Cray GPU Programming Environment Update

Seiji Nishimura

Cray Japan Inc.

Major Cray Hybrid Multi Petaflop Systems

Blue Waters:

Sustained Petascale Performance

- Production Science at Full Scale
- 244 XE Cabinets + 44 XK Cabinets
  - > 25K compute nodes
- 13.3 Petaflops (7.1 CPU + 6.2 GPU)
- 1.5 Petabytes of total memory
- 25 Petabytes Storage
  - 1 TB/sec IO
- Cray's scalable Linux Environment
- HPC-focused GPU/CPU Programming Environment

Titan:

A "Jaguar-Size" System with GPUs

- 200 cabinets
- 18,688 compute nodes
- 25x32x24 3D torus (22.5 TB/s global BW)
- 128 I/O blades (512 PCIe-2 @ 16 GB/s bidir)
- 1,278 TB of memory
- 4,352 sq. ft.
- 10 MW

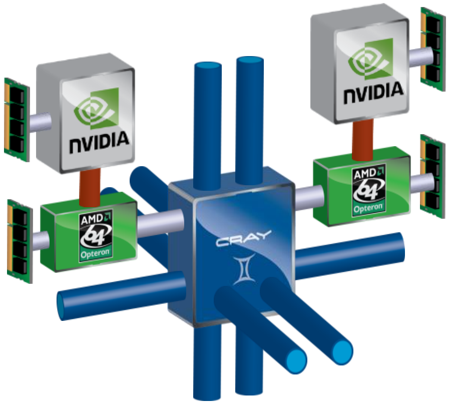
Piz Daint:

Top Supercomputer in Europe

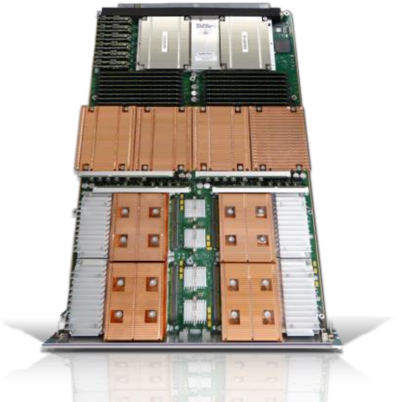
- Cray XC30
  - Aries routing
- 5272 Compute Nodes
  - one Intel® Xeon® E5-2670 and one NVIDIA® Tesla® K20X)
- 7.787 Petaflops
- 32 GB per node
- 169 TB DDR3
  - 32 TB non-ECC GDDR5
- 2.5 Petabytes Storage

The Cray Hybrid Architecture

- **CPU and Interconnect**
  - XC30:
    - Intel SandyBridge or IvyBridge
    - Cray Aries interconnect
  - XK7:
    - AMD Interlagos
    - Cray Gemini interconnect



- **NVIDIA GPUs**
  - Kepler (K20/K20X) GPUs
  - Atlas (K40) GPUs



- **Unified X86/GPU programming environment**

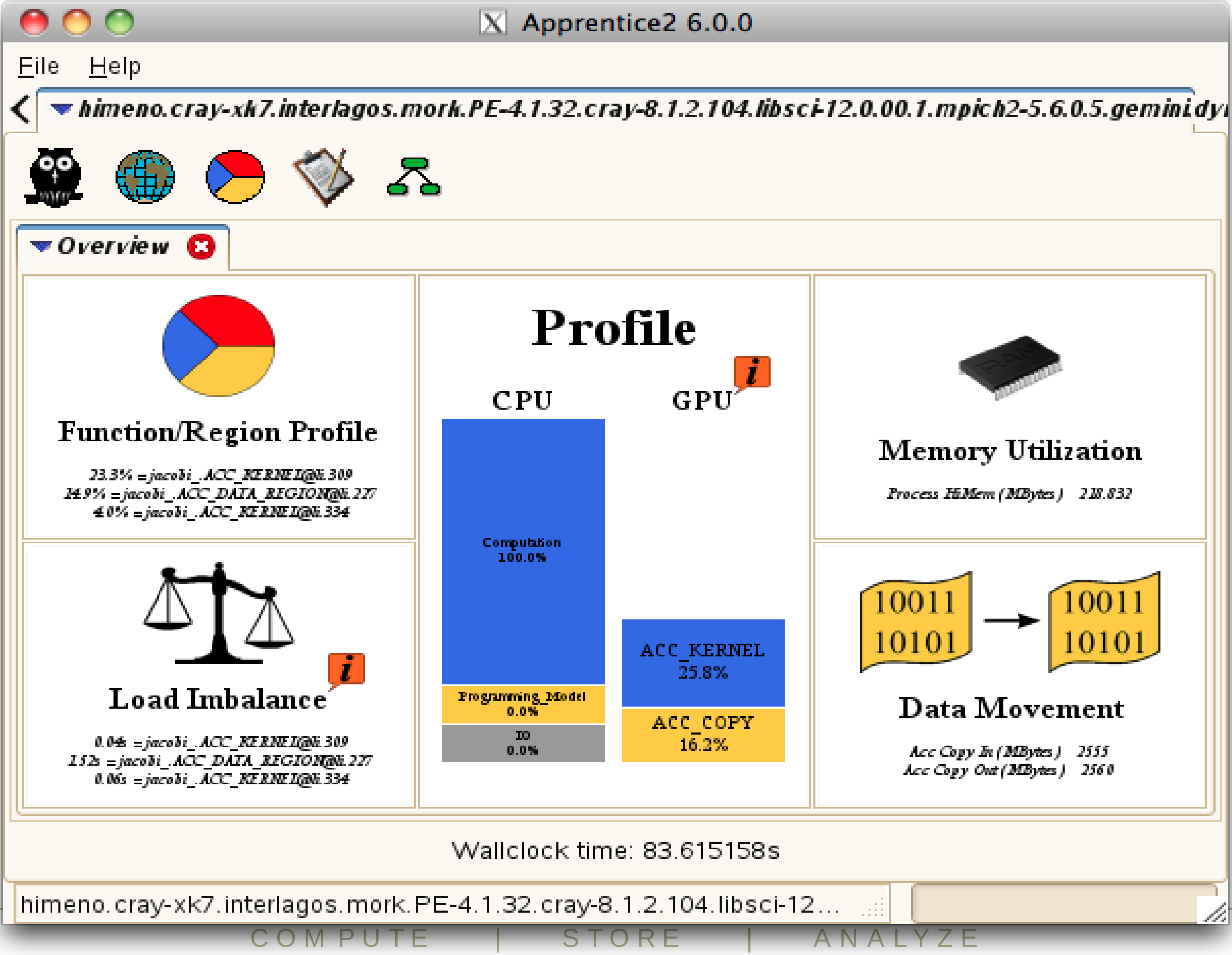
- **Fully compatible with Cray homogeneous XE/XC product line**

The Cray Compilation Environment

- **Cray technology focused on scientific applications**
  - Takes advantage of **automatic vectorization**
  - Takes advantage of **automatic shared memory parallelization**

- **OpenACC 2.0 compliant**
  - Compiles to PTX not CUDA
  - Single object file
  - CCE Identifies parallel loops within code regions
  - Splits the code into accelerator and host portions
  - Workshares loops running on accelerator
    - Make use of MIMD and SIMD style parallelism
  - Data movement
    - allocates/frees GPU memory at start/end of region
    - moves data to/from GPU
  - Debuggers see original program not CUDA intermediate

Cray Apprentice2 Overview with GPU Data



himeno.cray-xk7.interlagos.mork.PE-4.1.32.cray-8.1.2.104.libsci-12.0.00.1.mpich2-5.6.0.5.gemini.dy

Function/Region Profile

23.3% =jacobi_.ACC_KERNEL@li.309
14.9% =jacobi_.ACC_DATA_REGION@li.227
4.0% =jacobi_.ACC_KERNEL@li.334

Load Imbalance

0.04s =jacobi_.ACC_KERNEL@li.309
1.52s =jacobi_.ACC_DATA_REGION@li.227
0.06s =jacobi_.ACC_KERNEL@li.334

Profile

CPU: Computation 100.0%; Programming_Model 0.0%; IO 0.0%

GPU: ACC_KERNEL 25.8%; ACC_COPY 16.2%

Memory Utilization

Process HiMem (MBytes) 218.832

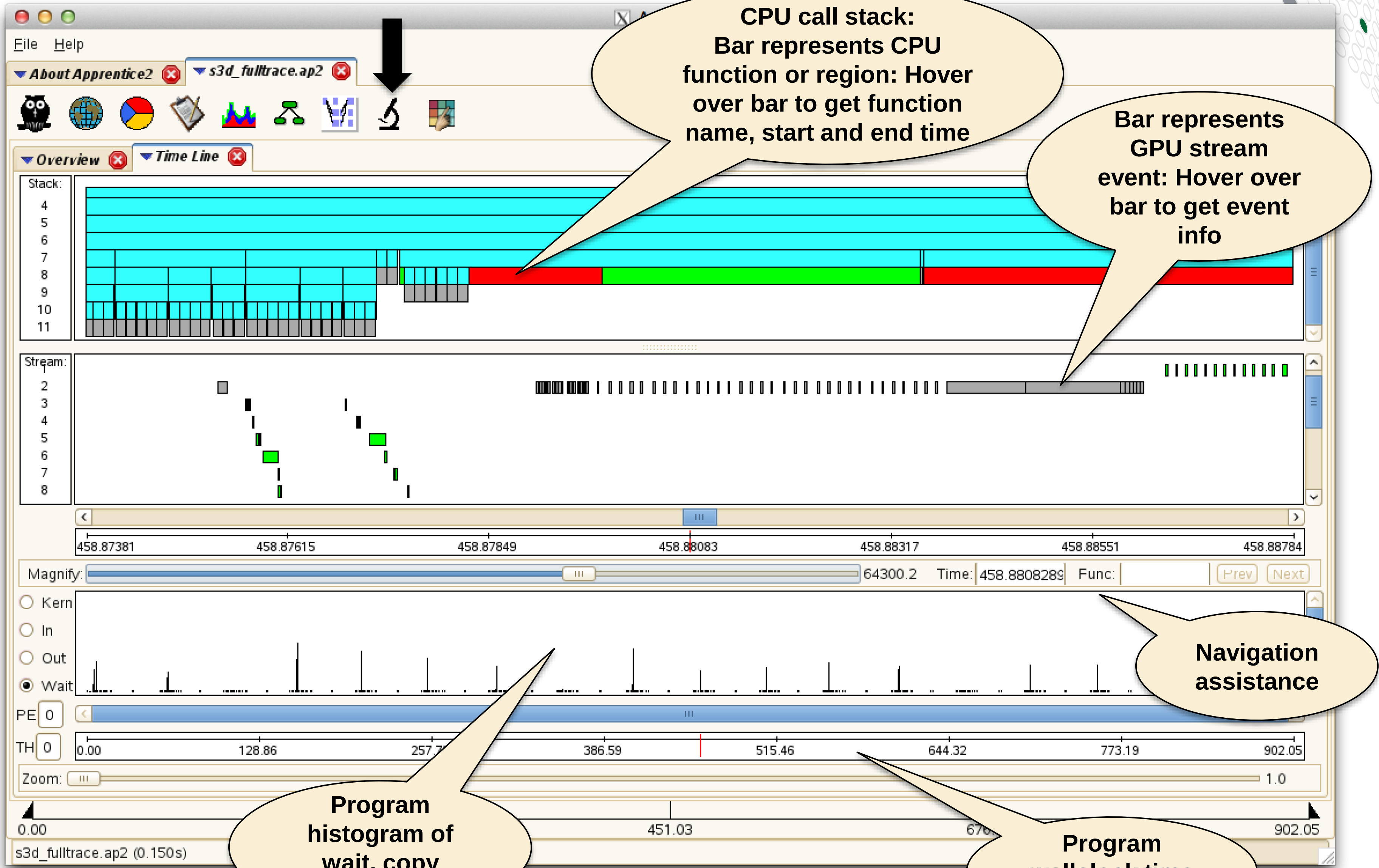
Data Movement

Acc Copy In (MBytes) 2555
Acc Copy Out (MBytes) 2560

Wallclock time: 83.615158s

himeno.cray-xk7.interlagos.mork.PE-4.1.32.cray-8.1.2.104.libsci-12...

GPU Program Timeline



Magnify: 64300.2 Time: 458.8808289 Func:

PE 0

TH 0

Zoom: 1.0

s3d_fulltrace.ap2 (0.150s)

CPU call stack: Bar represents CPU function or region: Hover over bar to get function name, start and end time

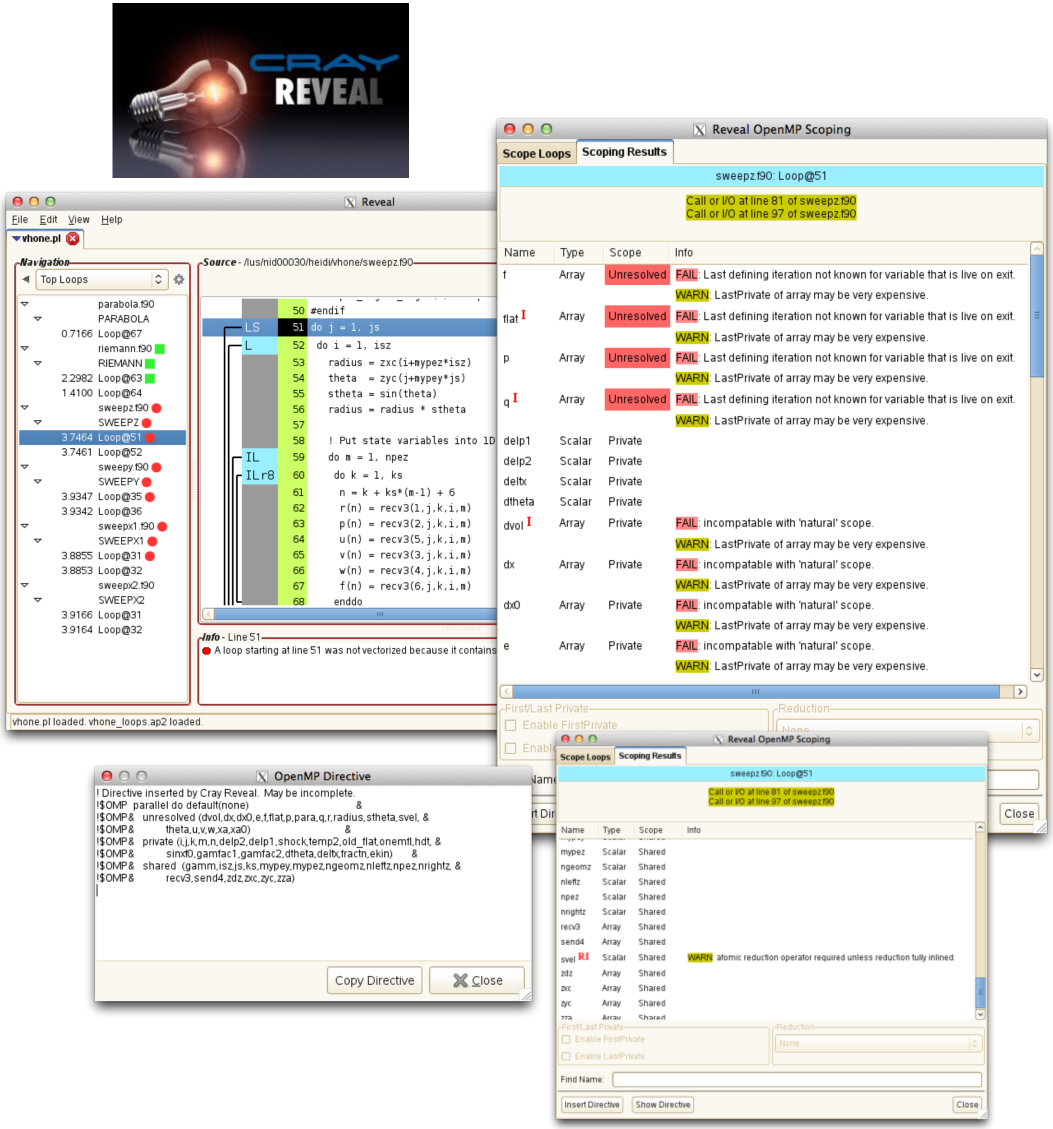
Bar represents GPU stream event: Hover over bar to get event info

Navigation assistance

Program histogram of wait, copy kernel time

Program wallclock time line

Simplifying the Task with Reveal



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

```
50 #endif
51 do j = 1, js
52  do i = 1, isz
53    radius = zxc(i+mypez*isz)
54    theta  = zyc(j+mypey*js)
55    stheta = sin(theta)
56    radius = radius * stheta
57
58    ! Put state variables into 1D
59    do m = 1, npez
60     do k = 1, ks
61      n = k + ks*(m-1) + 6
62      r(n) = recv3(1,j,k,i,m)
63      p(n) = recv3(2,j,k,i,m)
64      u(n) = recv3(5,j,k,i,m)
65      v(n) = recv3(3,j,k,i,m)
66      w(n) = recv3(4,j,k,i,m)
67      f(n) = recv3(6,j,k,i,m)
68     enddo
```

Info - Line 51

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

vhone.pl loaded. vhone_loops.ap2 loaded.

sweepz.f90: Loop@51

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

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

OpenMP Directive

```
! Directive inserted by Cray Reveal.  May be incomplete.
!$OMP  parallel do default(none)                                  &
!$OMP&   unresolved (dvol,dx,dx0,e,f,flat,p,para,q,r,radius,stheta,svel, &
!$OMP&          theta,u,v,w,xa,xa0)                               &
!$OMP&   private (i,j,k,m,n,delp2,delp1,shock,temp2,old_flat,onemfl,hdt, &
!$OMP&          sinxf0,gamfac1,gamfac2,dtheta,deltx,fractn,ekin)      &
!$OMP&   shared  (gamm,isz,js,ks,mypey,mypez,ngeomz,nleftz,npez,nrightz, &
!$OMP&          recv3,send4,zdz,zxc,zyc,zza)
```

sweepz.f90: Loop@51

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

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

- Navigate to relevant loops to parallelize
- Identify parallelization and scoping issues
- Get feedback on issues down the call chain (shared reductions, etc.)
- Optionally insert parallel directives into source
- Validate scoping correctness on existing directives

Summary

- **Cray provides a high level programming environment for acceletate Computing**
  - Fortran, C, and C++ compilers
    - **OpenACC directives to drive compiler optimization**
    - Compiler optimizations to take advantage of accelerator and multi-core X86 hardware appropriately

  - Cray **Reveal**
    - **Scoping analysis** tool to assist user in understanding their code and taking full advantage of SW and HW system

  - **Cray Performance Measurement and Analysis toolkit**
    - Single tool for GPU and CPU performance analysis with statistics for the whole application
  - **Parallel Debugger support** with allinea www.allinea.com DDT or TotalView
  - Auto-tuned Scientific Libraries support
    - Getting performance from the system … **no assembly required**

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Cray Inc. Overview

- **35+ year legacy focused on building the worlds fastest computer.**
- **1000+ employees world wide**
  - Growing in a tough economy
- **Cray XT6 first computer to deliver a PetaFLOP/s in a production environment**
  - Jaguar system at Oakridge
- **A full range of products**

OpenACC Accelerator Programming Model

- **Why a new model? There are already many ways to program:**
  - CUDA and OpenCL
    - All are quite low-level and closely coupled to the GPU
    - PGI CUDA Fortran: still CUDA just in a better base language
    - User needs to write specialized kernels:
      - **Hard** to write and debug
      - **Hard** to optimize for specific GPU
      - **Hard** to update (porting/functionality)
- **OpenACC Directives provide high-level approach**
  - **Simple programming model for hybrid systems**
  - **Easier to maintain/port/extend code**
    - Non-executable statements (comments, pragmas)
    - The **same source** code can be compiled for multicore CPU
  - Based on the work in the OpenMP Accelerator Subcommittee
  - PGI accelerator directives, CAPS HMPP
    - First steps in the right direction – Needed standardization
  - **Possible performance sacrifice**
    - A small performance gap is acceptable (do you still hand-code in assembly?)
    - Goal is to provide at least 80% of the performance obtained with hand coded CUDA
- **Compiler support: all complete in 2012**
  - Cray CCE, PGI, CAPS

The OpenACC™ API
QUICK REFERENCE GUIDE

The OpenACC Application Program Interface describes a collection of compiler directives to specify loops and regions of code in standard C, C++ and Fortran to be offloaded from a host CPU to an attached accelerator, providing portability across operating systems, host CPUs and accelerators.

Most OpenACC directives apply to the immediately following structured block or loop; a structured block is a single statement or a compound statement (C or C++) or a sequence of statements (Fortran) with a single entry point at the top and a single exit at the bottom.

CAPS

CRAY
THE SUPERCOMPUTER COMPANY

NVIDIA.

PGI

Version 1.0, November 2011

All four companies intend to work within OpenMP organization to merge OpenACC and create a common specification that extends OpenMP to support accelerators.

© 2011 OpenACC-standard.org all rights reserved.