Cray Performance Measurement and Analysis Tools

Heidi Poxon
Manager & Technical Lead, Performance Tools
Cray Inc.

7-9 November, 2011

CSCS
Swiss National Supercomputing Centre

More on Cray Performance Tools

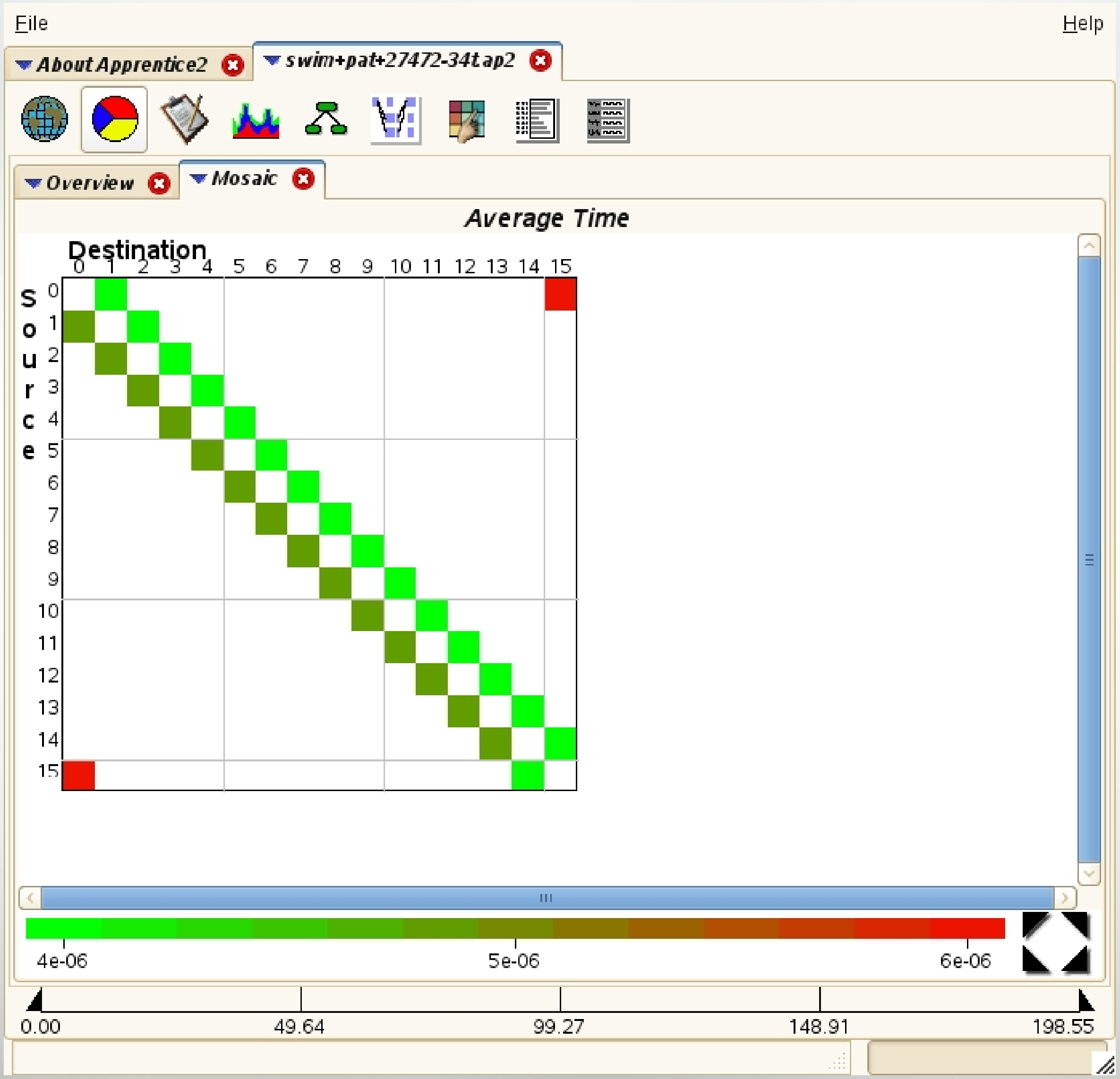
- Trace analysis and visualization
- Where to get help
- What’s Coming Next

Trace Analysis and Visualization

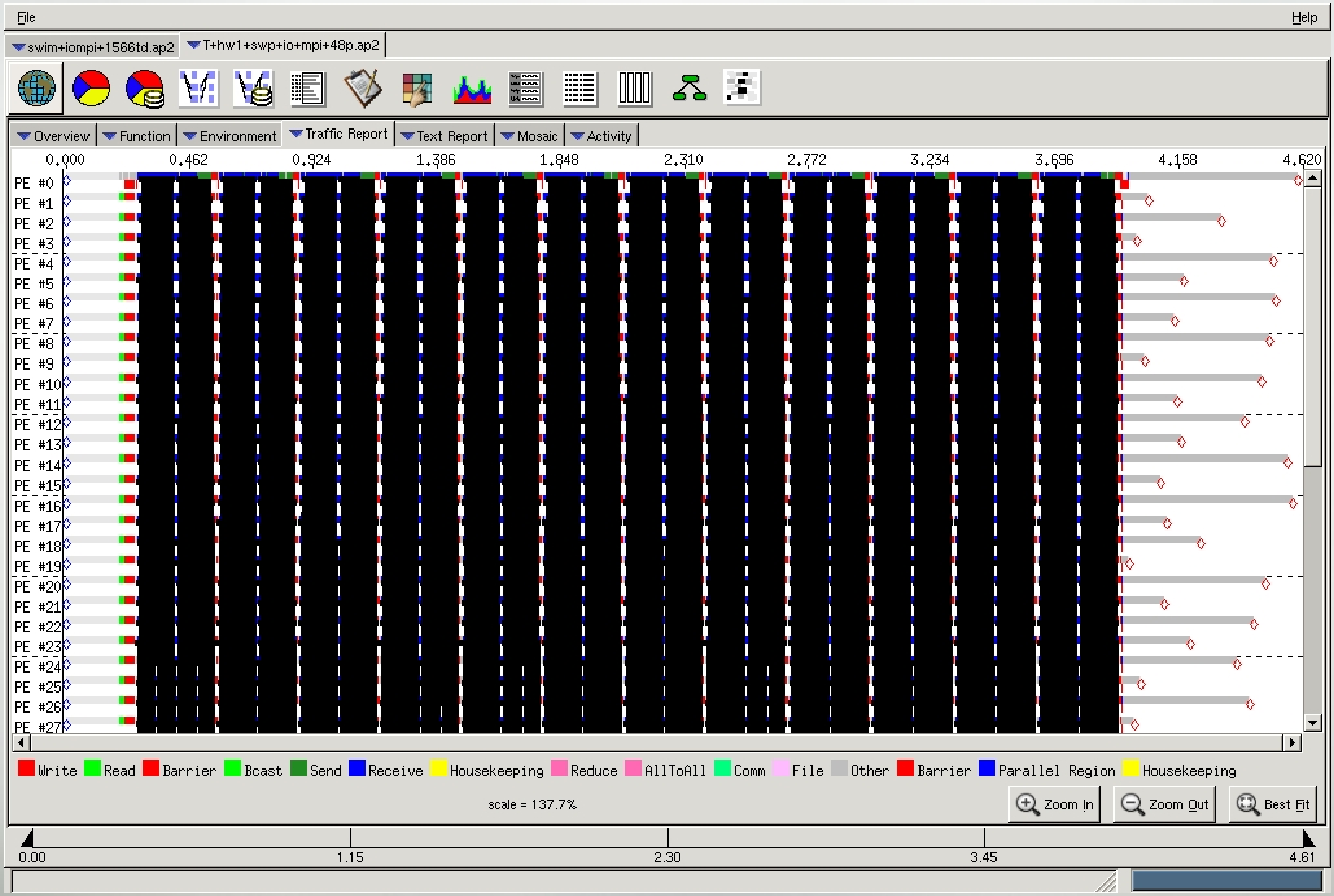
Tracing

- Only true function calls can be traced
  - Functions that are inlined by the compiler or that have local scope in a compilation unit cannot be traced
- Enabled with pat_build –g, -u, -T or –w options
- Full trace (sequence of events) enabled by setting PAT_RT_SUMMARY=0
  - **Warning:** trace files are not scalable
    - Tend to generate huge performance files
    - Not really recommended

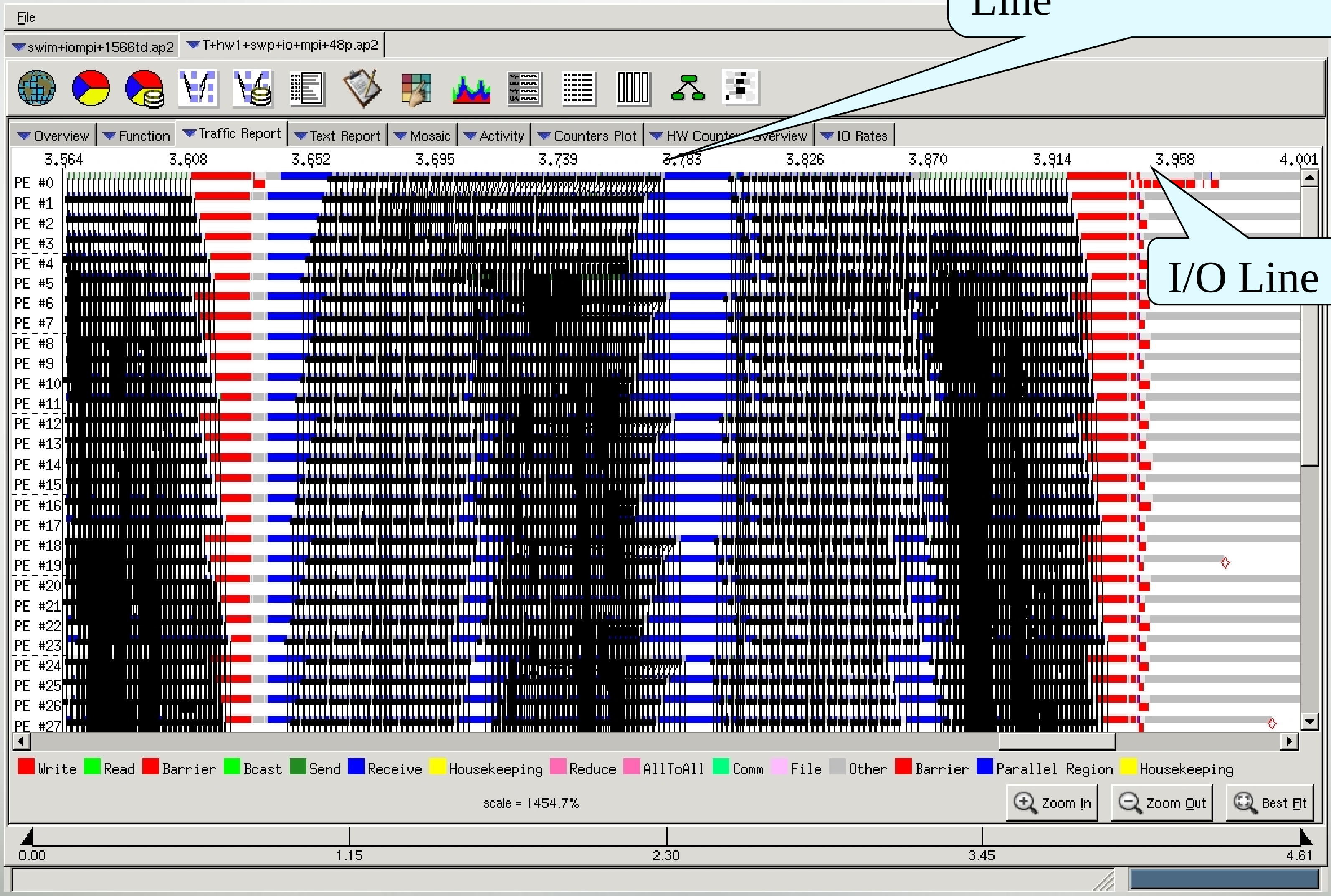
Mosaic View – Shows Communication Pattern



Time Line View (Sweep3D)



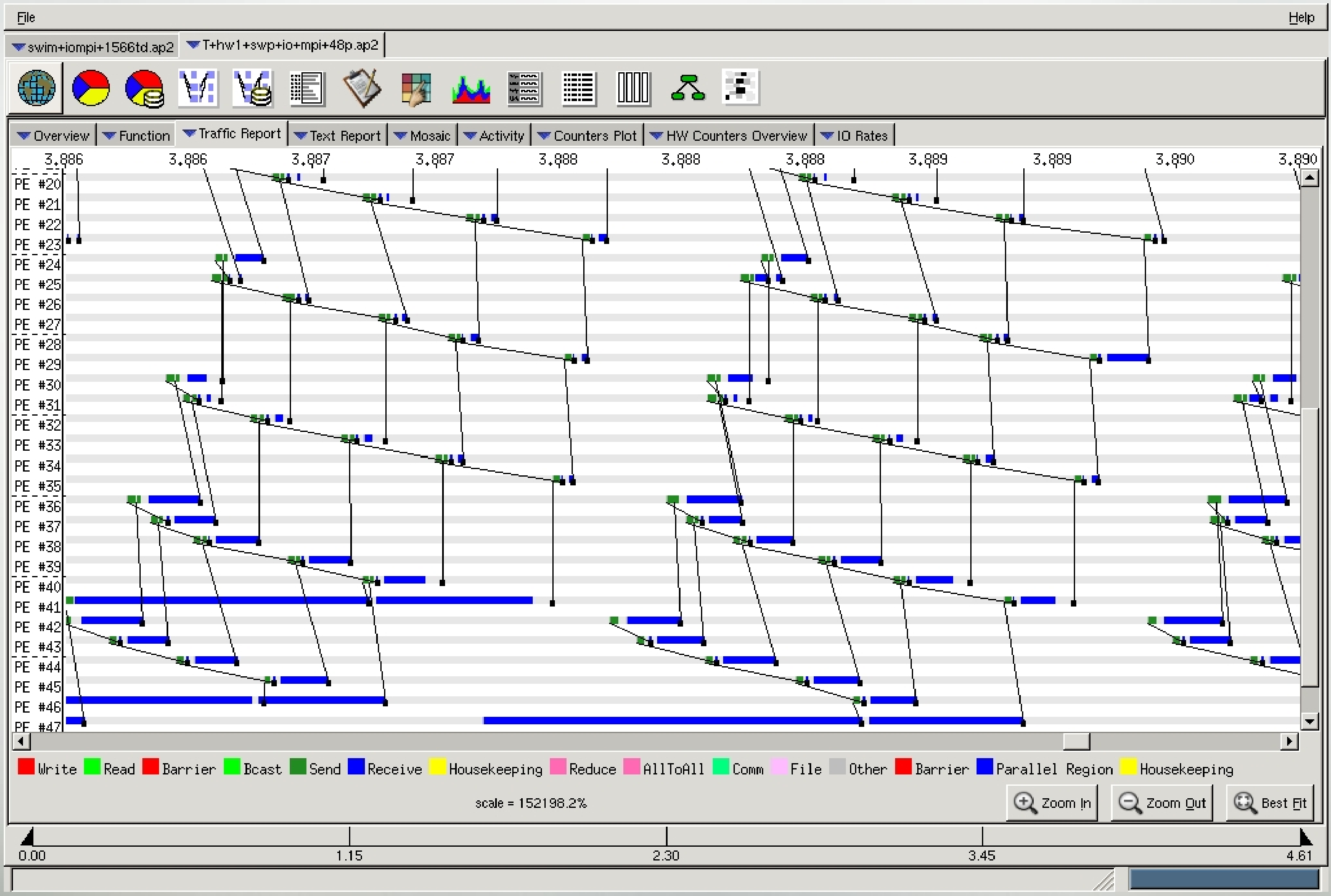
Time Line View (Zoom)



User Functions, MPI Line

I/O Line

Time Line View (Fine Grain Zoom)



Suggestions for Controlling Large Traces

Several environment variables are available to limit trace files to a more reasonable size:

- PAT_RT_CALLSTACK
  - Limit the depth to trace the call stack
- PAT_RT_HWPC
  - Avoid collecting hardware counters (unset)
- PAT_RT_RECORD_PE
  - Collect trace for a subset of the PEs
- PAT_RT_TRACE_FUNCTION_ARGS
  - Limit the number of function arguments to be traced
- PAT_RT_TRACE_FUNCTION_LIMITS
  - Avoid tracing indicated functions
- PAT_RT_TRACE_FUNCTION_MAX
  - Limit the maximum number of traces generated for all functions for a single process

Suggestions for Controlling Large Traces (2)

- PAT_RT_TRACE_THRESHOLD_PCT
  - Specifies a % of time threshold to enforce when executing in full trace mode
- PAT_RT_TRACE_THRESHOLD_TIME
  - Specifies a time threshold to enforce when executing in full trace mode
- Set PAT_RT_EXPFILE_MAX to the number of ranks (or any larger number)
  - Data for only 1 MPI rank stored in each .xf file
- Use pat_region API to start and stop tracing within a program

Controlling large traces - Additional API Functions

- int PAT_state (int state)
  - State can have one of the following:
    - PAT_STATE_ON
    - PAT_STATE_OFF
    - PAT_STATE_QUERY
- int PAT_record (int state)
  - Controls the state for all threads on the executing PE. As a rule, use PAT_record() unless there is a need for different behaviors for sampling and tracing
    - int PAT_sampling_state (int state)
    - int PAT_tracing_state (int state)
- int PAT_trace_function (const void *addr, int state)
  - Activates or deactivates the tracing of the instrumented function
- int PAT_flush_buffer (void)

Trace On / Trace Off Example

```
include "pat_apif.h“
!  Turn data recording off at the beginning of execution.
call PAT_record( PAT_STATE_OFF, istat )
...
!  Turn data recording on for two regions of interest.
call PAT_record( PAT_STATE_ON, istat )
...
call PAT_region_begin( 1, "step 1", istat )
...
call PAT_region_end( 1, istat )
...
call PAT_region_begin( 2, "step 2", istat )
...
call PAT_region_end( 2, istat )
...
!  Turn data recording off again.
call PAT_record( PAT_STATE_OFF, istat )
...
```

Where to Find Help on the Tools

Accessing Software Versions

- Software package information
  - Use **avail**, **list** or **help** parameters to module command
  - '**module help perftools**' shows release notes
- Version (same for pat_build, pat_report, pat_help)

```
% pat_build –V
CrayPat/X:  Version 5.2.3 Revision 8155  09/13/11 08:47:57
```

- Cray Apprentice^2 version
  - Displayed in top menu bar when running GUI

Release Notes

```
% module help perftools
----------- Module Specific Help for 'perftools/5.2.3' ---------
===================================================================
Perftools 5.2.3
==============
Release Date: September 15, 2011

Differences between CrayPat 5.2.2 release and 5.2.3 release
------------------------------------------------------------
 General
  * PAPI library supports counters in NVIDIA GPUs
  * PAPI library available as dynamically shared object
  * All installed PerfTools executable binary files are dynamically linked
. . .

Purpose
-------
. . .

Bugs Fixed
----------
. . .
Known Problem(s)
----------------
. . .
Product and OS Dependencies:
----------------------------
 . . .
Documentation:
  --------------
  See the following documents at http://docs.cray.com/
  Cray Performance Analysis Tools Release Overview and
      Installation Guide S-2474-52
  Using Cray Performance Analysis Tools S-2376-52
```

Online information

- User guide
  - http://docs.cray.com
- Man pages
- To see list of reports that can be generated

```
% pat_report –O –h
```

- Notes sections in text performance reports provide information and suggest further options

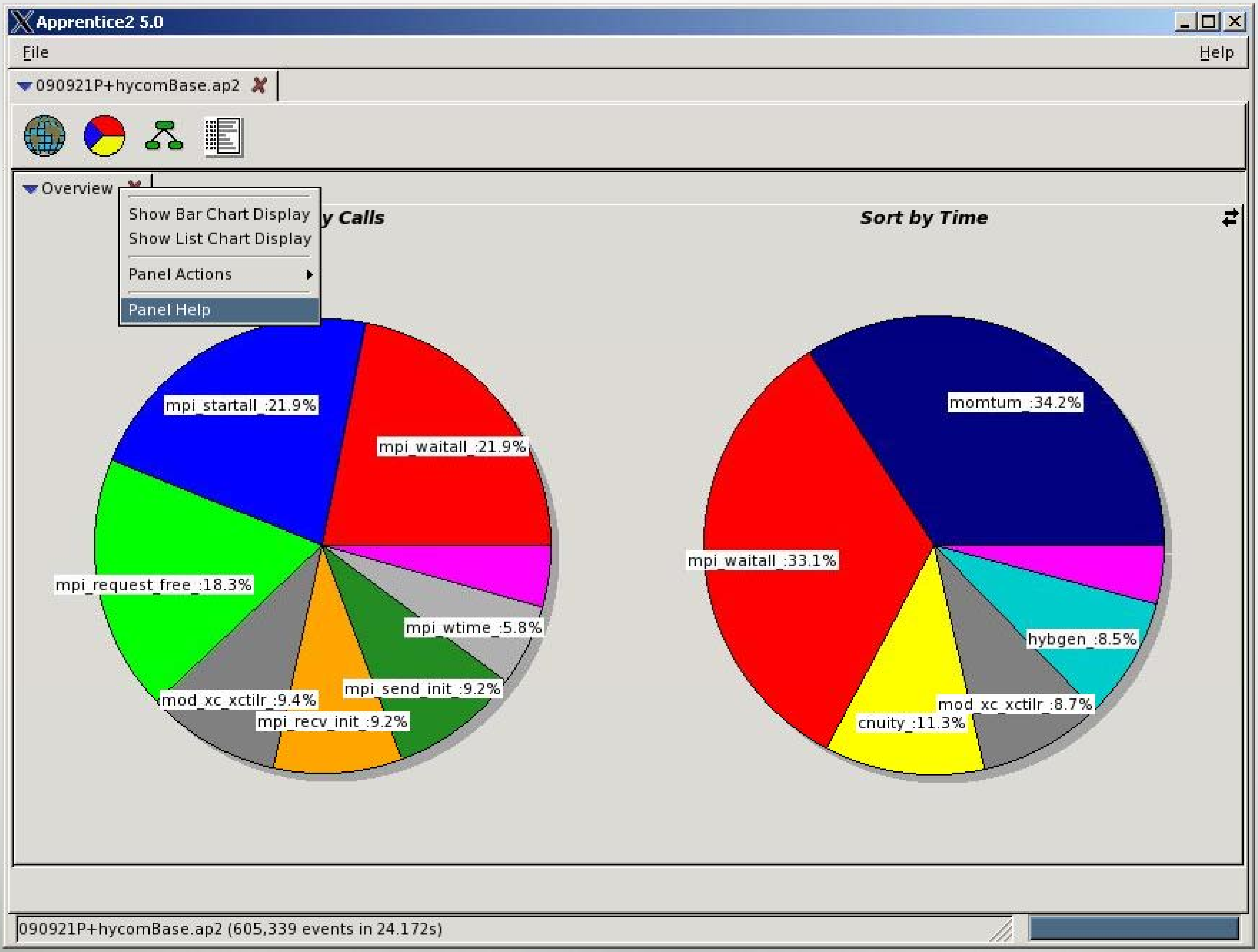
Online Information (2)

- Cray Apprentice^2 panel help
- pat_help – interactive help on the Cray Performance toolset
- FAQ available through pat_help

Man pages

- **intro_craypat**(1)
  - Introduces the craypat performance tool
- **pat_build(1)**
  - Instrument a program for performance analysis
- **pat_help(1)**
  - Interactive online help utility
- **pat_report(1)**
  - Generate performance report in both text and for use with GUI
- **hwpc**(5)
  - describes predefined hardware performance counter groups
- **intro_papi**(3)
  - Lists PAPI event counters
  - Use papi_avail or papi_native_avail utilities to get list of events when running on a specific architecture

Cray Apprentice^2 Panel Help



090921P+hycomBase.ap2 (605,339 events in 24.172s)

Top of Default Report from APA Sampling

```
CrayPat/X:  Version 5.0 Revision 2631 (xf 2571)  05/29/09 14:54:00

Number of PEs (MPI ranks):      48
Number of Threads per PE:       1
Number of Cores per Processor:  4

Execution start time:   Fri May 29 15:31:49 2009
System type and speed:   x86_64  2200 MHz
Current path to data file:
  /lus/nid00008/homer/sweep3d/sweep3d.mpi+samp.rts.ap2  (RTS)

Notes:
    Sampling interval was 10000 microseconds (100.0/sec)
    BSD timer type was ITIMER_PROF

  Trace option suggestions have been generated into a separate file
  from the data in the next table.  You can examine the file, edit
  it if desired, and use it to reinstrument the program like this:

            pat_build -O sweep3d.mpi+samp.rts.apa
```

pat_help

- Interactive by default, or use trailing '.' to just print a topic:
- Troubleshooting FAQ available
- Has counter and counter group information

```
% pat_help counters amd_fam15h groups
```

pat_help Example

```
    The top level CrayPat/X help topics are listed below.
    A good place to start is:

        overview

    If a topic has subtopics, they are displayed under the heading
    "Additional topics", as below.  To view a subtopic, you need
    only enter as many initial letters as required to distinguish
    it from other items in the list.  To see a table of contents
    including subtopics of those subtopics, etc., enter:

        toc

    To produce the full text corresponding to the table of contents,
    specify "all", but preferably in a non-interactive invocation:

        pat_help all . > all_pat_help
        pat_help report all . > all_report_help

  Additional topics:

    API                     execute
    balance                 experiment
    build                   first_example
    counters                overview
    demos                   report
    environment             run

pat_help (.=quit ,=back ^=up /=top ~=search)
=>
```

pat_help: FAQ

```
pat_help (.=quit ,=back ^=up /=top ~=search)
=> FAQ
  Additional topics that may follow "FAQ":

    Application Runtime                    Miscellaneous
    Availability and Module Environment    Processing Data with pat_report
    Building Applications                  Visualizing Data with Apprentice2
    Instrumenting with pat_build

pat_help FAQ (.=quit ,=back ^=up /=top ~=search)
=> I
  Additional topics that may follow ""Instrumenting with pat_build"":

        1. Can not access the file ...
        2. ERROR: Missing required ELF section 'link information' from the program 'FILE'.
        3. ERROR: Missing required ELF section 'string table' from the program '...'.
        4. FATAL: The link information was not found in the .note section of ...
        5. How can I find out the text size of functions?
        6. How can I list trace points from my instrumented binary?
        7. How can I lower the size of data files with pat_build?
        8. How can I NOT instrument some of my object file(s)?
        9. How do I get MPI rank order suggestions?
       10. How do I specify a directory containing object files?
       11. My error messaage is "xyz can not be traced because ... not writable"
       12. Problems with instrumented programs using both MPI and OpenMP?
       13. User sampling with compiler hooks present is not allowed
       14. WARNING: Entry point 'FUNCTION' can not be traced because it is a locally
           defined function
       15. WARNING: The function 'FUNCTION' can not be traced because a trace wrapper
           was not successfully created
       16. What is APA?
       17. Why am I getting an error with userTraceFunctions.c?
       18. Why does my binary take longer to run when using 'pat_build -u'?

pat_help FAQ "Instrumenting with pat_build"
(.=quit ,=back ^=up /=top ~=search) =>
```

THE SUPERCOMPUTER COMPANY

What’s Coming Next

Feature Highlights Coming in 2011 – 2012

- Perftools 5.3.0 (Dec 2011)
  - Loop work estimates enhancement: loops integrated into the call chain (exclusive loop times in the profile)
  - pat_region API can be used to bracket PGI accelerator directives and CUDA driver API code to collect accelerator performance statistics
  - Windows version of Cray Apprentice2

- Perftools 6.0 (3Q2012)
  - Reveal 1.0
  - Mac version of Cray Apprentice2 client and server
  - New program performance summary in Cray Apprentice2
  - Additional observations and suggestions related to memory traffic outliers

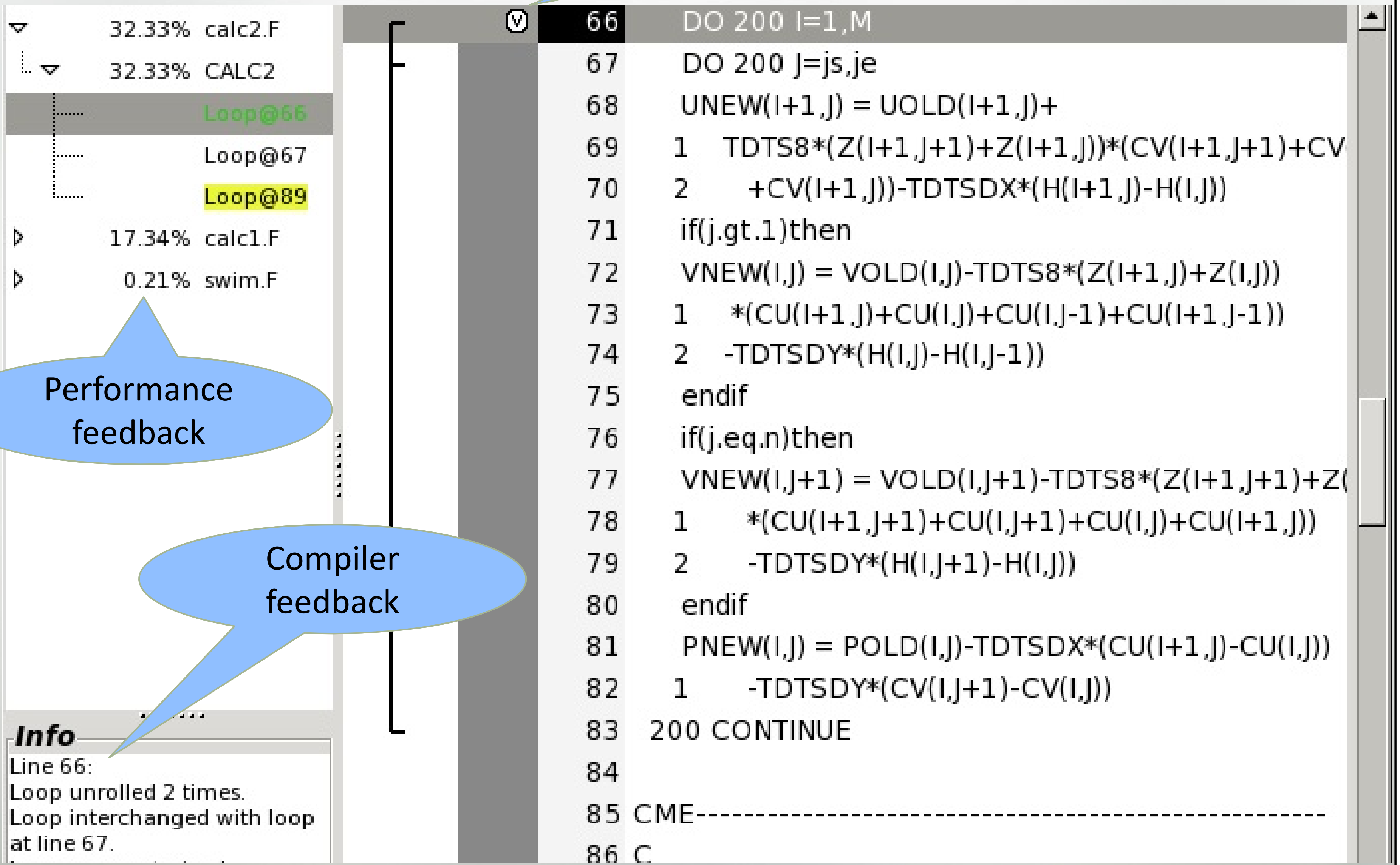
Reveal

New code restructuring and analysis assistant…

- Uses both the performance toolset and CCE's program library functionality to provide static and runtime analysis information and hints
- Assists user with the code optimization phase by correlating source code with analysis to help identify which areas are key candidates for optimization

- Key Features
  - Annotated source code with compiler optimization information
    - Highlighted loops that could not be optimized
    - Feedback on critical dependencies that prevent optimizations
  - Scoping analysis
    - Identify, shared, private and ambiguous arrays
      - Allow user to privatize ambiguous arrays
      - Allow user to override dependency analysis
  - Source code navigation based on performance data collected through CrayPat

Source Code – Loopmark



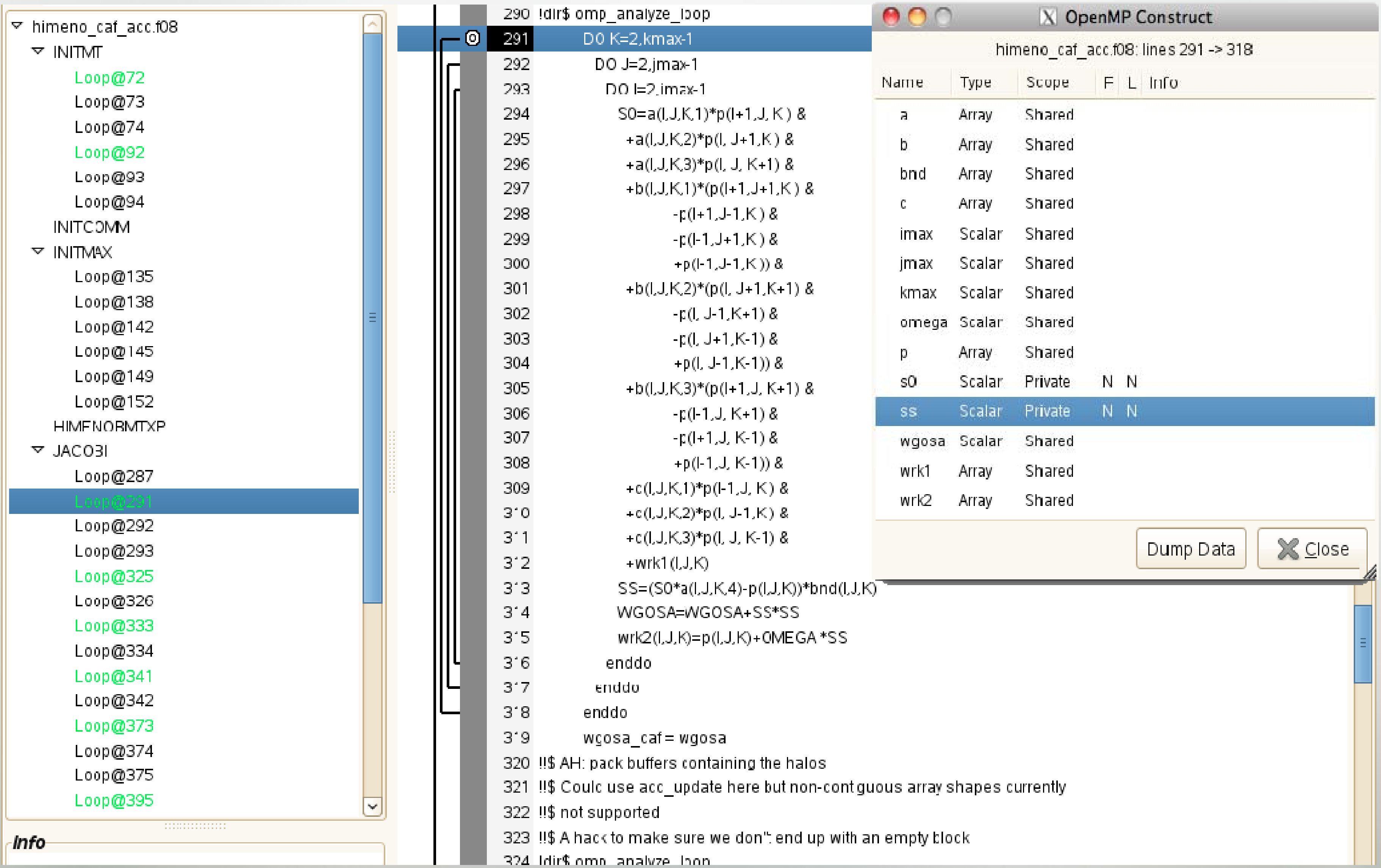
```
32.33% calc2.F
  32.33% CALC2
    Loop@66
    Loop@67
    Loop@89
17.34% calc1.F
 0.21% swim.F
```

Info

Line 66:
Loop unrolled 2 times.
Loop interchanged with loop at line 67.

```
66        DO 200 I=1,M
67        DO 200 J=js,je
68        UNEW(I+1,J) = UOLD(I+1,J)+
69      1   TDTS8*(Z(I+1,J+1)+Z(I+1,J))*(CV(I+1,J+1)+CV
70      2      +CV(I+1,J))-TDTSDX*(H(I+1,J)-H(I,J))
71        if(j.gt.1)then
72        VNEW(I,J) = VOLD(I,J)-TDTS8*(Z(I+1,J)+Z(I,J))
73      1    *(CU(I+1,J)+CU(I,J)+CU(I,J-1)+CU(I+1,J-1))
74      2   -TDTSDY*(H(I,J)-H(I,J-1))
75        endif
76        if(j.eq.n)then
77        VNEW(I,J+1) = VOLD(I,J+1)-TDTS8*(Z(I+1,J+1)+Z(
78      1      *(CU(I+1,J+1)+CU(I,J+1)+CU(I,J)+CU(I+1,J))
79      2      -TDTSDY*(H(I,J+1)-H(I,J))
80        endif
81        PNEW(I,J) = POLD(I,J)-TDTSDX*(CU(I+1,J)-CU(I,J))
82      1      -TDTSDY*(CV(I,J+1)-CV(I,J))
83    200 CONTINUE
84
85   CME-----------------------------------------------
86   C
```

Display Scoping Information for Selected Loop



```
▽ himeno_caf_acc.f08
  ▽ INITMT
      Loop@72
      Loop@73
      Loop@74
      Loop@92
      Loop@93
      Loop@94
    INITCOMM
  ▽ INITMAX
      Loop@135
      Loop@138
      Loop@142
      Loop@145
      Loop@149
      Loop@152
    HIMENORMTXP
  ▽ JACOBI
      Loop@287
      Loop@291
      Loop@292
      Loop@293
      Loop@325
      Loop@326
      Loop@333
      Loop@334
      Loop@341
      Loop@342
      Loop@373
      Loop@374
      Loop@375
      Loop@395
```

Info

```
290 !dir$ omp_analyze_loop
291        DO K=2,kmax-1
292          DO J=2,jmax-1
293            DO I=2,imax-1
294              S0=a(I,J,K,1)*p(I+1,J, K) &
295                +a(I,J,K,2)*p(I, J+1,K) &
296                +a(I,J,K,3)*p(I, J, K+1) &
297                +b(I,J,K,1)*(p(I+1,J+1,K) &
298                         -p(I+1,J-1,K) &
299                         -p(I-1,J+1,K) &
300                         +p(I-1,J-1,K)) &
301                +b(I,J,K,2)*(p(I, J+1,K+1) &
302                         -p(I, J-1,K+1) &
303                         -p(I, J+1,K-1) &
304                         +p(I, J-1,K-1)) &
305                +b(I,J,K,3)*(p(I+1,J, K+1) &
306                         -p(I-1,J, K+1) &
307                         -p(I+1,J, K-1) &
308                         +p(I-1,J, K-1)) &
309                +c(I,J,K,1)*p(I-1,J, K) &
310                +c(I,J,K,2)*p(I, J-1,K) &
311                +c(I,J,K,3)*p(I, J, K-1) &
312                +wrk1(I,J,K)
313              SS=(S0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)
314              WGOSA=WGOSA+SS*SS
315              wrk2(I,J,K)=p(I,J,K)+OMEGA *SS
316            enddo
317          enddo
318        enddo
319        wgosa_caf = wgosa
320 !!$ AH: pack buffers containing the halos
321 !!$ Could use acc_update here but non-contiguous array shapes currently
322 !!$ not supported
323 !!$ A hack to make sure we don't end up with an empty block
324 !dir$ omp_analyze_loop
```

OpenMP Construct

himeno_caf_acc.f08: lines 291 -> 318

Name	Type	Scope	F	L	Info
a	Array	Shared			
b	Array	Shared			
bnd	Array	Shared			
c	Array	Shared			
imax	Scalar	Shared			
jmax	Scalar	Shared			
kmax	Scalar	Shared			
omega	Scalar	Shared			
p	Array	Shared			
s0	Scalar	Private	N	N	
ss	Scalar	Private	N	N	
wgosa	Scalar	Shared			
wrk1	Array	Shared			
wrk2	Array	Shared			

THE SUPERCOMPUTER COMPANY

Questions

??