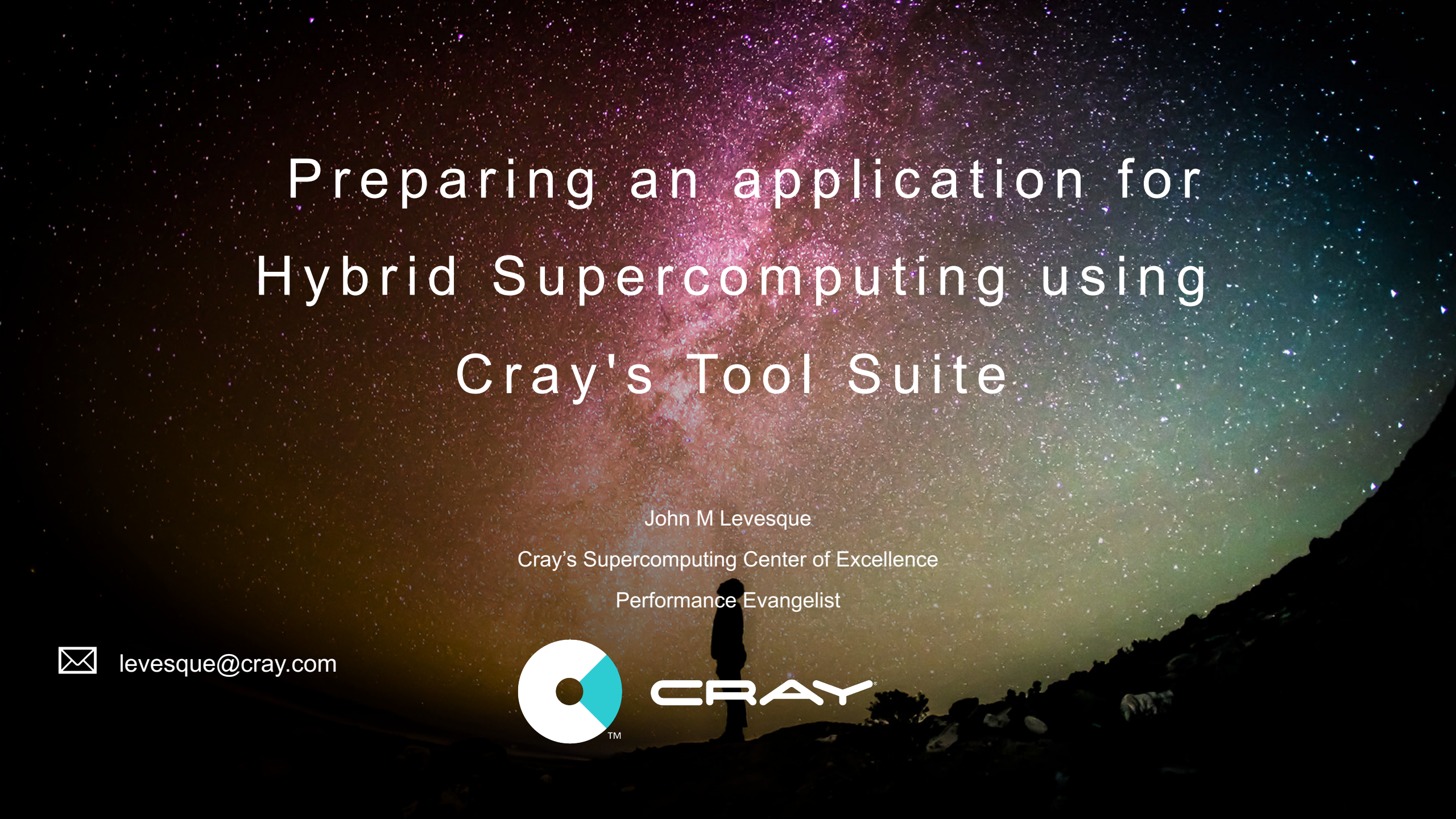
Preparing an application for Hybrid Supercomputing using Cray's Tool Suite

John M Levesque

Cray's Supercomputing Center of Excellence

Performance Evangelist

levesque@cray.com



Technical Data Rights

All materials contained in, attached to, or referenced by this document that are marked Cray Confidential or with a similar restrictive legend may not be disclosed in any form without the advance written permission of Cray, a Hewlett Packard Enterprise company. These data are submitted with limited rights under Government Contract No. B626589 and Lease Agreement 4000167127. These data may be reproduced and used by the Government with the express limitation that they will not, without written permission of Cray, be used for purposes of manufacture nor disclosed outside the Government.

This notice shall be marked on any reproduction of these data, in whole or in part.

Copyright and Trademark Acknowledgements

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CLUSTERSTOR,CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE, and REVEAL. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. ARM is a registered trademark of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ThunderX, ThunderX2, and ThunderX3 are trademarks or registered trademarks of Cavium Inc. in the U.S. and other countries. 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. Intel, the Intel logo, Intel Cilk, Intel True Scale Fabric, Intel VTune, Xeon, and Intel Xeon Phi are trademarks or registered trademarks of Intel Corporation in the U.S. and/or other countries. Lustre is a trademark of Xyratex. NVIDIA, Kepler, and CUDA are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and/or other countries.

Other trademarks used in this document are the property of their respective owners.

Forward Looking Statements

This presentation may contain forward-looking statements that involve risks, uncertainties and assumptions. If the risks or uncertainties ever materialize or the assumptions prove incorrect, the results of Hewlett Packard Enterprise Company and its consolidated subsidiaries ("Hewlett Packard Enterprise") may differ materially from those expressed or implied by such forward-looking statements and assumptions. All statements other than statements of historical fact are statements that could be deemed forward-looking statements, including but not limited to any statements regarding the expected benefits and costs of the transaction contemplated by this presentation; the expected timing of the completion of the transaction; the ability of HPE, its subsidiaries and Cray to complete the transaction considering the various conditions to the transaction, some of which are outside the parties' control, including those conditions related to regulatory approvals; projections of revenue, margins, expenses, net earnings, net earnings per share, cash flows, or other financial items; any statements concerning the expected development, performance, market share or competitive performance relating to products or services; any statements regarding current or future macroeconomic trends or events and the impact of those trends and events on Hewlett Packard Enterprise and its financial performance; any statements of expectation or belief; and any statements of assumptions underlying any of the foregoing. Risks, uncertainties and assumptions include the possibility that expected benefits of the transaction described in this presentation may not materialize as expected; that the transaction may not be timely completed, if at all; that, prior to the completion of the transaction, Cray's business may not perform as expected due to transaction-related uncertainty or other factors; that the parties are unable to successfully implement integration strategies; the need to address the many challenges facing Hewlett Packard Enterprise's businesses; the competitive pressures faced by Hewlett Packard Enterprise's businesses; risks associated with executing Hewlett Packard Enterprise's strategy; the impact of macroeconomic and geopolitical trends and events; the development and transition of new products and services and the enhancement of existing products and services to meet customer needs and respond to emerging technological trends; and other risks that are described in our Fiscal Year 2018 Annual Report on Form 10-K, and that are otherwise described or updated from time to time in Hewlett Packard Enterprise's other filings with the Securities and Exchange Commission, including but not limited to our subsequent Quarterly Reports on Form 10-Q. Hewlett Packard Enterprise assumes no obligation and does not intend to update these forward-looking statements.

Systems and Software being used

- Using 4-8 Intel Broadwell nodes – 44 cores/node
- Also using cce/9.0.0.23676 and cray-accel-nvidia60
- Perftools-base/7.1.0
- Using four nodes with 1 P100s on each node for first set of runs
- Using up to 512 nodes with P100s on each node for scaling runs

Current State of CCE/9.0.2

		CCE/9.0.X	CCE/9.0.x.classic
Fortran	Perftools-lite	YES	YES
	Perftools-lite-loops	MAYBE	MAYBE
	Perftools-lite-hbm	YES	YES
	Perftools-lite-events (GPU)	YES	YES
	Reveal	YES	YES
	Compiler Listing Info	YES	YES
C, C++	Perftools-lite	YES (-P)	YES (-P)
	Perftools-lite-loops	MAYBE	MAYBE
	Perftools-lite-hbm	YES (-P)	YES (-P)
	Perftools-lite-events (GPU)	YES (-P)	YES (-P)
	Reveal	NO	YES
	Compiler Listing Info	KIND OF	YES

When using perftool-lite tools with C++

- Your initial profile will probably be sparse
  - pat_report –P <statistics_file>
- Perftools-lite-loops turns of OpenMP, if the application call openMP API calls, the application will not compile

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

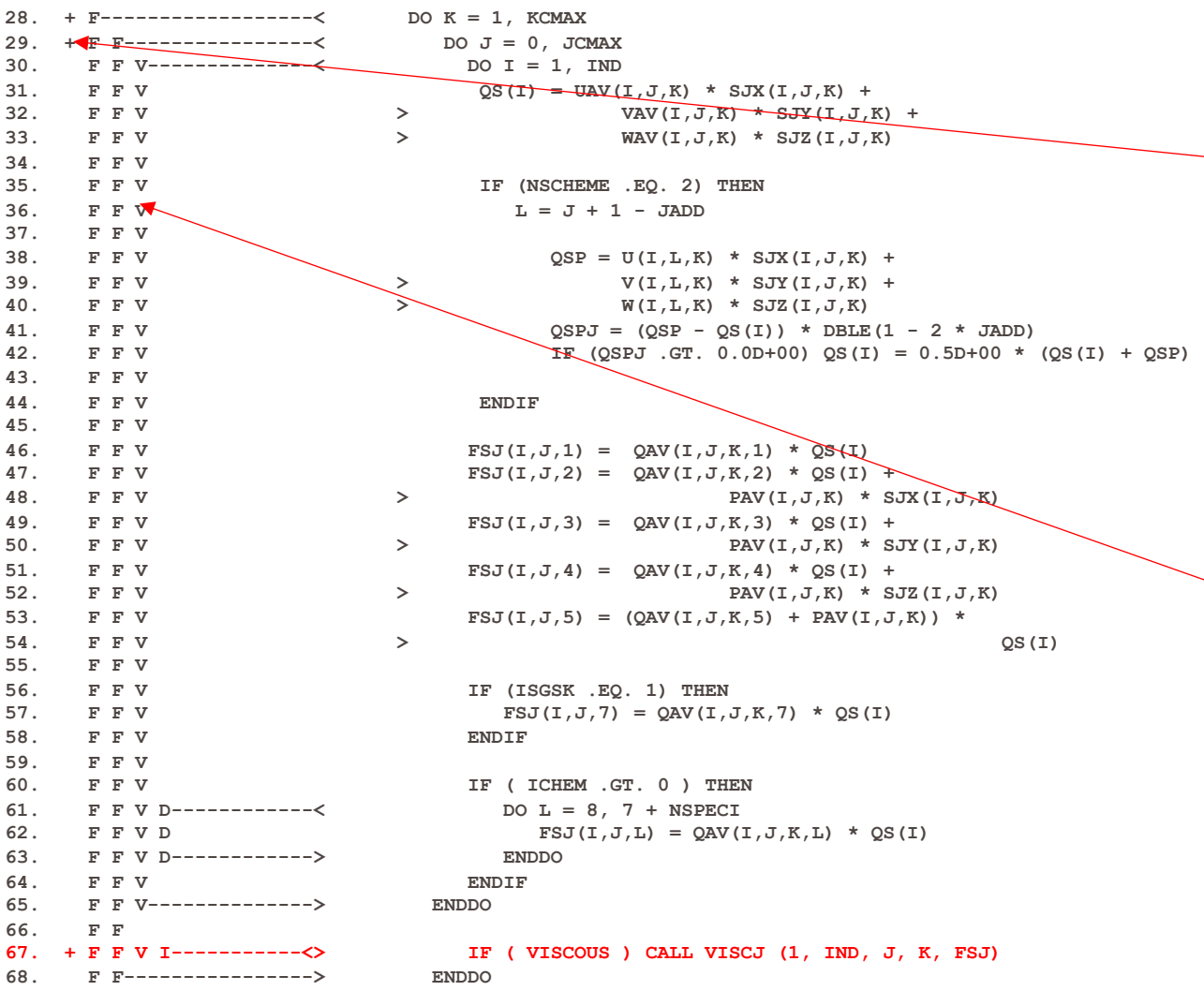
Perftools-lite profile – Run on 8 nodes–8 MPI tasks

```
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

Pat_report –Oct <statistics directory>

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

```
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_
```

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_
```

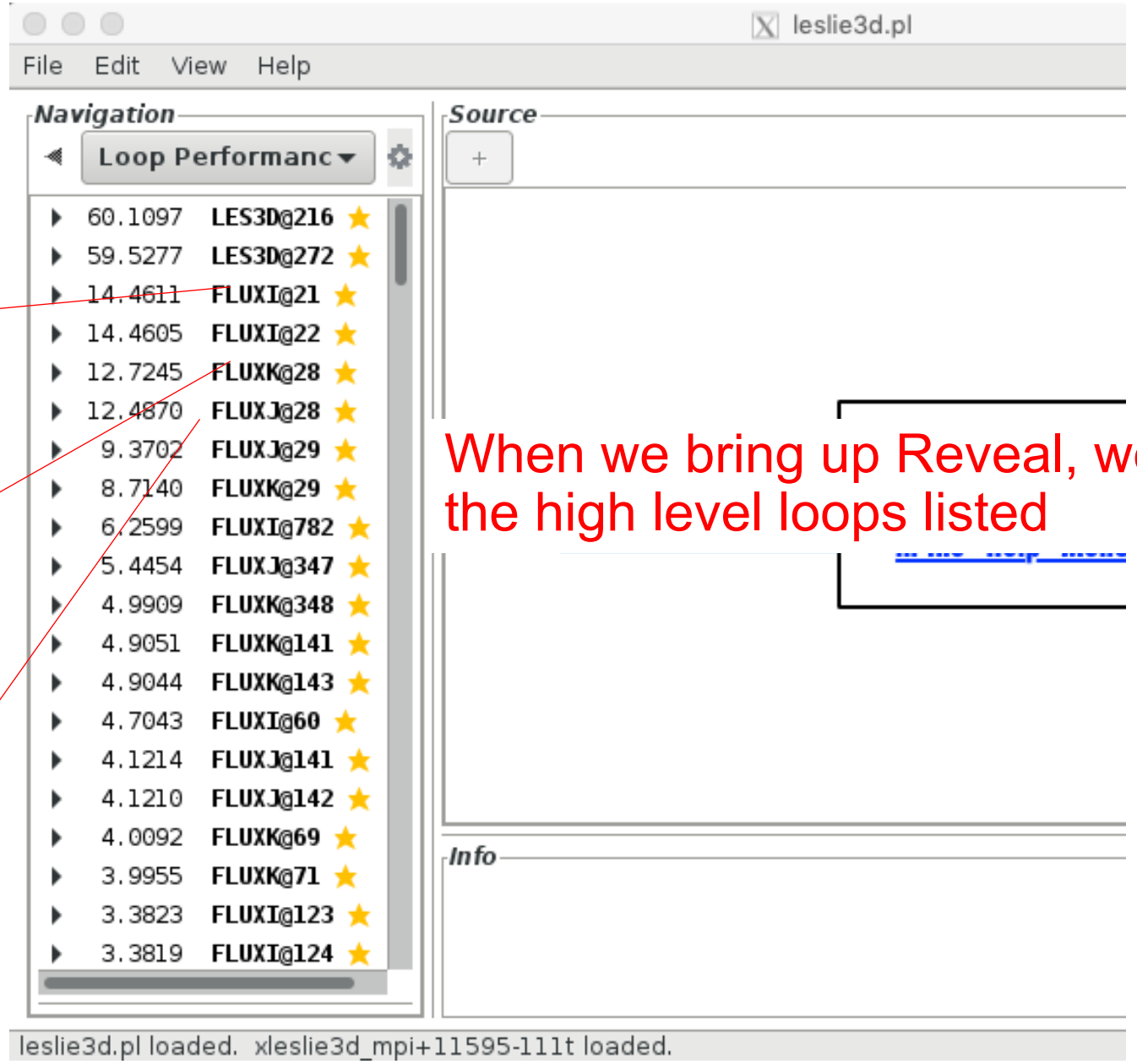
Using Reveal

- 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

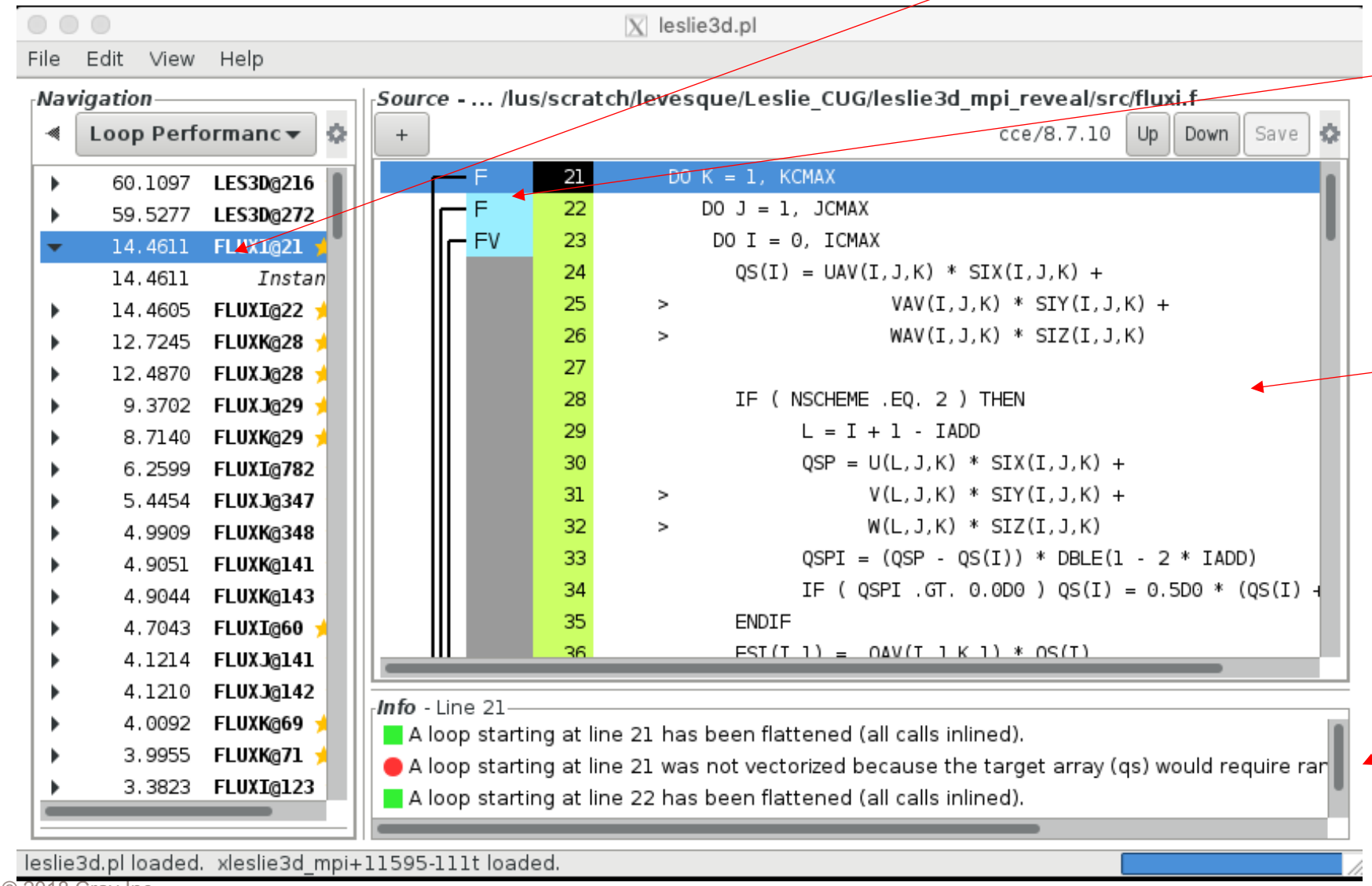
Perftools-lite-loops – Run on 8 nodes–1 MPI task/Node

```
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

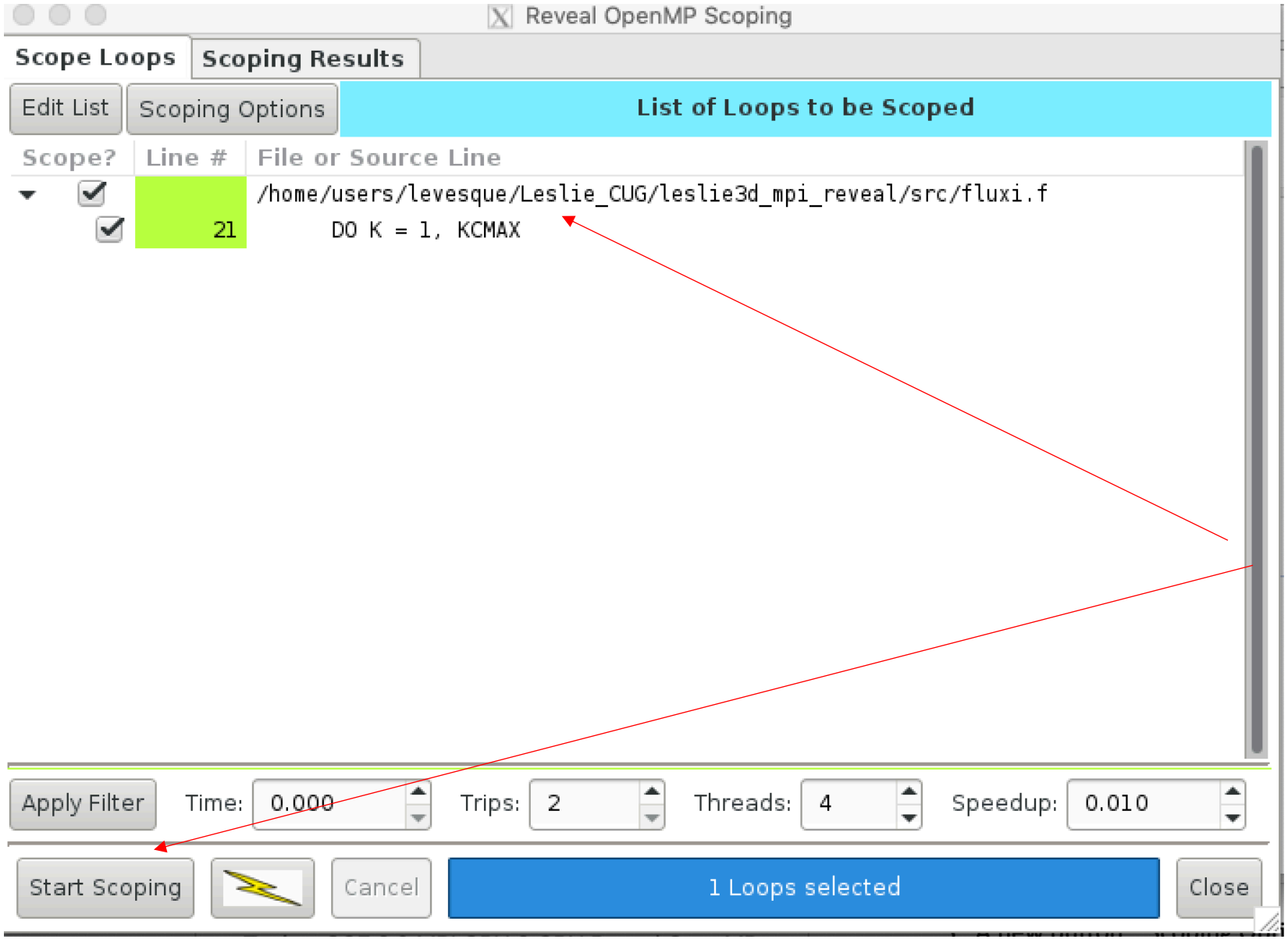
F- Flattened

V- Vectorized

Listing of loop

Diagnostics from compiler

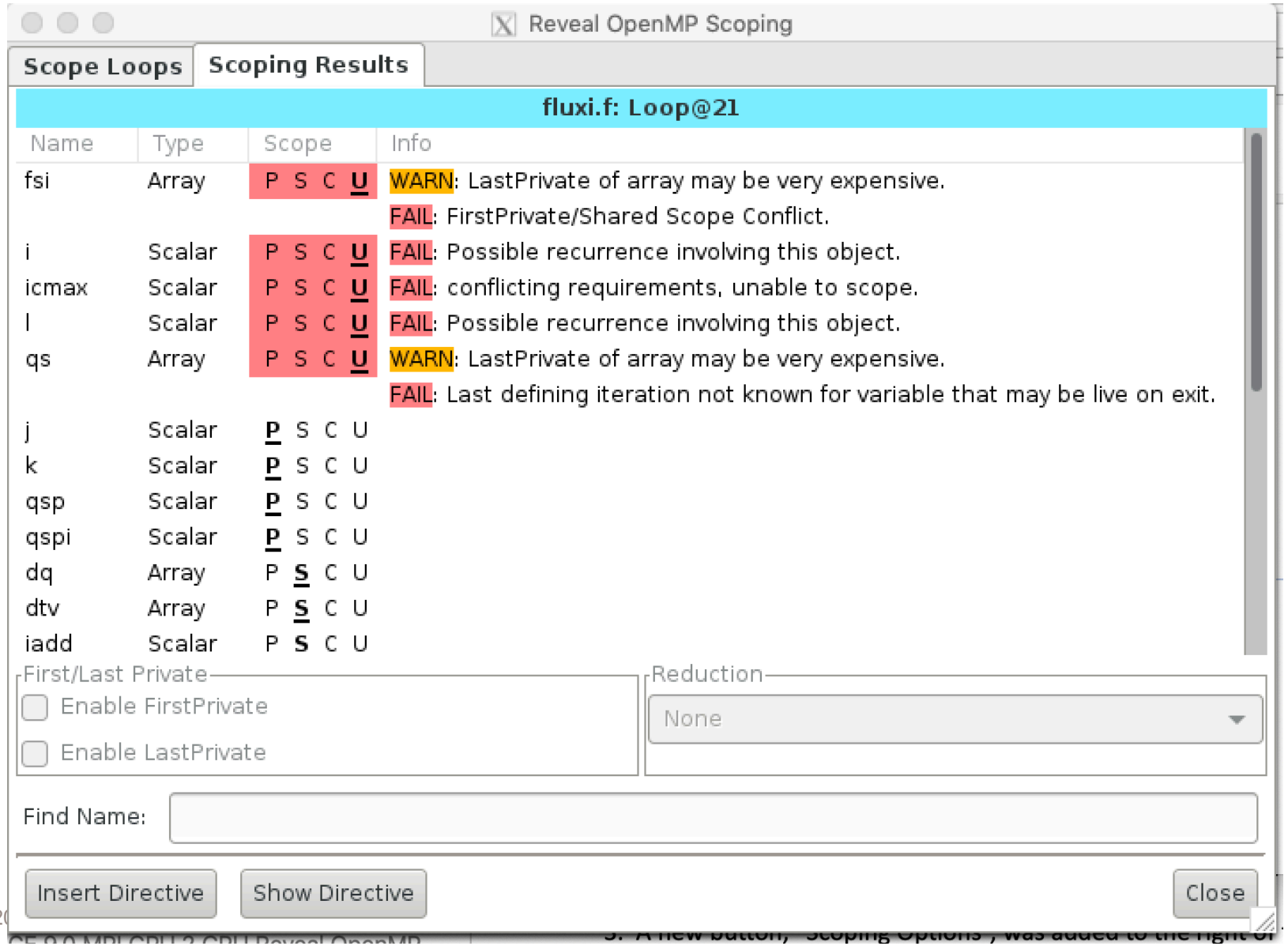
Scoping Window



Then up pops the Scoping window

Loop selected Just click on Start Scoping

Scoping results



P – Private

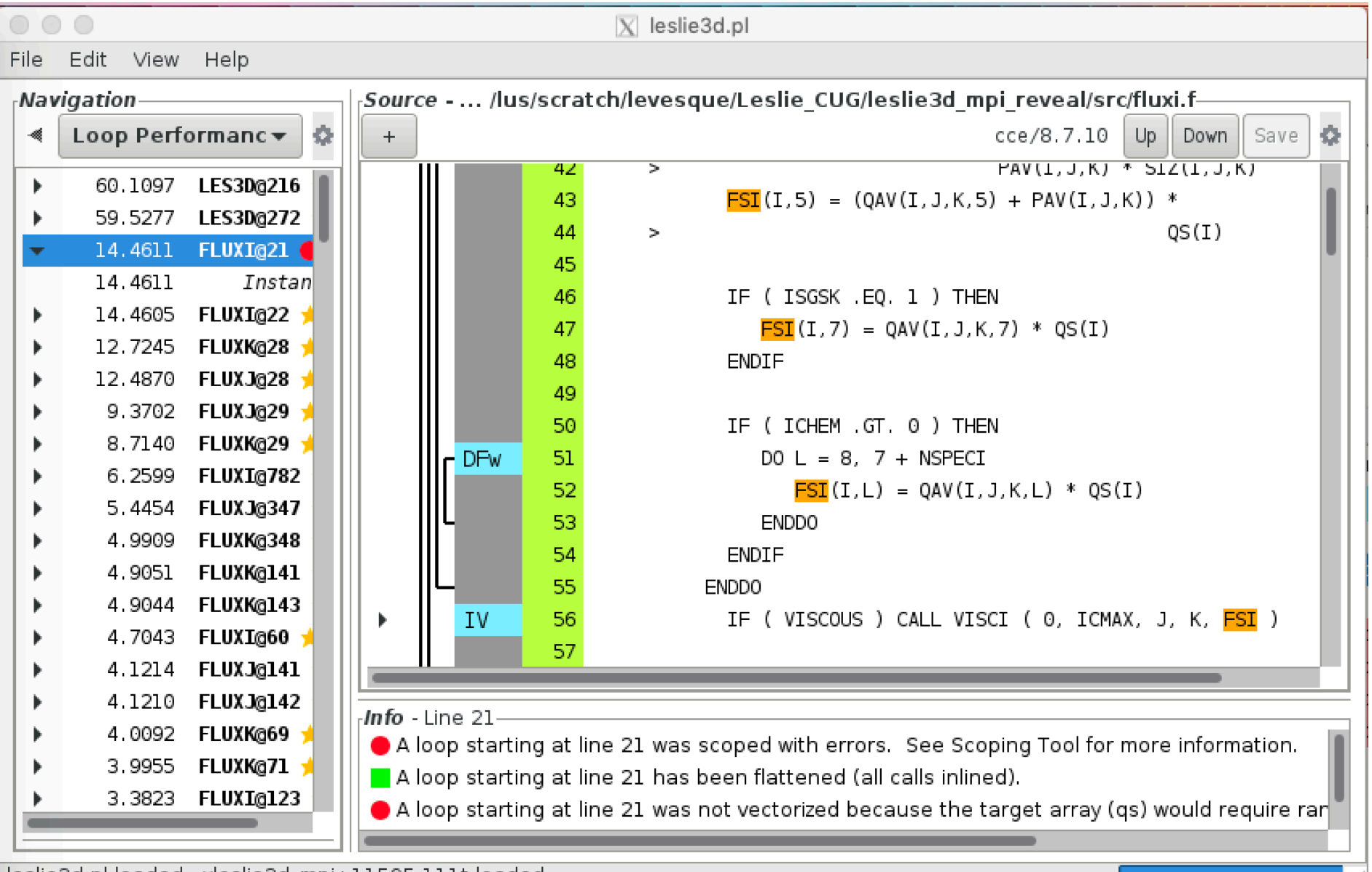
S – Shared

C – Conflict

U - Unresolved

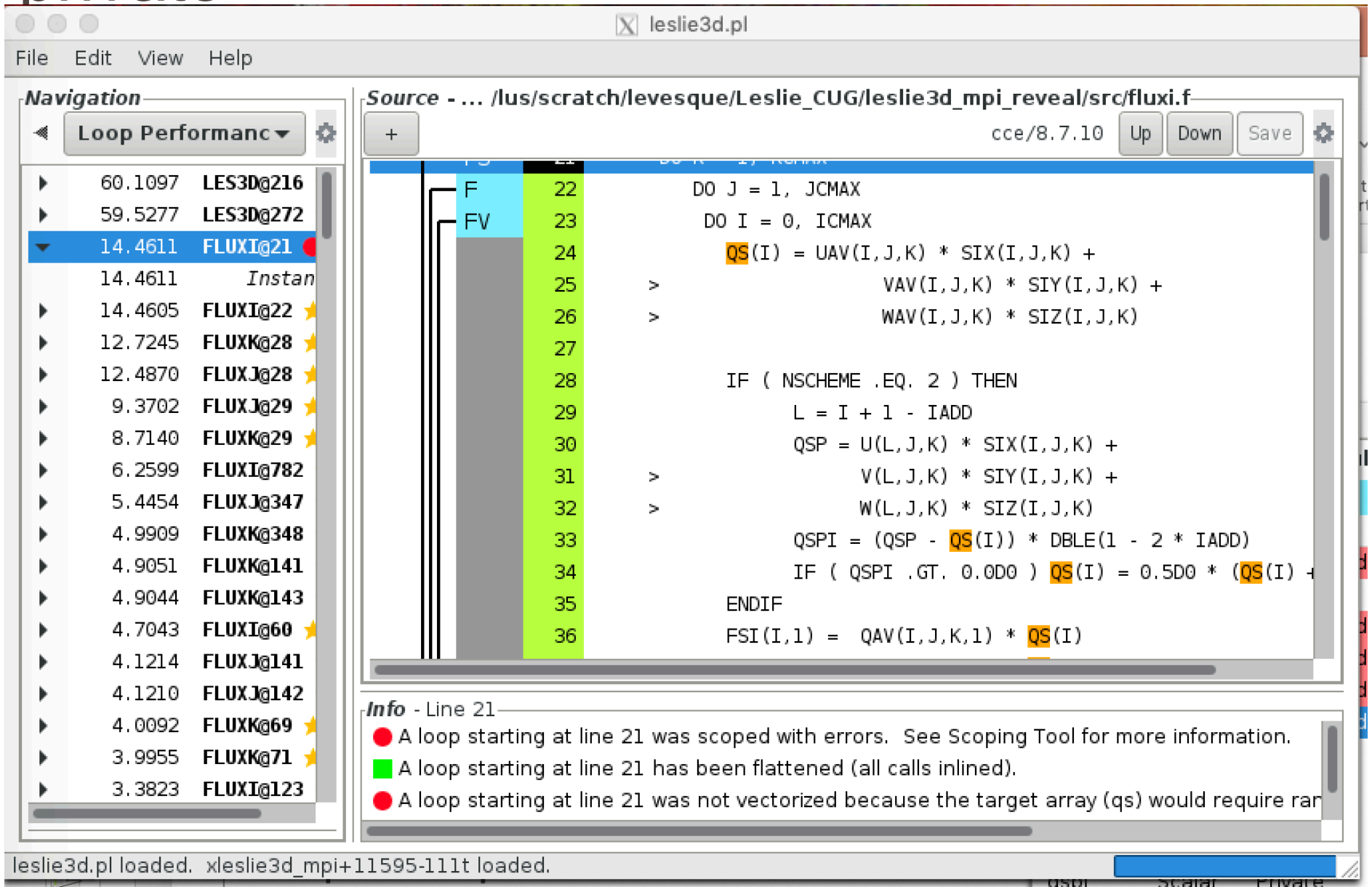
User can now change scope by selecting the appropriate letter

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

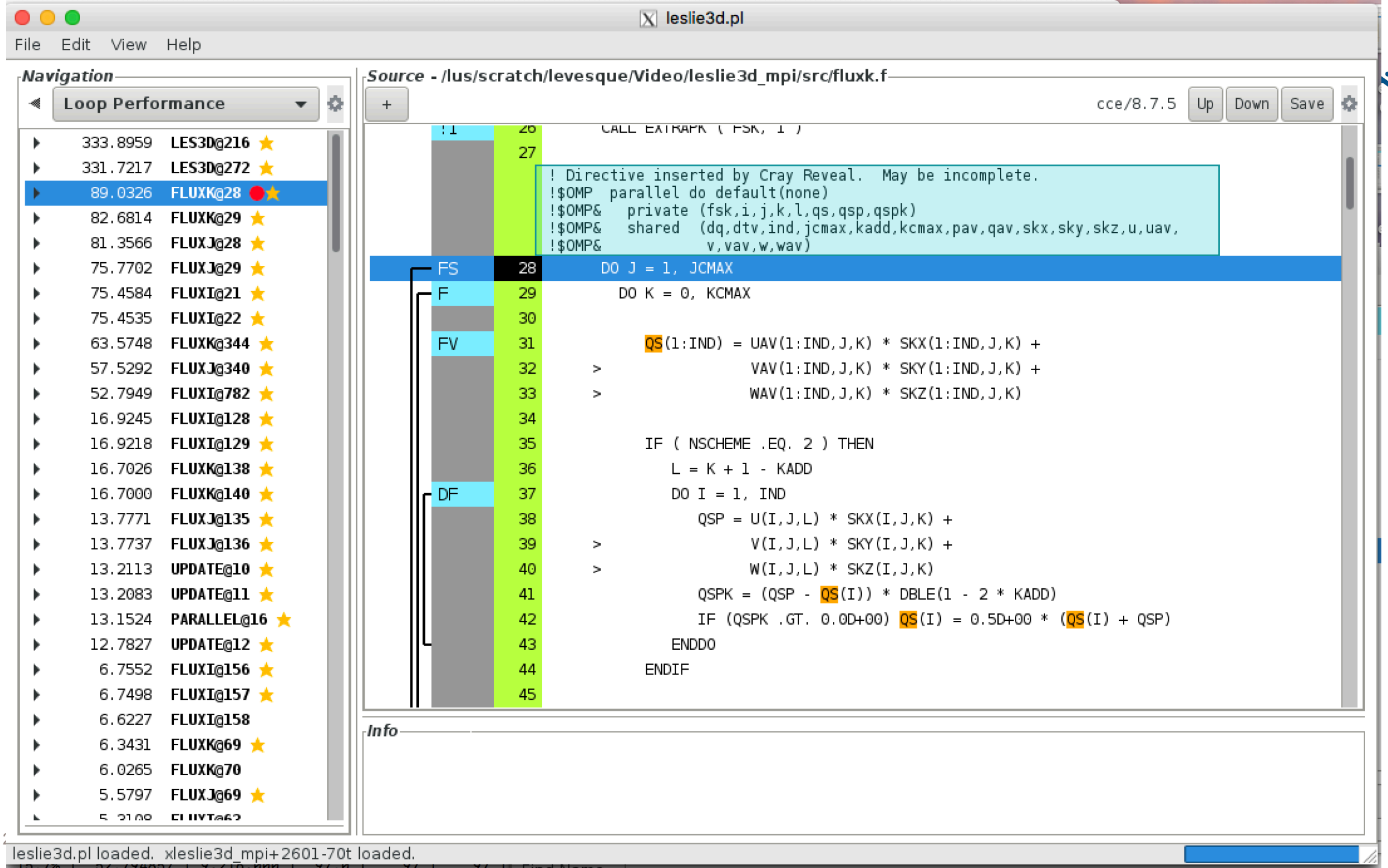
The compiler doesn't scope either FSI or QS as private



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

```
28.                        ! Directive inserted by Cray Reveal.  May be incomplete.
29.     M------------------< !$OMP  parallel do default(none)
30.     M                   !$OMP&   private (fsk,i,j,k,l,qs,qsp,qspk)
31.     M                   !$OMP&   shared  (dq,dtv,ind,jcmax,kadd,kcmax,pav,qav,skx,sky,skz,u,uav,
32.     M                   !$OMP&            v,vav,w,wav)
33.   + M mF--------------<       DO J = 1, JCMAX
34.   + M mF F------------<         DO K = 0, KCMAX
35.     M mF F V----------<           DO I = 1, IND
36.     M mF F V                        QS(I) = UAV(I,J,K) * SKX(I,J,K) +
37.     M mF F V                >                VAV(I,J,K) * SKY(I,J,K) +
38.     M mF F V                >                WAV(I,J,K) * SKZ(I,J,K)
39.     M mF F V
40.     M mF F V                        IF ( NSCHEME .EQ. 2 ) THEN
41.     M mF F V                           L = K + 1 - KADD
42.     M mF F V
43.     M mF F V
44.     M mF F V
45.     M mF F V
46.     M mF F V                              IF (QSPK .GT. 0.0D+00) QS(I) = 0.5D+00 * (QS(I) + QSP)
47.     M mF F V                        ENDIF
48.     M mF F V
49.     M mF F V                        FSK(I,K,1) =  QAV(I,J,K,1) * QS(I)
50.     M mF F V                        FSK(I,K,2) =  QAV(I,J,K,2) * QS(I) +
51.     M mF F V                >                                   PAV(I,J,K) * SKX(I,J,K)
52.     M mF F V                        FSK(I,K,3) =  QAV(I,J,K,3) * QS(I) +
53.     M mF F V                >                                   PAV(I,J,K) * SKY(I,J,K)
54.     M mF F V                        FSK(I,K,4) =  QAV(I,J,K,4) * QS(I) +
55.     M mF F V                >                                   PAV(I,J,K) * SKZ(I,J,K)
56.     M mF F V                        FSK(I,K,5) = (QAV(I,J,K,5) + PAV(I,J,K)) *
57.     M mF F V                >                                                    QS(I)
```

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

Perftools-lite profiles

Original running on 8 nodes x 1 MPI/Node

```
Table 1:  Profile by Function (limited entries shown)
  Samp% |      Samp | Imb.  |  Imb.  | Group
        |           | Samp  | Samp%  |  Function=[MAX10]
        |           |       |        |   PE=HIDE
 100.0% | 29,633.0  |   --  |    --  | Total
|-------------------------------------------------
|  98.5% | 29,183.5 |   --  |    --  | USER
||------------------------------------------------
||  27.1% |  8,030.6 | 43.4  |  0.6%  | fluxj_
||  26.9% |  7,958.6 | 47.4  |  0.7%  | fluxk_
||  22.8% |  6,745.0 | 79.0  |  1.3%  | fluxi_
||   5.2% |  1,551.4 | 21.6  |  1.6%  | extrapk_
||   5.1% |  1,499.5 | 18.5  |  1.4%  | extrapi_
||   4.6% |  1,376.2 | 24.8  |  2.0%  | update_
||   4.2% |  1,258.9 | 24.1  |  2.1%  | extrapj_
||================================================
|   1.4% |    407.9 |   --  |    --  | MPI
|================================================
```

Threaded is 26 times faster on 44 threads. However, the MPI is only using one thread, lets run 44 MPI tasks on the node

Threaded running on 8 nodes 1 MPIx44 threads /Node

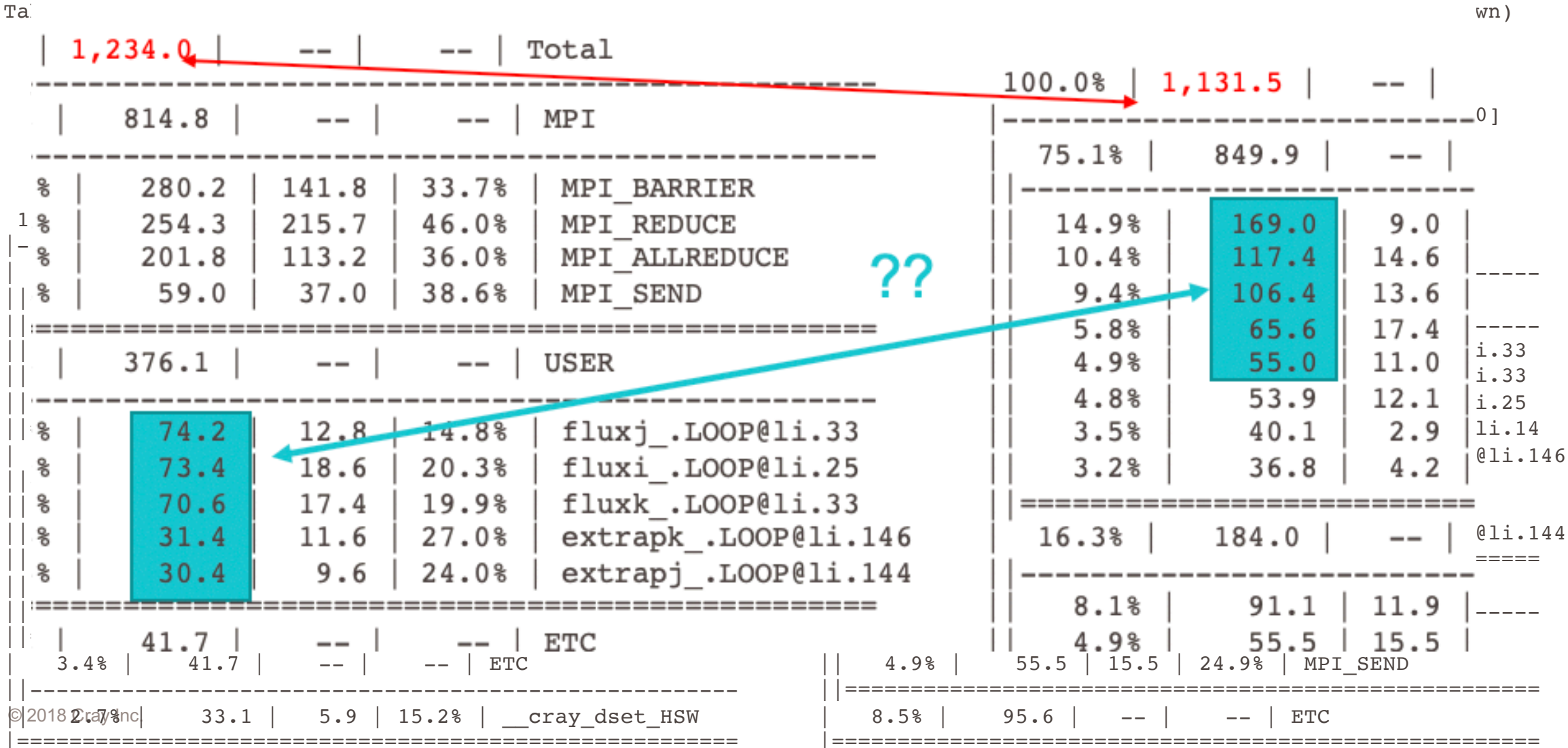
```
  Samp% |      Samp | Imb.  |  Imb.  | Group
        |           | Samp  | Samp%  |  Function=[MAX10]
        |           |       |        |   PE=HIDE
        |           |       |        |    Thread=HIDE

 100.0% | 1,131.5   |   --  |    --  | Total
|-------------------------------------------------
|  75.1% |   849.9  |   --  |    --  | USER
||------------------------------------------------
||  14.9% |   169.0  |  9.0  |  5.8%  | fluxk_.LOOP@li.33
||  10.4% |   117.4  | 14.6  | 12.7%  | fluxj_.LOOP@li.33
||   9.4% |   106.4  | 13.6  | 13.0%  | fluxi_.LOOP@li.25
||   5.8% |    65.6  | 17.4  | 23.9%  | update_.LOOP@li.14
||   4.9% |    55.0  | 11.0  | 19.0%  | extrapk_.LOOP@li.146
||   4.8% |    53.9  | 12.1  | 21.0%  | mpicx_
||   3.5% |    40.1  |  2.9  |  7.6%  | tmstep_
||   3.2% |    36.8  |  4.2  | 11.8%  | extrapj_.LOOP@li.144
||================================================
|  16.3% |   184.0  |   --  |    --  | MPI
||------------------------------------------------
||   8.1% |    91.1  | 11.9  | 13.2%  | MPI_REDUCE
||   4.9% |    55.5  | 15.5  | 24.9%  | MPI_SEND
||================================================
|   8.5% |    95.6  |   --  |    --  | ETC
|================================================
```

Perftools-lite profiles

Original running on 8 nodes x 44 MPI

Threaded running on 8 nodes 1 MPIx44 threads/Node



Why are the OpenMP loops doing poorly?

Table 3: Memory Bandwidth by Numanode (limited entries shown)

Memory Traffic GBytes	Local Memory Traffic GBytes	Remote Memory Traffic GBytes	Thread Time	Memory Traffic GBytes / Sec	Memory Traffic / Nominal Peak	Numanode Node Id=[max3,min3] PE=HIDE Thread=HIDE
184.47	173.59	10.89	11.578777	15.93	20.7%	numanode.0
183.50	173.59	9.91	11.569322	15.86	20.7%	nid.63
182.61	172.40	10.21	11.578777	15.77	20.5%	nid.61
178.55	167.75	10.80	11.563156	15.44	20.1%	nid.71
178.10	168.14	9.96	11.562097	15.40	20.1%	nid.62
178.08	168.07	10.01	11.564512	15.40	20.1%	nid.68
178.01	167.20	10.82	11.572032	15.38	20.0%	nid.70
60.36	14.73	45.62	9.073119	6.65	8.7%	numanode.1
60.36	14.73	45.62	9.072693	6.65	8.7%	nid.63
59.88	14.33	45.55	9.071553	6.60	8.6%	nid.62
59.48	14.19	45.29	9.068044	6.56	8.5%	nid.68
58.78	13.70	45.08	9.069259	6.48	8.4%	nid.70
58.67	13.87	44.81	9.071591	6.47	8.4%	nid.69
58.53	13.86	44.67	9.067146	6.46	8.4%	nid.71

The threads on socket 0 have great local memory bandwidth because they are accessing most of their data on socket 0.

The threads on socket 1 have bad remote memory bandwidth because they are accessing most of their data off socket 0.

Why is MPI getting better scaling than OpenMP on the node

- OpenMP run
  - setenv OMP_NUM_THREADS 44
  - aprun -n 8 -N 1 -d 44 ./xleslie3d_mpi > out8on8_44
- MPI run
  - setenv OMP_NUM_THREADS 1
  - aprun -n 352 -N 44 ./xleslie3d_mpi > all_mpi
- Improved OpenMP
  - setenv OMP_NUM_THREADS 22
  - aprun -n 8 -N 2 –S 1 -d 22 ./xleslie3d_mpi > out8on8_44

Running across two sockets will incur NUMA issues

Running with a MPI task on each socket and 22 threads/MPI task

Perftools-lite profiles

Original running on 8 nodes x 44 MPI

Table 1: Profile by Function (limited entries shown)

Samp%	Samp	Imb. Samp	Imb. Samp%	Group Function=[MAX10] PE=HIDE
100.0%	1,234.0	--	--	Total
66.0%	814.8	--	--	MPI
22.7%	280.2	141.8	33.7%	MPI_BARRIER
20.6%	254.3	215.7	46.0%	MPI_REDUCE
16.4%	201.8	113.2	36.0%	MPI_ALLREDUCE
4.8%	59.0	37.0	38.6%	MPI_SEND
30.5%	376.1	--	--	USER
6.0%	74.2	12.8	14.8%	fluxj_.LOOP@li.33
5.9%	73.4	18.6	20.3%	fluxi_.LOOP@li.25
5.7%	70.6	17.4	19.9%	fluxk_.LOOP@li.33
2.5%	31.4	11.6	27.0%	extrapk_.LOOP@li.146
2.5%	30.4	9.6	24.0%	extrapj_.LOOP@li.144
3.4%	41.7	--	--	ETC
2.7%	33.1	5.9	15.2%	__cray_dset_HSW

Threaded running on 8 nodes 2 MPIx22 threads/Node

Samp%	Samp	Imb. Samp	Imb. Samp%	Group Function=[MAX10] PE=HIDE Thread=HIDE
100.0%	843.8	--	--	Total
55.0%	463.9	--	--	USER
9.9%	83.1	9.9	11.3%	fluxk_.LOOP@li.33
9.2%	77.6	8.4	10.5%	fluxj_.LOOP@li.33
8.7%	73.3	14.7	17.8%	fluxi_.LOOP@li.25
3.6%	30.1	8.9	24.3%	extrapk_.LOOP@li.146
3.5%	29.8	10.2	27.3%	mpicx_
3.4%	28.4	6.6	20.0%	update_.LOOP@li.14
37.4%	315.4	--	--	MPI
19.4%	164.0	51.0	25.3%	MPI_REDUCE
9.4%	79.1	36.9	33.9%	MPI_ALLREDUCE
6.4%	54.1	27.9	36.3%	MPI_SEND
7.5%	63.2	--	--	ETC
2.9%	24.2	0.8	3.5%	__cray_dset_HSW

Let's look at perftools-lite-hbm

Table 5: Profile by Group, Function, and Line (limited entries shown)

Samp%	Samp	Imb. Samp	Imb. Samp%	MEM_LOAD_UOPS_RETIRED :HIT_LFB:precise=2	RESOURCE_STALLS :ANY	Group Function=[MAX10] Source Line PE=HIDE
100.0%	59.5	--	--	5,805,421,397,144,252	209,611,978,880	Total
97.1%	57.8	--	--	5,594,315,164,526,714	208,307,562,822	USER
23.5%	14.0	--	--	1,231,453,023,644,516	56,564,956,788	fluxj_ fluxj.f
4.2%	2.5	1.5	42.9%	246,290,604,776,946	11,331,973,206	line.36
7.8%	4.6	5.4	61.4%	387,028,093,089,632	18,250,631,160	line.72
8.0%	4.8	4.2	54.0%	316,659,349,107,561	16,880,190,381	line.81
23.3%	13.9	--	--	1,618,481,116,517,358	46,077,159,986	fluxi_ fluxi.f
5.7%	3.4	1.6	37.1%	316,659,348,989,901	9,236,821,683	line.29
1.5%	0.9	2.1	81.0%	70,368,744,186,054	5,096,471,788	line.41
1.1%	0.6	1.4	78.6%	70,368,744,181,663	2,885,621,509	line.42
1.7%	1.0	1.0	57.1%	105,553,116,300,240	5,639,566,321	line.48
7.8%	4.6	3.4	48.2%	668,503,069,752,688	13,084,700,740	line.61
4.8%	2.9	1.1	32.1%	316,659,348,925,684	7,922,852,884	line.68

Let's look at perftools-lite-hbm

```
28.                          ! Directive inserted by Cray Reveal.  May be incomplete.
29.    M---------------------< !$OMP  parallel do default(none)
30.    M                     !$OMP&   private (fsj,i,j,k,l,qs,qsp,qspj)
31.    M                     !$OMP&   shared  (dq,dtv,ind,jadd,jcmax,kcmax,pav,qav,sjx,sjy,sjz,u,uav,
32.    M                     !$OMP&            v,vav,w,wav)
33.  + M mF------------------<       DO K = 1, KCMAX
34.  + M mF F---------------<          DO J = 0, JCMAX
35.    M mF F
36.  + M mF F fVF----------<>             QS(1:IND) = UAV(1:IND,J,K) * SJX(1:IND,J,K) +
37.    M mF F                >                       VAV(1:IND,J,K) * SJY(1:IND,J,K) +
38.    M mF F                >                       WAV(1:IND,J,K) * SJZ(1:IND,J,K)
39.    M mF F
40.    M mF F                                   IF (NSCHEME .EQ. 2) THEN
41.    M mF F                                      L = J + 1 - JADD
42.    M mF F D-------------<                      DO I = 1, IND
43.    M mF F D                                       QSP = U(I,L,K) * SJX(I,J,K) +
44.    M mF F D              >                              V(I,L,K) * SJY(I,J,K) +
45.    M mF F D              >                              W(I,L,K) * SJZ(I,J,K)
46.    M mF F D                                       QSPJ = (QSP - QS(I)) * DBLE(1 - 2 *
47.    M mF F D                                       IF (QSPJ .GT. 0.0D+00) QS(I) = 0.5D+
48.    M mF F D------------->                      ENDDO
49.    M mF F                                   ENDIF
50.    M mF F
51.  + M mF F fF----------<>               FSJ(1:IND,J,1) =  QAV(1:IND,J,K,1) * QS(1:I
52.  + M mF F fF----------<>               FSJ(1:IND,J,2) =  QAV(1:IND,J,K,2) * QS(1:I
53.    M mF F                >                                  PAV(1:IND,J,K) * SJX(
54.  + M mF F fF----------<>               FSJ(1:IND,J,3) =  QAV(1:IND,J,K,3) * QS(1:IND) +
55.    M mF F                >                                  PAV(1:IND,J,K) * SJY(1:IND,J,K)
```

Array syntax is very poor for cache based systems.

The compiler is even fusing 36-38 with 51-55

Rewritten with loops

```
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) +
```

Let's look at perftools-lite-hbm

```
73.     M mF
74.   + M mF ib--------------<          DO J = 1, JCMAX
75.   + M mF ib ib-----------<             DO L = 1, 5
76.     M mF ib ib Vbr2-----<>                DQ(1:IND,J,K,L) = DQ(1:IND,J,K,L) -
77.     M mF ib ib                  >             DTV(1:IND,J,K) * (FSJ(1:IND,J,L) - FSJ(1:IND,J-1,L))
78.     M mF ib ib----------->             ENDDO
79.     M mF ib
80.     M mF ib                            IF (ISGSK .EQ. 1) THEN
81.     M mF ib                               DQ(1:IND,J,K,7) = DQ(1:IND,J,K,7) -
82.     M mF ib                     >             DTV(1:IND,J,K) * (FSJ(1:IND,J,7) - FSJ(1:IND,J-1,7))
83.     M mF ib                            ENDIF
84.     M mF ib
85.     M mF ib                            IF ( ICHEM .GT. 0 ) THEN
86.     M mF ib D-----------<                 DO L = 8, 7 + NSPECI
87.     M mF ib D                                DQ(1:IND,J,K,L) = DQ(1:IND,J,K,L) -
88.     M mF ib D                   >             DTV(1:IND,J,K) * (FSJ(1:IND,J,L) - FSJ(1:IND,J-1,L))
89.     M mF ib D----------->                 ENDDO
90.     M mF ib                            ENDIF
91.     M mF ib------------->           ENDDO
92.     M mF---------------->>        ENDDO
```

Be careful of compiler's' automatic blocking

Rewritten with loops and directives

```
70.     F                    !dir$ noblocking
71.   + F ir2--------------<         DO J = 1, JCMAX
72.     F ir2                !dir$ noblocking
73.     F ir2 iVr2---------<           DO I = 1, IND
74.     F ir2 iVr2           !dir$ noblocking
75.   + F ir2 iVr2 ir2-----<             DO L = 1, 5
76.     F ir2 iVr2 ir2                     DQ(I,J,K,L) = DQ(I,J,K,L) -
77.     F ir2 iVr2 ir2               >           DTV(I,J,K) * (FSJ(I,J,L) - FSJ(I,J-1,L))
78.     F ir2 iVr2 ir2----->               ENDDO
79.     F ir2 iVr2
80.     F ir2 iVr2                         IF (ISGSK .EQ. 1) THEN
81.     F ir2 iVr2                         DQ(I,J,K,7) = DQ(I,J,K,7) -
82.     F ir2 iVr2                   >           DTV(I,J,K) * (FSJ(I,J,7) - FSJ(I,J-1,7))
83.     F ir2 iVr2                         ENDIF
84.     F ir2 iVr2
85.     F ir2 iVr2                         IF ( ICHEM .GT. 0 ) THEN
86.     F ir2 iVr2 D-------<               DO L = 8, 7 + NSPECI
87.     F ir2 iVr2 D                          DQ(I,J,K,L) = DQ(I,J,K,L) -
88.     F ir2 iVr2 D                 >           DTV(I,J,K) * (FSJ(I,J,L) - FSJ(I,J-1,L))
89.     F ir2 iVr2 D------->               ENDDO
90.     F ir2 iVr2                         ENDIF
91.     F ir2 iVr2--------->             ENDDO
92.     F ir2-------------->           ENDDO
93.     F----------------->          ENDDO
```

Perftools-lite profiles

After removing array syntax

Table 1: Profile by Function (limited entries shown)

Samp%	Samp	Imb.	Imb.	Group
100.0%	803.9	--	--	Total
54.0%	433.8	--	--	USER
9.7%	78.0	6.0	7.6%	fluxk_.LOOP@li.32
8.5%	68.3	7.7	10.8%	fluxj_.LOOP@li.32
8.1%	65.1	5.9	8.8%	fluxi_.LOOP@li.25
3.5%	28.4	3.6	12.1%	extrapk_.LOOP@li.150
3.3%	26.8	12.2	33.5%	mpicx_
3.3%	26.6	2.4	9.0%	update_.LOOP@li.14
37.7%	302.8	--	--	MPI
22.0%	176.8	56.2	25.8%	MPI_REDUCE
10.2%	81.7	46.3	38.6%	MPI_ALLREDUCE
3.9%	31.2	14.8	34.2%	MPI_SEND
8.2%	66.0	--	--	ETC
3.1%	25.3	1.7	6.7%	cray_dset_HSW

Table 1: Profile by Function (limited entries shown)

Samp%	Samp	Imb.	Imb.	Group
100.0%	843.8	--	--	Total
55.0%	463.9	--	--	USER
9.9%	83.1	9.9	11.3%	fluxk
9.2%	77.6	8.4	10.5%	fluxj
8.7%	73.3	14.7	17.8%	fluxi
3.6%	30.1	8.9	24.3%	extra
3.5%	29.8	10.2	27.3%	mpicx
3.4%	28.4	6.6	20.0%	updat
37.4%	315.4	--	--	MPI
19.4%	164.0	51.0	25.3%	MPI_REDUCE
9.4%	79.1	36.9	33.9%	MPI_ALLREDUCE
6.4%	54.1	27.9	36.3%	MPI_SEND
7.5%	63.2	--	--	ETC
2.9%	24.2	0.8	3.5%	__cray_dset_HSW

So lets add OpenMP Accelerator directives

```
#ifdef OMP_TARGET
!$omp target teams distribute
!$omp&    private(fsi,i,j,k,l,qs,qsp,qspi)
#else
! Directive inserted by Cray Reveal.  May be incomplete.
!$OMP  parallel do default(none)
!$OMP&   private (fsi,i,j,k,l,qs,qsp,qspi)
!$OMP&   shared  (dq,dtv,iadd,icmax,jcmax,kcmax,pav,qav,six,siy,siz,u,
!$OMP&            uav,v,vav,w,wav)
#endif
      DO K = 1, KCMAX
         DO J = 1, JCMAX
          DO I = 0, ICMAX
            QS(I) = UAV(I,J,K) * SIX(I,J,K) +
     >                VAV(I,J,K) * SIY(I,J,K) +
     >                WAV(I,J,K) * SIZ(I,J,K)


            IF ( NSCHEME .EQ. 2 ) THEN
                 L = I + 1 - IADD
                 QSP = U(L,J,K) * SIX(I,J,K) +
     >                  V(L,J,K) * SIY(I,J,K) +
     >                  W(L,J,K) * SIZ(I,J,K)
                 QSPI = (QSP - QS(I)) * DBLE(1 - 2 * IADD)
                 IF ( QSPI .GT. 0.0D0 ) QS(I) = 0.5D0 * (QS(I) + QSP)
            ENDIF
```

Only need to worry about private variables

Don’t worry about data movement yet

Reveal put these directives in

So lets add OpenMP Accelerator directives

```
21.  + G----------------< !$omp target teams distribute
22.    G                 !$omp&    private(fsi,i,j,k,l,qs,qsp,qspi)
23.    G                 #else
24.    D                 ! Directive inserted by Cray Reveal.  May be incomplete.
25.    D                 !$OMP  parallel do default(none)
26.    D                 !$OMP&   private (fsi,i,j,k,l,qs,qsp,qspi)
27.    D                 !$OMP&   shared  (dq,dtv,iadd,icmax,jcmax,kcmax,pav,qav,six,siy,siz,u,
28.    D                 !$OMP&            uav,v,vav,w,wav)
29.    G                 #endif
30.  + G gF-------------<       DO K = 1, KCMAX
31.  + G gF F-----------<          DO J = 1, JCMAX
32.    G gF F
33.  + G gF F fgF------<>             QS(0:ICMAX) = UAV(0:ICMAX,J,K) * SIX(0:ICMAX,J,K) +
34.    G gF F                >                     VAV(0:ICMAX,J,K) * SIY(0:ICMAX,J,K) +
35.    G gF F                >                     WAV(0:ICMAX,J,K) * SIZ(0:ICMAX,J,K)
36.    G gF F
37.    G gF F                             IF ( NSCHEME .EQ. 2 ) THEN
38.    G gF F D---------<                    DO I = 0, ICMAX
39.    G gF F D                                 L = I + 1 - IADD
40.    G gF F D                                 QSP = U(L,J,K) * SIX(I,J,K) +
41.    G gF F D              >                        V(L,J,K) * SIY(I,J,K) +
42.    G gF F D              >                        W(L,J,K) * SIZ(I,J,K)
43.    G gF F D                                 QSPI = (QSP - QS(I)) * DBLE(1 - 2 * IADD)
44.    G gF F D                                 IF ( QSPI .GT. 0.0D0 ) QS(I) = 0.5D0 * (QS(I) + QSP)
45.    G gF F D--------->                    ENDDO
46.    G gF F                             ENDIF
```

So lets add Accelerator directives

Let the compiler figure out data movement

```
ftn-6418 ftn: ACCEL FLUXI, File = fluxi.f, Line = 21
  If not already present: allocate memory and copy whole array "siy" to accelerator, free at line 93 (acc_copyin).

ftn-6418 ftn: ACCEL FLUXI, File = fluxi.f, Line = 21
  If not already present: allocate memory and copy whole array "vav" to accelerator, free at line 93 (acc_copyin).

ftn-6418 ftn: ACCEL FLUXI, File = fluxi.f, Line = 21
  If not already present: allocate memory and copy whole array "six" to accelerator, free at line 93 (acc_copyin).

ftn-6418 ftn: ACCEL FLUXI, File = fluxi.f, Line = 21
  If not already present: allocate memory and copy whole array "uav" to accelerator, free at line 93 (acc_copyin).

ftn-6418 ftn: ACCEL FLUXI, File = fluxi.f, Line = 21
  If not already present: allocate memory and copy whole array "siz" to accelerator, free at line 93 (acc_copyin).

ftn-6418 ftn: ACCEL FLUXI, File = fluxi.f, Line = 21
  If not already present: allocate memory and copy whole array "wav" to accelerator, free at line 93 (acc_copyin).

ftn-6418 ftn: ACCEL FLUXI, File = fluxi.f, Line = 21
  If not already present: allocate memory and copy whole array "qav" to accelerator, free at line 93 (acc_copyin).
```

Approach for turning OpenMP threading into OpenMP Accelerator Directives

- Will also use perftools to investigate accelerator performance
  - Module load perftools instead of perftools-lite
    - Perftools-lite-gpu doesn't show much right now
  - After building application pat_build –u –g mpi <executable>
  - Run <executable+pat>
  - pat_report < directory created when running <executable+pat>>

Profile from first cut at OpenMP Offload

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

  Time% |        Time |     Imb. |  Imb. |    Calls | Group
        |             |     Time | Time% |          |  Function
100.0% | 192.065111 |       -- |    -- | 22,492.0 | Total
|-----------------------------------------------------------------------
|  96.3% | 184.972019 |       -- |    -- |  9,400.0 | OACC
||----------------------------------------------------------------------
||  17.1% |  32.847375 | 0.018724 |  0.1% |    200.0 | fluxi_.ACC_COPY@li.29
||  15.8% |  30.296096 | 0.020366 |  0.1% |    200.0 | fluxk_.ACC_COPY@li.24
||  15.8% |  30.291321 | 0.018862 |  0.1% |    200.0 | fluxj_.ACC_COPY@li.24
||  13.5% |  25.842807 | 0.003061 |  0.0% |    200.0 | fluxi_.ACC_COPY@li.266
||  12.4% |  23.890719 | 0.003862 |  0.0% |    200.0 | fluxj_.ACC_COPY@li.109
||  12.4% |  23.889958 | 0.001421 |  0.0% |    200.0 | fluxk_.ACC_COPY@li.106
||   4.1% |   7.804356 | 0.015985 |  0.3% |    200.0 | update_.ACC_COPY@li.10
||   1.0% |   1.962536 | 0.000358 |  0.0% |    200.0 | fluxi_.ACC_SYNC_WAIT@li.265
||======================================================================
|   2.9% |   5.591116 |       -- |    -- |  6,546.0 | USER
||----------------------------------------------------------------------
```

Copies to and from the GPU

Waiting for GPU kernel to finish

Approach for turning OpenMP threading into OpenMP Accelerator Directives

- First convert all OpenMP high level loops into OpenMP Accelerator Directives

- Next introduce DATA regions up the call tree

    - Insert appropriate data updates and move more loops to the accelerator

Perftools-lite loops – Run on 8 nodes–8 MPI tasks

Can we introduce DATA regions here?

```
Table 1:  Function Calltree View (limited entries shown)
  Time% |       Time |       Calls | Calltree
        |            |             |  PE=HIDE
 100.0% | 331.356157 |          -- | Total
|-------------------------------------------------------
| 100.0% | 331.356117 |         2.0 | les3d_
|  99.3% | 329.201866 |          -- |  les3d_.LOOP.3.li.216
3  98.7% | 327.029239 |          -- |   les3d_.LOOP.4.li.272
||||-----------------------------------------------------
4|||  31.6% | 104.588754 |     1,000.0 | fluxk_
|||||-----------------------------------------------------
5||||  20.2% |  66.813949 |          -- | fluxk_.LOOP.1.li.28
6||||        |            |             |  fluxk_.LOOP.2.li.29
7||||  20.2% |  66.813949 | 9,312,000.0 |   visck_
5||||   6.4% |  21.068680 |     1,000.0 | fluxk_(exclusive)
5||||   5.0% |  16.706125 |     1,000.0 | extrapk_
|||||=====================================================
4|||  29.2% |  96.783424 |     1,000.0 | fluxi_
|||||-----------------------------------------------------
5||||  16.9% |  55.929620 |          -- | fluxi_.LOOP.1.li.21
6||||        |            |             |  fluxi_.LOOP.2.li.22
7||||  16.9% |  55.929620 | 9,216,000.0 |   visci_
5||||   6.8% |  22.465390 |     1,000.0 | extrapi_
||||||----------------------------------------------------
6|||||   3.4% |  11.399343 |     1,000.0 | extrapi_(exclusive)
6|||||   3.3% |  11.066047 |          -- | extrapi_.LOOP.1.li.128
7|||||       |            |             |  extrapi_.LOOP.2.li.129
8|||||   3.3% |  11.066047 | 9,801,000.0 |   exi4_
```

```
||||||=====================================================
5||||   5.5% |  18.388414 |     1,000.0 | fluxi_(exclusive)
||||||=====================================================
4|||  28.4% |  93.988073 |     1,000.0 | fluxj_
|||||-----------------------------------------------------
5||||  18.3% |  60.687484 |          -- | fluxj_.LOOP.1.li.28
6||||       |            |             |  fluxj_.LOOP.2.li.29
7||||  18.3% |  60.687484 | 9,312,000.0 |   viscj_
5||||   5.9% |  19.520634 |     1,000.0 | fluxj_(exclusive)
5||||   4.2% |  13.779955 |     1,000.0 | extrapj_
|||||=====================================================
4|||   5.3% |  17.668610 |     1,000.0 | parallel_
|||||-----------------------------------------------------
5||||   4.0% |  13.109379 |          -- | parallel_.LOOP.1.li.16
6||||   4.0% |  13.109379 |     5,000.0 |  mpicx_
5||||   1.4% |   4.543585 |     4,000.0 | ghost_
|||||=====================================================
```

So we insert OpenMP 4.5 target data region

```
216.  +              !$omp target data map(tofrom:P,Q,SIY,VAV,SIX,UAV,SIZ,WAV,QAV,PAV,U,V,
217.                 !$omp&        W,T,Y,W,T11,T21,EKAV,EK,DS,DS1,T31,TAV,T12,T22,T32,T13,
218.                 !$omp&        T23,T33,HF,DTV,DQ)
220.  + 1----------<       DO WHILE ( NADV .LE. NEND .AND. TAU .LT. 35.0D+00 )
221.    1
222.    1                    IF ( MOD ( NADV, ITIME ) .EQ. 0 .OR. NADV .EQ. NSTART )
223.  + 1                 >                              CALL TMSTEP ( TMAX, RMACH )
224.    1
225.    1                    TIME = TIME + DT
226.    1
227.  + 1                    CALL ANALYSIS ( TAU, DELM )
228.    1
229.    1                    IF ( IAM .EQ. 0 .AND. MOD ( NADV, ITIME ) .EQ. 0 ) THEN
230.    1                       WRITE(6,1000) NADV, DT, TIME, TAU, DELM
231.    1                       WRITE(50,'(2(F10.5,1X))') TAU, DELM
232.    1                    ENDIF
233.    1              1000  FORMAT(' NADV = ', I6,
234.    1                   >        ' DT   = ', E10.4,
235.    1                   >        ' TIME = ', F8.5,
236.    1                   >        ' TAU  = ', F8.4,
237.    1                   >        ' DELM = ', F8.4)
238.    1
239.    1
240.    1                    IF ( IDYN .GT. 0 ) THEN
241.  + 1                       CALL EJECT ( 'LDKM, PLEASE RECOMPILE WITH -DLDKM!' )
242.    1                    ENDIF
243.    1
244.    1                    IADD = MOD ( NADV, 2 )
```

But then you have added a ton of work to do

- You now have two copies of the data that has been moved to the accelerator
  - Every place data is used that resides on the accelerator and on the host, the user is responsible for moving data back to host when computation requires an updated copy of the data
  - I know about unified memory; however, often times is will not do as good a job as the user and we do not have it right now
- Put as much computation on the accelerator as possible
- In this case halo exchanges require some treatment

Within the Data region

```
316.    1 2
317.    1 2                     IF ( ISTART .EQ. 1 .AND. INFLOW .EQ. 1 ) THEN
318.    1 2                        I = 0
319.    1 2          #ifdef GPU
320.    1 2          !$omp target teams distribute
321.    1 2          !$omp&           map(to:bound)
322.    1 2          !$omp&            map(from:q,t)
323.    1 2          !$omp&            private(l)
324.    1 2          #else
325.    D            !$OMP PARALLEL DO PRIVATE(L)
326.    1 2          #endif
327.    1 2 D------<                DO K = 1, KCMAX
328.    1 2 D 4----<                   DO J = 1, JCMAX
329.    1 2 D 4                           Q(I,J,K,1,M) = BOUND(J,K,1)
330.    1 2 D 4                           Q(I,J,K,2,M) = BOUND(J,K,1) * BOUND(J,K,2)
331.    1 2 D 4                           Q(I,J,K,3,M) = BOUND(J,K,1) * BOUND(J,K,3)
332.    1 2 D 4                           Q(I,J,K,4,M) = BOUND(J,K,1) * BOUND(J,K,4)
333.    1 2 D 4                           T(I,J,K)     = BOUND(J,K,5)
334.    1 2 D 4                           IF ( ICHEM .GT. 0 ) THEN
335.    1 2 D 4 5--<                         DO NS = 1, NSPECI
336.    1 2 D 4 5                               L = 7 + NS
337.    1 2 D 4 5                               Q(I,J,K,L,M) = BOUND(J,K,1) * BOUND(J,K,L)
338.    1 2 D 4 5-->                         ENDDO
339.    1 2 D 4                           ENDIF
340.    1 2 D 4                           IF ( ISGSK .EQ. 1 ) THEN
341.    1 2 D 4                              Q(I,J,K,7,M) = BOUND(J,K,1) * BOUND(J,K,7)
342.    1 2 D 4                           ENDIF
343.    1 2 D 4---->                   ENDDO
344.    1 2 D------>                ENDDO
345.    1 2                      ENDIF
346.    1 2
```

Any computation within the data block must either be put on the accelerator or the data read has to be updated on the host.

This loop we move to the Accelerator

Halo exchange within the Data region

```
346.     1 2
347.     1 2           ! Exchange boundary information
348.     1 2
349.     1 2           #ifdef GPU
350.     1 2           !$omp target update from(q,hf)
351.     1 2           #endif
352.     1 2                     IF ( NSCHEME .EQ. 2 ) THEN
353.     1 2           !              CALL PARALLEL(2000 + N, 1)
354.   + 1 2                       CALL PARALLEL(2000 + N)
355.     1 2                     ELSE
356.     1 2           !              CALL PARALLEL(2000 + N, 2)
357.   + 1 2                       CALL PARALLEL(2000 + N)
358.     1 2                     ENDIF
359.     1 2           #ifdef GPU
360.     1 2           !$omp target update to(q,u,v,w)
361.     1 2           #endif
```

Perftools loaded – pat_build –u –g mpi <exe>

```
Table 1:  Profile by Function Group and Function
  Time% |      Time |     Imb. |  Imb. |    Calls | Group
        |           |     Time | Time% |          |  Function
        |           |          |       |          |   PE=HIDE
        |           |          |       |          |    Thread=HIDE
 100.0% | 48.808788 |       -- |    -- | 20,933.0 | Total
|-------------------------------------------------------------------------
|  85.5% | 41.721597 |       -- |    -- | 11,003.0 | OACC
||------------------------------------------------------------------------
||  27.7% | 13.512695 | 0.040378 |  0.6% |    200.0 | les3d_.ACC_COPY@li.355
||  20.3% |  9.926572 | 0.001944 |  0.0% |    200.0 | les3d_.ACC_COPY@li.345
||   7.2% |  3.491549 | 0.003118 |  0.2% |    200.0 | fluxi_.ACC_SYNC_WAIT@li.92
||   6.9% |  3.380262 | 0.003574 |  0.2% |    200.0 | fluxj_.ACC_SYNC_WAIT@li.108
||   6.8% |  3.340281 | 0.017049 |  1.0% |    200.0 | fluxk_.ACC_SYNC_WAIT@li.105
||   4.3% |  2.099495 | 0.006041 |  0.6% |    200.0 | fluxj_.ACC_COPY@li.32
||   4.3% |  2.089182 | 0.006879 |  0.7% |    200.0 | fluxk_.ACC_COPY@li.31
||   1.7% |  0.835638 | 0.000934 |  0.2% |    200.0 | extrapj_.ACC_SYNC_WAIT@li.240
||   1.5% |  0.748967 | 0.007268 |  1.9% |    200.0 | extrapk_.ACC_SYNC_WAIT@li.238
||   1.5% |  0.718938 | 0.010417 |  2.9% |      1.0 | les3d_.ACC_COPY@li.216
||   1.2% |  0.567096 | 0.008023 |  2.8% |      1.0 | les3d_.ACC_COPY@li.415
||========================================================================
|  12.2% |  5.978121 |       -- |    -- |  6,647.0 | USER
||------------------------------------------------------------------------
||   2.4% |  1.194200 | 0.012415 |  2.1% |      1.0 | grid_
||   2.2% |  1.082590 | 0.000672 |  0.1% |     21.0 | tmstep_
||   1.6% |  0.765046 | 0.011128 |  2.9% |      1.0 | setiv_
||   1.2% |  0.595023 | 0.000736 |  0.2% |  1,010.0 | mpicx_
||   1.0% |  0.500016 | 0.059047 | 21.1% |    202.0 | parallel_
||   1.0% |  0.471803 | 0.287358 | 75.7% |      1.0 | flowio_
||========================================================================
|   1.7% |  0.845256 |       -- |    -- |  3,261.0 | MPI
||------------------------------------------------------------------------
||   1.2% |  0.563471 | 0.175119 | 47.4% |      2.0 | MPI_REDUCE
|=========================================================================
```

ACC_COPY is data movement to and from the accelerator

ACC_SYNC_WAIT is waiting for a kernel to finish

47.7% of the time is in updates for halo exchange

```
343.     1 2           ! Exchange boundary information
344.     1 2           #ifdef OMP_TARGET
345.     1 2           !$omp target update from(q,hf)
346.     1 2           #endif
347.     1 2                     IF ( NSCHEME .EQ. 2 ) THEN
348.     1 2           !             CALL PARALLEL(2000 + N, 1)
349.   + 1 2                        CALL PARALLEL(2000 + N)
350.     1 2                     ELSE
351.     1 2           !             CALL PARALLEL(2000 + N, 2)
352.   + 1 2                        CALL PARALLEL(2000 + N)
353.     1 2                     ENDIF
354.     1 2           #ifdef OMP_TARGET
355.     1 2           !$omp target update to(q,hf)
356.     1 2           #endif
```

So what is going on within PARALLEL

From our Call Tree --- pat_report –Oct –T.

```
3||   3.3% |  1.589344 |   200.0 | parallel_
||||-------------------------------------------------------------
4|||   1.5% |  0.728695 | 1,000.0 | mpicx_
|||||-----------------------------------------------------------
5||||   1.2% |  0.589571 | 1,000.0 | mpicx_(exclusive)
5||||   0.3% |  0.135674 | 1,000.0 | MPI_SEND
5||||   0.0% |  0.002217 | 1,000.0 | MPI_IRECV
5||||   0.0% |  0.001233 | 1,000.0 | MPI_WAIT
```

Down into PARALLEL and MPICX

```
      IF(NCY .LT. NPY .OR. (NCY .EQ. NPY .AND. JPERIODIC)) THEN
         JCNT = 0
         DO K = 1-NLEVELS, KCMAX+NLEVELS
            DO J = JCMAX-NLEVELS+1, JCMAX
               DO I = 1-NLEVELS, ICMAX+NLEVELS
                  JCNT = JCNT + 1
                  S_N(JCNT) = VAR(I,J,K)
               END DO
            ENDDO
         END DO
         CALL MPI_SEND(S_N,
     >                 NLEVELS * NSIZE1 * NSIZE3,
     >                 MPI_DOUBLE_PRECISION,
     >                 NORTH,
     >                 MSGFLAG + MSGTSN,
     >                 MPI_COMM_WORLD,
     >                 IERR )
         IF ( IERR .NE. 0 ) CALL EJECT ('MPI_SEND FAILED: S_N')
       END IF
```

Need to pack buffers on the accelerator

Packing on the GPU and sending the halo

```
738.                        #ifdef OMP_TARGET
739.     G----------------< !$omp target teams distribute
740.     G                  #endif
741.     G g-------------<          DO K = 1, NLEVELS
742.   + G g 3-----------<             DO J = 1-NLEVELS, JCMAX+NLEVELS
743.     G g 3              #ifdef OMP_TARGET
744.   + G g 3              !$omp parallel do private(kcnt) private(i)
745.     G g 3              #endif
746.     G g 3 g---------<                DO I = 1-NLEVELS, ICMAX+NLEVELS
747.     G g 3 g                             KCNT = (k-1)*(JCMAX+2*NLEVELS)*(ICMAX+2*NLEVELS)
748.     G g 3 g                  >                    +(j-1+NLEVELS)*(ICMAX+2*NLEVELS)+i
749.     G g 3 g                             S_I(KCNT) = VAR(I,J,K)
750.     G g 3 g--------->                END DO
751.     G g 3----------->                ENDDO
752.     G g------------->             END DO
753.     G                  #ifdef OMP_TARGET
754.     G----------------> !$omp end target teams distribute
755.                        !$omp target update from(S_I)
756.                        #endif
757.   +                          CALL MPI_SEND(S_I,
758.                            >                NLEVELS * NSIZE1 * NSIZE2,
759.                            >                MPI_DOUBLE_PRECISION,
760.                            >                IN,
761.                            >                MSGFLAG + MSGTSI,
762.                            >                MPI_COMM_WORLD,
763.                            >                IERR )
764.   +                            IF ( IERR .NE. 0 ) CALL EJECT ('MPI_SEND FAILED: S_I')
765.                             END IF
```

We do have to update the host prior to the MPI_SEND

Posting the receive on the Host

We don’t have to do anything prior to or after the IRECV

```
706.
707.                          IF(NCZ .GT.   1 .OR. (NCZ .EQ.   1 .AND. KPERIODIC)) THEN
708.  +                          CALL MPI_IRECV(R_I,
709.                         >                  NLEVELS * NSIZE1 * NSIZE2,
710.                         >                  MPI_DOUBLE_PRECISION,
711.                         >                  IN,
712.                         >                  MSGFLAG + MSGTRI,
713.                         >                  MPI_COMM_WORLD,
714.                         >                  MSGIDRI,
715.                         >                  IERR )
```

Receiving and unpacking on the GPU

```
785.  +                          CALL MPI_WAIT( MSGIDRI, MPI_STATUS, IERR)
786.
787.                       #ifdef OMP_TARGET
788.                       !$omp target update to(R_I)
789.    G--------------<   !$omp target teams distribute
790.    G                  #endif
791.    G g------------<           DO K = 1-NLEVELS, 0
792.  + G g 3----------<             DO J = 1-NLEVELS, JCMAX+NLEVELS
793.    G g 3              #ifdef OMP_TARGET
794.  + G g 3              !$omp parallel do private(kcnt) private(i)
795.    G g 3              #endif
796.    G g 3 g--------<                 DO I = 1-NLEVELS, ICMAX+NLEVELS
797.    G g 3 g                           KCNT = (k-1+NLEVELS)*(JCMAX+2*NLEVELS)*(ICMAX+2*NLEVELS)
798.    G g 3 g                >                              +(j-1+NLEVELS)*(ICMAX+2*NLEVELS)+i
799.    G g 3 g                             VAR(I,J,K) = R_I(KCNT)
800.    G g 3 g-------->                 END DO
801.    G g 3---------->               END DO
802.    G g------------>           END DO
803.    G                  #ifdef OMP_TARGET
804.    G--------------> !$omp end target teams distribute
805.                       #endif
```

After the MPI_WAIT we need to update the GPU with the communicated array

Two times faster

```
Table 1:  Profile by Function Group and Function
  Time% |      Time |     Imb. |  Imb. |    Calls | Group
        |           |     Time | Time% |          |  Function
        |           |          |       |          |   PE=HIDE
        |           |          |       |          |    Thread=HIDE
 100.0% | 25.786538 |       -- |    -- | 35,083.0 | Total
|-------------------------------------------------------------------------------
|  73.3% | 18.898477 |       -- |    -- | 21,113.0 | OACC
||------------------------------------------------------------------------------
||  13.5% |  3.491808 | 0.000623 |  0.0% |    200.0 | fluxi_.ACC_SYNC_WAIT@li.92
||  13.1% |  3.378351 | 0.003587 |  0.2% |    200.0 | fluxj_.ACC_SYNC_WAIT@li.108
||  12.9% |  3.338869 | 0.016283 |  1.0% |    200.0 | fluxk_.ACC_SYNC_WAIT@li.105
||   8.2% |  2.124079 | 0.004317 |  0.4% |    200.0 | fluxj_.ACC_COPY@li.32
||   8.2% |  2.104135 | 0.005343 |  0.5% |    200.0 | fluxk_.ACC_COPY@li.31
||   3.2% |  0.835128 | 0.000431 |  0.1% |    200.0 | extrapj_.ACC_SYNC_WAIT@li.240
||   2.9% |  0.746907 | 0.005521 |  1.5% |    200.0 | extrapk_.ACC_SYNC_WAIT@li.238
||   2.2% |  0.564556 | 0.006842 |  2.4% |      1.0 | les3d_.ACC_COPY@li.409
||   1.7% |  0.439470 | 0.000434 |  0.2% |    200.0 | update_.ACC_SYNC_WAIT@li.45
||   1.6% |  0.420604 | 0.007835 |  3.7% |      1.0 | les3d_.ACC_COPY@li.216
||   1.4% |  0.360217 | 0.000020 |  0.0% |  1,010.0 | mpicx_.ACC_COPY@li.322
||   1.2% |  0.315081 | 0.000720 |  0.5% |    200.0 | extrapi_.ACC_SYNC_WAIT@li.172
||==============================================================================
|  22.4% |  5.785730 |       -- |    -- | 10,687.0 | USER
||------------------------------------------------------------------------------
||   4.8% |  1.248279 | 0.000063 |  0.0% |      1.0 | grid_
||   4.2% |  1.081927 | 0.000590 |  0.1% |     21.0 | tmstep_
||   3.0% |  0.785531 | 0.001003 |  0.3% |      1.0 | setiv_
||   2.7% |  0.703152 | 0.067870 | 17.6% |    202.0 | parallel_
||   1.9% |  0.482106 | 0.297350 | 76.3% |      1.0 | flowio_
||   1.8% |  0.460976 | 0.066984 | 25.4% |    303.0 | ghost_
||   1.6% |  0.415937 | 0.000363 |  0.2% |    100.0 | analysis_
||==============================================================================
|   3.3% |  0.838605 |       -- |    -- |  3,261.0 | MPI
||------------------------------------------------------------------------------
||   2.2% |  0.559752 | 0.178772 | 48.4% |      2.0 | MPI_REDUCE
||==============================================================================
|   1.0% |  0.263651 | 0.000053 |  0.0% |     13.0 | ETC
||------------------------------------------------------------------------------
||   1.0% |  0.263651 | 0.000053 |  0.0% |     13.0 | _END
|==============================================================================
```

Now most of the time is spent in kernel execution. Next step would be to investigate the performance of the top kernels to see if they can be improved

So we use CRAY_ACC_DEBUG=2 and rerun

```
ACC: Start transfer 5 items from src/fluxj.f:32
ACC:       allocate, copy to acc 'sjx' (31990464 bytes)
ACC:       allocate, copy to acc 'sjx' (31990464 bytes)
ACC:       allocate, copy to acc 'sjy' (31990464 bytes)
ACC:       allocate, copy to acc 'sjy' (31990464 bytes)
ACC:       allocate, copy to acc 'sjz' (31990464 bytes)
ACC:       allocate, copy to acc 'sjz' (31990464 bytes)
ACC:       allocate <internal> (227672064 bytes)
ACC:       allocate <internal> (147456 bytes)
ACC: End transfer (to acc 95971392 bytes, to host 0 bytes)
```

Didn’t add those arrays into the data map

```
126.                    #ifdef OMP_TARGET
127.                    !$omp declare target (exj2,exj4)
128.  +                 !$omp target data map(tofrom:SIY,VAV,SIX,UAV,SIZ,WAV,QAV,PAV,U,V,W,T,
129.                    !$omp&         Y,W,T11,T21,Q,HF,DS,DS1,T31,TAV,T12,T22,T32,T13,T23,
130.                    !$omp&         T33,HF,DTV,DQ)
131.                    #endif
```

Because they are used down the call chain in VICJ

```
511.    V
512.    V                   SX = SJX(I,J,K)
513.    V                   SY = SJY(I,J,K)
514.    V                   SZ = SJZ(I,J,K)
515.    V
516.    V                   FSJ(I,J,2) = FSJ(I,J,2) + TXX * SX + TXY * SY + TXZ * SZ
517.    V                   FSJ(I,J,3) = FSJ(I,J,3) + TYX * SX + TYY * SY + TYZ * SZ
518.    V                   FSJ(I,J,4) = FSJ(I,J,4) + TZX * SX + TZY * SY + TZZ * SZ
519.    V                   FSJ(I,J,5) = FSJ(I,J,5) +
520.    V                 >               (TXX*UAVE + TXY*VAVE + TXZ*WAVE + QX) * SX +
521.    V                 >               (TYX*UAVE + TYY*VAVE + TYZ*WAVE + QY) * SY +
522.    V                 >               (TZX*UAVE + TZY*VAVE + TZZ*WAVE + QZ) * SZ
523.    V---------------->              ENDDO
```

There - those went away

```
Table 1:  Profile by Function Group and Function
  Time% |       Time |     Imb. |  Imb. |    Calls | Group
        |            |     Time | Time% |          |  Function
        |            |          |       |          |   PE=HIDE
        |            |          |       |          |    Thread=HIDE
 100.0% | 21.612385 |       -- |    -- | 35,083.0 | Total
|-------------------------------------------------------------------------
|  68.0% | 14.693815 |       -- |    -- | 21,113.0 | OACC
||------------------------------------------------------------------------
||  16.2% |  3.494655 | 0.000677 |  0.0% |    200.0 | fluxi_.ACC_SYNC_WAIT@li.92
||  15.6% |  3.380081 | 0.001161 |  0.1% |    200.0 | fluxj_.ACC_SYNC_WAIT@li.108
||  15.4% |  3.332006 | 0.024111 |  1.4% |    200.0 | fluxk_.ACC_SYNC_WAIT@li.105
||   3.9% |  0.833878 | 0.000218 |  0.1% |    200.0 | extrapj_.ACC_SYNC_WAIT@li.240
||   3.4% |  0.743858 | 0.001672 |  0.4% |    200.0 | extrapk_.ACC_SYNC_WAIT@li.238
||   2.7% |  0.580301 | 0.006628 |  2.3% |      1.0 | les3d_.ACC_COPY@li.409
||   2.0% |  0.439370 | 0.000235 |  0.1% |    200.0 | update_.ACC_SYNC_WAIT@li.45
||   2.0% |  0.438281 | 0.008003 |  3.6% |      1.0 | les3d_.ACC_COPY@li.216
||   1.7% |  0.360211 | 0.000058 |  0.0% |  1,010.0 | mpicx_.ACC_COPY@li.322
||   1.5% |  0.315138 | 0.000423 |  0.3% |    200.0 | extrapi_.ACC_SYNC_WAIT@li.172
||=========================================================================
|  26.7% |  5.778250 |       -- |    -- | 10,687.0 | USER
||------------------------------------------------------------------------
||   5.8% |  1.253559 | 0.018548 |  2.9% |      1.0 | grid_
||   5.0% |  1.085428 | 0.001729 |  0.3% |     21.0 | tmstep_
||   3.7% |  0.804346 | 0.024287 |  5.9% |      1.0 | setiv_
||   3.0% |  0.640157 | 0.070052 | 19.7% |    202.0 | parallel_
||   2.3% |  0.506806 | 0.070618 | 24.5% |    303.0 | ghost_
||   2.2% |  0.475394 | 0.290094 | 75.8% |      1.0 | flowio_
||   1.9% |  0.416498 | 0.001701 |  0.8% |    100.0 | analysis_
||=========================================================================
|   4.1% |  0.876445 |       -- |    -- |  3,261.0 | MPI
||------------------------------------------------------------------------
||   2.6% |  0.556783 | 0.176866 | 48.2% |      2.0 | MPI_REDUCE
||=========================================================================
|   1.2% |  0.263798 | 0.000062 |  0.0% |     13.0 | ETC
||------------------------------------------------------------------------
||   1.2% |  0.263798 | 0.000062 |  0.0% |     13.0 | _END
|=========================================================================
```

These copies in les3d do the initialization of the GPU and bring the results back at the end

Still can do better – use use_device_ptr and setenv MPICH_RDMA_ENABLED_CUDA 1

```
#ifdef OMP_TARGET
!$omp target enter data map(to:VAR)
!$omp&          map(alloc:R_N,R_W,R_S,R_E,R_I,R_O,S_N,S_W,S_S,
!$OMP&                    S_E,S_I,S_O)
!$omp target data
!$omp&  use_device_ptr(R_N,R_W,R_S,R_E,R_I,R_O,S_N,S_W,S_S,
!$OMP&                    S_E,S_I,S_O)
#endif
```

```
      CALL MPI_IRECV(R_W,
     >               NLEVELS * NSIZE2 * NSIZE3,
     >               MPI_DOUBLE_PRECISION,
     >               WEST,
     >               MSGFLAG + MSGTRW,
     >               MPI_COMM_WORLD,
     >               MSGIDRW,
     >               IERR)
```

Much easier to code –

Do not need target update clauses

Still can do better

```
#ifdef OMP_TARGET
!$omp target enter data map(to:VAR)
!$omp&          map(alloc:R_N,R_W,R_S,R_E,R_I,R_O,S_N,S_W,S_S,
!$OMP&                    S_E,S_I,S_O)
!$omp target data
!$omp&  use_device_ptr(R_N,R_W,R_S,R_E,R_I,R_O,S_N,S_W,S_S,
!$OMP&                    S_E,S_I,S_O)
#endif
```

```
      CALL MPI_SEND(S_E,
     >              NLEVELS * NSIZE2 * NSIZE3,
     >              MPI_DOUBLE_PRECISION,
     >              EAST,
     >              MSGFLAG + MSGTSE,
     >              MPI_COMM_WORLD,
     >              IERR )
```

Much easier to code

Timings of various version of Leslie3D on four nodes

	original 64 MPI Tasks	Initial Port 4 MPI Tasks>	Raising Data Region x 4	Halo Packing on GPU x 4
fluxi	2.7	63.1	2.05	2.05
fluxj	2.4	57.3	2.12	2.12
fluxk	2.6	57.3	2.76	2.07
update	0.72	9.4	0.24	0.24
leslie3d	2.52	2.6	14.81	3.49
parallel	0.94	1.2	1.04	1.04
tmstep	0.62	1.1	0.78	1
Total	12.5	192	23.8	12.01

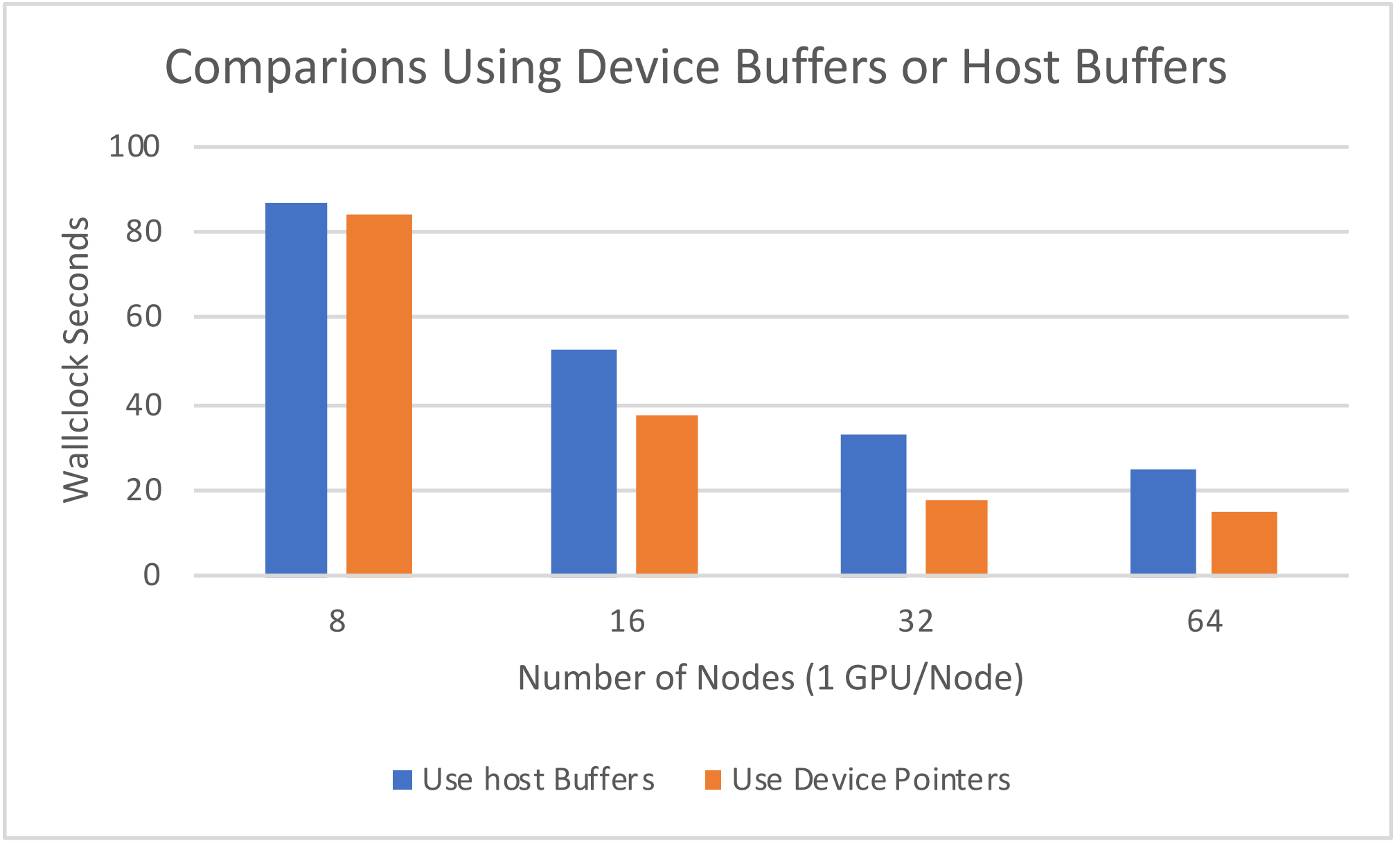
Timings of various version of Leslie3D on four nodes

	original 64 MPI Tasks	Initial Port 4 MPI Tasks>	Raising Data Region x 4	Halo Packing on GPU x 4
fluxi	2.7	63.1	2.05	2.05
fluxj	2.4	57.3	2.12	2.12
fluxk	2.6	57.3	2.76	2.07
update	0.72	9.4	0.24	0.24
leslie3d	2.52	2.6	14.81	3.49
parallel	0.94	1.2	1.04	1.04
tmstep	0.62	1.1	0.78	1
Total	12.5	192	23.8	12.01

Timings of various version of Leslie3D on four nodes

	original 64 MPI Tasks	Initial Port 4 MPI Tasks>	Raising Data Region x 4	Halo Packing on GPU x 4
fluxi	2.7	63.1	2.05	2.05
fluxj	2.4	57.3	2.12	2.12
fluxk	2.6	57.3	2.76	2.07
update	0.72	9.4	0.24	0.24
leslie3d	2.52	2.6	14.81	3.49
parallel	0.94	1.2	1.04	1.04
tmstep	0.62	1.1	0.78	1
Total	12.5	192	23.8	12.01

All Packing done on the GPU



CRAY_ACC_DEBUG Environment Variable

- Three levels of verbosity
  - 1 High Level overview of kernels executed and data transferred
  - 2 Breaks down data transfer by each variable
  - 3 The whole Kitchen sink

setenv CRAY_ACC_DEBUG 1

```
ACC: Transfer 29 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:21
ACC: Transfer 15 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:114
ACC: Transfer 1 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:141
ACC: Transfer 29 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:21
ACC: Transfer 15 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:114
ACC: Transfer 1 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:141
ACC: Execute kernel extrapi_$ck_L141_7 async(auto) from src/fluxi.f:141
ACC: Execute kernel extrapi_$ck_L141_7 async(auto) from src/fluxi.f:141
ACC: Wait async(auto) from src/fluxi.f:172
ACC: Transfer 1 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:172
ACC: Execute kernel extrapi_$ck_L173_9 async(auto) from src/fluxi.f:173
ACC: Wait async(auto) from src/fluxi.f:172
```

setenv CRAY_ACC_DEBUG 2

```
ACC: Start transfer 29 items from src/fluxi.f:21
ACC:         present 'dq' (255923712 bytes)
ACC:         present 'ds' (95971392 bytes)
ACC:         present 'ds1' (95971392 bytes)
ACC:         present 'dtv' (31990464 bytes)
ACC:         present 'hf' (127961856 bytes)
ACC:         present 'pav' (31990464 bytes)
ACC:         present 'q' (511847424 bytes)
ACC:         present 'qav' (255923712 bytes)
ACC:         present 'six' (31990464 bytes)
ACC:         present 'siy' (31990464 bytes)
ACC:         present 'siz' (31990464 bytes)
ACC:         present 't' (31990464 bytes)
```

setenv CRAY_ACC_DEBUG 3

```
ACC: Start transfer 29 items from src/fluxi.f:21
ACC:   flags: RETURN_ACC_TIME
ACC:
ACC:   Trans 1
ACC:       Simple transfer of 'dq' (144 bytes)
ACC:            host ptr 78b538
ACC:            acc  ptr 0
ACC:            flags: DOPE_VECTOR DV_ONLY_DATA ALLOCATE COPY_HOST_TO_ACC ACQ_PRESENT REG_PRESENT
ACC:            Transferring dope vector
ACC:              dim:1 lowbound:-2 extent:198 stride_mult:1
ACC:              dim:2 lowbound:-2 extent:198 stride_mult:198
ACC:              dim:3 lowbound:-2 extent:102 stride_mult:39204
ACC:              dim:4 lowbound:1 extent:8 stride_mult:3998808
ACC:              DV size=255923712 (scale:8, elem_size:8)
ACC:              total mem size=255923712 (dv:0 obj:255923712)
ACC:            host region 4dac8780 to 5ced9d80 found in present table index 0 (ref count 2)
ACC:            memory found in present table (2aaaf2000000, base 2aaaf2000000)
ACC:            new acc ptr 2aaaf2000000
```

Now we need to work on the kernels

- We are barely beating the two socket node even with all the work we did.
- The kernels are not optimal, they need some work
- This is left as an exercise for the student

So what have we learned

- Perftools is excellent for identifying issues in existing applications for improving threading, vectorization and scalar optimization
- Reveal can help with the difficult job of scoping variables in potential parallelizable loops
  - More difficult if not impossible with C++
- Moving to the GPU is difficult; however, it can be done in steps that are more manageable
  - Perftools identifies the bottlenecks in the GPU application very quickly
- GPU direct is best way to do the message passing – in this case it only matters at scale


Questions?
CRAY