Introduction to the Cray Programming Environment

October 10, 2019

Heidi Poxon

CRAY®

a Hewlett Packard Enterprise company

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.

© 2019 Cray, a Hewlett Packard Enterprise company

Copyright and Trademark Acknowledgements

©2016-2019 Cray, a Hewlett Packard Enterprise company. All Rights Reserved.

Portions Copyright Advanced Micro Devices, Inc. (“AMD”) Confidential and Proprietary.

The following are trademarks of Cray, and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray: 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: 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.

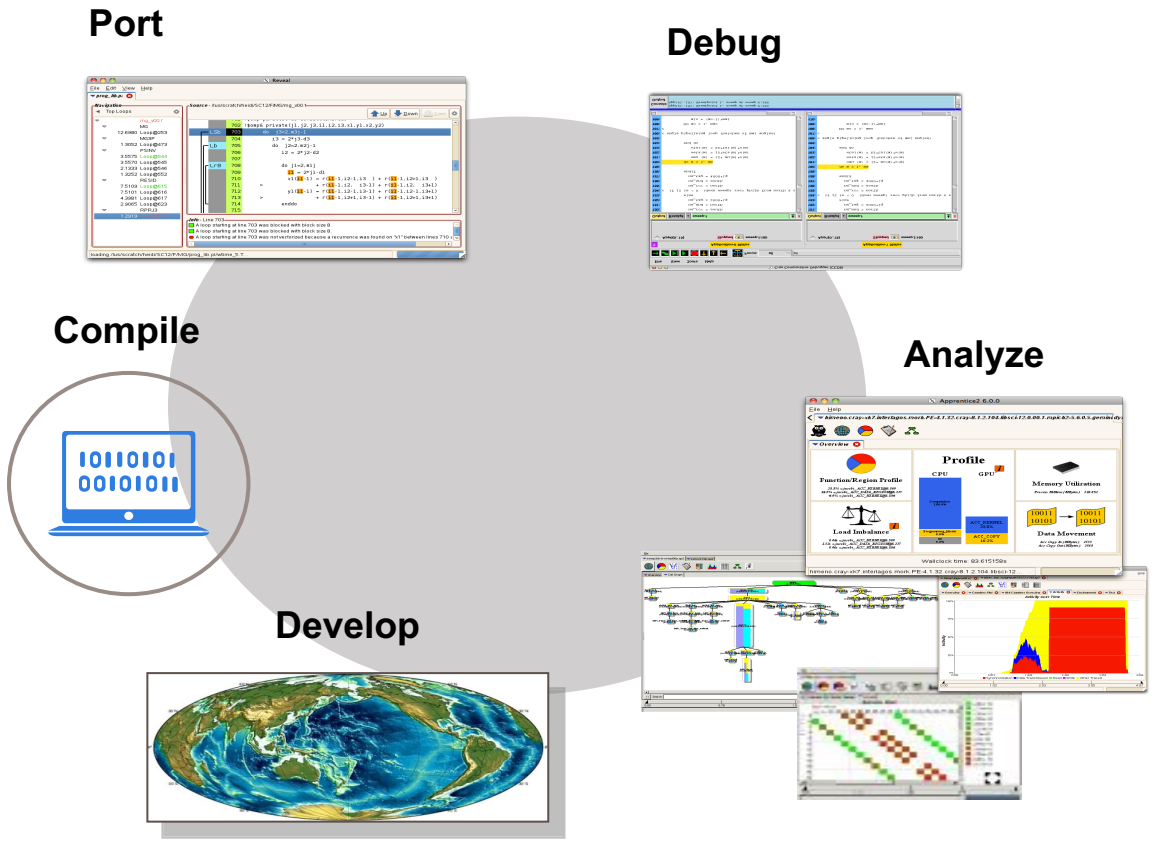
Other trademarks used in this document are the property of their respective owners.

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.

© 2019 Cray, a Hewlett Packard Enterprise company

Cray PE



- **Intuitive behavior** and best performance with the least amount of effort
- **Mature programming environment** to develop, debug, analyze, and optimize applications **for production supercomputing**
- Complete developer software stack with unique functionality built from **close interaction with users**

© 2019 Cray, a Hewlett Packard Enterprise company

Cray Developer Environment for Cray XC

Programming Languages

- Languages
- Fortran
- C
- C++
- Chapel
- Python
- R

Programming Models

- Distributed Memory
- Cray MPI
- SHMEM
- Shared Memory / GPU
- OpenMP
- PGAS & Global View
- UPC
- Fortran coarrays
- Coarray C++
- Chapel
- Global Arrays

Programming Environments

- PrgEnv-
- Cray Compiling Environment PrgEnv-cray
- GNU PrgEnv-gnu
- 3rd Party compilers (Intel, Allinea, PGI) PrgEnv-???

Optimized Libraries

- Scientific Libraries
- BLAS
- LAPACK
- ScaLAPACK
- Iterative Refinement Toolkit
- libsci_acc
- FFTW
- I/O Libraries
- NetCDF
- HDF5

Tools

- Environment setup
- Modules
- Performance Analysis
- CrayPAT
- Cray Apprentice^2
- Porting
- Reveal
- CCDB

Tools (continued)

- Debuggers
- gdb4hpc
- TotalView
- DDT
- Debugging Support
- Valgrind4hpc
- STAT
- Abnormal Termination Processing (ATP)

Analytics / AI **

- AI Toolboxes
- Cray Urika XC AI - Analytics
- Chapel AI
- DL Frameworks
- Cray PE DL Scalability Plugin

Cray Developed
Cray added value to 3rd party
3rd party package
Licensed ISV SW

© 2019 Cray, a Hewlett Packard Enterprise company

Cray Developer Environment for Cray CS

Programming Languages	Programming Models	Programming Environments	Optimized Libraries	Tools	Tools (continued)	Analytics / AI **
Languages	Distributed Memory	PrgEnv-	Scientific Libraries	Environment setup	Debuggers	AI Toolboxes
Fortran	Intel MPI MVAPICH	Cray Compiling Environment PrgEnv-cray	BLAS	Modules	gdb4hpc	Cray Urika CS AI - Analytics
C	Shared Memory / GPU		LAPACK		TotalView	Chapel AI
C++	OpenMP		ScaLAPACK	Performance Analysis	DDT	
Chapel	PGAS & Global View		Iterative Refinement Toolkit	CrayPAT	Debugging Support	DL Frameworks
	UPC Fortran coarrays Coarray C++ Chapel		libsci_acc	Cray Apprentice^2	Valgrind4hpc	Cray PE DL Scalability Plugin
			FFTW	Porting		
				Reveal		
				CCDB		

Cray Developed
Cray added value to 3rd party
3rd party package
Licensed ISV SW

© 2019 Cray, a Hewlett Packard Enterprise company

Cray's PE Vision for Accelerated Computing

- **Most important hurdle** for widespread adoption of accelerated computing in HPC **is programming difficulty**
  - Need a single programming model that is **portable across machine types**
    - **Portable** expression of heterogeneity and multi-level parallelism
    - Programming model and optimization should not be significantly difference for "accelerated" nodes and multi-core x86 processors
    - **Allow users to maintain a single code base**

- Cray's approach to Accelerator Programming is to provide an **ease of use** tightly coupled high level programming environment with compilers, libraries, and tools that can **hide the complexity** of the system

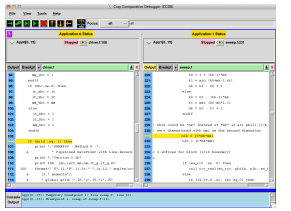
- **Ease of use** is possible with
  - Compiler making it **feasible for users** to write applications in **Fortran, C, and C++**
  - Tools to help users port and optimize for hybrid systems
  - Auto-tuned scientific libraries

© 2019 Cray, a Hewlett Packard Enterprise company

Cray PE Technology Applied to Accelerators

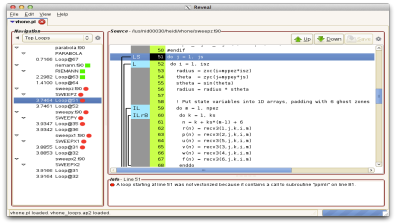
- **Fortran, C, and C++ compilers**
  - OpenMP target offload directives to drive compiler optimization
    - Compiler optimizations take advantage of accelerator and multi-core X86 hardware appropriately
  - Support for new programming frameworks, such as **RAJA and Kokkos**
  - Advanced users **can mix HIP or CUDA functions with compiler-generated accelerator code**

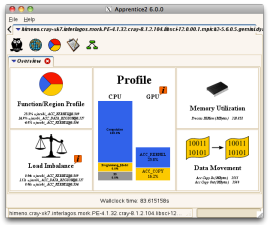
- **Debugger** support with allinea www.allinea.com DDT, ROGUE WAVE SOFTWARE TotalView, or Cray gdb4hpc & CCDB



- **Scientific Libraries** tuned to take advantage of multi-core and accelerators appropriately

- **Cray Reveal**, built upon compiler knowledge of the application
  - Scoping tool to help users port and optimize applications





- **Cray Performance tools** for whole-program view of performance

- **Cray MPI GPU to GPU** for direct communications between GPUs on node and off node

© 2019 Cray, a Hewlett Packard Enterprise company

Environment Setup

© 2019 Cray, a Hewlett Packard Enterprise company

Programmability Focused Environment

- Modules simplify build environment
  - Complexity of compile and link lines (-h –l –l –L) reduced
- Multiple product versions, compilers, and compiler versions available on system at the same time offers more flexibility and convenience
- Product agnostic drivers (cc, CC, ftn) are used to compile for supported Programming Environments
  - Customer-integrated and Cray libraries share the same driver interface
- Support available to plug 3rd party software into Cray software environment (craypkg-gen)

© 2019 Cray, a Hewlett Packard Enterprise company

Which Software Versions Are Available?

```
user@login:~> module avail cce
--------------------------- /opt/cray/pe/modulefiles ---------------------------
cce/8.7.0          cce/8.7.3          cce/8.7.6          cce/8.7.9
cce/8.7.1          cce/8.7.4          cce/8.7.7          cce/8.7.10
cce/8.7.2          cce/8.7.5          cce/8.7.8          cce/9.0.0(default)


user@login:~> module avail PrgEnv
--------------------------- /opt/cray/pe/modulefiles ---------------------------
PrgEnv-cray/6.0.3           PrgEnv-gnu/6.0.3           PrgEnv-intel/6.0.3
PrgEnv-cray/6.0.4           PrgEnv-gnu/6.0.4           PrgEnv-intel/6.0.4
PrgEnv-cray/6.0.5(default)  PrgEnv-gnu/6.0.5(default)  PrgEnv-intel/6.0.5(default)


user@login:~> module -S avail fftw
--------------------------- /opt/cray/pe/modulefiles ---------------------------
cray-fftw/3.3.6.1          cray-fftw/3.3.6.4          cray-fftw/3.3.8.2
cray-fftw/3.3.6.2          cray-fftw/3.3.6.5          cray-fftw/3.3.8.3(default)
cray-fftw/3.3.6.3          cray-fftw/3.3.8.1          fftw/2.1.5.9
```

© 2019 Cray, a Hewlett Packard Enterprise company

Targeting Processors / Accelerators

```
user@login:~> module avail craype

------------------------ /opt/cray/pe/craype/2.6.1/modulefiles -----------------
craype-accel-host       craype-hugepages2M     craype-intel-knc
craype-accel-nvidia20   craype-hugepages4M     craype-ivybridge
craype-accel-nvidia35   craype-hugepages8M     craype-mic-knl
craype-accel-nvidia52   craype-hugepages16M    craype-network-aries
craype-accel-nvidia60   craype-hugepages32M    craype-network-none
craype-broadwell        craype-hugepages64M    craype-sandybridge
craype-haswell          craype-hugepages128M   craype-x86-cascadelake
craype-hugepages1G      craype-hugepages256M   craype-x86-skylake
craype-hugepages2G      craype-hugepages512M

---------------------------------- /opt/cray/pe/modulefiles ---------------------
craype/2.6.1(default)                    craype-dl-plugin-py3/19.09.1(default)
craype-dl-plugin-py2/19.09.1(default)
```

© 2019 Cray, a Hewlett Packard Enterprise company

Choosing Different Compilers on Cray XC

- To access a different compiler:
  - Load or swap to the corresponding Programming Environment (PE) module
    - PrgEnv-cray for CCE
    - PrgEnv-intel for Intel
    - PrgEnv-gnu for GNU
  - Once one of these is loaded, you can then select a compiler version
    - CCE: **module avail cce**
    - GNU: **module avail gcc**
- **With PE 19.06 the default linking for all compilers (CCE, Intel, GCC, etc.) is now dynamic**
  - Static linking still a non-default option, where supported

© 2019 Cray, a Hewlett Packard Enterprise company

Using the Compilers

- Cray Systems come with compiler wrappers to simplify building parallel applications (similar the mpicc/mpif90)
  - Fortran Compiler: **ftn**
  - C Compiler: **cc**
  - C++ Compiler: **CC**

- Using these wrappers ensures that your code is built for the compute nodes and linked against important libraries
  - Cray MPT (MPI, Shmem, etc.)
  - Cray LibSci (BLAS, LAPACK, etc.)
  - …

- Do not call the Cray, Intel, GNU compilers directly

- Cray Compiler wrappers try to hide the complexities of using the proper header files and libraries

© 2019 Cray, a Hewlett Packard Enterprise company

Release Notes Easily Accessible

```
user@login:~> module help cce

----------- Module Specific Help for 'cce/9.0.0' -----------------

The modulefile, cce, defines the system paths and environment
variables needed to run the Cray Compile Environment.

Type "module avail cce" to see if other versions of this product
are available on this system. Use "module switch" to change versions.

Cray Compiling Environment (CCE) 9.0.0
======================================

Release Date:
-------------
June 20, 2019

Purpose:
--------
CCE 9.0.0 provides Fortran, C, and C++ compilers for Cray XC systems and Cray CS systems.
  - With this release, the default C and C++ compilers are based on Clang/LLVM
  - The previous C and C++ compilers are also provided and referred to as Cray Classic C and C++
  - See S-5212 Cray Compiling Environment Release Overview (9.0) for additional information

The following key enhancements are included in this CCE release:
 - Dynamic linking is the default link mode
      - This change is in craype included with the June PE (19.06) release
      - Applies to all compilers (CCE, Intel, GCC, etc.) when the new craype is loaded
      - Override the default with the -static flag on the command line,
         or by setting CRAYPE_LINK_TYPE=static in the environment
  - C++17 support for both Classic and Clang-based C and C++ compilers
  -...
```

© 2019 Cray, a Hewlett Packard Enterprise company

Cray MPI

© 2019 Cray, a Hewlett Packard Enterprise company

Cray MPI Overview

- ANL MPICH3.2 implementation base
- Fully compliant with the **MPI-3.1** Specification
- Optimize for scale in the following areas: I/O, collectives, point-to-point, one-sided communication
- Optimization examples
  - SMP-aware collectives
  - High performance single-copy on-node communication via Cray's XPMEM
  - MPI rank reordering
  - Supports GPU to GPU direct communication
  - **Ran with 2,001,150 ranks on Trinity in 2016**
- Highly tunable through environment variables
  - Defaults should generally be best, but some cases benefit from fine tuning

© 2019 Cray, a Hewlett Packard Enterprise company

Example Environment Variable Control

- By default, MPI uses 2M hugepages for Aries GNI mailboxes internally
- Used even if user didn't select hugepages or if user selected different hugepage size
- User can change the 2M size by setting the MPICH_GNI_HUGEPAGE_SIZE environment variable
- Other buffers that get allocated within MPI (e.g. temporary buffers for collectives) will abide by the hugepage module the user has selected

© 2019 Cray, a Hewlett Packard Enterprise company

MPICH_GNI_HUGEPAGE_SIZE

```
Specifies the hugepage size in bytes
that will be used for the GNI internal mailbox memory. The
default size is 2MB. Jobs that scale to high process counts
and have a high connectivity pattern may benefit from using
a larger hugepage size for this memory, as this can reduce
the number of Aries PTE misses. If setting
MPICH_GNI_HUGEPAGE_SIZE to a larger value, you may also want
to increase the MPICH_GNI_MBOXES_PER_BLOCK value.

The supported values are 2M, 4M, 8M, 16M, 32M, 64M, 128M,
256M, and 512M. For CLE 6.0.UP05 and later, values of 1G and
2G are also supported.

Default: 2M
```

See the MPI man page for environment variable descriptions

© 2019 Cray, a Hewlett Packard Enterprise company

Interaction with Cray PE and other MPI Libraries

- Integrated into the Cray Programming Environment
  - Compiler drivers manage compile flags and linking automatically
  - Profiling through Cray performance tools

- MPICH ABI compatibility
  - Programs built with other MPICH ABI compatible vendors like Intel MPI can get native Aries performance without recompiling
  - http://wiki.mpich.org/mpich/index.php/ABI_Compatibility_Initiative
  - `cray-mpich-abi` module exposes this feature

© 2019 Cray, a Hewlett Packard Enterprise company

Cray Performance Tools

© 2019 Cray, a Hewlett Packard Enterprise company

Cray Performance Tools

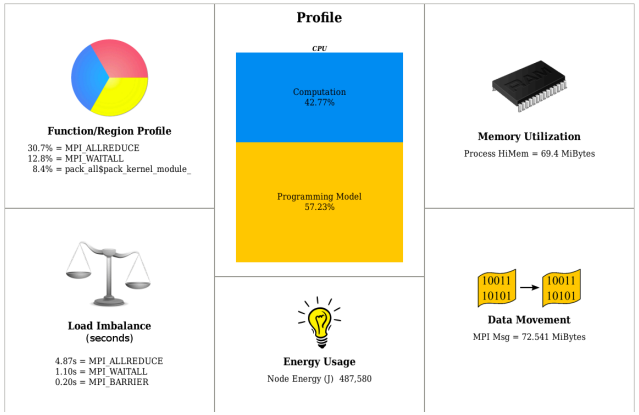
- Reduce the time investment associated with porting and tuning applications on Cray systems
- Analyze whole-program behavior across many nodes to identify critical performance bottlenecks within a program
- Improve profiling experience by using simple and/or advanced interfaces for a wealth of capability that targets analyzing the largest HPC jobs

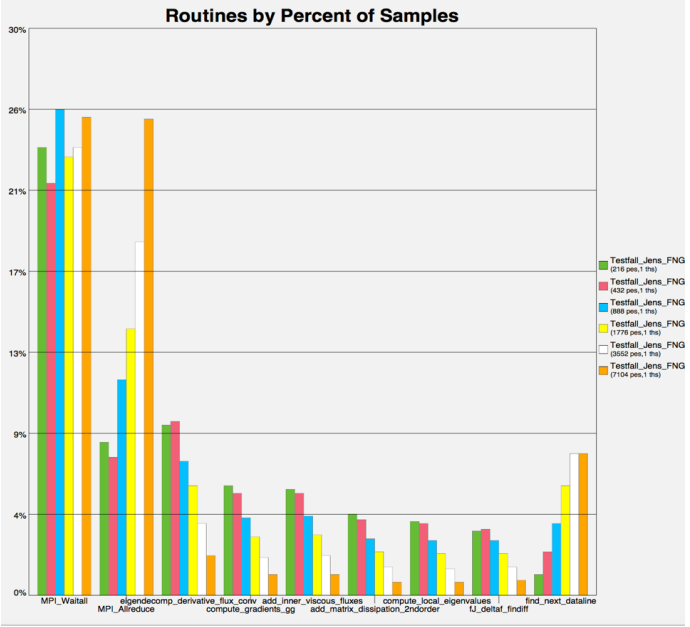
© 2019 Cray, a Hewlett Packard Enterprise company

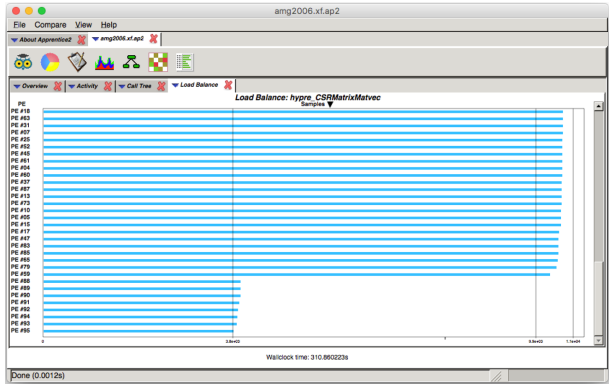
Cray Tools Provide Various Levels of Detail

Table 1: Time and Bytes Transferred for Accelerator Regions

Host Time%	Host Time	Acc Time	Acc Copy In (MiBytes)	Acc Copy Out (MiBytes)	Events	Function=[max10] PE=HIDE Thread=HIDE
100.0%	173.71	163.57	2,544	2,544	1,071,949	Total
50.4%	87.64	83.26	--	--	320,970	cudaLaunchKernel
13.7%	23.71	22.55	2,544	--	285,936	CloverleafCudaChunk::unpackBuffer
11.8%	20.46	19.37	--	2,544	285,938	CloverleafCudaChunk::packBuffer
10.0%	17.32	15.54	--	--	67,975	CloverleafCudaChunk::update_array
5.9%	10.29	9.53	--	--	23,640	CloverleafCudaChunk::advec_mom_kernel
3.3%	5.65	5.26	--	0.09	29,550	thrust::tuple<> thrust::cuda_cub::__extrema::extrema<>
1.9%	3.30	3.08	--	--	11,820	void thrust::cuda_cub::core::AgentLauncher<>::launch<> const








```
/* Turn data recording on for two regions of interest. */
PAT_record(PAT_STATE_ON);

PAT_region_begin(1, "step 1");
...
PAT_region_end(1);
PAT_region_begin(2, "step 2");
...
PAT_region_end(2);

/* Turn data recording off again. */
PAT_record(PAT_STATE_OFF);
```

© 2019 Cray, a Hewlett Packard Enterprise company

Two Modes of Use

- Lite modes: simple interface for convenience

- Advanced interface for in-depth performance investigation and tuning assistance

- Both offer:
  - Whole program analysis across many nodes
  - Indication of causes of problems
  - Ability to easily switch between the two interfaces

© 2019 Cray, a Hewlett Packard Enterprise company

What About Different Compilers?

Cray Performance Tools support the following compilers

- Cray (CCE), Intel, GCC, and Arm Allinea compilers on Cray XC systems
  - AMD compiler support coming next year

- Cray (CCE) compiler on Cray CS systems

© 2019 Cray, a Hewlett Packard Enterprise company

Using the Simple Interface

- `user@login> module load perftools-lite`

- Build program

- Run program

- View report sent to STDOUT (and .rpt file in experiment directory)
  - Example data directory: `stencil_order+49144-225s/`

© 2019 Cray, a Hewlett Packard Enterprise company

Find Top Bottlenecks

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

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

 100.0% | 463.147240 |        -- |     -- | 21621.0 |Total
|-------------------------------------------------------------------------
|  52.0% | 240.974379 |        -- |     -- | 21523.0 |MPI
||------------------------------------------------------------------------
||  47.7% | 221.142266 | 36.214468 |  14.1% | 10740.0 |mpi_recv
||   4.3% |  19.829001 | 25.849906 |  56.7% | 10740.0 |MPI_SEND
||========================================================================
|  43.3% | 200.474690 |        -- |     -- |    32.0 |USER
||------------------------------------------------------------------------
||  41.0% | 189.897060 | 58.716197 |  23.6% |    12.0 |sweep_
||   1.6% |   7.579876 |  1.899097 |  20.1% |    12.0 |source_
||========================================================================
|   4.7% |  21.698147 |        -- |     -- |    39.0 |MPI_SYNC
||------------------------------------------------------------------------
|   4.3% |  20.091165 | 20.005424 |  99.6% |    32.0 | mpi_allreduce_(sync)
||========================================================================
|   0.0% |   0.000024 |        -- |     -- |    27.0 |SYSCALL
```

MPI utilization Observation:

The time spent processing MPI communications is relatively high. Functions and callsites responsible for consuming the most time can be found in the table generated by

`pat_report -O callers+src`

(within the MPI group)

© 2019 Cray, a Hewlett Packard Enterprise company

Find Any Program Load Imbalance

Table 1: Profile by Function Group and Function

Time%	Time	Imb. Time	Imb. Time%	Calls	Group Function PE=HIDE
100.0%	1.957703	--	--	42,970.8	Total
60.0%	1.174021	--	--	3,602.0	USER
30.8%	0.603850	0.176924	23.0%	1,198.0	function3_
19.2%	0.375117	0.128748	26.0%	1,200.0	function2_
9.1%	0.178111	0.081880	32.0%	1,200.0	function1_
36.0%	0.704928	--	--	9,613.0	MPI_SYNC
25.8%	0.505174	0.385130	76.2%	9,596.0	mpi_barrier_(sync)
10.2%	0.199537	0.199518	100.0%	1.0	mpi_init_(sync)
4.0%	0.078736	--	--	29,754.8	MPI
2.3%	0.045351	0.003531	7.3%	9,596.0	MPI_BARRIER
1.1%	0.021520	0.051295	71.6%	8,756.9	MPI_ISEND

Look for function execution imbalance as well as late arrivers to synchronization points

© 2019 Cray, a Hewlett Packard Enterprise company

Reduce Communication Distance

```
MPI Grid Detection:

    There appears to be point-to-point MPI communication in a 96 X 8
    grid pattern. The 52% of the total execution time spent in MPI
    functions might be reduced with a rank order that maximizes
    communication between ranks on the same node. The effect of several
    rank orders is estimated below.

    A file named MPICH_RANK_ORDER.Grid was generated along with this
    report and contains usage instructions and the Custom rank order
    from the following table.

         Rank     On-Node      On-Node    MPICH_RANK_REORDER_METHOD
        Order    Bytes/PE    Bytes/PE%
                             of Total
                             Bytes/PE

       Custom  2.385e+09       95.55%   3
          SMP  1.880e+09       75.30%   1
         Fold  1.373e+06        0.06%   2
    RoundRobin 0.000e+00        0.00%   0
```

© 2019 Cray, a Hewlett Packard Enterprise company

Use Auto-Generated MPI Rank Order File

```
# The 'USER_Time_hybrid' rank order in this file targets nodes with multi-core
# processors, based on Sent Msg Total Bytes collected for:
#
# Program:  /lus/nid00023/malice/craypat/WORKSHOP/bh2o-demo/Rank/sweep3d/src/sweep3d
# Ap2 File: sweep3d.gmpi-u.ap2
# Number PEs:   768
# Max PEs/Node: 16
#
# To use this file, make a copy named MPICH_RANK_ORDER, and set the
# environment variable MPICH_RANK_REORDER_METHOD to 3 prior to
# executing the program.
#
0,532,64,564,32,572,96,540,8,596,72,524,40,604,24,588
104,556,16,628,80,636,56,620,48,516,112,580,88,548,120,612
1,403,65,435,33,411,97,443,9,467,25,499,105,507,41,475
73,395,81,427,57,459,17,419,113,491,49,387,89,451,121,483
6,436,102,468,70,404,38,412,14,444,46,476,110,508,78,500
86,396,30,428,62,460,54,492,118,420,22,452,94,388,126,484
129,563,193,531,161,571,225,539,241,595,233,523,249,603,185,555
153,587,169,627,137,635,201,619,177,515,145,579,209,547,217,611
7,405,71,469,39,437,103,413,47,445,15,509,79,477,31,501
111,397,63,461,55,429,87,421,23,493,119,389,95,453,127,485
134,402,198,434,166,410,230,442,238,466,174,506,158,394,246,474
190,498,254,426,142,458,150,386,182,418,206,490,214,450,222,482
128,533,192,541,160,565,232,525,224,573,240,597,184,557,248,605
168,589,200,517,152,629,136,549,176,637,144,621,208,581,216,613
5,439,37,407,69,447,101,415,13,471,45,503,29,479,77,511
53,399,85,431,21,463,61,391,109,423,93,455,117,495,125,487
2,530,34,562,66,538,98,522,10,570,42,554,26,594,50,602
18,514,74,586,58,626,82,546,106,634,90,578,114,618,122,610
135,315,167,339,199,347,259,307,231,371,239,379,191,331,247,299
175,363,159,323,143,355,255,291,207,275,183,283,151,267,215,223
133,406,197,438,165,470,229,414,245,446,141,478,237,502,253,398
157,510,189,462,173,430,205,390,149,422,213,454,181,494,221,486
130,316,260,340,194,372,162,348,226,308,234,380,242,332,250,300
202,364,186,324,154,356,138,292,170,276,178,284,210,218,268,146
4,535,36,543,68,567,100,527,12,599,44,575,28,559,76,607
52,591,20,631,60,639,84,519,108,623,92,551,116,583,124,615
3,440,35,432,67,400,99,408,11,464,43,496,27,472,51,504
19,392,75,424,59,456,83,384,107,416,91,488,115,448,123,480
132,401,196,441,164,409,228,433,236,465,204,473,244,393,188,497
252,505,140,425,212,457,156,385,172,417,180,449,148,489,220,481
131,534,195,542,163,566,227,526,235,574,203,598,243,558,187,606
251,590,211,630,179,638,139,622,155,550,171,518,219,582,147,614
761,660,737,652,705,668,745,692,673,700,641,684,713,644,753,724
729,732,681,756,721,716,764,676,697,748,689,657,740,665,649,708
760,528,736,536,704,560,744,520,672,568,712,592,752,552,640,600
728,584,680,624,720,512,696,632,688,616,664,544,608,656,648,576
762,659,738,651,706,667,746,643,714,691,674,699,754,683,730,723
722,731,763,658,642,755,739,675,707,650,682,715,698,666,690,747
257,345,265,313,281,305,273,337,609,369,577,377,617,329,513,529
545,297,633,361,625,321,585,537,601,289,553,353,593,521,569,561
256,373,261,341,264,349,280,317,272,381,269,309,285,333,277,365
352,301,320,325,288,357,328,304,360,312,376,293,296,368,336,344
258,338,266,346,282,314,274,370,766,306,710,378,742,330,678,362
646,298,750,322,718,354,758,290,734,662,686,670,726,702,694,654
262,375,263,343,270,311,271,351,286,319,278,342,287,350,279,374
294,318,358,383,359,310,295,382,326,303,327,367,366,335,302,334
765,661,709,663,741,653,711,669,767,655,743,671,749,695,679,703
677,727,751,693,647,701,717,687,757,685,733,725,719,735,645,759
```

© 2019 Cray, a Hewlett Packard Enterprise company

View Memory Traffic per NUMA Domain

MPI

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

Memory Traffic GBytes	Local Memory Traffic GBytes	Remote Memory Traffic GBytes	Memory Traffic GBytes / Sec	Memory Traffic / Nominal Peak	Numanode Node Id=[max3,min3] PE=HIDE
172.95	171.48	1.48	8.75	11.4%	numanode.0
172.77	171.48	1.30	8.90	11.6%	nid.68
172.09	170.61	1.48	9.02	11.7%	nid.63
171.20	169.93	1.27	9.71	12.6%	nid.62
162.51	161.07	1.43	8.26	10.8%	nid.71
162.28	160.82	1.46	8.22	10.7%	nid.72
161.75	160.29	1.46	8.19	10.7%	nid.70
168.69	166.81	1.89	8.53	11.1%	numanode.1
168.69	166.81	1.89	8.67	11.3%	nid.62
167.74	166.03	1.71	8.61	11.2%	nid.63
166.66	164.88	1.78	8.67	11.3%	nid.61
161.68	160.07	1.61	8.17	10.6%	nid.71
161.60	159.99	1.62	8.23	10.7%	nid.70
157.32	156.01	1.31	8.72	11.4%	nid.72

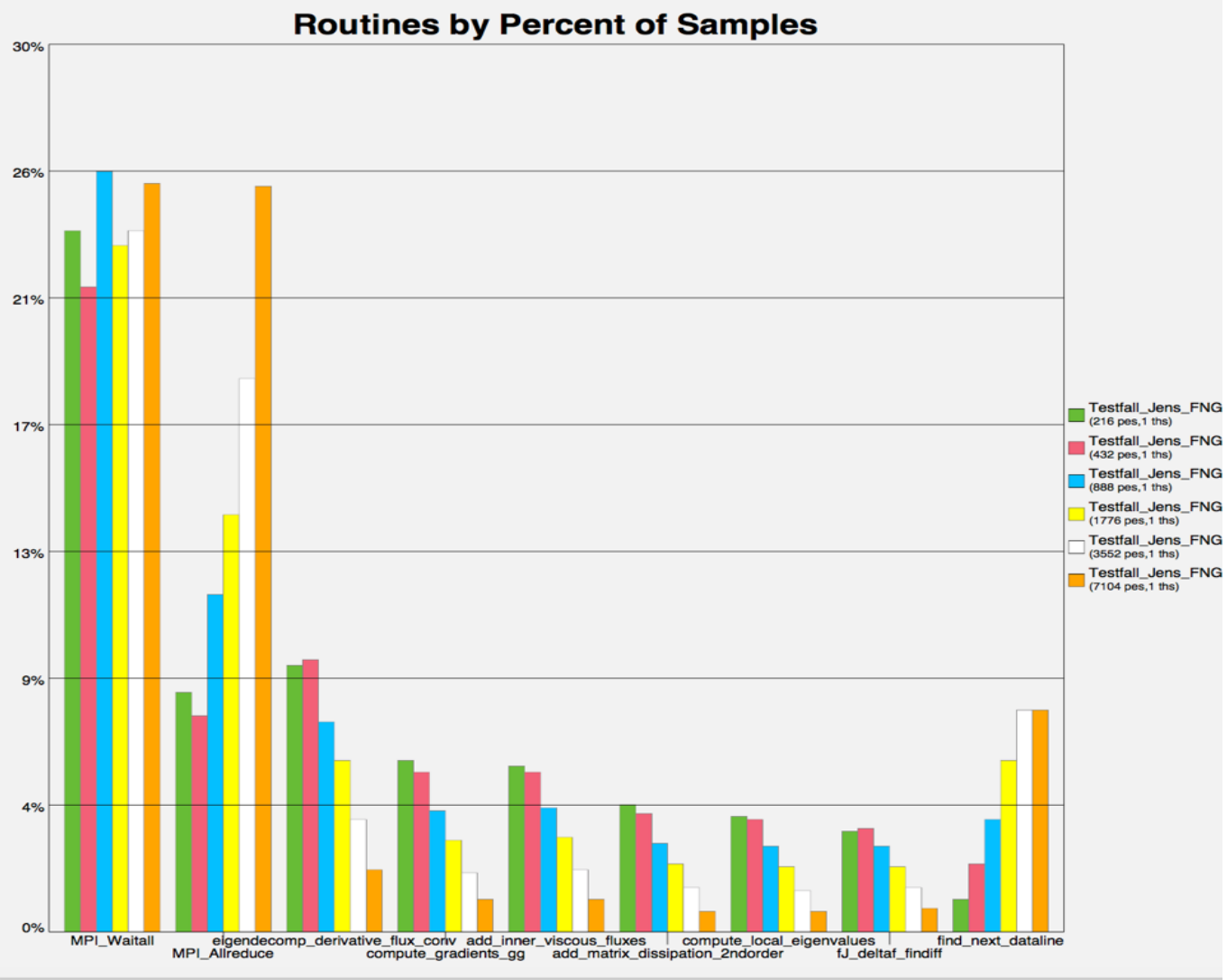
MPI + OpenMP

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

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

© 2019 Cray, a Hewlett Packard Enterprise company

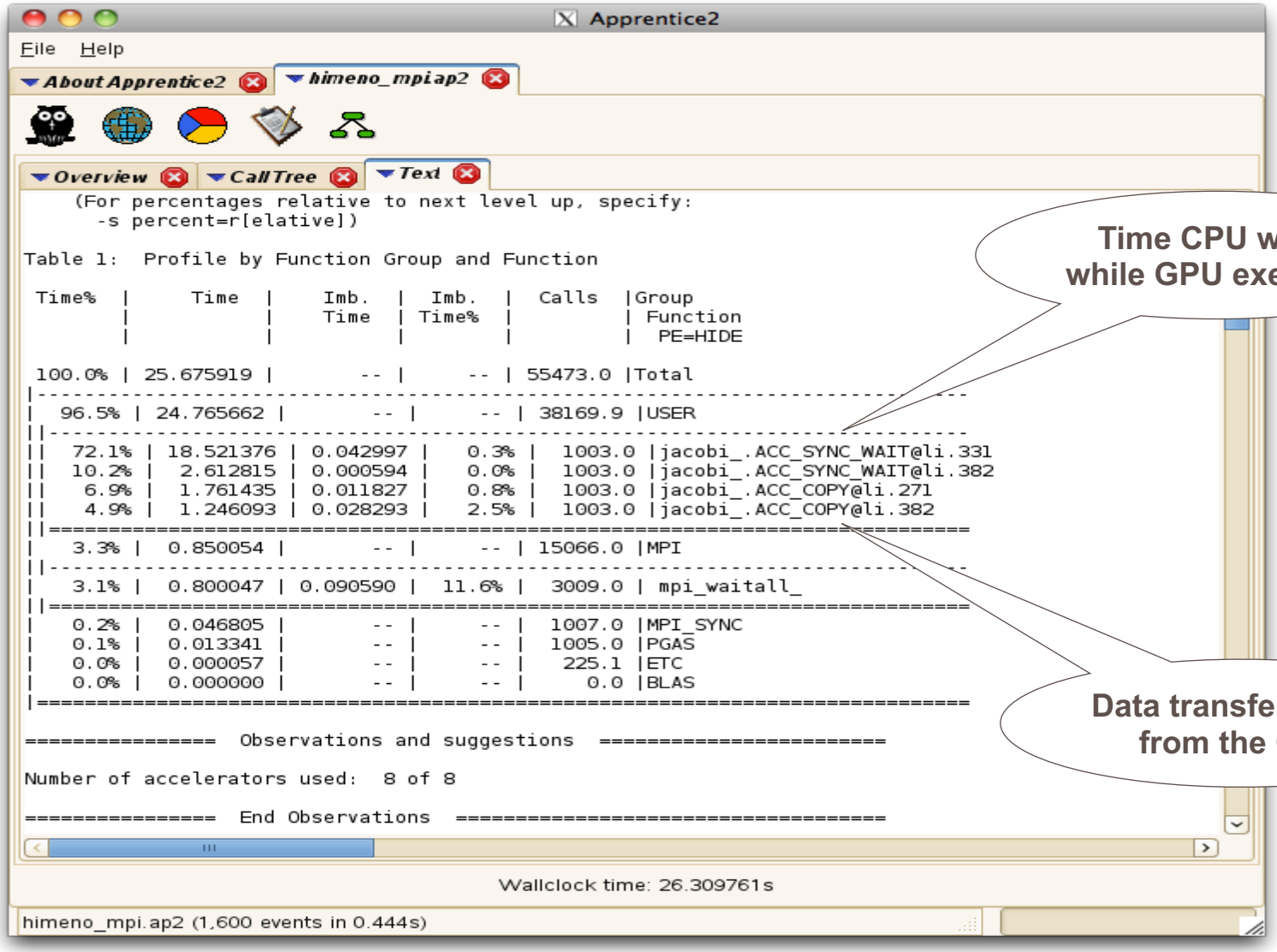
Check Scaling



- pat_view takes multiple experiment directories as input
- Helpful when assessing performance differences between runs
- Good for function or overall program scaling analysis

© 2019 Cray, a Hewlett Packard Enterprise company

View Application Profile with GPU Information



© 2019 Cray, a Hewlett Packard Enterprise company

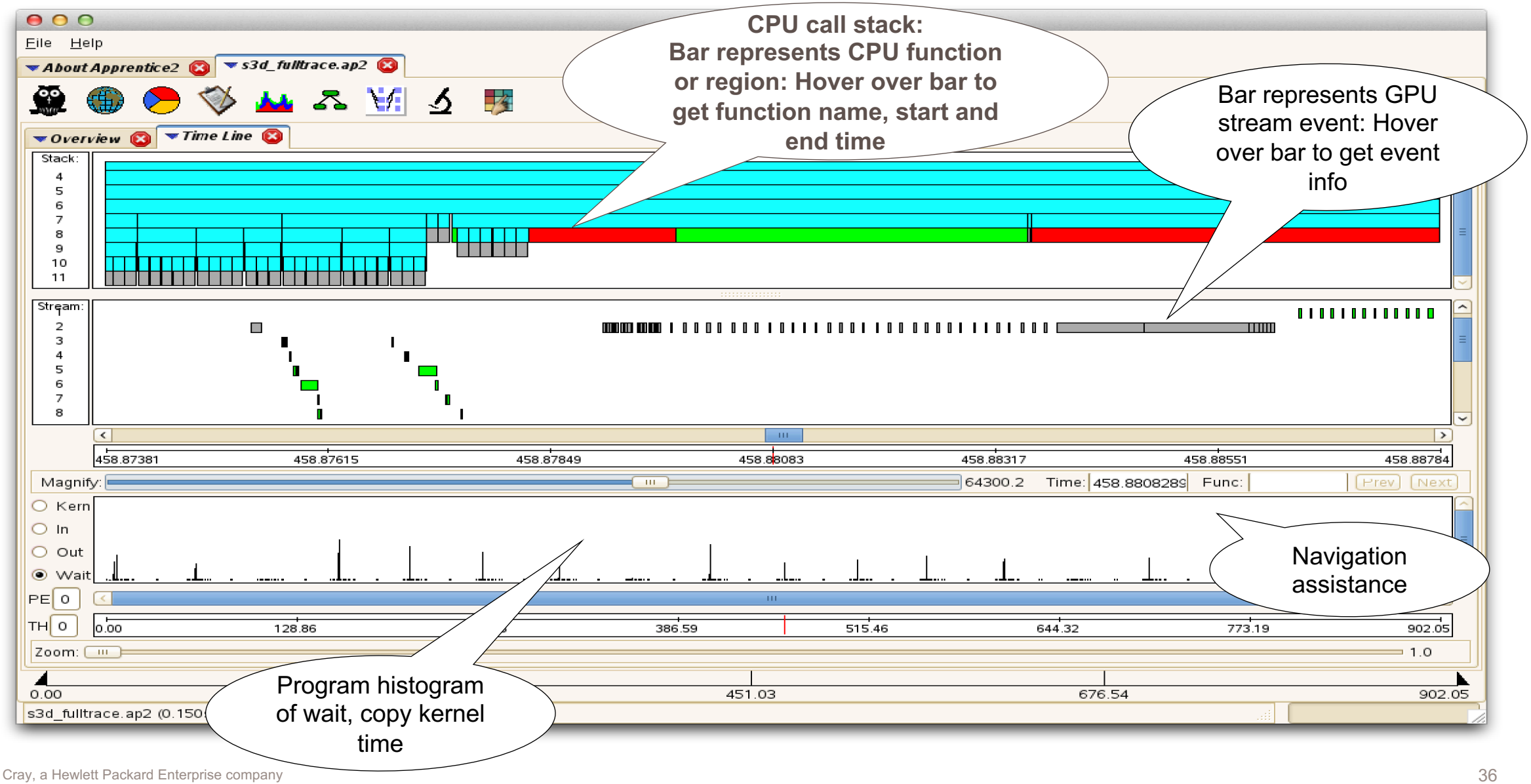
Focus on Accelerator Statistics

```
Table 1:  Time and Bytes Transferred for Accelerator Regions
  Host  |    Host  |     Acc  | Acc Copy  | Acc Copy  | Calls  |Calltree
 Time%  |    Time  |    Time  |       In  |      Out  |        | PE=HIDE
        |          |          | (MBytes)  | (MBytes)  |        |
 100.0% |    2.750 |    2.015 |  2812.760 |    13.568 |    103 |Total
|-------------------------------------------------------------------------------------------
-
| 100.0% |    2.750 |    2.015 |  2812.760 |    13.568 |    103 |lbm3d2p_d_
|        |          |          |           |           |        | lbm3d2p_d_.ACC_DATA_REGION@li.104
|||-----------------------------------------------------------------------------------------
-
3||  63.5% |    1.747 |    1.747 |  2799.192 |        -- |      1 |lbm3d2p_d_.ACC_COPY@li.104
3||  22.1% |    0.609 |    0.088 |    12.304 |    12.304 |     36 |streaming_
||||---------------------------------------------------------------------------------------
-
4|||  20.6% |    0.566 |    0.046 |    12.304 |    12.304 |     27 |streaming_exchange_
5|||        |          |          |           |           |        | streaming_exchange_.ACC_DATA_REGION@li.526
6|||  18.8% |    0.517 |       -- |        -- |        -- |      1 |
streaming_exchange_.ACC_DATA_REGION@li.526(exclusive)
4|||   1.6% |    0.043 |    0.042 |        -- |        -- |      9 |streaming_.ACC_DATA_REGION@li.907
5|||   1.1% |    0.031 |    0.031 |        -- |        -- |      4 | streaming_.ACC_REGION@li.909
```

Host time (wallclock)
Time spent on GPU (wallclock)
Data copies to and from GPU
Time to copy data

© 2019 Cray, a Hewlett Packard Enterprise company

Analyze CPU and GPU Overlap

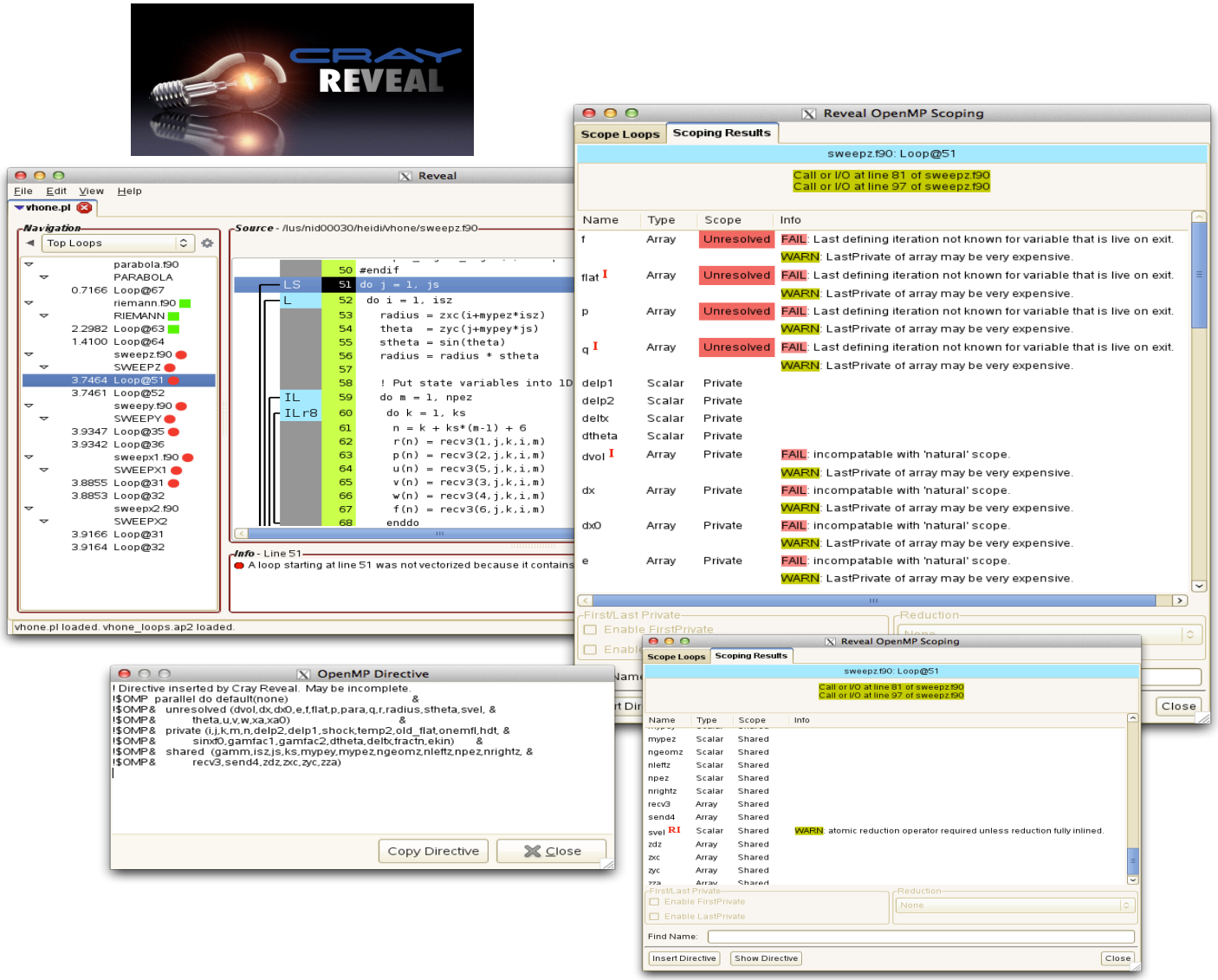


© 2019 Cray, a Hewlett Packard Enterprise company

Adding OpenMP with Cray Reveal

© 2019 Cray, a Hewlett Packard Enterprise company

Reveal



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

© 2019 Cray, a Hewlett Packard Enterprise company

The Problem – How Do I Parallelize This Loop?

- How do I know this is a good loop to parallelize?
- What prevents me from parallelizing this loop?
- Can I get help building a directive?

```
subroutine sweepz
…
do j = 1, js
 do i = 1, isz
   radius = zxc(i+mypez*isz)
   theta  = zyc(j+mypey*js)
   do m = 1, npez
    do k = 1, ks
     n = k + ks*(m-1) + 6
     r(n) = recv3(1,j,k,i,m)
     p(n) = recv3(2,j,k,i,m)
     u(n) = recv3(5,j,k,i,m)
     v(n) = recv3(3,j,k,i,m)
     w(n) = recv3(4,j,k,i,m)
     f(n) = recv3(6,j,k,i,m)
    enddo
   enddo
   …
   call ppmlr
   do k = 1, kmax
     n = k + 6
     xa (n) = zza(k)
     dx (n) = zdz(k)
     xa0(n) = zza(k)
     dx0(n) = zdz(k)
     e  (n) = p(n)/(r(n)*gamm)+0.5 &
        *(u(n)**2+v(n)**2+w(n)**2)
   enddo
   call ppmlr
…
 enddo
enddo
```

```
subroutine ppmlr

call boundary
call flatten
call paraset(nmin-4, nmax+5, para, dx, xa)

call parabola(nmin-4,nmax+4,para,p,dp,p6,pl,flat)
call parabola(nmin-4,nmax+4, para,r,dr,r6,rl,flat)
call parabola(nmin-4,nmax+4,para,u,du,u6,ul,flat)

call states(pl,ul,rl,p6,u6,r6,dp,du,dr,plft,ulft,&
            rlft,prgh,urgh,rrgh)
call riemann(nmin-3,nmax+4,gam,prgh,urgh,rrgh,&
             plft,ulft,rlft pmid umid)
call evolve(umid, pmid)  ← contains more calls

call remap ← contains more calls

call volume(nmin,nmax,ngeom,radius,xa,dx,dvol)

call remap ← contains more calls

return
end
```

© 2019 Cray, a Hewlett Packard Enterprise company

Loop Work Estimates

Gather loop statistics using the Cray performance tools and the Cray Compiling Environment (CCE) to determine which loops have the most work

- Helps identify high-level serial loops to parallelize
  - Based on runtime analysis, approximates how much work exists within a loop

© 2019 Cray, a Hewlett Packard Enterprise company

Create Loop Profile to Find Tuning Candidates

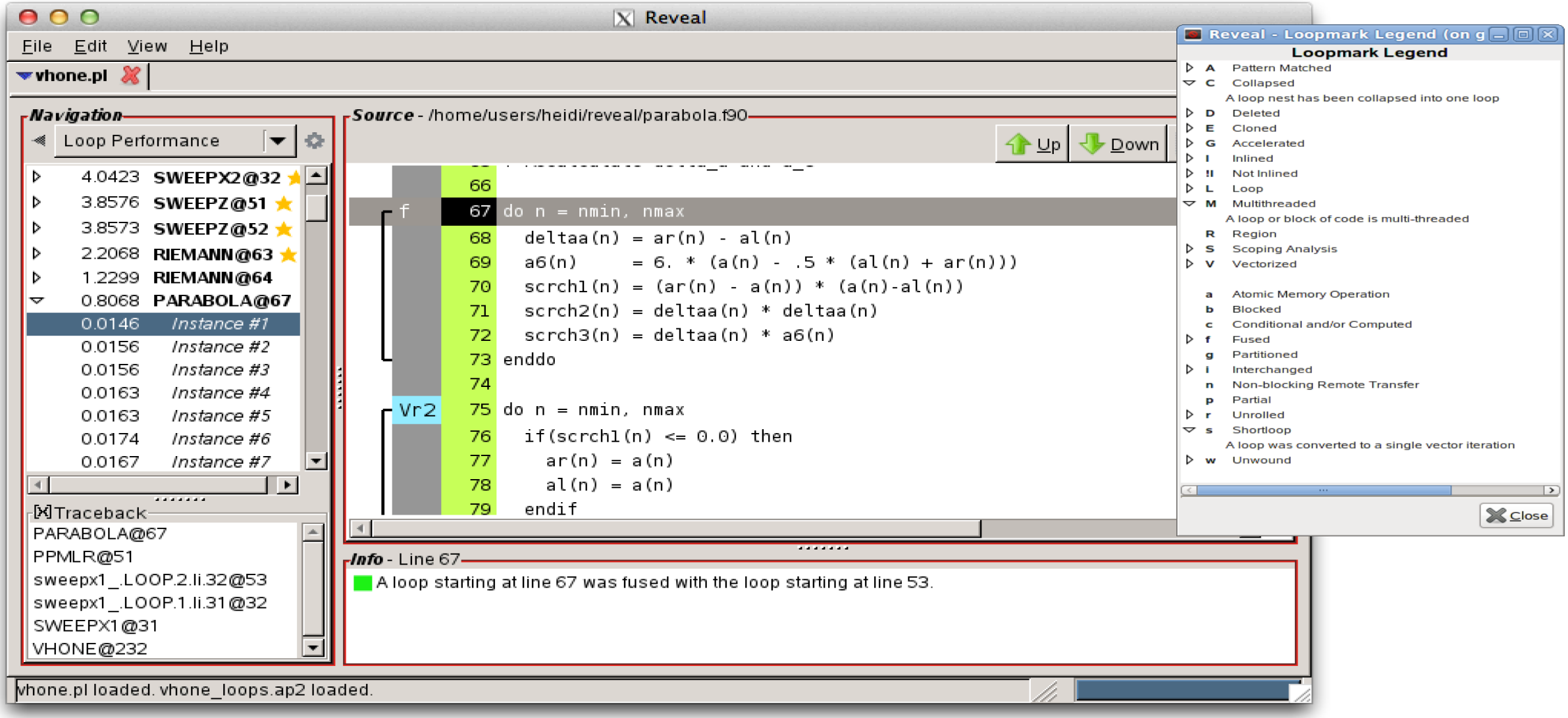
Use with compiler listing to understand compiler generated optimizations

Inclusive and Exclusive Time in Loops

Loop Incl Time%	Loop Incl Time	Loop Hit	Loop Trips Avg	Loop Trips Min	Loop Trips Max	Function=/.LOOP[.] PE=HIDE
99.4%	333.895923	1	500.0	500	500	les3d_.LOOP.3.li.216
98.7%	331.721721	500	2.0	2	2	les3d_.LOOP.4.li.272
26.5%	89.032566	1,000	96.0	96	96	fluxk_.LOOP.1.li.28
24.6%	82.681435	96,000	97.0	97	97	fluxk_.LOOP.2.li.29
24.2%	81.356609	1,000	96.0	96	96	fluxj_.LOOP.1.li.28
22.5%	75.770180	96,000	97.0	97	97	fluxj_.LOOP.2.li.29
22.5%	75.458432	1,000	96.0	96	96	fluxi_.LOOP.1.li.21
22.5%	75.453469	96,000	96.0	96	96	fluxi_.LOOP.2.li.22
18.9%	63.574836	9,312,000	96.0	96	96	visck_.LOOP.1.li.344
17.1%	57.529187	9,312,000	96.0	96	96	viscj_.LOOP.1.li.340
15.7%	52.794857	9,216,000	97.0	97	97	visci_.LOOP.1.li.782
5.0%	16.924522	1,000	99.0	99	99	extrapi_.LOOP.1.li.128

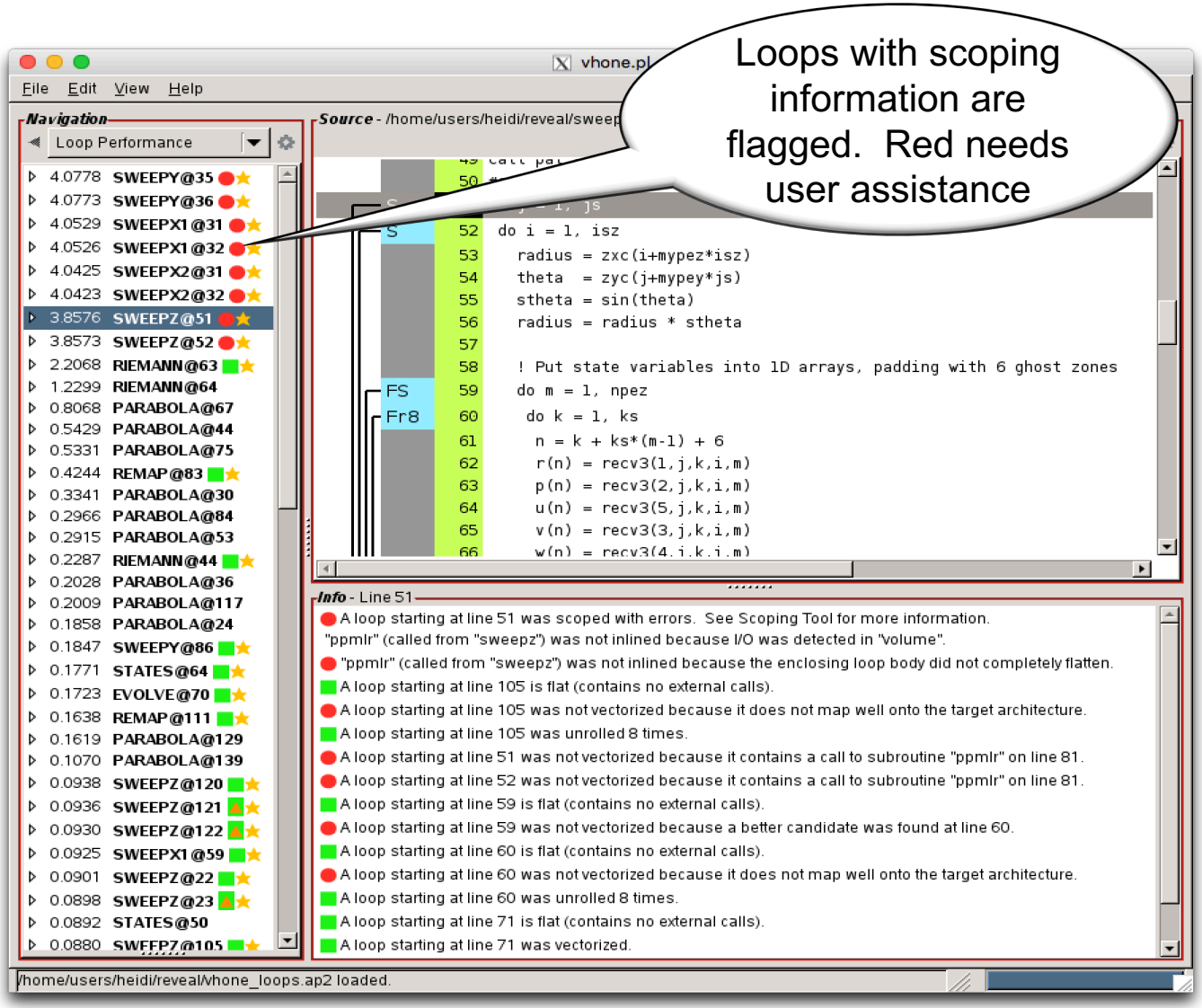
© 2019 Cray, a Hewlett Packard Enterprise company

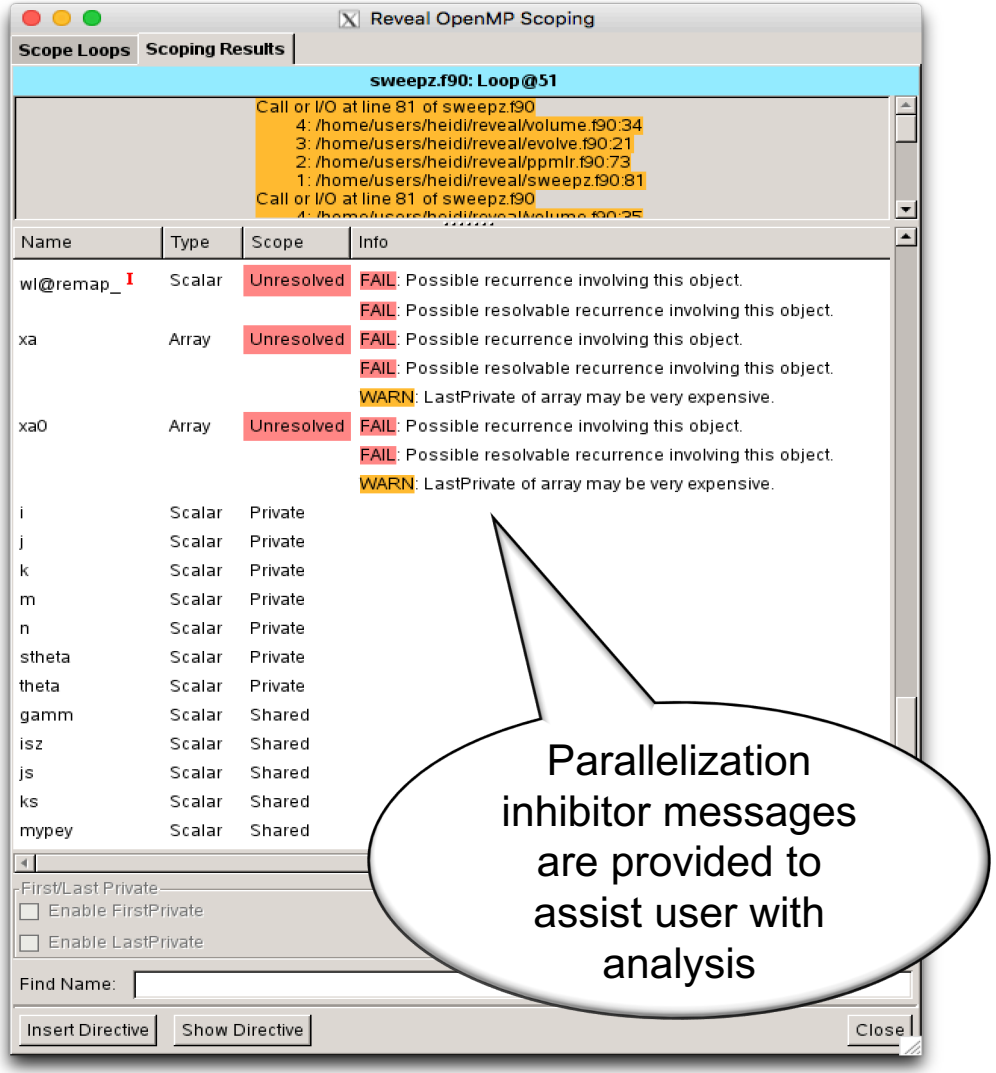
View Source and Optimization Information



© 2019 Cray, a Hewlett Packard Enterprise company

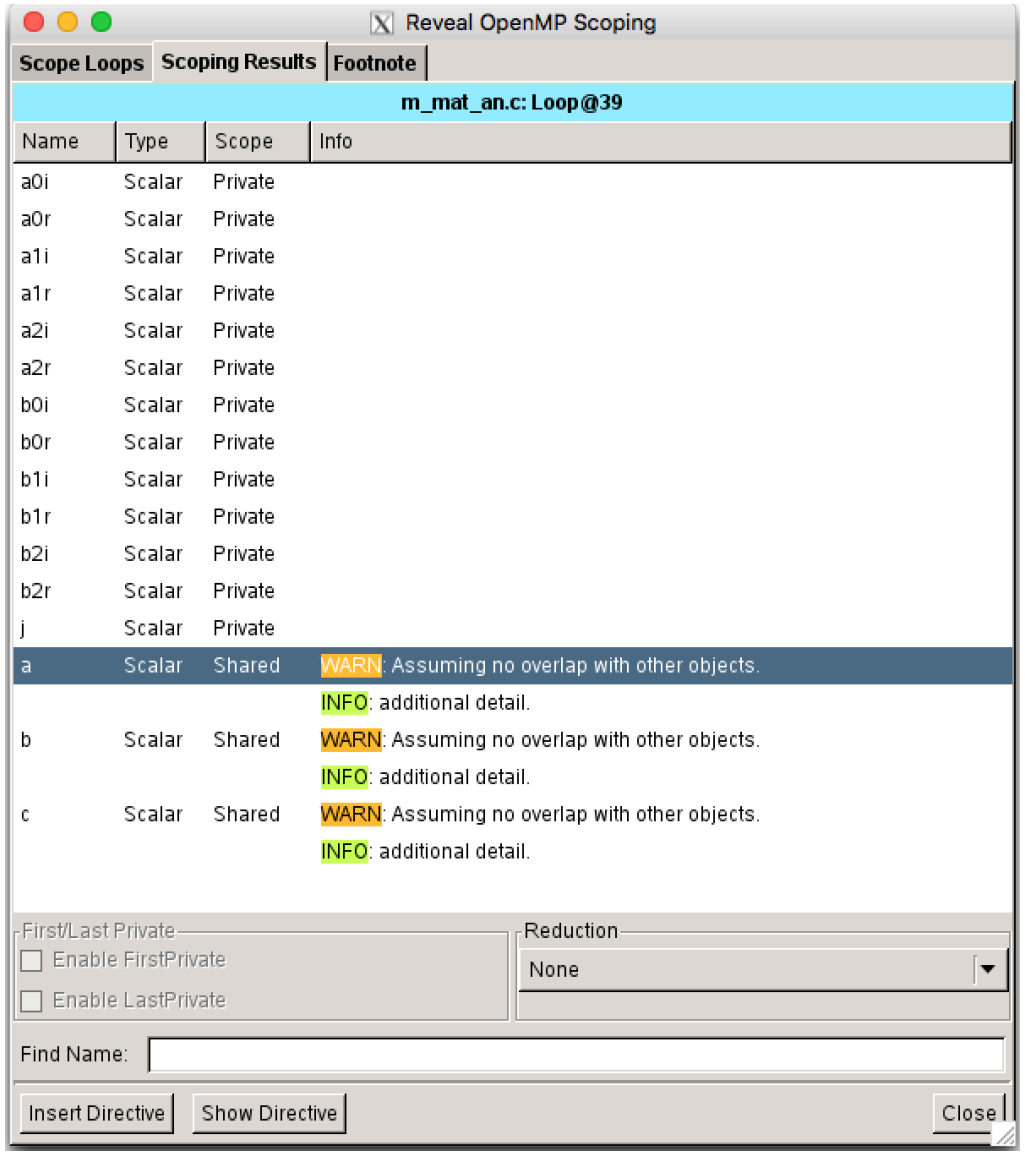
Review Scoping Results

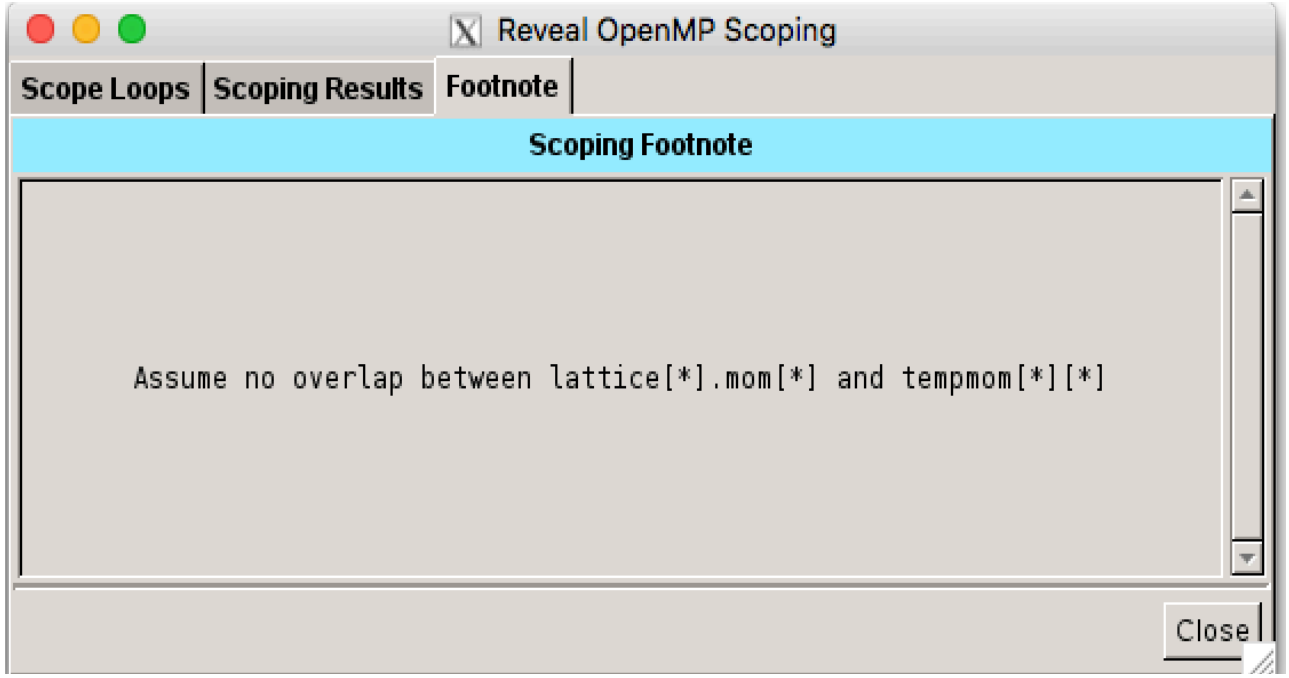




© 2019 Cray, a Hewlett Packard Enterprise company

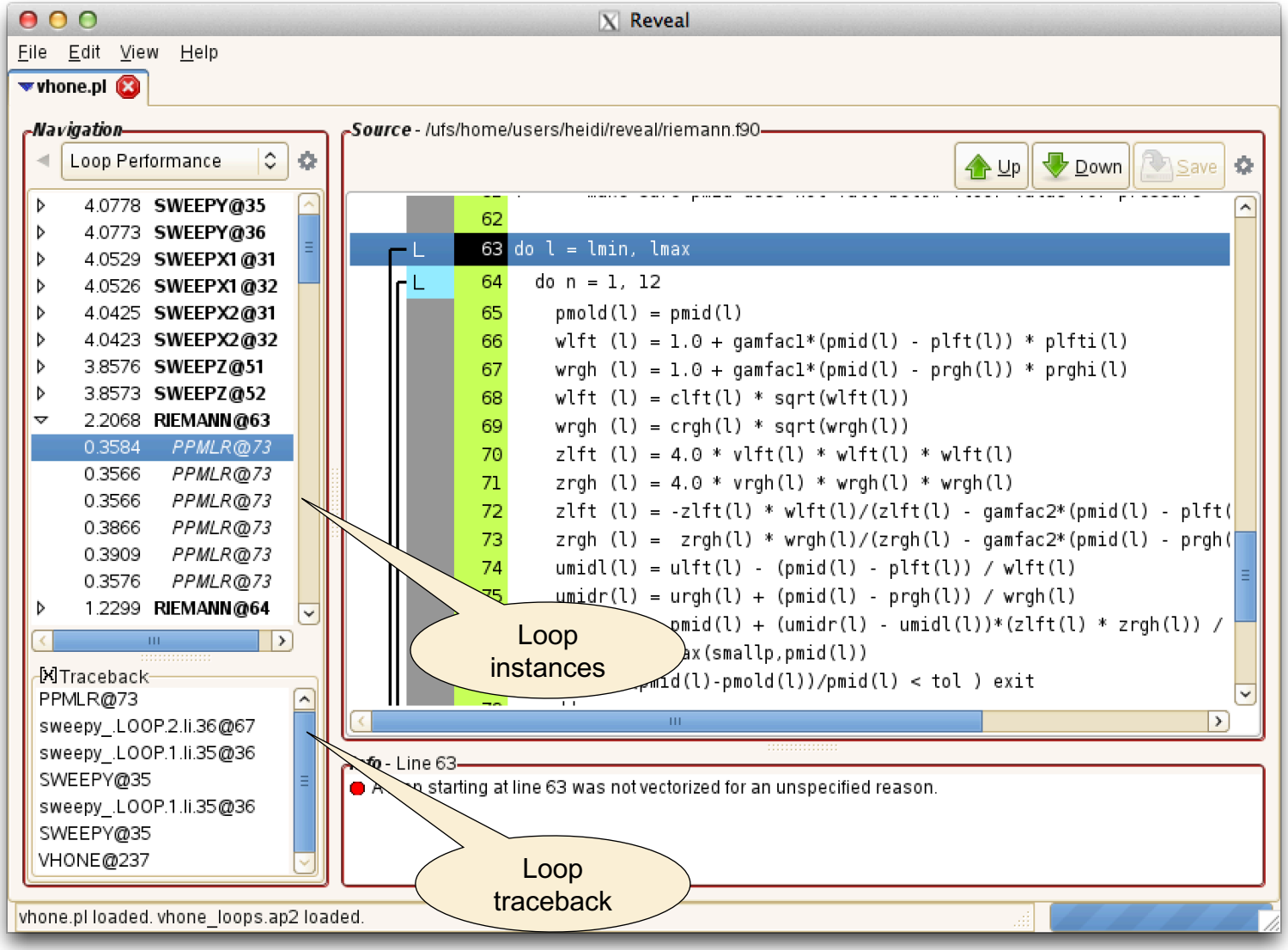
Review Scoping Results (continued)





© 2019 Cray, a Hewlett Packard Enterprise company

View Loops through Call Chain



© 2019 Cray, a Hewlett Packard Enterprise company

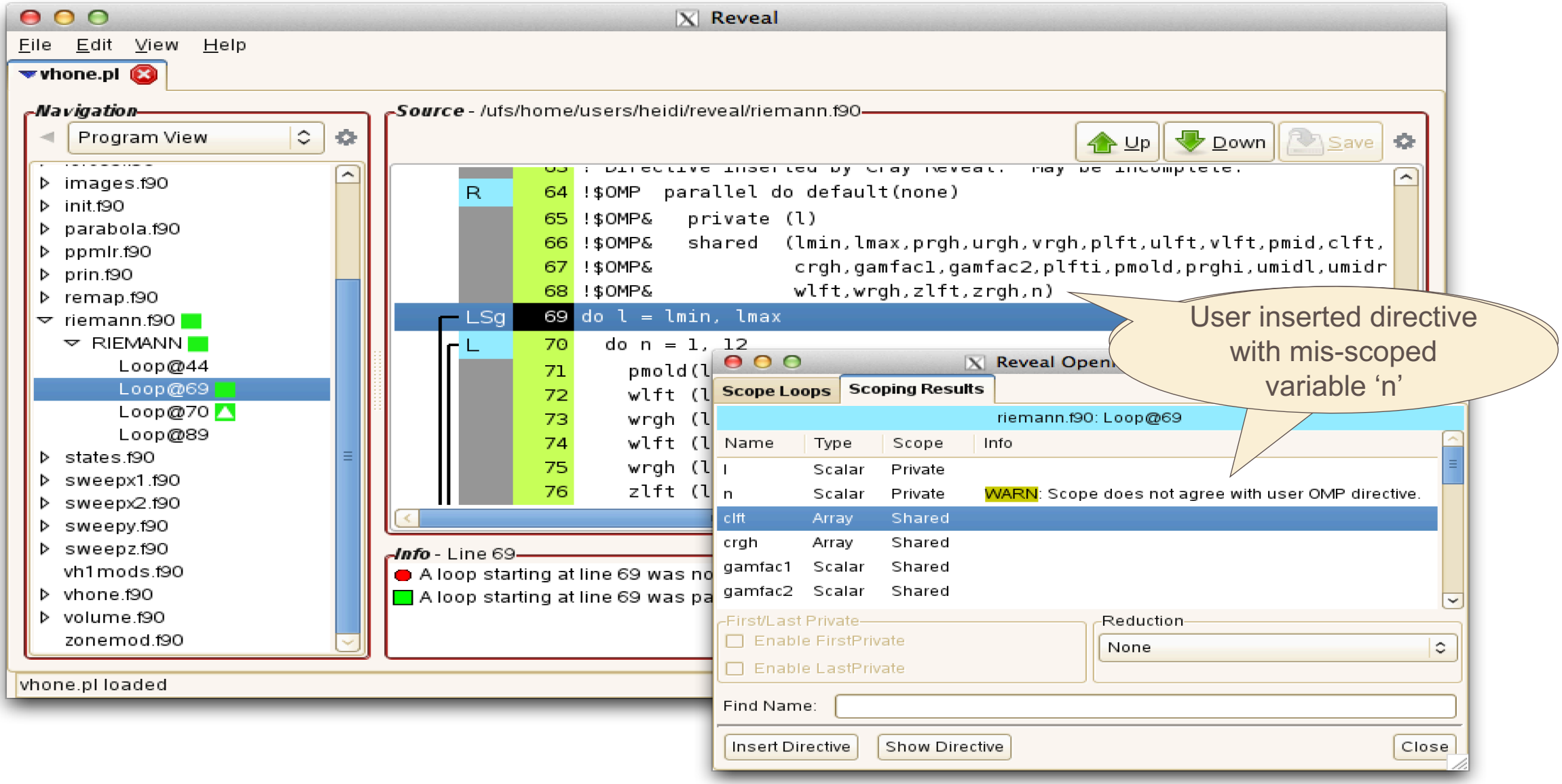
Generate OpenMP Directives

```
! Directive inserted by Cray Reveal.  May be incomplete.
!$OMP  parallel do default(none)                                        &
!$OMP&   unresolved (dvol,dx,dx0,e,f,flat,p,para,q,r,radius,svel,u,v,w,  &
!$OMP&              xa,xa0)                                              &
!$OMP&   private (i,j,k,m,n,$$_n,delp2,delp1,shock,temp2,old_flat,       &
!$OMP&             onemfl,hdt,sinxf0,gamfac1,gamfac2,dtheta,deltx,fractn, &
!$OMP&             ekin)                                                &
!$OMP&   shared  (gamm,isy,js,ks,mypey,ndim,ngeomy,nlefty,npey,nrighty, &
!$OMP&             recv1,send2,zdy,zxc,zya)
do k = 1, ks
 do i = 1, isy
  radius = zxc(i+mypey*isy)

  ! Put state variables into 1D arrays, padding with 6 ghost zones
  do m = 1, npey
   do j = 1, js
    n = j + js*(m-1) + 6
    r(n) = recv1(1,k,j,i,m)
    p(n) = recv1(2,k,j,i,m)
    u(n) = recv1(4,k,j,i,m)
    v(n) = recv1(5,k,j,i,m)
    w(n) = recv1(3,k,j,i,m)
    f(n) = recv1(6,k,j,i,m)
   enddo
  enddo

  do j = 1, jmax
    n = j + 6
```

Reveal generates OpenMP directive with illegal clause marking variables that need addressing

© 2019 Cray, a Hewlett Packard Enterprise company

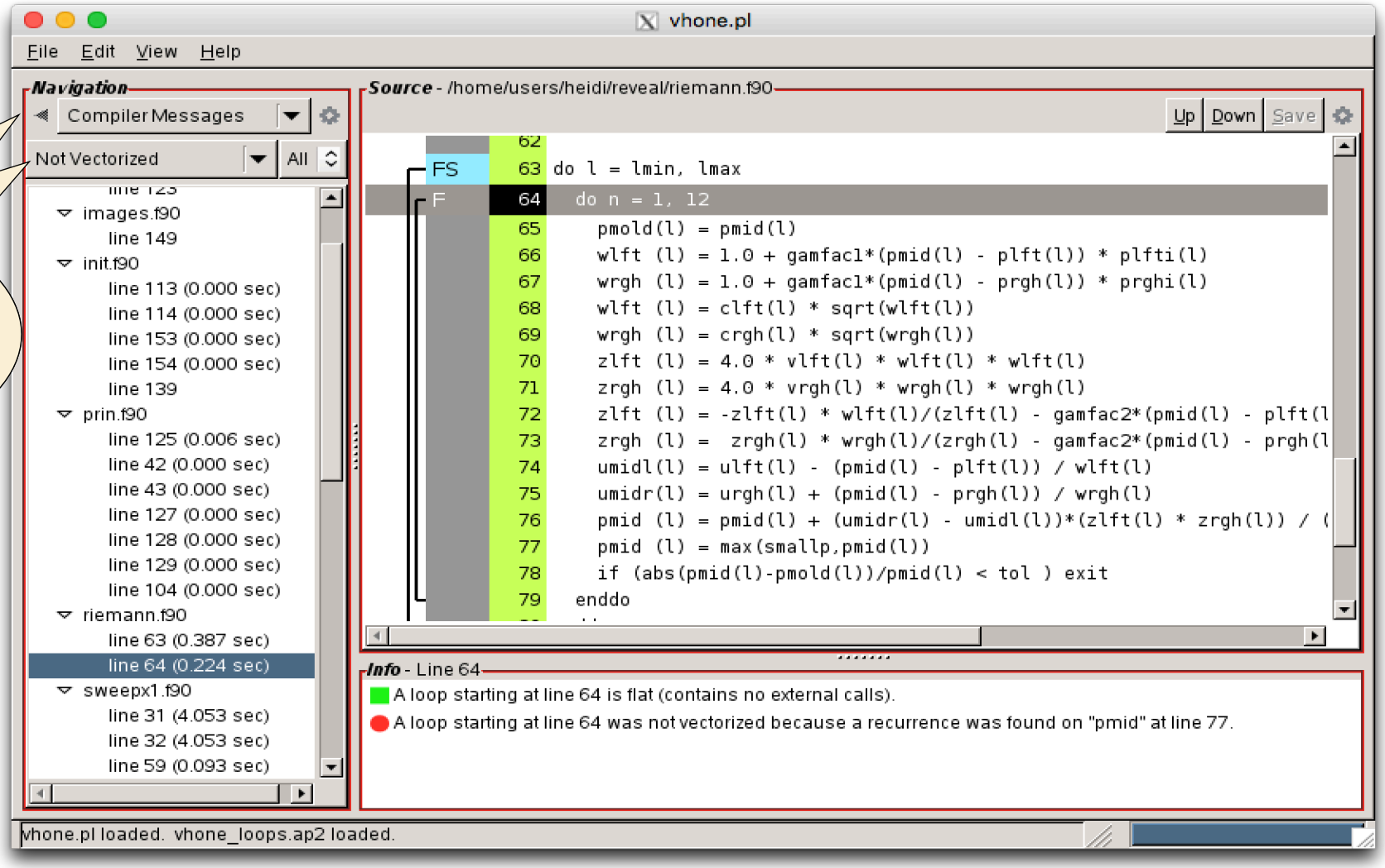
Validate User Inserted Directives



© 2019 Cray, a Hewlett Packard Enterprise company

Look For Vectorization Opportunities

Choose “Compiler Messages” view to access message filtering, then select desired type of message



© 2019 Cray, a Hewlett Packard Enterprise company

Cray Debugging Tools

© 2019 Cray, a Hewlett Packard Enterprise company

Scalable Debugging on Cray Systems

- Cray's focus is to build tools around traditional debuggers with innovative techniques for productivity and scalability
- Scalable Solutions based on MRNet from University of Wisconsin

- **STAT - Stack Trace Analysis Tool**
  - Scalable generation of a single merged stack backtrace for the application
  - GUI based tool (stat-gui/stat-view) along with cli tools (stat-cl)
  - Gain insight into application behavior at a function level

- **ATP - Abnormal Termination Processing**
  - Scalable core file generation and analysis when application crashes
  - Generates a merged stack backtrace akin to stat
  - Selection algorithm to dump unique core files

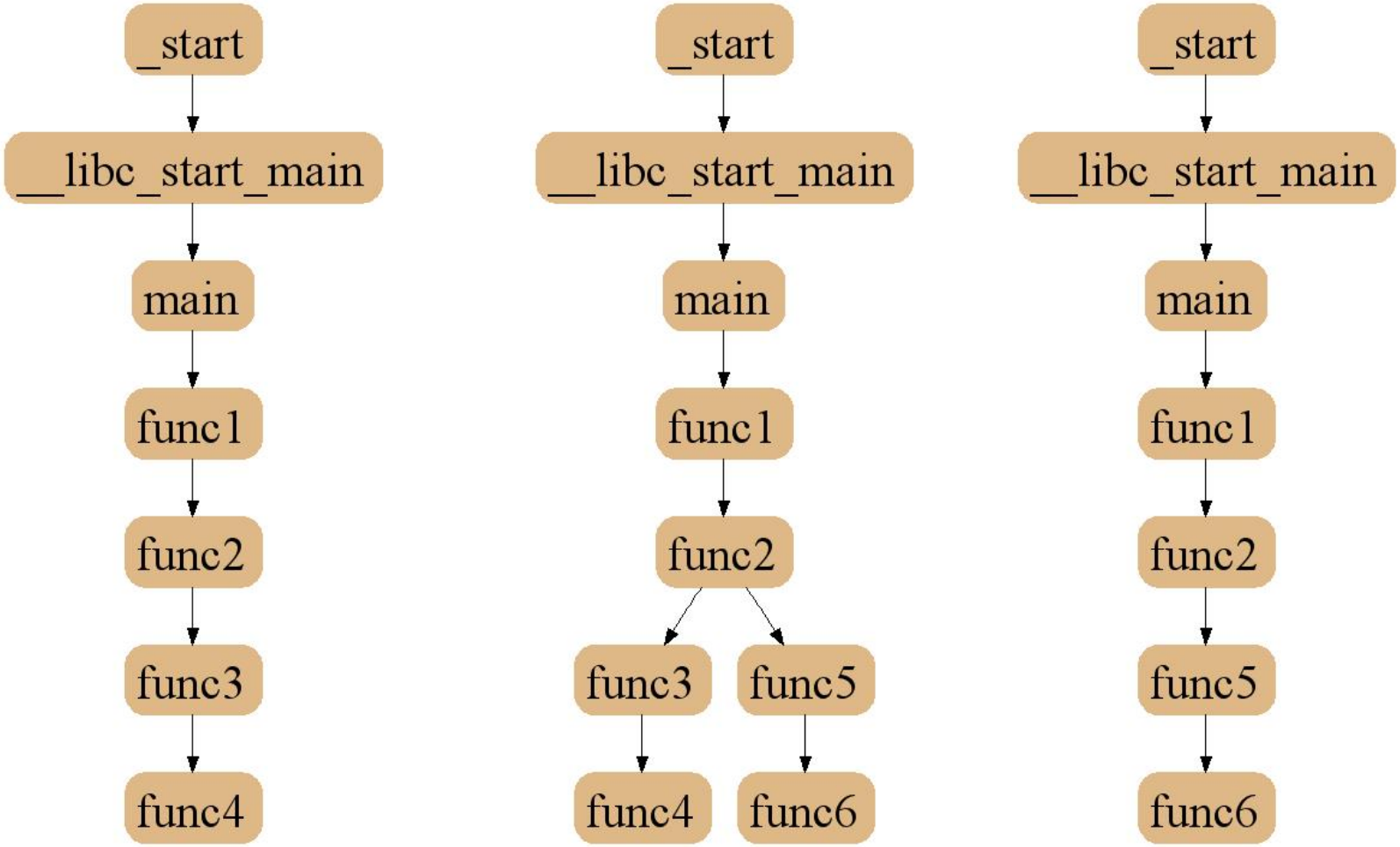
- **gdb4hpc**
  - Conventional CLI based interactive parallel debugger
  - Look and feel of gdb – syntax is inspired by gdb!
  - Debug your application at scale

- **CCDB - Comparative debugging**
  - A data-centric paradigm
  - Compare two applications side-by-side
    - Focus on the data – not state and internal operations
  - GUI tool that interacts with gdb4hpc

- **Valgring4hpc**
  - Parallel valgrind based debugging tool (memcheck)
  - Aids in detection of memory leaks and errors in parallel applications
  - Aggregates like errors across PEs/threads

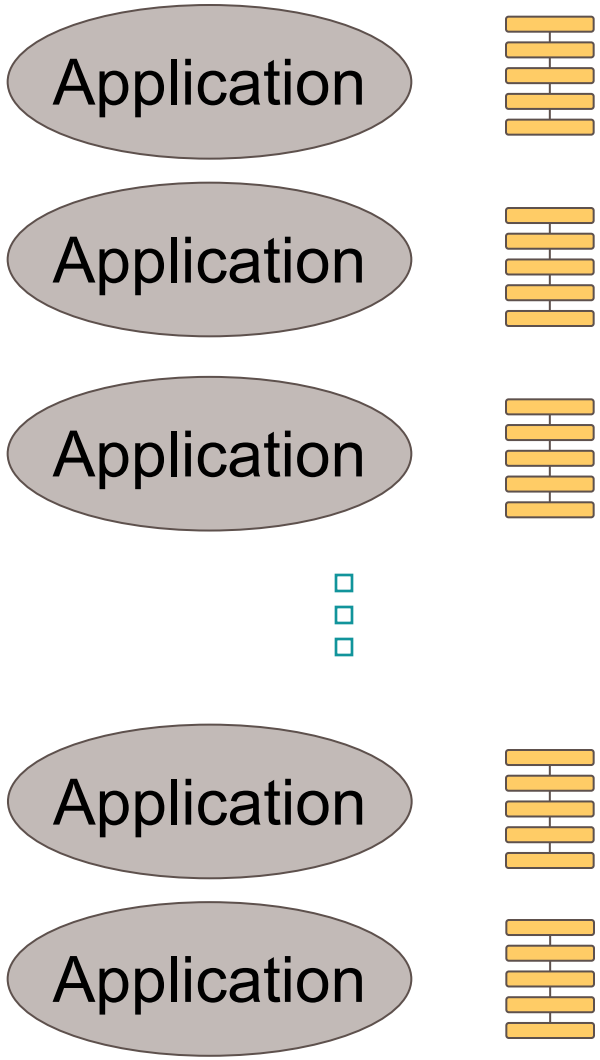
© 2019 Cray, a Hewlett Packard Enterprise company

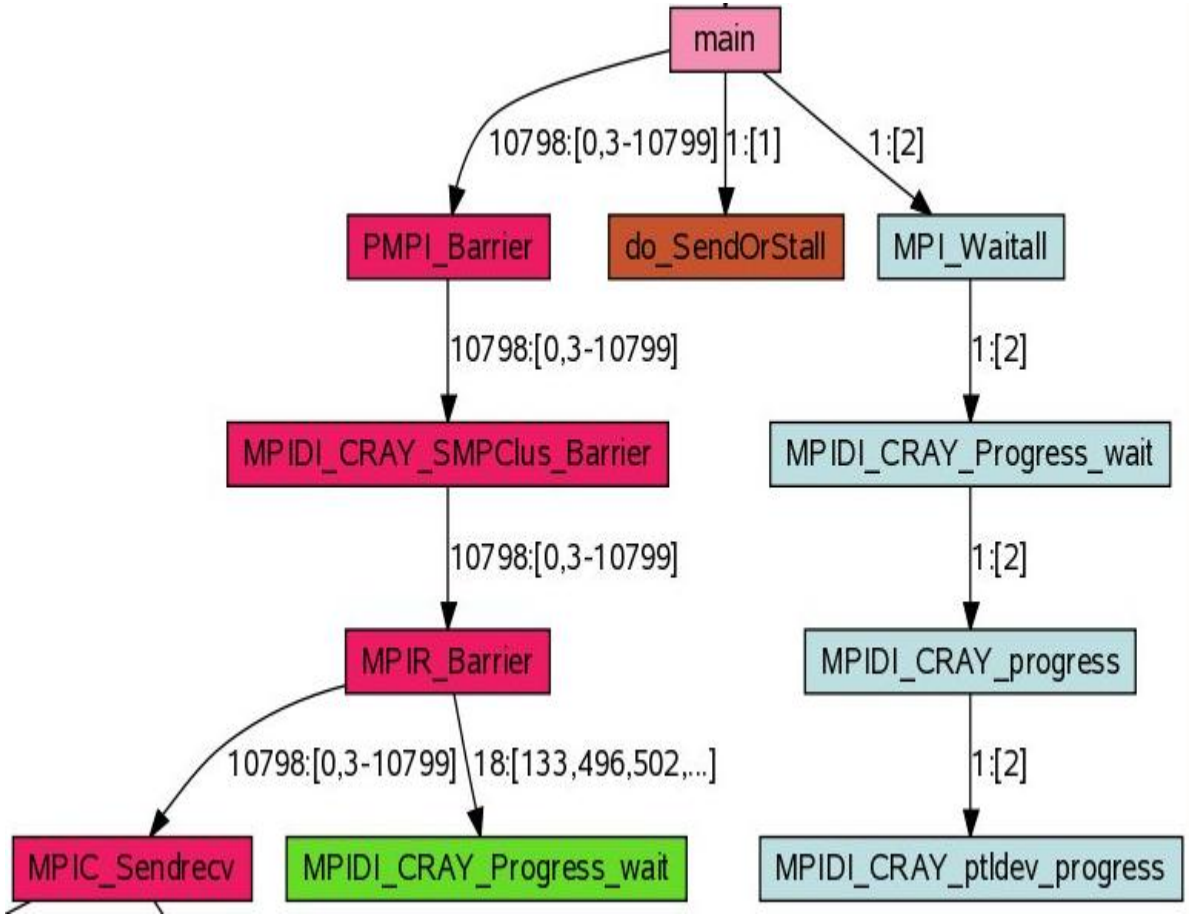
Stack Trace Merge Example



© 2019 Cray, a Hewlett Packard Enterprise company

2D-Trace/Space Analysis





© 2019 Cray, a Hewlett Packard Enterprise company

ATP

- ATP signal handler runs within an application to catch fatal errors
  - It handles the following signals:
    - SIGQUIT, SIGILL, SIGTRAP, SIGABRT, SIGFPE, SIGBUS, SIGSEGV, SIGSYS, SIGXCPU, SIGXFSZ
    - Setting the environment variables MPICH_ABORT_ON_ERROR and SHMEM_ABORT_ON_ERROR will cause a signal to be thrown and captured for MPI and SHMEM fatal errors

- ATP daemon running on the compute node captures signals, starts termination processing
  - All application processes are notified
  - Generates a stacktrace
  - Creates a single merged stack trace file

- The stack trace file is viewed with the stat-view tool

© 2019 Cray, a Hewlett Packard Enterprise company

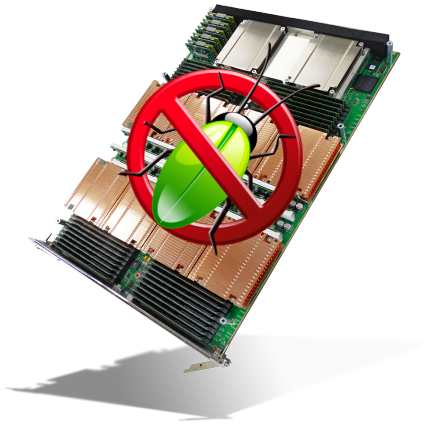
ATP Can Hold Dying Application

- ATP is able to hold a dying application in stasis in order to allow the user to attach to it with a debugger
  - To do so, set the ATP_HOLD_TIME environment variable to the number of minutes desired
- Once attached, the debugging session can last as long as the batch system allows
  - Which in turn depends on the compute node resources you requested when you began your session
  - Use ATP_HOLD_TIME to define the time you need to attach to the application, *not* the total time needed for the debugging session
- If ATP_HOLD_TIME is set, core dumping is disabled

© 2019 Cray, a Hewlett Packard Enterprise company

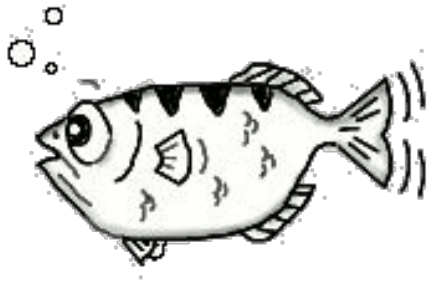
gdb4hpc

- **Why is a tool like this necessary?**
  - Code developers are already familiar with gdb
  - Commercial debuggers are GUI based
  - Using gdb on each individual PE/thread is painful
    - How to attach gdb?
    - How to communicate with gdb?
  - Focus on delivering a tool with scalability in mind



- **Traditional parallel debugger**
  - Compilers: CCE, GNU, Intel, clang/flang
  - Languages: C, C++, Fortran, UPC
  - Programming models: MPI, SHMEM, OpenMP, pthreads
  - Provides a command set similar to gdb

- **Built on top of gdb**
  - Have a gdb instance for each PE/thread
  - Glued together with a communication tree
  - gdb was modified to support Fortran and UPC



© 2019 Cray, a Hewlett Packard Enterprise company

Using gdb4hpc

- `user@login> module load gdb4hpc`
- `user@login> man gdb4hpc`

```
user@login:~> gdb4hpc

gdb4hpc 3.0 - Cray Line Mode Parallel Debugger
With Cray Comparative Debugging Technology.
Copyright 2007-2018 Cray Inc. All Rights Reserved.
Copyright 1996-2016 University of Queensland. All Rights Reserved.

Type "help" for a list of commands.
Type "help <cmd>" for detailed help about a command.
dbg all>
```

- Recommend compiling with `-g -O0` for best experience

© 2019 Cray, a Hewlett Packard Enterprise company

gdb4hpc Example

```
$ gdb4hpc
…

dbg all> break jacobi
App1{0..127}: Breakpoint 1: file himeno.f, line 209.

dbg all> c
App1{0..127}: Breakpoint 1, jacobi at himeno.f:209

dbg all> l
App1{0..127}: 209             subroutine jacobi(nn,gosa)
App1{0..127}: 210       C***********************************************
App1{0..127}: 211             IMPLICIT NONE
App1{0..127}: 212       C
App1{0..127}: 213             include 'mpif.h'
App1{0..127}: 214             include 'param.h'
App1{0..127}: 215       C
App1{0..127}: 216             integer :: nn,i,j,k,loop,ierr

dbg all> backtrace
App1{0..127}: #0  0x0000000000402020 in jacobi at himeno.f:209
App1{0..127}: #1  0x00000000004012de in himenobmtxp at himeno.f:91

dbg all> print npe
App1{0..127}: 128

dbg all> p jmax
App1{0..3,12..19,28..35,44..51,60..67,76..83,92..99,108..115,124..127}: 129
App1{4..11,20..27,36..43,52..59,68..75,84..91,100..107,116..123}: 130
```

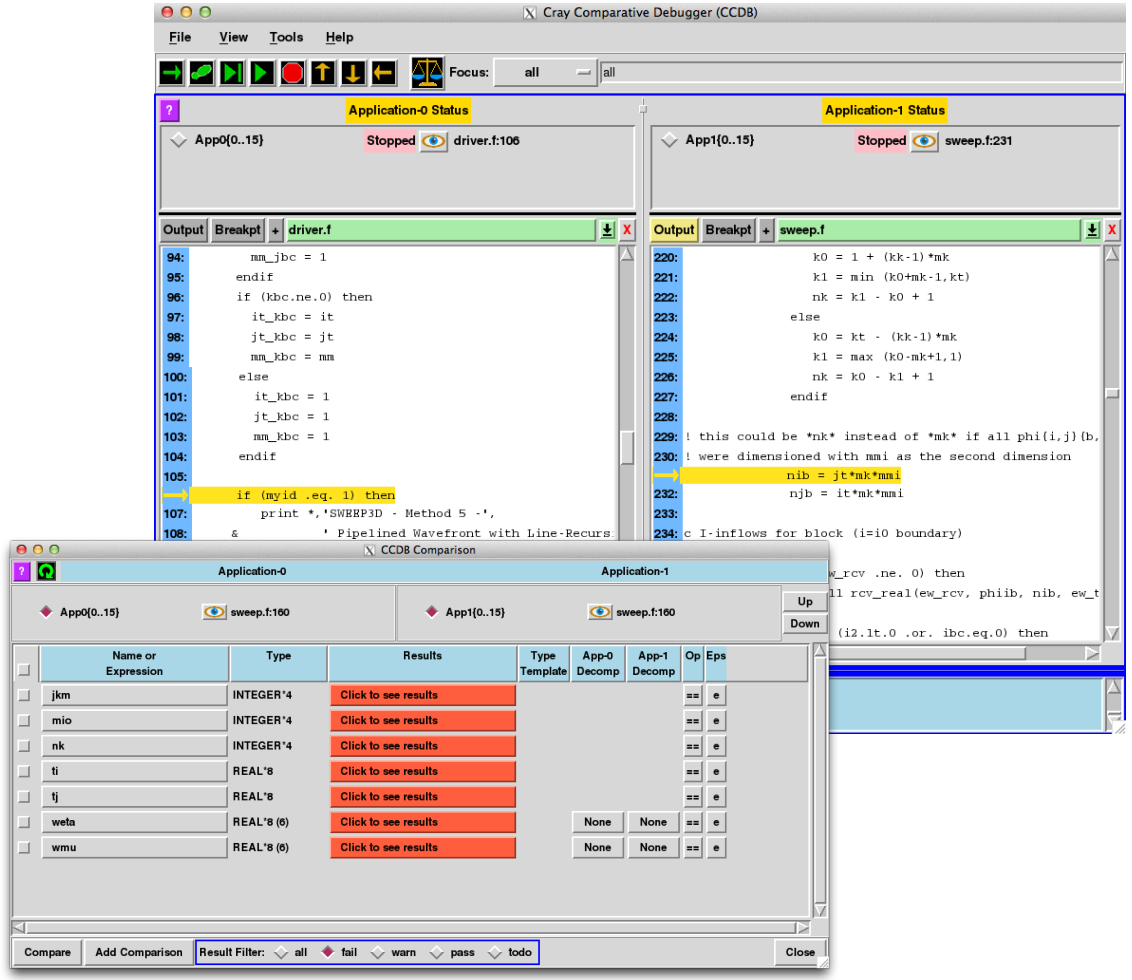
© 2019 Cray, a Hewlett Packard Enterprise company

Comparative Debugger

CRAY
a Hewlett Packard Enterprise company

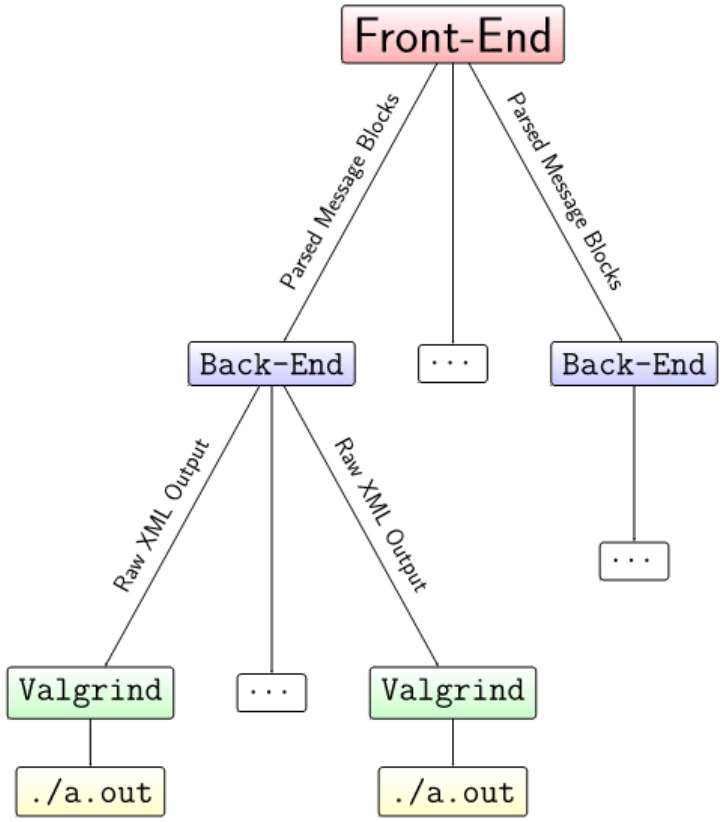
THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

- **What is comparative debugging?**
  - **Data centric** approach instead of the traditional control-centric paradigm
  - Two applications, same data - **simultaneous execution of both**
  - Key idea: The data should match
  - Quickly isolate deviating variables
- **Comparative debugging tool**
  - NOT a traditional debugger!
  - Assists with comparative debugging
    - **Focus on data** – not state and internal operations
    - Creates automatic comparisons
    - Based on symbol name and type
    - Allows user to create own comparisons
    - **Error and warning epsilon tolerance**
    - **Scalable**
- **How does this help me?**
  - Algorithm re-writes
  - Language ports
  - Different libraries/compilers
  - New architectures



© 2019 Cray, a Hewlett Packard Enterprise company

Valgrind4hpc



- Scalable Valgrind tool for analyzing parallel applications
- Focus: parallelizing memcheck and helgrind tools
- Memory leak/error checking, thread safety checking, bounds checking
- Performs analysis at every application rank and removes duplicates matching output blocks to provide concise summaries of program behavior
- Available via module
  - `user@login> module load valgrind4hpc`

© 2019 Cray, a Hewlett Packard Enterprise company

Cray PE DL Scalability Plugin

© 2019 Cray, a Hewlett Packard Enterprise company

The Cray PE DL Scalability Plugin Overview

- Goal was to design a solution for scaling TensorFlow (specifically synchronous SGD) to significantly larger node counts than existing methods allowed
  - Should require **minimal changes to user training scripts** and provide a **more friendly user experience**
- Achieve the **best possible TensorFlow performance on Cray Systems**
- **Maintain accuracy** for a given number of steps and hyper-parameter setup allowing for significantly reduced time-to-accuracy through scaling
- Can run on a range of x86-64 CPUs and currently supports Nvidia GPUs
- Ideally have a **portable solution** that would work with other deep learning frameworks

© 2019 Cray, a Hewlett Packard Enterprise company

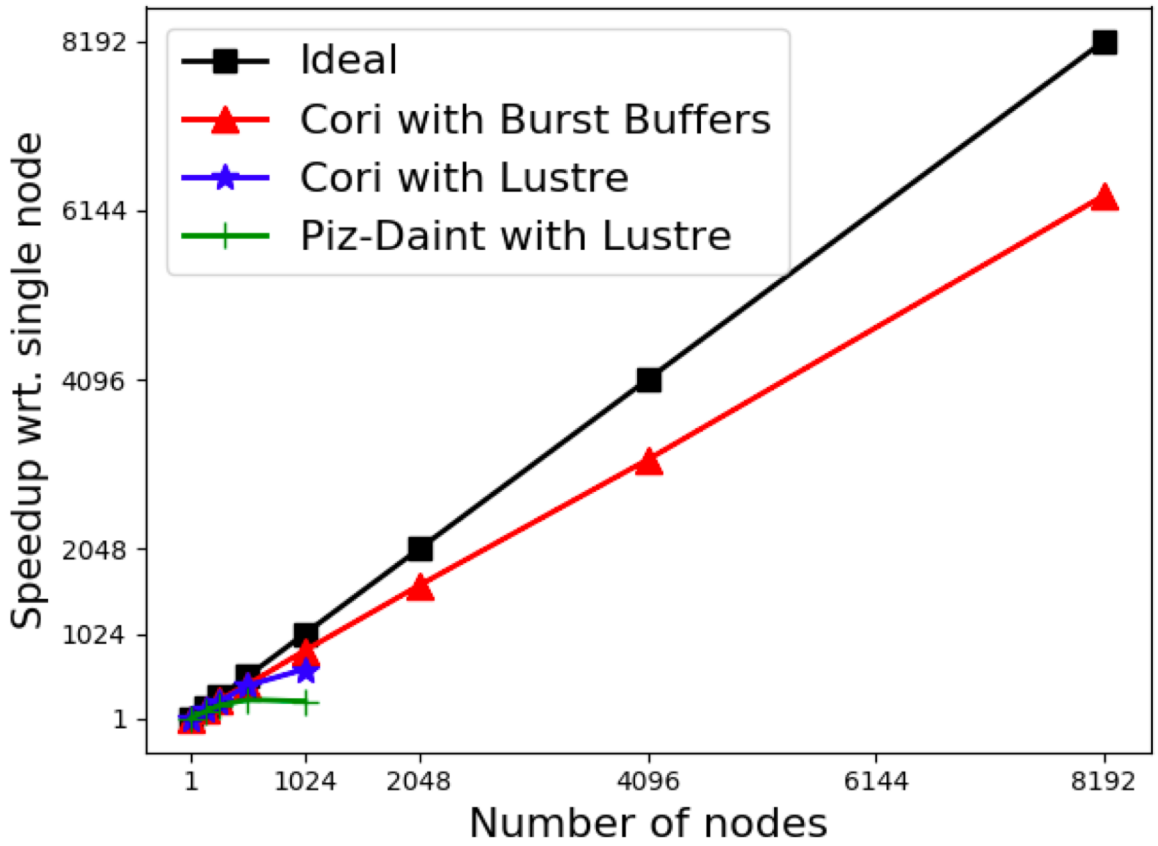
Using the Plugin

- Available on Cray XC and CS systems
  - `user@login> module load craype-dl-plugin-py3`
- Plugin has both a C and Python 3 API and supports multiple DL datatypes
- Can be used on single and multi-GPU nodes of various topologies
- Can be used with popular DL frameworks or integrated into a project via its API
- Compatible with TensorFlow and PyTorch frameworks

© 2019 Cray, a Hewlett Packard Enterprise company

CosmoFlow Scaling Performance

- **Achieved 77% scaling efficiency at 8192 nodes on Cori**
  - Fully synchronous SGD
  - Speedup of 6324X
- Measure walltime per epoch (throughput)
  - Captures end-to-end capability including:
    - Single-node computation
      - Training and validation
    - Communication
    - I/O



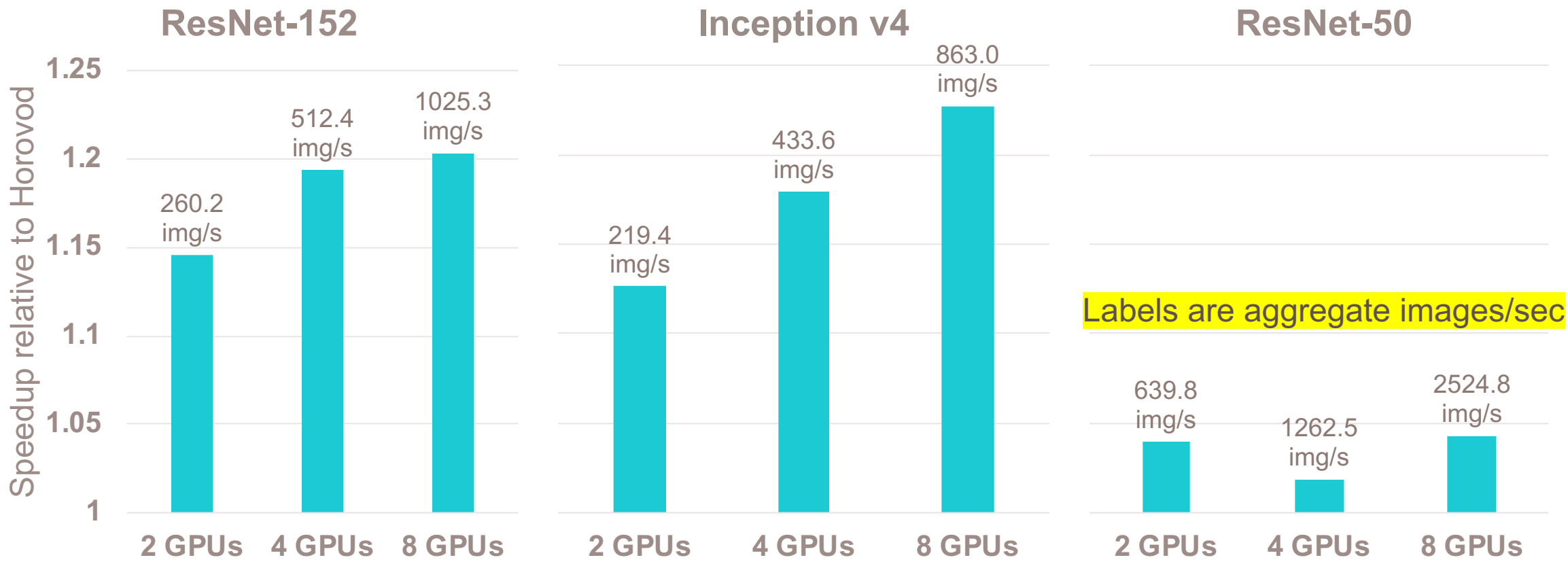
Note: global batch size = # nodes (local batch size of 1)

© 2019 Cray, a Hewlett Packard Enterprise company

Cray DL Plugin Perf. vs. Horovod on CS-Storm

Higher on the y-axis is better

Speedup = 1 is equivalent perf.



* TensorFlow 1.13.1 with tf_cnn_benchmarks, minibatch size = 32, synthetic data. Horovod 0.16.

© 2019 Cray, a Hewlett Packard Enterprise company

QUESTIONS?