Leaders in parallel software development tools

Accelerated Debugging

Allinea DDT and OpenACC on the XK6

David Lecomber

CTO, Allinea Software

david@allinea.com

Some history

- A long time ago (2007) in a galaxy far, far away....
  - The CUDA programming model is introduced
    - Powerful, efficient and C based
    - Understood and adopted by new groups of experts
    - Existing codes modified to extract SIMD parallelism and introduce CUDA kernels
    - Performance of codes is optimized by overlapping device and host, or rearranging memory usage inside device
  - The first CUDA bug is created

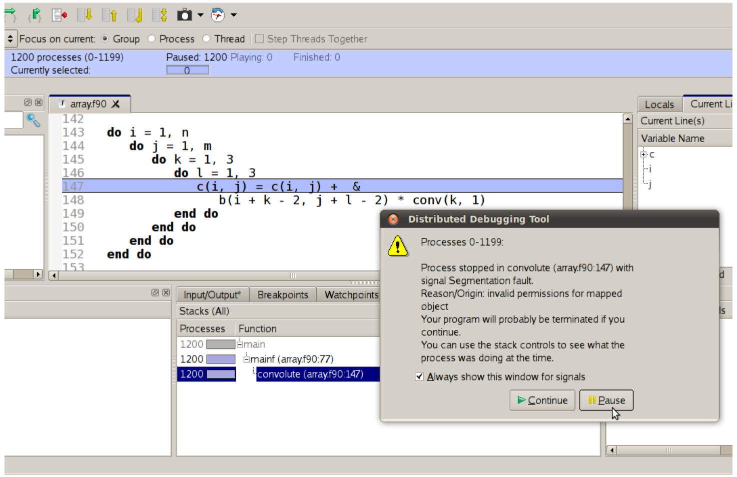
Allinea Software

- HPC development tools company
  - Flagship product Allinea DDT
    - The leading debugger in parallel computing
    - The scalable debugger
      - Record holder for debugging software on largest machines
      - Production use at extreme scale – and desktop
    - Wide customer base
      - Blue-chip engineering, government and academic research
      - Strong collaborative relationships with customers and partners
  - Leaders in performance and usability

allinea
www.allinea.com

Allinea DDT in a nutshell

- Graphical source level debugger for
  - Parallel, multi-threaded, scalar or hybrid code
  - C, C++, F90, Co-Array Fortran, UPC
- Strong feature set
  - Memory debugging
  - Data analysis
- Managing concurrency
  - Emphasizing differences
  - Collective control



1200 processes (0-1199) Paused: 1200 Playing: 0 Finished: 0
Currently selected: 0

```
142
143   do i = 1, n
144      do j = 1, m
145         do k = 1, 3
146            do l = 1, 3
147               c(i, j) = c(i, j) +  &
148                  b(i + k - 2, j + l - 2) * conv(k, 1)
149            end do
150         end do
151      end do
152   end do
153
```

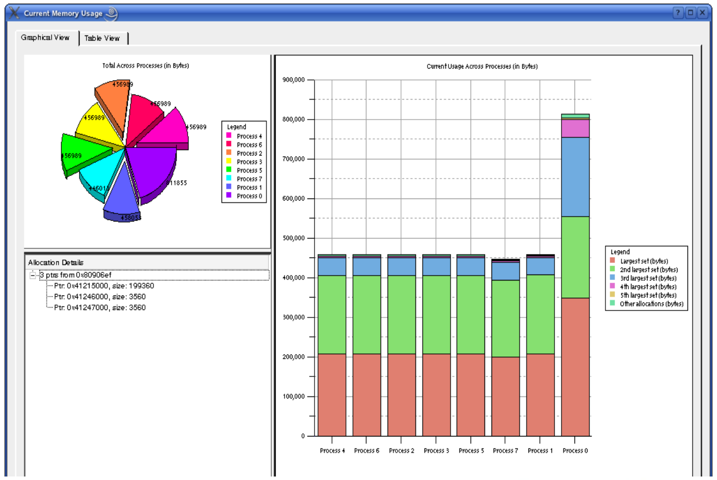
Processes	Function
1200	main
1200	mainf (array.f90:77)
1200	convolute (array.f90:147)

Distributed Debugging Tool

Processes 0-1199:

Process stopped in convolute (array.f90:147) with signal Segmentation fault.
Reason/Origin: invalid permissions for mapped object
Your program will probably be terminated if you continue.
You can use the stack controls to see what the process was doing at the time.

[✓] Always show this window for signals

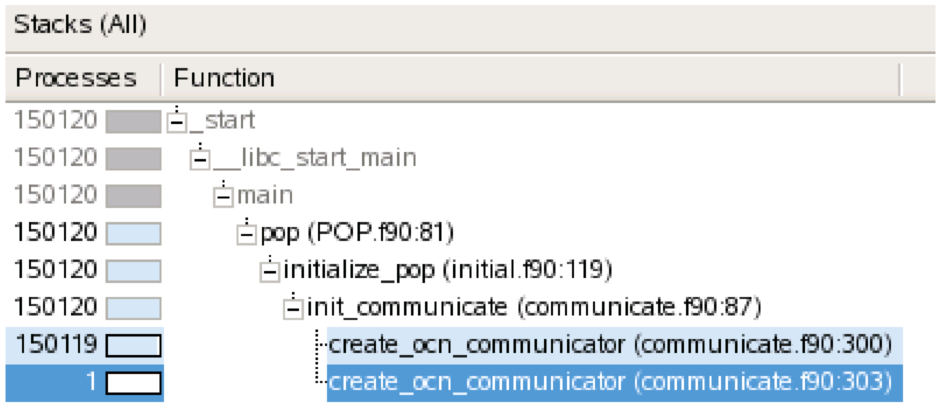


Allocation Details
- 3 ptrs from 0x80906ef
  - Ptr: 0x41215000, size: 199360
  - Ptr: 0x41246000, size: 3560
  - Ptr: 0x41247000, size: 3560

www.allinea.com

May 2011 - CUG 2011

- Petascale debugging becomes real
- Allinea DDT 3.0 – lightning speed – 100,000 cores and beyond
  - Record holder for largest machines
  - Debugging at scale becomes fast, simple and **routine**
  - Production use at extreme scale on Cray XE and XT systems



Stacks (All)

Processes	Function
150120	_start
150120	__libc_start_main
150120	main
150120	pop (POP.f90:81)
150120	initialize_pop (initial.f90:119)
150120	init_communicate (communicate.f90:87)
150119	create_ocn_communicator (communicate.f90:300)
1	create_ocn_communicator (communicate.f90:303)

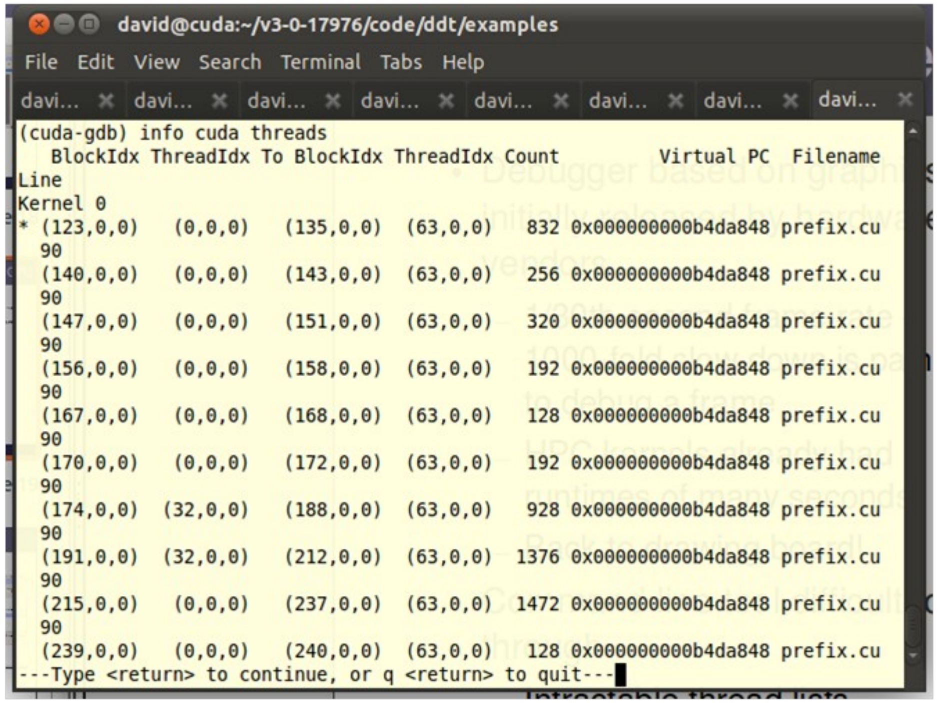
www.allinea.com

Lift-off – beyond petascale

- Fairbanks, May 2011 – Cray XK6 announced
  - Large GPU systems firmly on the agenda
  - Allinea and ORNL collaborate to ensure GPU applications debuggable at scale
    - Petascale debugging, but with GPUs
    - Core needs identified and key features and performance specified
  - How would the XK6 be programmed?
    - Candidate pragma languages to remove CUDA burden

How do we fix GPU bugs?

- Print statements
  - Too intrusive
- Command line debugger?
  - A good start:
    - Variables, source code
    - Large thread counts overwhelming
  - Too complex
- A graphical debugger...

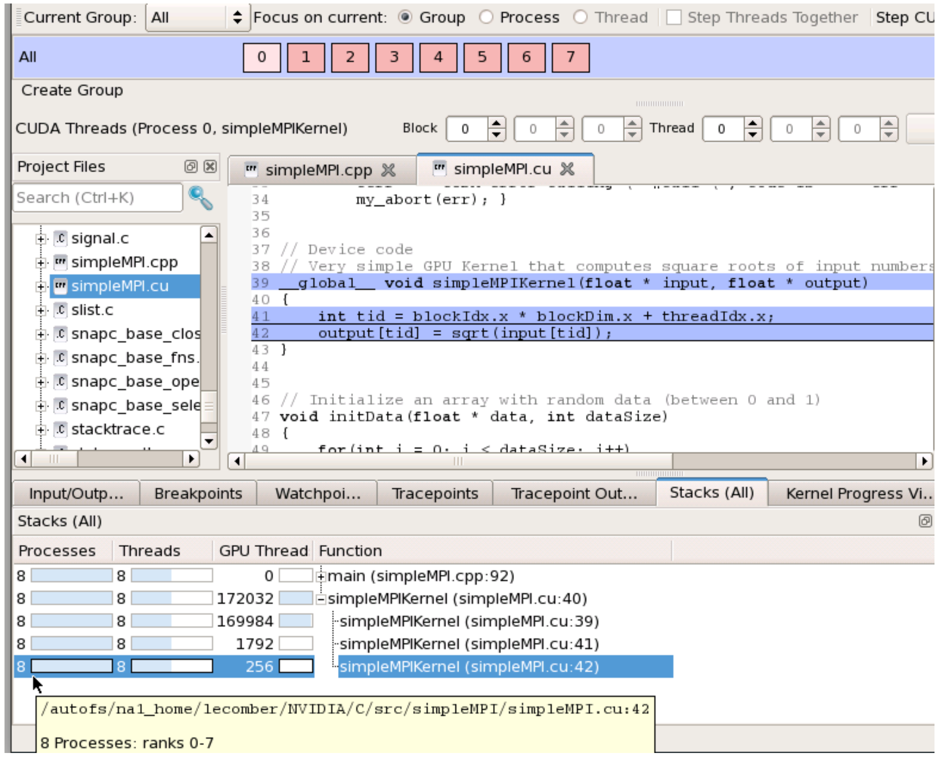

david@cuda:~/v3-0-17976/code/ddt/examples
File Edit View Search Terminal Tabs Help
davi... davi... davi... davi... davi... davi... davi... davi...

```
(cuda-gdb) info cuda threads
  BlockIdx ThreadIdx To BlockIdx ThreadIdx Count         Virtual PC  Filename
Line
Kernel 0
* (123,0,0)   (0,0,0)   (135,0,0)  (63,0,0)   832 0x000000000b4da848 prefix.cu
  90
  (140,0,0)   (0,0,0)   (143,0,0)  (63,0,0)   256 0x000000000b4da848 prefix.cu
  90
  (147,0,0)   (0,0,0)   (151,0,0)  (63,0,0)   320 0x000000000b4da848 prefix.cu
  90
  (156,0,0)   (0,0,0)   (158,0,0)  (63,0,0)   192 0x000000000b4da848 prefix.cu
  90
  (167,0,0)   (0,0,0)   (168,0,0)  (63,0,0)   128 0x000000000b4da848 prefix.cu
  90
  (170,0,0)   (0,0,0)   (172,0,0)  (63,0,0)   192 0x000000000b4da848 prefix.cu
  90
  (174,0,0)  (32,0,0)   (188,0,0)  (63,0,0)   928 0x000000000b4da848 prefix.cu
  90
  (191,0,0)  (32,0,0)   (212,0,0)  (63,0,0)  1376 0x000000000b4da848 prefix.cu
  90
  (215,0,0)   (0,0,0)   (237,0,0)  (63,0,0)  1472 0x000000000b4da848 prefix.cu
  90
  (239,0,0)   (0,0,0)   (240,0,0)  (63,0,0)   128 0x000000000b4da848 prefix.cu
---Type <return> to continue, or q <return> to quit---
```

allinea
www.allinea.com

GPU debugging with Allinea DDT

- Almost like debugging a CPU – we can still:
  - Run through to a crash
  - Step through and observe
- CPU-like debugging features
  - Double click to set breakpoints
  - Hover the mouse for more information
  - Step a warp, block or kernel
  - Follow threads through the kernel
- Simultaneously debugs CPU code
- CUDA Memcheck feature detects read/write errors



Current Group: All | Focus on current: ● Group ○ Process ○ Thread ☐ Step Threads Together | Step CU

All 0 1 2 3 4 5 6 7

Create Group

CUDA Threads (Process 0, simpleMPIKernel) Block 0 0 0 Thread 0 0 0

Project Files: signal.c, simpleMPI.cpp, simpleMPI.cu, slist.c, snapc_base_clos, snapc_base_fns., snapc_base_ope, snapc_base_sele, stacktrace.c

simpleMPI.cpp | simpleMPI.cu

```
        my_abort(err); }


// Device code
// Very simple GPU Kernel that computes square roots of input numbers
__global__ void simpleMPIKernel(float * input, float * output)
{
    int tid = blockIdx.x * blockDim.x + threadIdx.x;
    output[tid] = sqrt(input[tid]);
}


// Initialize an array with random data (between 0 and 1)
void initData(float * data, int dataSize)
{
    for(int i = 0; i < dataSize; i++)
```

Input/Outp... | Breakpoints | Watchpoi... | Tracepoints | Tracepoint Out... | Stacks (All) | Kernel Progress Vi..

Stacks (All)

Processes	Threads	GPU Thread	Function
8	8	0	main (simpleMPI.cpp:92)
8	8	172032	simpleMPIKernel (simpleMPI.cu:40)
8	8	169984	simpleMPIKernel (simpleMPI.cu:39)
8	8	1792	simpleMPIKernel (simpleMPI.cu:41)
8	8	256	simpleMPIKernel (simpleMPI.cu:42)

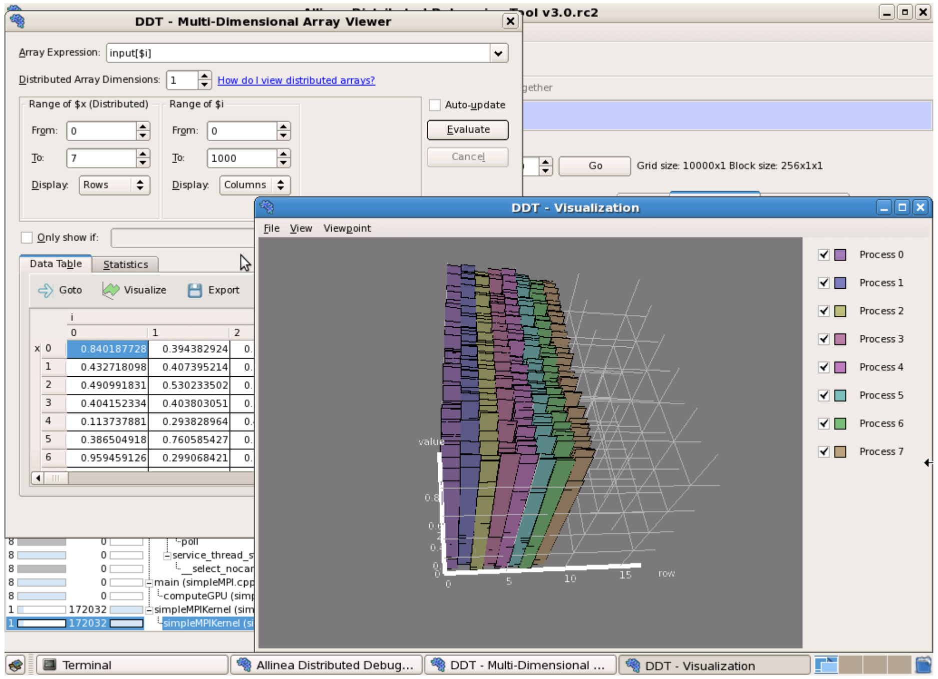
/autofs/na1_home/lecomber/NVIDIA/C/src/simpleMPI/simpleMPI.cu:42

8 Processes: ranks 0-7

allinea
www.allinea.com

Examining GPU data

- Debugger reads host and device memory
  - Shows all memory classes: shared, constant, local, global, register..
  - Able to examine variables
  - … or plot larger arrays directly from device memory



DDT - Multi-Dimensional Array Viewer

Array Expression: input[$i]

Distributed Array Dimensions: 1 How do I view distributed arrays?

Range of $x (Distributed) — From: 0 To: 7 Display: Rows

Range of $i — From: 0 To: 1000 Display: Columns

Auto-update

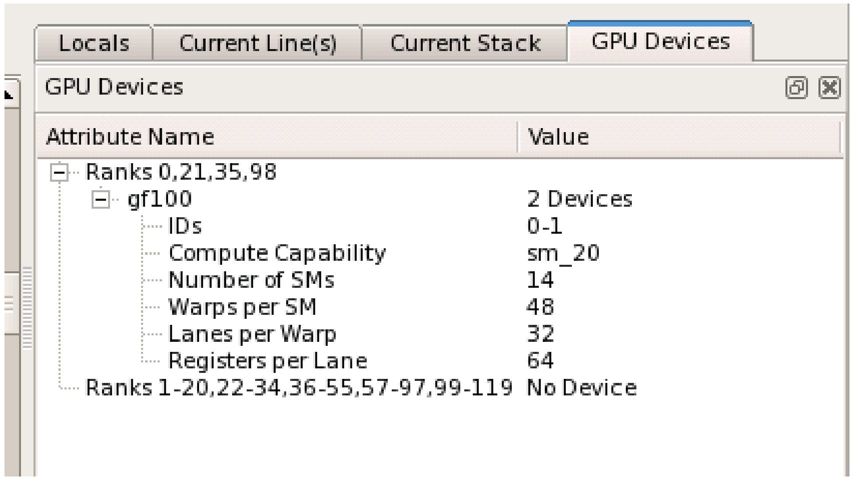
Only show if:

Grid size: 10000x1 Block size: 256x1x1

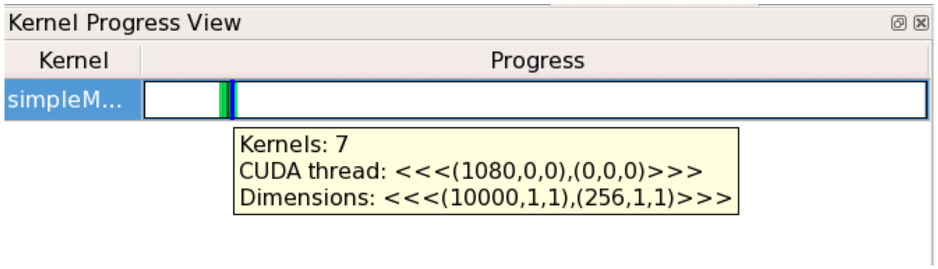
	i 0	1	2
x 0	0.840187728	0.394382924	0.
1	0.432718098	0.407395214	0.
2	0.490991831	0.530233502	0.
3	0.404152334	0.403803051	0.
4	0.113737881	0.293828964	0.
5	0.386504918	0.760585427	0.
6	0.959459126	0.299068421	0.

```
8        0   poll
8        0   service_thread_s
8        0     __select_nocar
8        0   main (simpleMPI.cpp
8        0     computeGPU (simp
1   172032   simpleMPIKernel (sim
1   172032     simpleMPIKernel (si
```

Overviews of GPUs



Attribute Name	Value
Ranks 0,21,35,98	
gf100	2 Devices
IDs	0-1
Compute Capability	sm_20
Number of SMs	14
Warps per SM	48
Lanes per Warp	32
Registers per Lane	64
Ranks 1-20,22-34,36-55,57-97,99-119	No Device



Kernels: 7
CUDA thread: <<<(1080,0,0),(0,0,0)>>>
Dimensions: <<<(10000,1,1),(256,1,1)>>>

- Device overview shows system properties
  - Helps optimize grid sizes
  - Handy for bug fixing – and detecting hardware failure!
- Kernel progress view
  - Shows progress through kernels
  - Click to select a thread

allinea
www.allinea.com

A New Hope

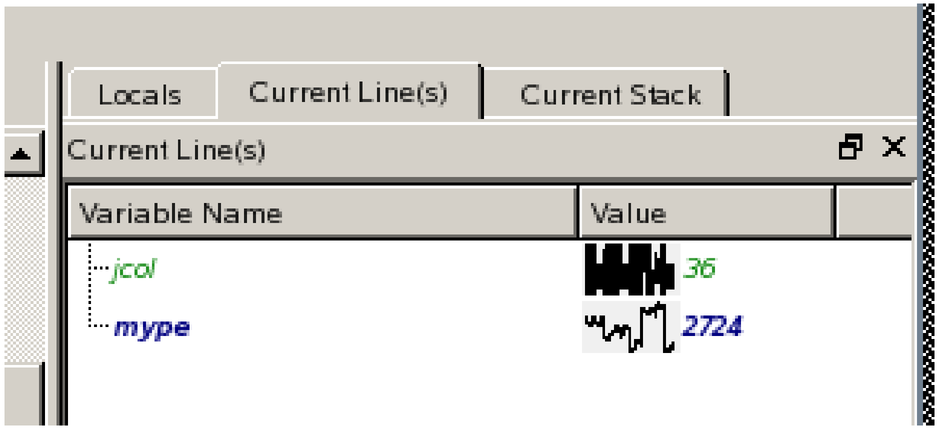
- Seattle, November 2011
  - CAPS, Cray, NVIDIA and PGI announce new standard for accelerator programming
    - Access CUDA compute power easily
    - A common standard
  - Allinea supports debugging Cray OpenACC compiler

OpenACC
DIRECTIVES FOR ACCELERATORS

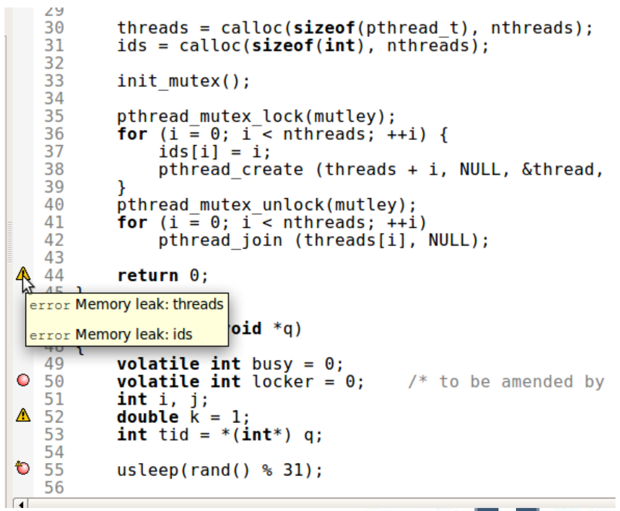
www.allinea.com

November 2011

- Allinea DDT 3.1 – innovation for all scales
  - Sparklines – automatic data comparison
  - Offline debugging – scalable hands-free
  - Static analysis – automatic detection of coding errors
  - Cray XK6 support
  - Cray UPC, CoArray and OpenACC support



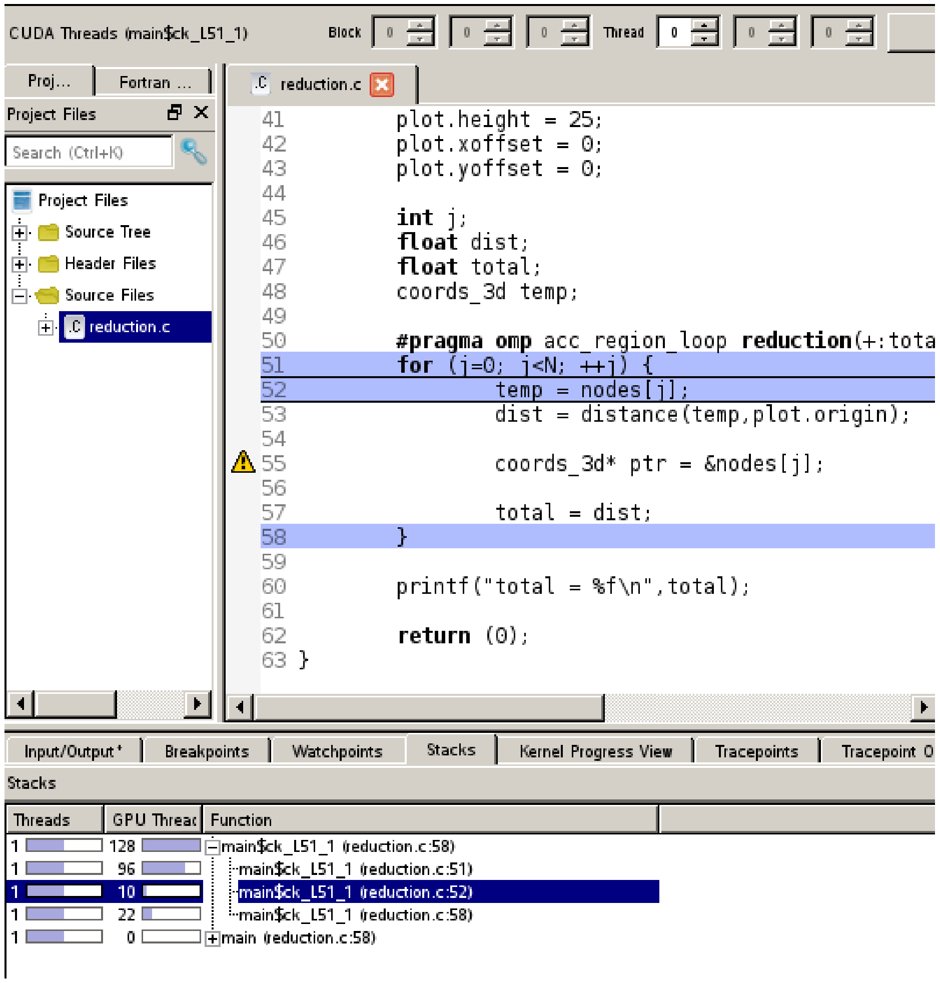
Variable Name	Value
jcol	36
mype	2724



```
    threads = calloc(sizeof(pthread_t), nthreads);
    ids = calloc(sizeof(int), nthreads);

    init_mutex();

    pthread_mutex_lock(mutley);
    for (i = 0; i < nthreads; ++i) {
        ids[i] = i;
        pthread_create (threads + i, NULL, &thread,
    }
    pthread_mutex_unlock(mutley);
    for (i = 0; i < nthreads; ++i)
        pthread_join (threads[i], NULL);

    return 0;
}
error Memory leak: threads
error Memory leak: ids        oid *q)
{
    volatile int busy = 0;
    volatile int locker = 0;     /* to be amended by
    int i, j;
    double k = 1;
    int tid = *(int*) q;

    usleep(rand() % 31);
```

www.allinea.com

OpenACC debugging



CUDA Threads (main$ck_L51_1) Block 0 0 0 Thread 0 0 0

```
        plot.height = 25;
        plot.xoffset = 0;
        plot.yoffset = 0;

        int j;
        float dist;
        float total;
        coords_3d temp;

        #pragma omp acc_region_loop reduction(+:tota
        for (j=0; j<N; ++j) {
                temp = nodes[j];
                dist = distance(temp,plot.origin);

                coords_3d* ptr = &nodes[j];

                total = dist;
        }

        printf("total = %f\n",total);

        return (0);
}
```

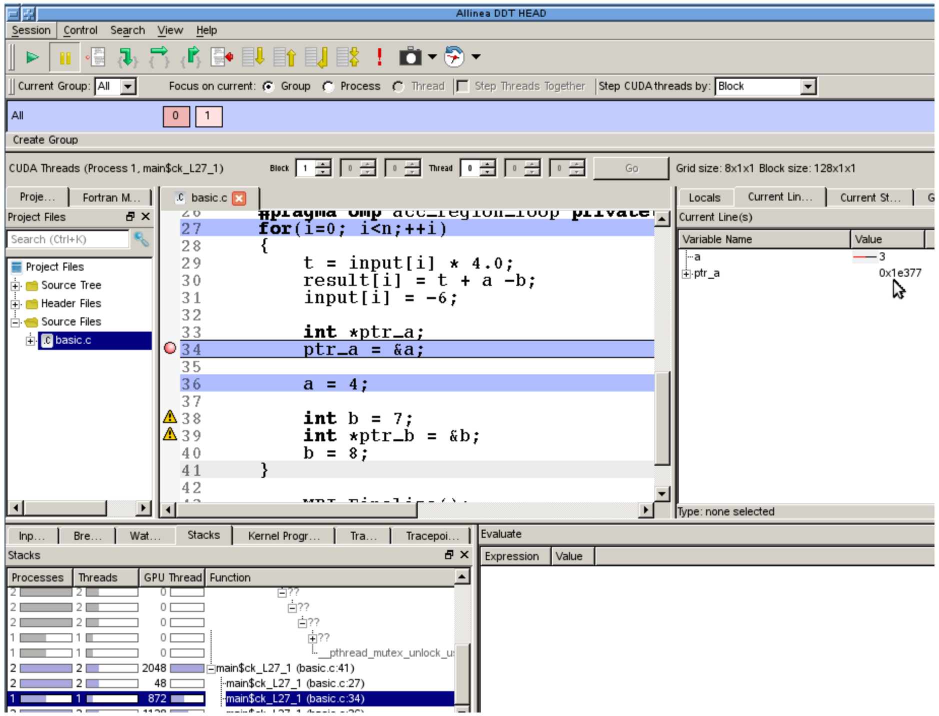
Threads	GPU Threac	Function
1	128	main$ck_L51_1 (reduction.c:58)
1	96	main$ck_L51_1 (reduction.c:51)
1	10	main$ck_L51_1 (reduction.c:52)
1	22	main$ck_L51_1 (reduction.c:58)
1	0	main (reduction.c:58)

- On device debugging with Allinea DDT
  - Variables – arrays, pointers, full F90 and C support
  - Set breakpoints and step warps and blocks
- Requires Cray compiler for on-device debugging
  - Other compilers to follow
- Identical to CUDA
  - Full warp/block/kernel controls

allinea
www.allinea.com

XK6 Status

- Interlagos update
  - No problem – tested at scale!
- Last known major GPU issues fixed
  - Driver fixes deployed April 2012
    - 3-way debug sessions to diagnose and fix a device-hang
- Full OpenACC and CUDA support



Current Group: All | Focus on current: Group ● Process Thread | Step Threads Together | Step CUDA threads by: Block

CUDA Threads (Process 1, main$ck_L27_1) Block 1 0 0 Thread 0 0 0 Go Grid size: 8x1x1 Block size: 128x1x1

```
#pragma omp acc_region_loop private
for(i=0; i<n;++i)
{
    t = input[i] * 4.0;
    result[i] = t + a -b;
    input[i] = -6;

    int *ptr_a;
    ptr_a = &a;

    a = 4;

    int b = 7;
    int *ptr_b = &b;
    b = 8;
}

MPI_Finalize();
```

Variable Name	Value
a	3
ptr_a	0x1e377

Type: none selected

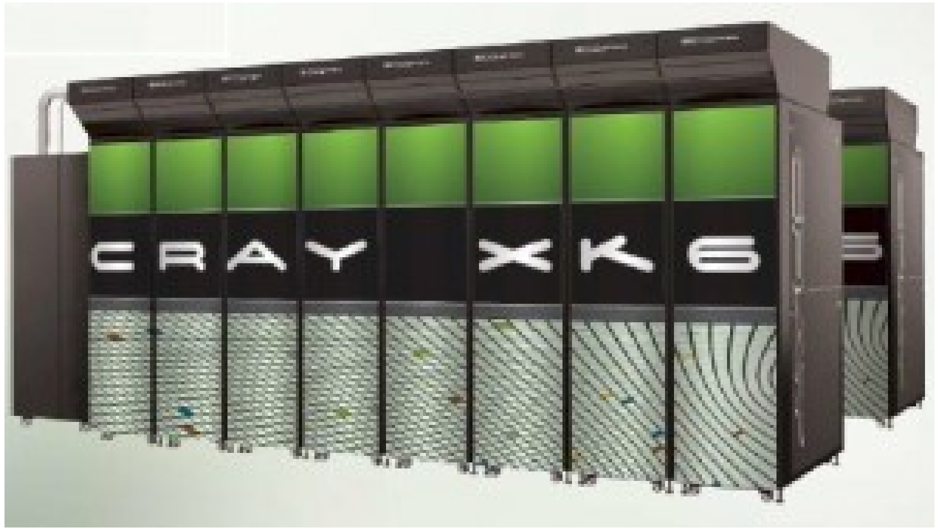
Stacks

```
Processes  Threads  GPU Thread  Function
2          2           0                  ??
2          2           0                    ??
2          2           0                      ??
1          1           0                        ??
1          1           0                          __pthread_mutex_unlock_u
2          2        2048        main$ck_L27_1 (basic.c:41)
2          2          48          main$ck_L27_1 (basic.c:27)
1          1         872          main$ck_L27_1 (basic.c:34)
```

allinea
www.allinea.com

What's next?

- Large Cray XK6 systems in (or almost in) place
  - ORNL Titan – 300,000 CPU cores
  - NCSA Blue Waters – 380,000 CPU cores
- Allinea DDT chosen for both systems – at scale
  - Watch out for improvements over next 6 months!
  - Kepler support for system upgrades



allinea
www.allinea.com