057,072.0 | 180,630.67 | pe.3
||  21.8% |  7.783913 | 8,526.0 | 1,540,057,072.0 | 180,630.67 | pe.0
|======================================================================
```

MESSAGE SIZES – MIN, MAX, AVE WITH MPI

Pat_report <experiment file>

```
------------------------------------------------------------------------------
  MPI Msg Bytes%                              54.2%
  MPI Msg Bytes                       956,672,000.0
  MPI Msg Count                            10,100.0 msgs
  16<= MsgSz <256 Count                         0.0 msgs
  256<= MsgSz <4KiB Count                       0.0 msgs
  4KiB<= MsgSz <64KiB Count                 6,060.0 msgs
  64KiB<= MsgSz <1MiB Count                 4,040.0 msgs
  16MiB<= MsgSz Count                           0.0 msgs
==============================================================================
  MPI_SEND / mpicx_ / parallel_ / les3d_ / pe.18
------------------------------------------------------------------------------
  MPI Msg Bytes%                              53.6%
  MPI Msg Bytes                       946,976,000.0
  MPI Msg Count                            10,100.0 msgs
  16<= MsgSz <256 Count                         0.0 msgs
  256<= MsgSz <4KiB Count                       0.0 msgs
  4KiB<= MsgSz <64KiB Count                 6,060.0 msgs
  64KiB<= MsgSz <1MiB Count                 4,040.0 msgs
  16MiB<= MsgSz Count                           0.0 msgs
==============================================================================
  MPI_SEND / mpicx_ / parallel_ / les3d_ / pe.27
------------------------------------------------------------------------------
  MPI Msg Bytes%                              35.3%
  MPI Msg Bytes                       623,776,000.0
  MPI Msg Count                             8,080.0 msgs
  16<= MsgSz <256 Count                         0.0 msgs
  256<= MsgSz <4KiB Count                       0.0 msgs
  4KiB<= MsgSz <64KiB Count                 6,060.0 msgs
  64KiB<= MsgSz <1MiB Count                 2,020.0 msgs
```

MESSAGE SIZES – MIN, MAX, AVE WITH MPI

Pat_report –Oct <experiment file>

```
Table 1:  Function Calltree View
  Time% |       Time |    Calls | Calltree
        |            |          |  PE=HIDE
 100.0% | 35.764334 |       -- | Total
|-----------------------------------------------------
|  99.5% | 35.572095 |      1.0 | les3d_
||----------------------------------------------------
||  23.9% |  8.547970 |       -- | fluxk_
|||---------------------------------------------------
3||  19.3% |  6.902694 |    200.0 | fluxk_.REGION@li.33
4||  19.3% |  6.895061 |    200.0 |  fluxk_.LOOP@li.38
3||   4.6% |  1.642442 |       -- | extrapk_
4||   4.6% |  1.640225 |    200.0 |  extrapk_.REGION@li.156
5||   4.6% |  1.639311 |    200.0 |   extrapk_.LOOP@li.158
|||===================================================
||  18.9% |  6.751193 |       -- | fluxi_
|||---------------------------------------------------
3||  17.1% |  6.117161 |    200.0 | fluxi_.REGION@li.25
4||  17.1% |  6.112560 |    200.0 |  fluxi_.LOOP@li.30
3||   1.8% |  0.631296 |       -- | extrapi_
4||   1.8% |  0.628479 |    200.0 |  extrapi_.REGION@li.137
5||   1.1% |  0.409303 |    200.0 |   extrapi_.LOOP@li.141
|||===================================================
||  18.5% |  6.629379 |    202.0 | parallel_
|||---------------------------------------------------
3||  15.8% |  5.663738 | 1,010.0 | mpicx_
||||--------------------------------------------------
4|||  10.8% |  3.845383 | 9,522.9 | MPI_SEND
4|||   3.0% |  1.062270 | 1,010.0 | mpicx_(exclusive)
4|||   1.9% |  0.670306 | 9,522.9 | MPI_WAIT
||||==================================================
```

LETS LOOK AT LOOP INSTRUMENTATION

- Module swap perftools perftools-lite-loops
- DO EVER DO THIS WHEN RUNNING FOR REAL _ MAKE CODE RUN SLOWER

MESSAGE SIZES – MIN, MAX, AVE WITH MPI

Perftools-lite-loops place profile into standard out

```
Table 1:  Inclusive and Exclusive Time in Loops (from compiler-inserted loop instrumentation)
  Loop | Loop Incl |      Time |  Loop Exec |  Loop |  Loop |  Loop | Function=/.LOOP[.]
  Incl |      Time |     (Loop |            | Trips | Trips | Trips |  PE=HIDE
 Time% |           |     Adj.) |            |   Avg |   Min |   Max |
|-----------------------------------------------------------------------------
| 96.0% | 62.483595 |  0.001207 |          1 | 100.0 |   100 |   100 | les3d_.LOOP.3.li.218
| 89.8% | 58.428053 | 12.470257 |        100 |   2.0 |     2 |     2 | les3d_.LOOP.4.li.274
| 18.9% | 12.303912 |  0.000742 |        200 |  96.0 |    96 |    96 | fluxk_.LOOP.1.li.38
| 17.7% | 11.551030 |  0.000195 |        200 |  27.4 |    27 |    28 | fluxi_.LOOP.1.li.30
| 17.7% | 11.550835 |  7.489365 |      5,486 |  96.0 |    96 |    96 | fluxi_.LOOP.2.li.31
| 15.3% |  9.965937 |  0.000450 |        200 |  27.4 |    27 |    28 | fluxj_.LOOP.1.li.38
| 14.0% |  9.103169 |  8.076826 |     19,200 |  28.4 |    28 |    29 | fluxk_.LOOP.2.li.39
| 12.0% |  7.804776 |  7.170560 |      5,486 |  97.0 |    97 |    97 | fluxj_.LOOP.2.li.39
| 10.8% |  7.058184 |  0.007562 |        202 |   5.0 |     5 |     5 | parallel_.LOOP.1.li.16
|  8.6% |  5.625883 |  5.625883 |    526,629 |  97.0 |    97 |    97 | visci_.LOOP.1.li.809
|  8.5% |  5.515949 |  5.515949 |    545,829 |  96.0 |    96 |    96 | visck_.LOOP.1.li.367
|  8.5% |  5.515182 |  0.000325 |        200 |  28.4 |    28 |    29 | extrapk_.LOOP.1.li.160
|  8.5% |  5.514857 |  4.410280 |      5,686 |  99.0 |    99 |    99 | extrapk_.LOOP.2.li.162
|  8.3% |  5.375106 |  0.000202 |        200 |  31.1 |    30 |    32 | extrapj_.LOOP.1.li.161
|  8.3% |  5.374903 |  4.204186 |      6,229 |  97.0 |    97 |    97 | extrapj_.LOOP.2.li.162
|  7.2% |  4.688454 |  4.688454 |    532,114 |  96.0 |    96 |    96 | viscj_.LOOP.1.li.367
|  5.7% |  3.739388 |  0.000092 |        200 |  31.1 |    30 |    32 | extrapi_.LOOP.1.li.141
|  5.7% |  3.739296 |  3.739296 |      6,229 |  99.0 |    99 |    99 | extrapi_.LOOP.2.li.142
|  5.3% |  3.470722 |  1.246789 |    526,629 |  96.0 |    96 |    96 | fluxi_.LOOP.4.li.69
|  4.9% |  3.200002 |  0.023447 |     19,200 |  27.4 |    27 |    28 | fluxk_.LOOP.4.li.79
|  4.9% |  3.176555 |  0.457949 |    526,629 |  96.0 |    96 |    96 | fluxk_.LOOP.5.li.81
|  4.2% |  2.718606 |  2.718606 | 50,556,343 |   5.0 |     5 |     5 | fluxk_.LOOP.6.li.83
|  3.9% |  2.519885 |  2.519885 |    557,229 | 100.0 |   100 |   100 | exk4_.LOOP.1.li.280
|  3.4% |  2.223934 |  2.223934 | 50,556,343 |   5.0 |     5 |     5 | fluxi_.LOOP.5.li.71
|  3.4% |  2.205782 |  2.205782 |    597,943 | 100.0 |   100 |   100 | exj4_.LOOP.1.li.280
```

Sampling run 1/100 sec

Column

1 Inclusive loop percent
2 Inclusive loop time
3 ?????
4 Time loop is executed
5 Avg trip count
6 Min trip count
7 Max trip count
8 Location of loop

MESSAGE SIZES – MIN, MAX, AVE WITH MPI (PAT_REPORT -OCT

```
Table 1:  Calltree with Loop Inclusive Time
   Incl |  Incl |  Loop Exec |  Loop | Calltree
  Time% |  Time |            | Trips |  PE=HIDE
        |       |            |   Avg |
 100.0% | 65.09 |         -- |    -- | Total
|-------------------------------------------------------------
|  84.4% | 54.97 |         -- |    -- | les3d_
||------------------------------------------------------------
||  96.0% | 62.48 |          1 | 100.0 | les3d_.LOOP.3.li.218
|||-----------------------------------------------------------
3||  89.8% | 58.43 |        100 |   2.0 | les3d_.LOOP.4.li.274
||||----------------------------------------------------------
4|||  17.6% | 11.45 |         -- |    -- | fluxk_
|||||---------------------------------------------------------
5||||  18.9% | 12.30 |        200 |  96.0 | fluxk_.LOOP.1.li.38
||||||--------------------------------------------------------
6|||||  14.0% |  9.10 |     19,200 |  28.4 | fluxk_.LOOP.2.li.39
|||||||-------------------------------------------------------
7||||||   1.6% |  1.03 |    545,829 |  96.0 | fluxk_.LOOP.3.li.40
|||||||=======================================================
6|||||   4.9% |  3.20 |     19,200 |  27.4 | fluxk_.LOOP.4.li.79
|||||||-------------------------------------------------------
7||||||   4.9% |  3.18 |    526,629 |  96.0 | fluxk_.LOOP.5.li.81
||||||||------------------------------------------------------
8|||||||   4.2% |  2.72 | 50,556,343 |   5.0 | fluxk_.LOOP.6.li.83
||||||=========================================================
5||||   5.3% |  3.42 |         -- |    -- | exk4_
||||||--------------------------------------------------------
6|||||   3.9% |  2.52 |    557,229 | 100.0 | exk4_.LOOP.1.li.280
||||||========================================================
```

Sampling run 1/100 sec

Column

1 Inclusive percent

2 Inclusive time

3 Number of times element encountered

4 Ave trip count

5 Program element

HOW WOULD WE USE PERFTOOLS TO PARALLELIZE THIS APPLICATION?

WHEN USING PERFTOOL-LITE TOOLS WITH C++

- Your initial profile will probably be sparse
  - pat_report –P <statistics_file>
- Perftools-lite-loops turns off OpenMP, if the application calls openMP API calls, they will be ignored

USING PERFTOOLS-LITE (OR -LOOPS OR -HBM)

- module load perftools-lite or perftools-lite-loops or perftools-lite-hbm
  - Module perftools-base should already be loaded
- Build application
- Run application
- Statistics report comes out within standard out
  - Also generates a directory of profile data to be examined with different options

PERFTOOLS-LITE PROFILE – RUN ON 8 NODES–1 MPI TASK/NODE

```
Table 1:  Profile by Function

  Samp% |     Samp | Imb. |  Imb. | Group
        |          | Samp | Samp% |  Function=[MAX10]
        |          |      |       |   PE=HIDE

 100.0% | 4,061.6 |   -- |    -- | Total
|-------------------------------------------------
|  95.0% | 3,859.9 |   -- |    -- | USER
||------------------------------------------------
||  21.4% |   869.6 |  8.4 |  1.1% | fluxj_
||  20.7% |   842.8 |  7.2 |  1.0% | fluxi_
||  19.9% |   808.2 |  5.8 |  0.8% | fluxk_
||   7.9% |   318.9 |  6.1 |  2.2% | extrapk_
||   7.5% |   303.2 |  7.8 |  2.8% | extrapi_
||   6.8% |   274.2 |  3.8 |  1.5% | update_
||   6.1% |   249.4 |  4.6 |  2.1% | extrapj_
||   1.0% |    40.5 |  5.5 | 13.7% | mpicx_
||================================================
|   4.1% |   165.2 |   -- |    -- | MPI
||------------------------------------------------
||   2.2% |    91.1 |  5.9 |  6.9% | MPI_REDUCE
||   1.1% |    44.9 | 31.1 | 46.8% | MPI_SEND
|=================================================
```

Exclusive time

Sampling is in

100th of a second

Imbalance

Only showing items that take up more than 1% of time – you can override with -T

PERFTOOLS-LITE PROFILE – RUN ON 8 NODES–8 MPI TASKS

```
Table 2:  Profile by Group, Function, and Line
  Samp% |     Samp | Imb. |  Imb. | Group
        |          | Samp | Samp% |  Function=[MAX10]
        |          |      |       |   Source
        |          |      |       |    Line
        |          |      |       |     PE=HIDE

 100.0% | 4,061.6 |   -- |    -- | Total
|-----------------------------------------------------------------------
|  95.0% | 3,859.9 |   -- |    -- | USER
||----------------------------------------------------------------------
||  21.4% |   869.6 |   -- |    -- | fluxj_
3|        |         |      |       |  Leslie_CUG/leslie3d_mpi/src/fluxj.f
||||--------------------------------------------------------------------
4|||   1.6% |    63.6 |  8.4 | 13.3% | line.31
4|||  14.0% |   567.5 | 15.5 |  3.0% | line.67
4|||   2.7% |   109.6 | 12.4 | 11.6% | line.76
||||====================================================================
||  20.7% |   842.8 |   -- |    -- | fluxi_
3|        |         |      |       |  Leslie_CUG/leslie3d_mpi/src/fluxi.f
||||--------------------------------------------------------------------
4|||   1.9% |    77.1 | 18.9 | 22.5% | line.24
4|||  14.2% |   578.8 | 21.2 |  4.0% | line.56
4|||   2.1% |    85.5 | 10.5 | 12.5% | line.63
||||====================================================================
||  19.9% |   808.2 |   -- |    -- | fluxk_
3|        |         |      |       |  Leslie_CUG/leslie3d_mpi/src/fluxk.f
||||--------------------------------------------------------------------
4|||   1.7% |    69.0 | 13.0 | 18.1% | line.31
4|||  12.2% |   497.5 | 21.5 |  4.7% | line.65
4|||   3.2% |   129.6 | 18.4 | 14.2% | line.74
```

Profile on line level within a routine – lets look at this routine

PERFTOOLS-LITE PROFILE – RUN ON 8 NODES–8 MPI TASKS

```
28.  + F------------------<       DO K = 1, KCMAX
29.  + F F----------------<          DO J = 0, JCMAX
30.    F F V--------------<            DO I = 1, IND
31.    F F V                             QS(I) = UAV(I,J,K) * SJX(I,J,K) +
32.    F F V                      >              VAV(I,J,K) * SJY(I,J,K) +
33.    F F V                      >              WAV(I,J,K) * SJZ(I,J,K)
34.    F F V
35.    F F V                             IF (NSCHEME .EQ. 2) THEN
36.    F F V                                L = J + 1 - JADD
37.    F F V
38.    F F V                                   QSP = U(I,L,K) * SJX(I,J,K) +
39.    F F V                      >                  V(I,L,K) * SJY(I,J,K) +
40.    F F V                      >                  W(I,L,K) * SJZ(I,J,K)
41.    F F V                                   QSPJ = (QSP - QS(I)) * DBLE(1 - 2 * JADD)
42.    F F V                                   IF (QSPJ .GT. 0.0D+00) QS(I) = 0.5D+00 * (QS(I) + QSP)
43.    F F V
44.    F F V                             ENDIF
45.    F F V
46.    F F V                            FSJ(I,J,1) =  QAV(I,J,K,1) * QS(I)
47.    F F V                            FSJ(I,J,2) =  QAV(I,J,K,2) * QS(I) +
48.    F F V                      >                           PAV(I,J,K) * SJX(I,J,K)
49.    F F V                            FSJ(I,J,3) =  QAV(I,J,K,3) * QS(I) +
50.    F F V                      >                           PAV(I,J,K) * SJY(I,J,K)
51.    F F V                            FSJ(I,J,4) =  QAV(I,J,K,4) * QS(I) +
52.    F F V                      >                           PAV(I,J,K) * SJZ(I,J,K)
53.    F F V                            FSJ(I,J,5) = (QAV(I,J,K,5) + PAV(I,J,K)) *
54.    F F V                      >                                                   QS(I)
55.    F F V
56.    F F V                            IF (ISGSK .EQ. 1) THEN
57.    F F V                               FSJ(I,J,7) = QAV(I,J,K,7) * QS(I)
58.    F F V                            ENDIF
59.    F F V
60.    F F V                            IF ( ICHEM .GT. 0 ) THEN
61.    F F V D------------<                DO L = 8, 7 + NSPECI
62.    F F V D                                FSJ(I,J,L) = QAV(I,J,K,L) * QS(I)
63.    F F V D------------>                ENDDO
64.    F F V                            ENDIF
65.    F F V-------------->          ENDDO
66.    F F
67.  + F F V I----------<>              IF ( VISCOUS ) CALL VISCJ (1, IND, J, K, FSJ)
68.    F F---------------->          ENDDO
```

+ indicates further information

You get this listing by using –hlist-a

Line 67 from –hlist=a

I indicates the call was inlined , V loop Vectorized, F Flattened

PERFTOOLS-LITE PROFILE – RUN ON 8 NODES–8 MPI TASKS

```
ftn-6315 ftn: VECTOR FLUXJ, File = fluxj.f, Line = 28
  A loop starting at line 28 was not vectorized because the target array (qs) would require rank expansion.

ftn-3182 ftn: IPA FLUXJ, File = fluxj.f, Line = 28, Column = 7
  Loop has been flattened.

ftn-6315 ftn: VECTOR FLUXJ, File = fluxj.f, Line = 29
  A loop starting at line 29 was not vectorized because the target array (qs) would require rank expansion.

ftn-3182 ftn: IPA FLUXJ, File = fluxj.f, Line = 29, Column = 10
  Loop has been flattened.

ftn-6204 ftn: VECTOR FLUXJ, File = fluxj.f, Line = 30
  A loop starting at line 30 was vectorized.

ftn-6002 ftn: SCALAR FLUXJ, File = fluxj.f, Line = 61
  A loop starting at line 61 was eliminated by optimization.

ftn-6383 ftn: VECTOR FLUXJ, File = fluxj.f, Line = 67
  A loop starting at line 67 requires an estimated 25 vector registers at line 67; 2 of these have been preemptively forced to
  memory.

ftn-6204 ftn: VECTOR FLUXJ, File = fluxj.f, Line = 67
  A loop starting at line 67 was vectorized.
```

PERFTOOLS-LITE PROFILE – RUN ON 8 NODES–8 MPI TASKS

```
Table 1:  Function Calltree View
  Samp% |     Samp | Calltree
        |          |  PE=HIDE
 100.0% | 4,061.6 | Total
|-----------------------------------
|  85.1% | 3,456.6 | les3d_
||-----------------------------------
||  27.8% | 1,127.1 | fluxk_
|||-----------------------------------
3||  19.9% |   808.2 | fluxk_(exclusive)
3||   7.9% |   318.9 | extrapk_
|||==================================
||  27.6% | 1,119.0 | fluxj_
|||-----------------------------------
3||  21.4% |   869.6 | fluxj_(exclusive)
3||   6.1% |   249.4 | extrapj_
|||==================================
||  20.7% |   842.8 | fluxi_
||   2.8% |   115.4 | parallel_
3|   2.1% |    87.0 |  mpicx_
||||-----------------------------------
4|||   1.1% |    44.9 | MPI_SEND
4|||   1.0% |    40.5 | mpicx_(exclusive)
||||=================================
||   2.3% |    92.0 | flowio_
3|   2.2% |    91.1 |  MPI_REDUCE
||   1.1% |    45.9 | tmstep_
||===================================
|   7.5% |   303.2 | extrapi_
|   6.8% |   274.2 | update_
```

Pat_report –Oct <statistics directory>

Level in the Call Tree, everything 3 and higher is called from this || (2)

PERFTOOLS-LITE-LOOPS – RUN ON 8 NODES–8 MPI TASKS

Table 1: Inclusive and Exclusive Time in Loops (from -hprofile_generate)

Loop Incl Time%	Loop Incl Time	Loop Hit	Loop Trips Avg	Loop Trips Min	Loop Trips Max	Function=/.LOOP[.] PE=HIDE
96.9%	60.109652	1	100.0	100	100	les3d_.LOOP.3.li.216
96.0%	59.527700	100	2.0	2	2	les3d_.LOOP.4.li.272
23.3%	14.461074	200	96.0	96	96	fluxi_.LOOP.1.li.21
23.3%	14.460550	19,200	96.0	96	96	fluxi_.LOOP.2.li.22
20.5%	12.724507	200	96.0	96	96	fluxk_.LOOP.1.li.28
20.1%	12.486995	200	96.0	96	96	fluxj_.LOOP.1.li.28
15.1%	9.370239	19,200	97.0	97	97	fluxj_.LOOP.2.li.29
14.0%	8.713997	19,200	97.0	97	97	fluxk_.LOOP.2.li.29
10.1%	6.259900	1,843,200	97.0	97	97	visci_.LOOP.1.li.782
8.8%	5.445447	1,862,400	96.0	96	96	viscj_.LOOP.1.li.347
8.0%	4.990874	1,862,400	96.0	96	96	visck_.LOOP.1.li.348
7.9%	4.905076	200	97.0	97	97	extrapk_.LOOP.1.li.141
7.9%	4.904410	19,400	99.0	99	99	extrapk_.LOOP.2.li.143
7.6%	4.704280	1,843,200	96.0	96	96	fluxi_.LOOP.4.li.60
6.6%	4.121449	200	99.0	99	99	extrapj_.LOOP.1.li.141
6.6%	4.121028	19,800	97.0	97	97	extrapj_.LOOP.2.li.142
6.5%	4.009192	19,200	96.0	96	96	fluxk_.LOOP.4.li.69
6.4%	3.995474	1,843,200	96.0	96	96	fluxk_.LOOP.5.li.71
5.5%	3.382283	200	99.0	99	99	extrapi_.LOOP.1.li.123
5.5%	3.381946	19,800	99.0	99	99	extrapi_.LOOP.2.li.124
5.0%	3.115988	19,200	96.0	96	96	fluxj_.LOOP.4.li.71
5.0%	3.102463	1,843,200	96.0	96	96	fluxj_.LOOP.5.li.73

This Table shows most important loops, the columns are percent of time, inclusive time, number of times the loop was executed, Avg, Min, Max iteration counts and location within the source

HOW DO I KNOW WHAT THE IMPORTANT LOOPS ARE?

- Pat_report –O calltree < directory produced by perftools-lite-loops run>
- Produces call tree with DO loops included

PERFTOOLS-LITE-LOOPS – RUN ON 8 NODES–8 MPI TASKS

```
Table 1:  Function Calltree View
  Time% |       Time |       Calls | Calltree
        |            |             |  PE=HIDE
 100.0% | 60.437329 |          -- | Total
|-----------------------------------------------------------
| 100.0% | 60.437286 |         2.0 | les3d_
||----------------------------------------------------------
||  96.8% | 58.504830 |          -- | les3d_.LOOP.3.li.216
3|  95.8% | 57.923177 |          -- |  les3d_.LOOP.4.li.272
||||--------------------------------------------------------
4|||  30.9% | 18.681501 |       200.0 | fluxi_
|||||-------------------------------------------------------
5||||  12.1% |  7.338945 |       200.0 | fluxi_(exclusive)
5||||  11.4% |  6.863554 |          -- | fluxi_.LOOP.1.li.21
6||||        |           |             |  fluxi_.LOOP.2.li.22
7||||  11.4% |  6.863554 | 1,843,200.0 |   visci_
5||||   7.4% |  4.479002 |       200.0 | extrapi_
||||||------------------------------------------------------
6|||||   3.8% |  2.320974 |       200.0 | extrapi_(exclusive)
6|||||   3.6% |  2.158027 |          -- | extrapi_.LOOP.1.li.123
7|||||        |           |             |  extrapi_.LOOP.2.li.124
8|||||   3.6% |  2.158027 | 1,960,200.0 |   exi4_
|||||=======================================================
4|||  28.3% | 17.098408 |       200.0 | fluxk_
|||||-------------------------------------------------------
5||||  11.3% |  6.809059 |       200.0 | fluxk_(exclusive)
5||||   9.4% |  5.654640 |          -- | fluxk_.LOOP.1.li.28
6||||        |           |             |  fluxk_.LOOP.2.li.29
7||||   9.4% |  5.654640 | 1,862,400.0 |   visck_
5||||   7.7% |  4.634709 |       200.0 | extrapk_
||||||------------------------------------------------------
6|||||   4.9% |  2.959637 |          -- | extrapk_.LOOP.1.li.141
7|||||        |           |             |  extrapk_.LOOP.2.li.143
8|||||   4.9% |  2.937827 | 1,900,800.0 |   exk4_
6|||||   2.8% |  1.675072 |       200.0 | extrapk_(exclusive)
|||||=======================================================
4|||  26.6% | 16.077506 |       200.0 | fluxj_
|||||-------------------------------------------------------
5||||  10.2% |  6.136293 |          -- | fluxj_.LOOP.1.li.28
6||||        |           |             |  fluxj_.LOOP.2.li.29
7||||  10.2% |  6.136293 | 1,862,400.0 |   viscj_
5||||  10.1% |  6.090073 |       200.0 | fluxj_(exclusive)
5||||   6.4% |  3.851140 |       200.0 | extrapj_
```

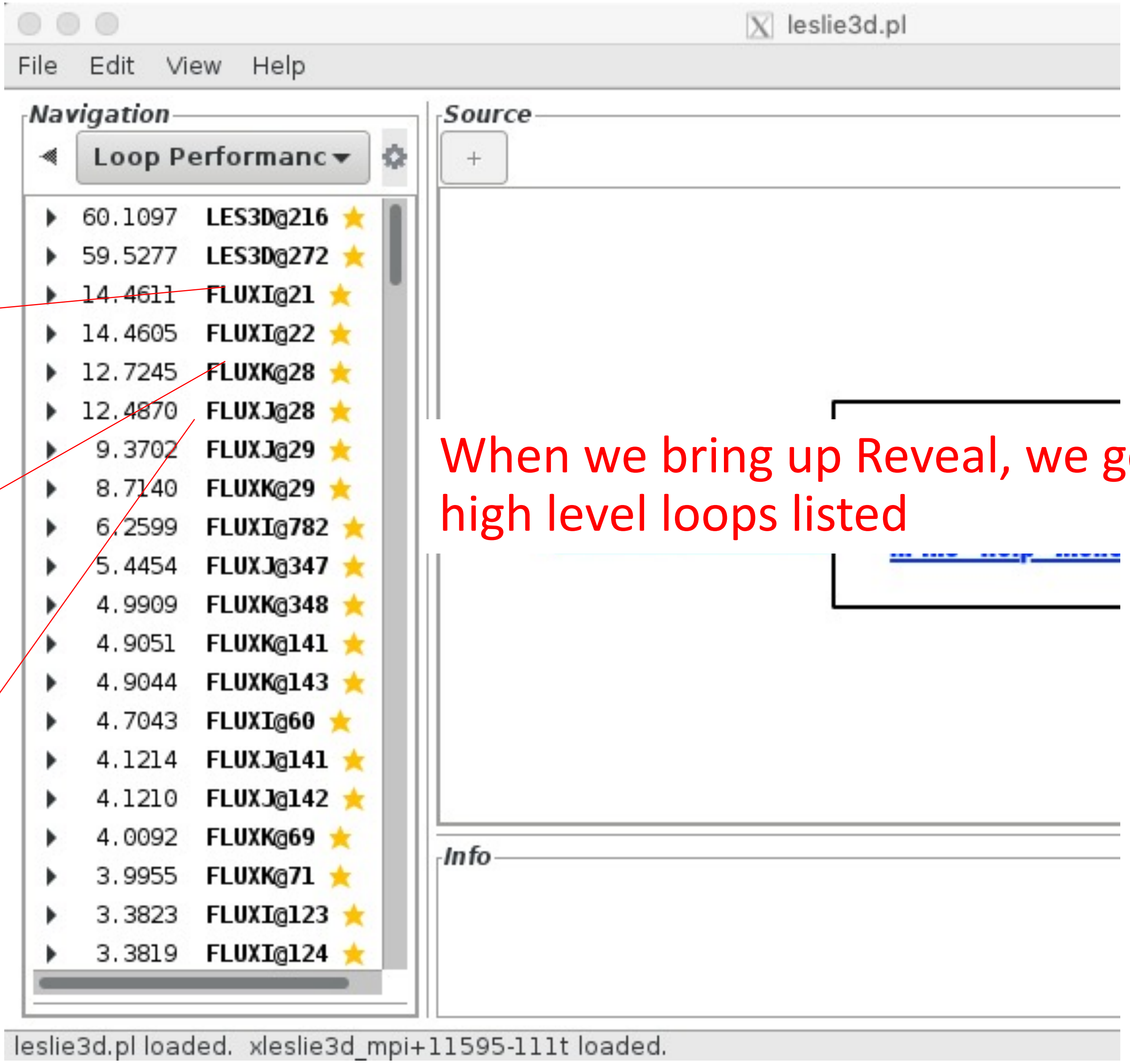
USING REVEAL

- Need program library and perftools-lite-loops output
  - ftn -hlist=a –hpl=leslie3d.pl
  - reveal leslie3d.pl < perftools-lite-loops data directory>

DO NOT HAVE any PERFTOOLS-LITE or PERFTOOLS modules loaded when you build the program library

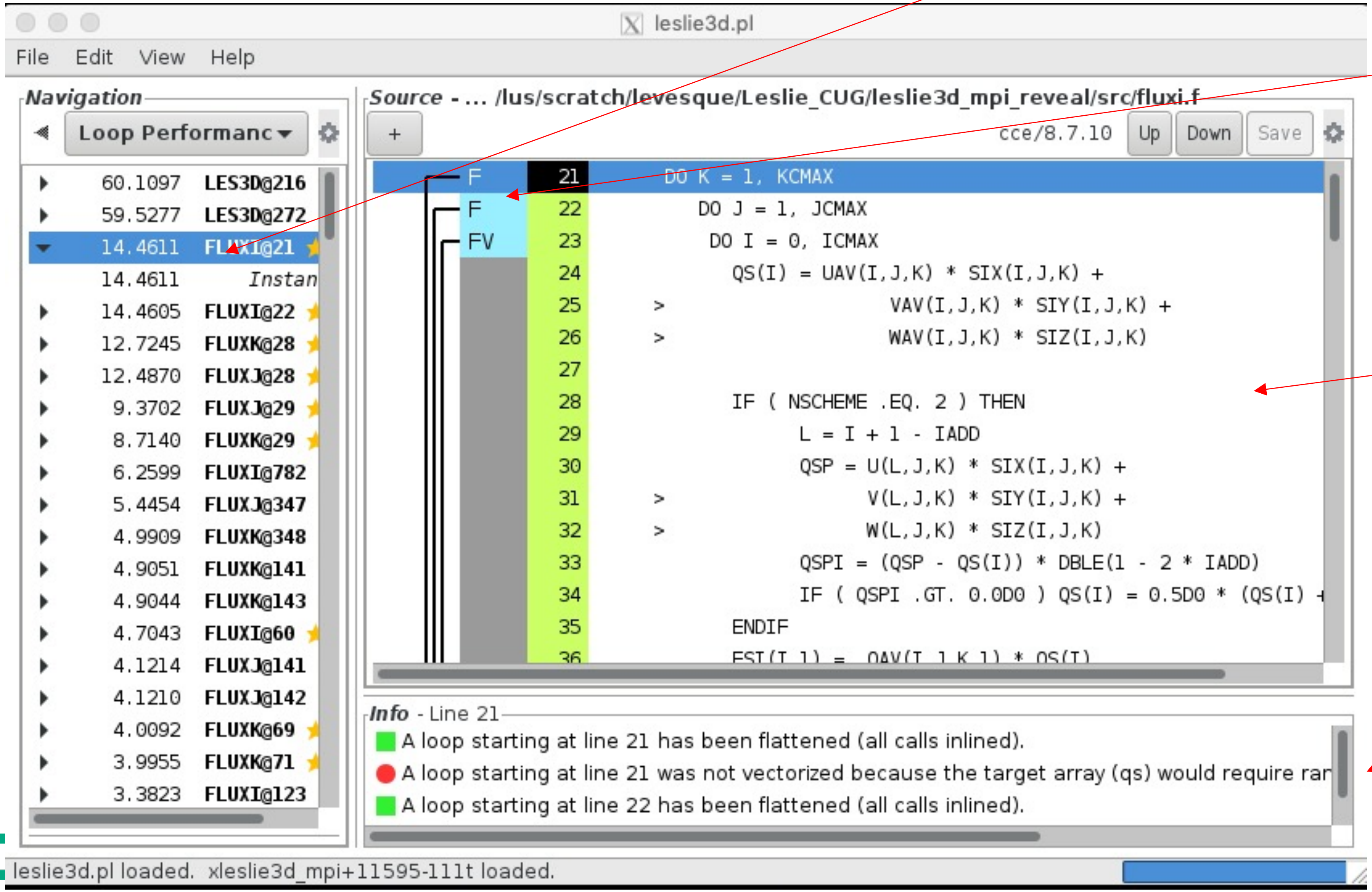
PERFTOOLS-LITE-LOOPS – RUN ON 8 NODES–1 MPI TASK/NODE

```
Table 1:  Function Calltree View
  Time%  |      Time  |       Calls  | Calltree
         |            |              |  PE=HIDE
 100.0%  | 60.437329  |          --  | Total
|-------------------------------------------------------------
| 100.0%  | 60.437286  |          2.0  | les3d_
||------------------------------------------------------------
||  96.8%  | 58.504830  |          --  | les3d_.LOOP.3.li.216
3|  95.8%  | 57.923177  |          --  |  les3d_.LOOP.4.li.272
|||-----------------------------------------------------------
4|||  30.9%  | 18.681501  |       200.0  | fluxi_
|||||---------------------------------------------------------
5||||  12.1%  |  7.338945  |       200.0  | fluxi_(exclusive)
5||||  11.4%  |  6.863554  |          --  | fluxi_.LOOP.1.li.21
6||||         |            |              |  fluxi_.LOOP.2.li.22
7||||  11.4%  |  6.863554  | 1,843,200.0  |   visci_
5||||   7.4%  |  4.479002  |       200.0  | extrapi_
||||||--------------------------------------------------------
6|||||   3.8%  |  2.320974  |       200.0  | extrapi_(exclusive)
6|||||   3.6%  |  2.158027  |          --  | extrapi_.LOOP.1.li.123
7|||||         |            |              |  extrapi_.LOOP.2.li.124
8|||||   3.6%  |  2.158027  | 1,960,200.0  |   exi4_
|||||=========================================================
4|||  28.3%  | 17.098408  |       200.0  | fluxk_
|||||---------------------------------------------------------
5||||  11.3%  |  6.809059  |       200.0  | fluxk_(exclusive)
5||||   9.4%  |  5.654640  |          --  | fluxk_.LOOP.1.li.28
6||||         |            |              |  fluxk_.LOOP.2.li.29
7||||   9.4%  |  5.654640  | 1,862,400.0  |   visck_
5||||   7.7%  |  4.634709  |       200.0  | extrapk_
||||||--------------------------------------------------------
6|||||   4.9%  |  2.959637  |          --  | extrapk_.LOOP.1.li.141
7|||||         |            |              |  extrapk_.LOOP.2.li.143
8|||||   4.9%  |  2.937827  | 1,900,800.0  |   exk4_
6|||||   2.8%  |  1.675072  |       200.0  | extrapk_(exclusive)
|||||=========================================================
4|||  26.6%  | 16.077506  |       200.0  | fluxj_
|||||---------------------------------------------------------
5||||  10.2%  |  6.136293  |          --  | fluxj_.LOOP.1.li.28
6||||         |            |              |  fluxj_.LOOP.2.li.29
7||||  10.2%  |  6.136293  | 1,862,400.0  |   viscj_
5||||  10.1%  |  6.090073  |       200.0  | fluxj_(exclusive)
5||||   6.4%  |  3.851140  |       200.0  | extrapj_
```



When we bring up Reveal, we get the high level loops listed

CLICK ON IMPORTANT LOOP – RIGHT CLICK TO SCOPE



Annotation of optimization of code

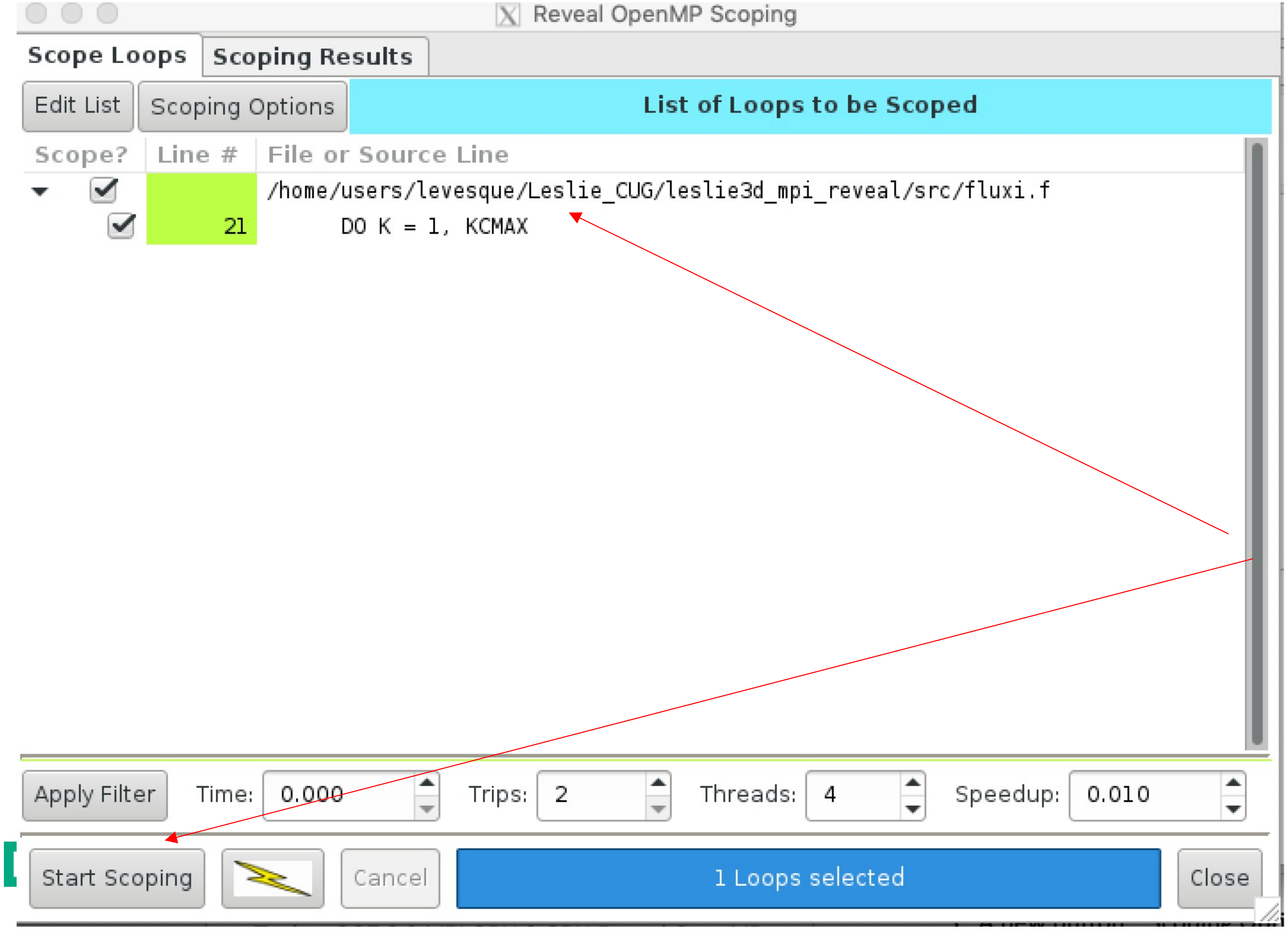
F- Flattened

V- Vectorized

Listing of loop

Diagnostics from compiler

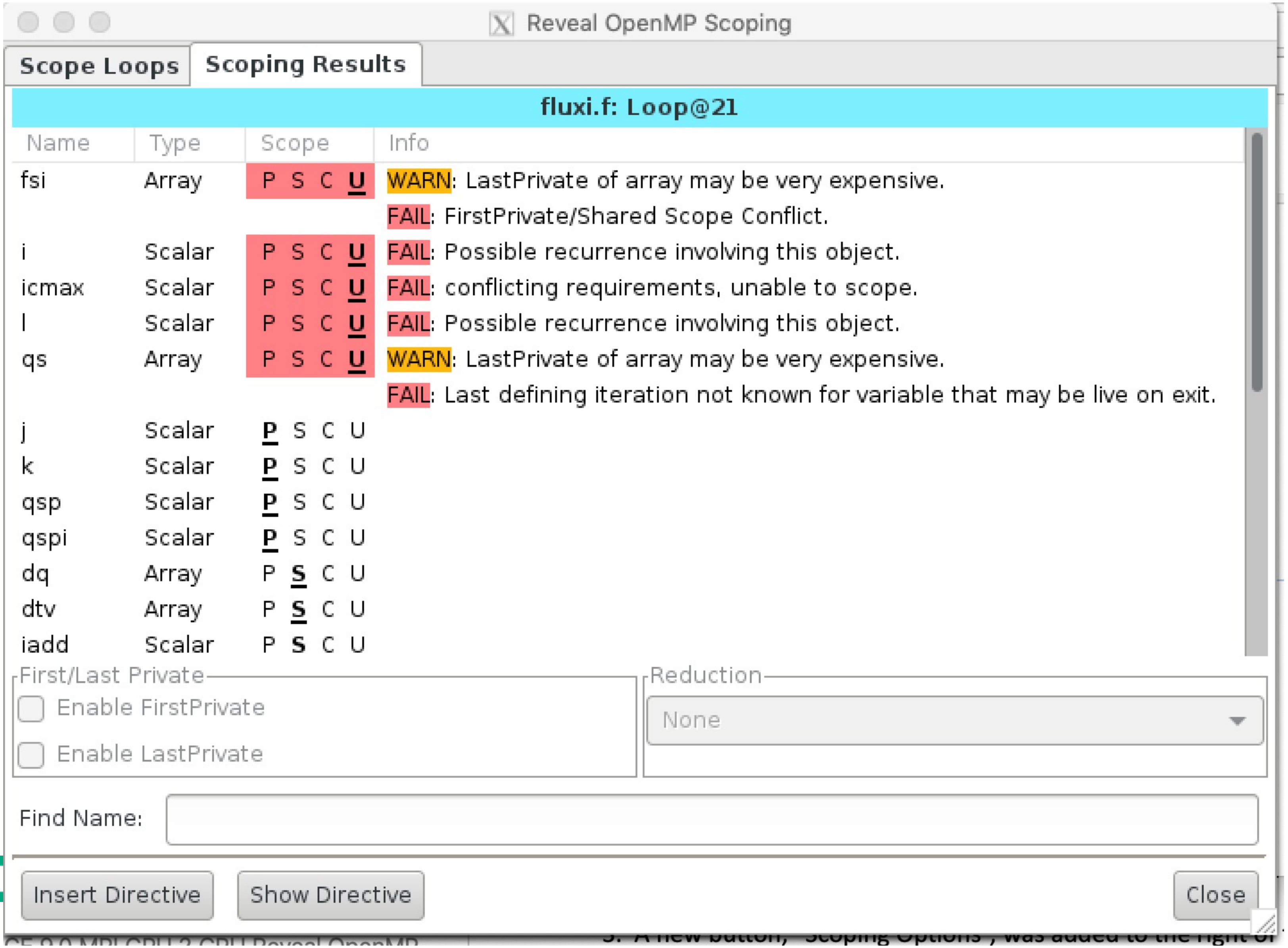
SCOPING WINDOW



Then up pops the Scoping window

Loop selected Just click on Start Scoping

SCOPING RESULTS



P – Private

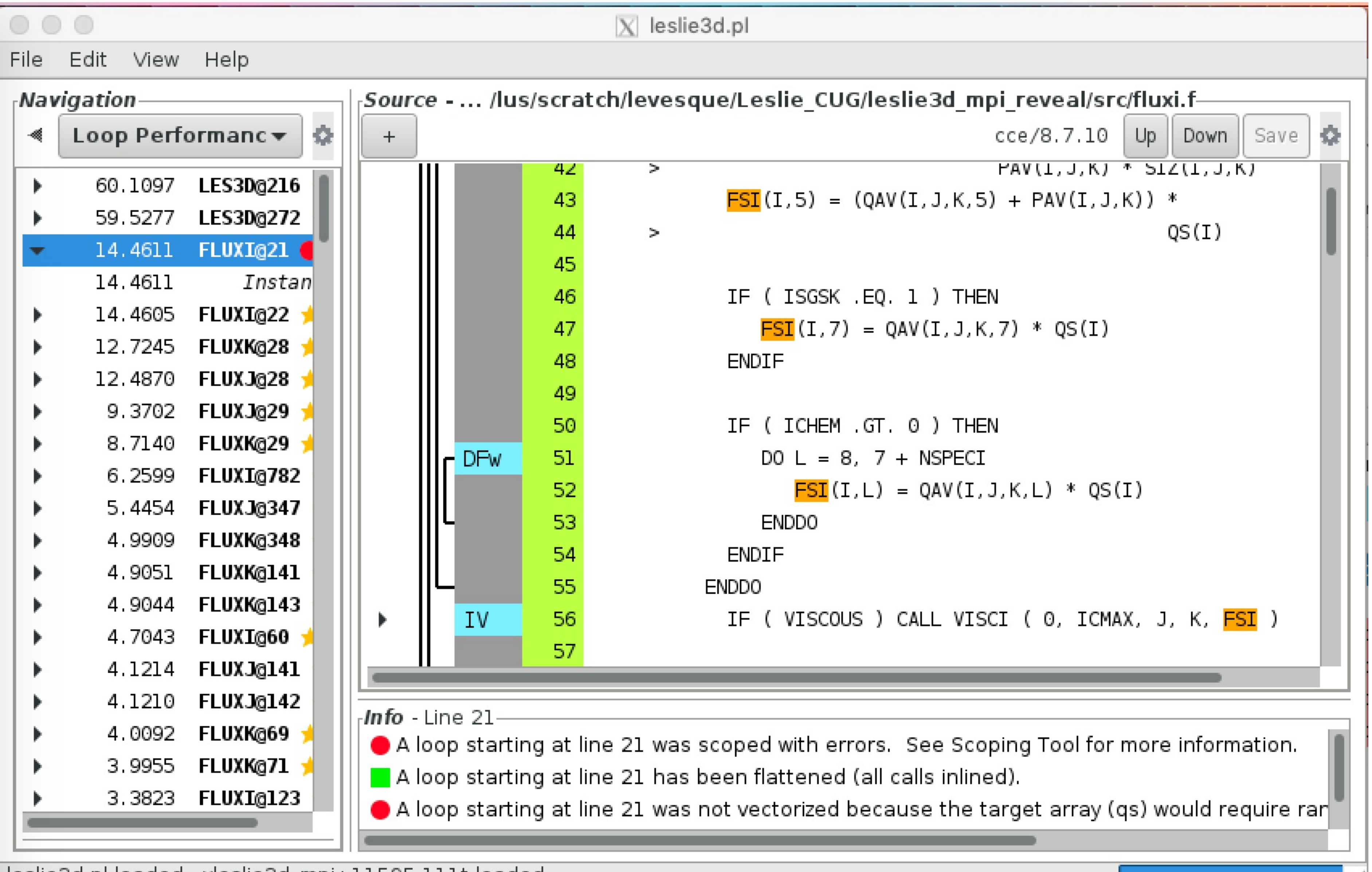
S – Shared

C – Conflict

U - Unresolved

User can now change scope by selecting the appropriate letter

ARRAY CONSTANTS ARE DIFFICULT TO SCOPE, ESPECIALLY WHEN PASSED TO A ROUTINE



If you select a variable in the scoping window, all occurrences are highlighted in the main window

WHAT CAN THE USER DO TO HELP REVEAL

- Trace each variable that is unresolved and decide whether it is a potential race condition or can be scoped private or shared
  - If okay, make private or shared
- Once you have resolved all the unresolved variables then select the Insert Directives
- Couple definitions
  - Array Constant – An array not referenced by the parallel loop index
  - Array Constant reduction - example

With respect to K - A(I,J) is an array reduction

```
do j=1,n
  do I=1,n
    do k=1,n
    a(i,j) = a(i,j) + b(i,k)*c(j,k)
    enddo
  enddo
enddo
```

SOME SIMPLE SCOPING RULES

- A scalar or an array not dependent on the loop being parallelized (array constant) should be private or ordered dependency
  - If the scalar is set prior to being used each time through the loop, then it is private. If it is not then it will result in a race condition
  - All elements of a array constant must be set prior to being used each pass through the loop. In those cases where not all the elements of the array are set prior to being used, first value getting is required.

SOME SIMPLE SCOPING RULES

- All arrays dependent upon the loop being parallelized should be shared. The compiler must perform data dependency analysis on the arrays to assure that there is no order dependency

- A reduction variable or constant array reduction is a special case that is identified by most compilers and must be in a reduction clause.

- Any variable that is just read is a shared variable

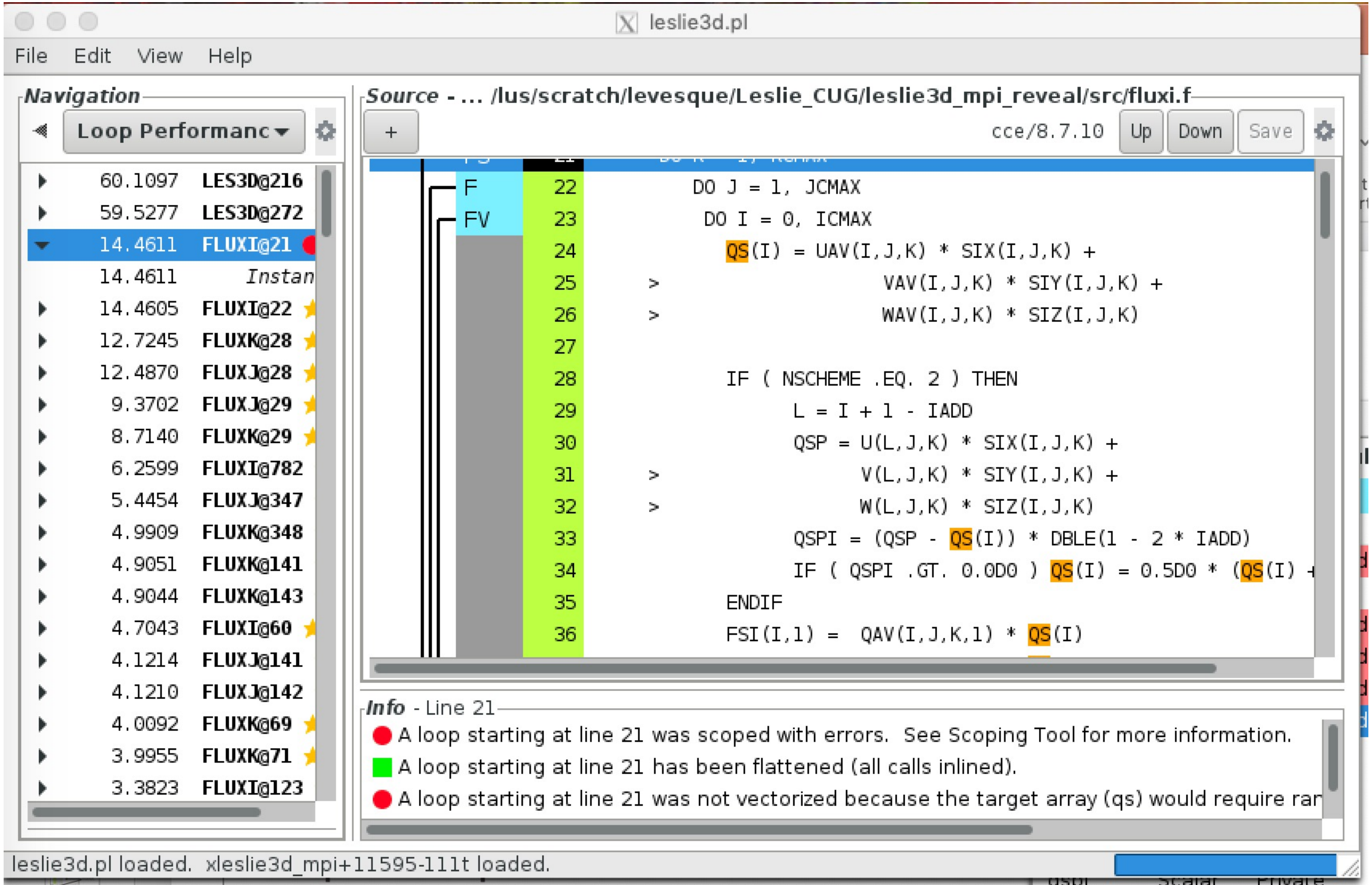
COMPLICATIONS DOWN THE CALL CHAIN AND MODULES

- No scoping directives can be inserted within those routines called from a parallelized loop.
  - All global variables must be shared
  - All variables allocated on stack must be private
- No private variables are allowed within a COMMON block or a module unless they are noted on a THREADPRIVATE directive

LASTPRIVATE SAVING AND FIRSTPRIVATE GETTING

- These two issues can drive you mad
  - Last Value saving is when the last value of a private variable is used outside the parallel loop. I can say that I have seen this in about .00001 % of the applications I have looked at – very rare.
    - When you select last value saving, the compiler has to make sure that the master thread either execute the last pass through the loop or have the thread, that does the last past, copy its private variable to the master
    - Last value saving can have a big performance hit

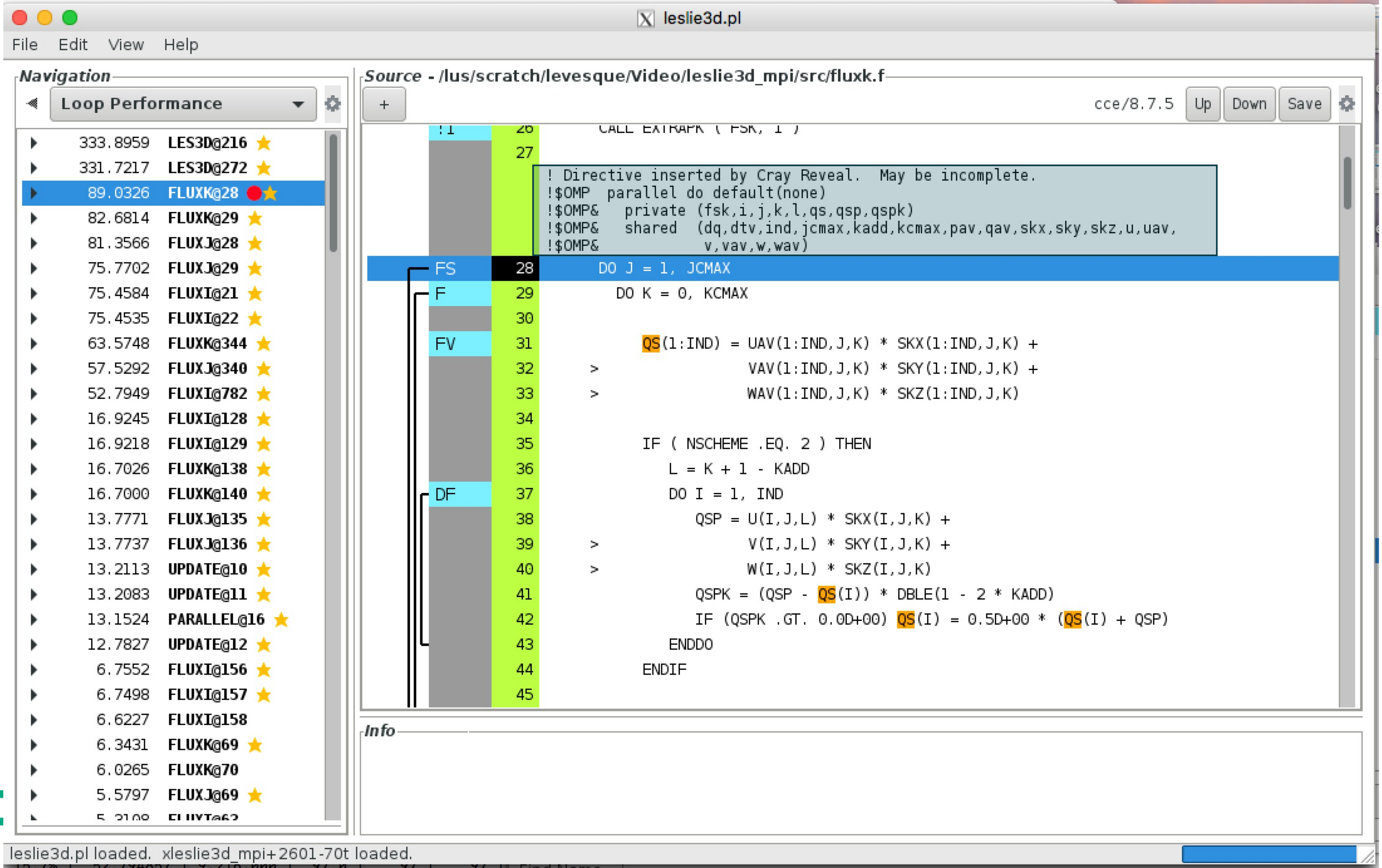
THE COMPILER DOESN'T SCOPE EITHER FSI OR QS AS PRIVATE



AFTER ANALYZING THE VARIABLES THAT ARE UNRESOLVED

- If you are confident of your changes and all unresolved are resolved then click on INSERT Directives
  - CAUTION – if you are wrong you will be inserting a race condition

- If you cannot confidently resolve the unresolved, then go on to another loop



leslie3d.pl
File Edit View Help
Navigation
Loop Performance
333.8959 LES3D@216
331.7217 LES3D@272
89.0326 FLUXK@28
82.6814 FLUXK@29
81.3566 FLUXJ@28
75.7702 FLUXJ@29
75.4584 FLUXI@21
75.4535 FLUXI@22
63.5748 FLUXK@344
57.5292 FLUXJ@340
52.7949 FLUXI@782
16.9245 FLUXI@128
16.9218 FLUXI@129
16.7026 FLUXK@138
16.7000 FLUXK@140
13.7771 FLUXJ@135
13.7737 FLUXJ@136
13.2113 UPDATE@10
13.2083 UPDATE@11
13.1524 PARALLEL@16
12.7827 UPDATE@12
6.7552 FLUXI@156
6.7498 FLUXI@157
6.6227 FLUXI@158
6.3431 FLUXK@69
6.0265 FLUXK@70
5.5797 FLUXJ@69
Source - /lus/scratch/levesque/Video/leslie3d_mpi/src/fluxk.f
+
cce/8.7.5
Up
Down
Save
! Directive inserted by Cray Reveal. May be incomplete.
!$OMP parallel do default(none)
!$OMP& private (fsk,i,j,k,l,qs,qsp,qspk)
!$OMP& shared (dq,dtv,ind,jcmax,kadd,kcmax,pav,qav,skx,sky,skz,u,uav,
!$OMP& v,vav,w,wav)
FS 28 DO J = 1, JCMAX
F 29 DO K = 0, KCMAX
30
FV 31 QS(1:IND) = UAV(1:IND,J,K) * SKX(1:IND,J,K) +
32 > VAV(1:IND,J,K) * SKY(1:IND,J,K) +
33 > WAV(1:IND,J,K) * SKZ(1:IND,J,K)
34
35 IF (NSCHEME .EQ. 2) THEN
36 L = K + 1 - KADD
DF 37 DO I = 1, IND
38 QSP = U(I,J,L) * SKX(I,J,K) +
39 > V(I,J,L) * SKY(I,J,K) +
40 > W(I,J,L) * SKZ(I,J,K)
41 QSPK = (QSP - QS(I)) * DBLE(1 - 2 * KADD)
42 IF (QSPK .GT. 0.0D+00) QS(I) = 0.5D+00 * (QS(I) + QSP)
43 ENDDO
44 ENDIF
45
Info
leslie3d.pl loaded. xleslie3d_mpi+2601-70t loaded.

REVEAL HELPED ANALYZE THIS LOOP

```
                           ! Directive inserted by Cray Reveal.  May be incomplete.
     M-------------------< !$OMP  parallel do default(none)
     M                     !$OMP&   private (fsk,i,j,k,l,qs,qsp,qspk)
     M                     !$OMP&   shared  (dq,dtv,ind,jcmax,kadd,kcmax,pav,qav,skx,sky,skz,u,uav,
     M                     !$OMP&            v,vav,w,wav)
   + M mF----------------<       DO J = 1, JCMAX
   + M mF F--------------<         DO K = 0, KCMAX
     M mF F V------------<          DO I = 1, IND
     M mF F V                         QS(I) = UAV(I,J,K) * SKX(I,J,K) +
     M mF F V                    >                 VAV(I,J,K) * SKY(I,J,K) +
     M mF F V                    >                 WAV(I,J,K) * SKZ(I,J,K)
     M mF F V
     M mF F V                         IF ( NSCHEME .EQ. 2 ) THEN
     M mF F V                            L = K + 1 - KADD
     M mF F V
     M mF F V
     M mF F V
     M mF F V
     M mF F V                               IF (QSPK .GT. 0.0D+00) QS(I) = 0.5D+00 * (QS(I) + QSP)
     M mF F V                         ENDIF
     M mF F V
     M mF F V                         FSK(I,K,1) =  QAV(I,J,K,1) * QS(I)
     M mF F V                         FSK(I,K,2) =  QAV(I,J,K,2) * QS(I) +
     M mF F V                    >                             PAV(I,J,K) * SKX(I,J,K)
     M mF F V                         FSK(I,K,3) =  QAV(I,J,K,3) * QS(I) +
     M mF F V                    >                             PAV(I,J,K) * SKY(I,J,K)
     M mF F V                         FSK(I,K,4) =  QAV(I,J,K,4) * QS(I) +
     M mF F V                    >                             PAV(I,J,K) * SKZ(I,J,K)
     M mF F V                         FSK(I,K,5) = (QAV(I,J,K,5) + PAV(I,J,K)) *
     M mF F V                    >                                               QS(I)
```

Notice that Reveal doesn't use default shared, primarily to illustrate that it has handled all variables

OPENMP VERSUS MPI

Running on 7 nodes with 48 threads

Table 1: Profile by Function

Samp%	Samp	Imb. Samp	Imb. Samp%	Group Function=[MAX10] PE=HIDE Thread=HIDE
100.0%	2,038.3	--	--	Total
48.6%	990.9	--	--	USER
11.3%	231.0	8.0	3.9%	tmstep_
6.0%	122.0	10.0	8.8%	parallel_
5.7%	115.3	28.7	23.3%	ghost_
4.0%	81.9	4.1	5.6%	fluxi_.LOOP@li.30
3.6%	72.7	6.3	9.3%	fluxk_.LOOP@li.38
3.6%	72.6	7.4	10.8%	fluxj_.LOOP@li.38
3.5%	71.9	85.1	63.3%	wallbc_
42.3%	862.1	--	--	MPI
17.3%	352.9	27.1	8.3%	MPI_REDUCE
16.3%	332.1	69.9	20.3%	MPI_SEND
5.9%	121.1	30.9	23.7%	MPI_ALLREDUCE
7.5%	152.3	--	--	ETC
1.6%	32.6	--	--	OMP

Running on 7 nodes with 1 threads

Table 1: Profile by Function

Samp%	Samp	Imb. Samp	Imb. Samp%	Group Function=[MAX10] PE=HIDE
100.0%	9,623.4	--	--	Total
79.7%	7,667.6	--	--	USER
20.5%	1,971.0	302.0	15.5%	fluxi_.LOOP@li.30
18.8%	1,812.7	427.3	22.3%	fluxk_.LOOP@li.38
17.5%	1,688.4	230.6	14.0%	fluxj_.LOOP@li.38
5.2%	496.1	30.9	6.8%	extrapk_.LOOP@li.158
4.6%	438.0	33.0	8.2%	extrapj_.LOOP@li.159
4.3%	414.4	5.6	1.5%	update_.LOOP@li.18
2.4%	229.0	6.0	3.0%	tmstep_
19.5%	1,878.0	--	--	MPI
8.7%	835.7	789.3	56.7%	MPI_SEND
5.7%	550.4	802.6	69.2%	MPI_WAIT
3.7%	354.4	26.6	8.1%	MPI_REDUCE

OPENMP VERSUS MPI

Running on 7 nodes with 48 threads

Table 1: Profile by Function

Samp%	Samp	Imb. Samp	Imb. Samp%	Group Function=[MAX10] PE=HIDE Thread=HIDE
100.0%	2,038.3	--	--	Total
48.6%	990.9	--	--	USER
11.3%	231.0	8.0	3.9%	tmstep_
6.0%	122.0	10.0	8.8%	parallel_
5.7%	115.3	28.7	23.3%	ghost_
4.0%	81.9	4.1	5.6%	fluxi_.LOOP@li.30
3.6%	72.7	6.3	9.3%	fluxk_.LOOP@li.38
3.6%	72.6	7.4	10.8%	fluxj_.LOOP@li.38
3.5%	71.9	85.1	63.3%	wallbc_
42.3%	862.1	--	--	MPI
17.3%	352.9	27.1	8.3%	MPI_REDUCE
16.3%	332.1	69.9	20.3%	MPI_SEND
5.9%	121.1	30.9	23.7%	MPI_ALLREDUCE
7.5%	152.3	--	--	ETC
1.6%	32.6	--	--	OMP

Running on 7 nodes with 224 MPI tasks*

Table 1: Profile by Function

Samp%	Samp	Imb. Samp	Imb. Samp%	Group Function=[MAX10] PE=HIDE
100.0%	6,633.3	--	--	Total
92.2%	6,114.0	--	--	MPI
48.9%	3,246.3	1,556.7	32.6%	MPI_ALLREDUCE
22.9%	1,521.1	3,025.9	66.8%	MPI_REDUCE
11.8%	785.4	530.6	40.5%	MPI_SEND
7.8%	518.0	579.0	53.0%	MPI_WAIT
6.9%	459.7	--	--	USER
1.7%	112.1	70.9	38.9%	fluxi_.LOOP@li.30
1.3%	88.4	25.6	22.6%	fluxk_.LOOP@li.38
1.2%	81.2	33.8	29.5%	fluxj_.LOOP@li.38

* When I ran 336 Mpi tasks – it seg faulted

WHY IS MPI GETTING BETTER SCALING THAN OPENMP ON THE NODE

- OpenMP run
  - setenv OMP_NUM_THREADS 48
  - srun –n 7 –N7 ./xleslie3d_mpi
  - MPI run
  - setenv OMP_NUM_THREADS 1
  - srun –n 336 –N7 ./xleslie3d_mpi
  - Improved OpenMP
  - setenv OMP_NUM_THREADS 24
  - srun –n 14 –N7 ./xleslie3d_mpi
  - setenv OMP_NUM_THREADS 12
  - srun –n 28 –N7 ./xleslie3d_mpi

Running across four NUMA domains will incur NUMA issues

Running with a MPI task on each NUMA domain and 12 threads/MPI task

SRUN –N 7 –N7 OMP_NUM_THREADS = 48

```
Table 4:  Memory Bandwidth by Numanode
Memory |    Read |   Write |    Thread |  Memory |  Memory | Numanode
 Traffic |  Memory |  Memory |      Time | Traffic | Traffic |  Node Id=[max3,min3]
  GBytes | Traffic | Traffic |           |  GBytes |       / |   PE=HIDE
         |  GBytes |  GBytes |           |   / Sec | Nominal |    Thread=HIDE
         |         |         |           |         |    Peak |
|-----------------------------------------------------------------------------
|  692.02 |  546.44 |  145.58 | 20.905607 |   33.10 |   12.9% | numanode.0
||----------------------------------------------------------------------------
||  692.02 |  546.44 |  145.58 | 20.826853 |   33.23 |   13.0% | nid.19
||  660.64 |  523.35 |  137.29 | 20.905607 |   31.60 |   12.3% | nid.23
||  636.23 |  502.95 |  133.28 | 20.851091 |   30.51 |   11.9% | nid.17
||   98.04 |   79.26 |   18.78 | 14.413192 |    6.80 |    2.7% | nid.21
||   52.58 |   42.79 |    9.80 | 14.388447 |    3.65 |    1.4% | nid.22
||============================================================================
|  537.23 |  424.03 |  113.20 | 20.827913 |   25.79 |   10.1% | numanode.1
||----------------------------------------------------------------------------
||  537.23 |  424.03 |  113.20 | 20.827913 |   25.79 |   10.1% | nid.18
||  289.46 |  229.36 |   60.10 | 14.449583 |   20.03 |    7.8% | nid.20
||   49.34 |   40.25 |    9.09 | 14.358161 |    3.44 |    1.3% | nid.19
||   34.87 |   28.08 |    6.79 | 14.431710 |    2.42 |    0.9% | nid.23
||============================================================================
|  492.50 |  388.74 |  103.76 | 20.828108 |   23.65 |    9.2% | numanode.2
||----------------------------------------------------------------------------
||  492.50 |  388.74 |  103.76 | 20.826837 |   23.65 |    9.2% | nid.22
||  416.20 |  328.86 |   87.34 | 20.828108 |   19.98 |    7.8% | nid.20
||  414.88 |  327.01 |   87.87 | 20.817194 |   19.93 |    7.8% | nid.21
||   80.90 |   64.72 |   16.18 | 14.427566 |    5.61 |    2.2% | nid.17
||   41.14 |   33.01 |    8.13 | 14.378477 |    2.86 |    1.1% | nid.18
||============================================================================
|  207.95 |  164.01 |   43.94 | 14.437465 |   14.40 |    5.6% | numanode.3
||----------------------------------------------------------------------------
```

SRUN –N 14 –N7 OMP_NUM_THREADS = 24

```
able 4:  Memory Bandwidth by Numanode
  Memory |    Read |   Write |     Thread |  Memory |  Memory | Numanode
 Traffic |  Memory |  Memory |       Time | Traffic | Traffic |  Node Id=[max3,min3]
  GBytes | Traffic | Traffic |            |  GBytes |       / |   PE=HIDE
         |  GBytes |  GBytes |            |   / Sec | Nominal |    Thread=HIDE
         |         |         |            |         |    Peak |
|-----------------------------------------------------------------------------
|  374.82 |  302.81 |   72.01 | 17.936564 |   20.90 |    8.2% | numanode.0
||----------------------------------------------------------------------------
||  374.82 |  302.81 |   72.01 | 17.918537 |   20.92 |    8.2% | nid.17
||  368.30 |  297.39 |   70.91 | 17.919944 |   20.55 |    8.0% | nid.22
||  364.44 |  294.10 |   70.33 | 17.936564 |   20.32 |    7.9% | nid.20
||   50.46 |   38.63 |   11.83 | 17.918304 |    2.82 |    1.1% | nid.19
||   29.65 |   25.08 |    4.57 | 11.629136 |    2.55 |    1.0% | nid.23
||    4.53 |    3.78 |    0.75 | 11.629711 |    0.39 |    0.2% | nid.18
||============================================================================
|  356.36 |  287.94 |   68.42 | 17.943760 |   19.86 |    7.8% | numanode.1
||----------------------------------------------------------------------------
||  356.36 |  287.94 |   68.42 | 17.916977 |   19.89 |    7.8% | nid.19
||  348.44 |  281.47 |   66.97 | 17.942056 |   19.42 |    7.6% | nid.18
||  343.78 |  277.59 |   66.19 | 17.933473 |   19.17 |    7.5% | nid.17
||  318.21 |  255.83 |   62.38 | 17.943760 |   17.73 |    6.9% | nid.23
||    4.24 |    3.56 |    0.68 | 11.611935 |    0.37 |    0.1% | nid.20
||============================================================================
|  376.72 |  303.36 |   73.37 | 17.940639 |   21.00 |    8.2% | numanode.2
||----------------------------------------------------------------------------
||  376.72 |  303.36 |   73.37 | 17.935383 |   21.00 |    8.2% | nid.23
||  371.79 |  300.44 |   71.35 | 17.926640 |   20.74 |    8.1% | nid.21
||  367.23 |  295.95 |   71.28 | 17.921852 |   20.49 |    8.0% | nid.22
||  355.70 |  288.58 |   67.13 | 11.620607 |   30.61 |   12.0% | nid.19
||  351.35 |  282.50 |   68.84 | 17.938276 |   19.59 |    7.7% | nid.20
||   34.92 |   29.00 |    5.91 | 11.593323 |    3.01 |    1.2% | nid.17
||============================================================================
|  367.95 |  296.27 |   71.69 | 17.937994 |   20.51 |    8.0% | numanode.3
||----------------------------------------------------------------------------
```

SRUN –N 28 –N7 OMP_NUM_THREADS = 12

able 4: Memory Bandwidth by Numanode

Memory Traffic GBytes	Read Memory Traffic GBytes	Write Memory Traffic GBytes	Thread Time	Memory Traffic GBytes / Sec	Memory Traffic / Nominal Peak	Numanode Node Id=[max3,min3] PE=HIDE Thread=HIDE
228.24	183.59	44.65	16.163097	14.12	5.5%	numanode.0
228.24	183.59	44.65	16.148021	14.13	5.5%	nid.17
223.37	180.53	42.84	16.163097	13.82	5.4%	nid.22
198.67	160.15	38.51	16.155601	12.30	4.8%	nid.23
193.59	156.32	37.27	16.159357	11.98	4.7%	nid.18
191.25	154.41	36.83	16.159708	11.83	4.6%	nid.20
190.05	154.30	35.75	16.162672	11.76	4.6%	nid.21
234.78	187.65	47.14	16.175862	14.51	5.7%	numanode.1
233.23	187.65	45.58	16.166679	14.43	5.6%	nid.18
232.62	185.48	47.14	16.175862	14.38	5.6%	nid.22
205.87	165.90	39.97	16.161616	12.74	5.0%	nid.19
200.60	161.63	38.97	16.154478	12.42	4.9%	nid.17
156.37	127.68	28.69	9.974652	15.68	6.1%	nid.20
156.16	127.78	28.38	9.981468	15.64	6.1%	nid.21
234.62	187.48	47.13	16.176280	14.50	5.7%	numanode.2
234.62	187.48	47.13	16.167319	14.51	5.7%	nid.21
232.10	185.04	47.05	16.176280	14.35	5.6%	nid.20
224.69	181.56	43.12	16.176174	13.89	5.4%	nid.18
200.39	162.84	37.55	16.152474	12.41	4.8%	nid.17
198.10	160.08	38.03	16.160609	12.26	4.8%	nid.19
156.38	127.36	29.02	9.975610	15.68	6.1%	nid.22
198.06	160.32	37.74	16.171106	12.25	4.8%	numanode.3

SRUN –N 336 –N7 OMP_NUM_THREADS = 1

```
Table 3:  Memory Bandwidth by Numanode
  Memory |    Read |   Write |     Thread |  Memory |  Memory | Numanode
 Traffic |  Memory |  Memory |       Time | Traffic | Traffic |  Node Id=[max3,min3]
  GBytes | Traffic | Traffic |            |  GBytes |       / |   PE=HIDE
         |  GBytes |  GBytes |            |   / Sec | Nominal |
         |         |         |            |         |    Peak |
|-----------------------------------------------------------------------------
|  339.06 |  282.68 |   56.38 | 72.507205 |    4.68 |    1.8% | numanode.0
||----------------------------------------------------------------------------
||  339.06 |  282.68 |   56.38 | 71.368068 |    4.75 |    1.9% | nid.18
||  332.41 |  276.58 |   55.83 | 70.427465 |    4.72 |    1.8% | nid.21
||  298.48 |  250.14 |   48.35 | 70.281285 |    4.25 |    1.7% | nid.22
||  284.96 |  239.95 |   45.02 | 72.507205 |    3.93 |    1.5% | nid.17
||  257.74 |  215.10 |   42.63 | 70.606828 |    3.65 |    1.4% | nid.20
||  253.79 |  211.00 |   42.78 | 69.717114 |    3.64 |    1.4% | nid.23
||============================================================================
|  298.16 |  246.34 |   51.82 | 73.258873 |    4.07 |    1.6% | numanode.1
||----------------------------------------------------------------------------
||  298.16 |  246.34 |   51.82 | 73.258873 |    4.07 |    1.6% | nid.17
||  291.14 |  244.89 |   46.25 | 70.993035 |    4.10 |    1.6% | nid.20
||  264.68 |  220.22 |   44.47 | 69.997530 |    3.78 |    1.5% | nid.22
||  255.54 |  211.79 |   43.75 | 69.723811 |    3.67 |    1.4% | nid.23
||  255.31 |  213.63 |   41.68 | 70.415287 |    3.63 |    1.4% | nid.21
||  222.18 |  184.48 |   37.71 | 70.866074 |    3.14 |    1.2% | nid.19
||============================================================================
|  317.07 |  267.29 |   49.78 | 72.149334 |    4.39 |    1.7% | numanode.2
||----------------------------------------------------------------------------
||  317.07 |  267.29 |   49.78 | 69.717081 |    4.55 |    1.8% | nid.23
||  301.05 |  251.67 |   49.38 | 70.680637 |    4.26 |    1.7% | nid.19
||  299.67 |  250.23 |   49.44 | 70.813903 |    4.23 |    1.7% | nid.20
||  294.69 |  246.52 |   48.17 | 71.229050 |    4.14 |    1.6% | nid.18
||  265.58 |  220.85 |   44.74 | 70.201836 |    3.78 |    1.5% | nid.21
||  248.40 |  208.18 |   40.22 | 72.149334 |    3.44 |    1.3% | nid.17
||============================================================================
|  367.85 |  307.58 |   60.27 | 73.012099 |    5.04 |    2.0% | numanode.3
||----------------------------------------------------------------------------
```