Leaders in parallel software development tools

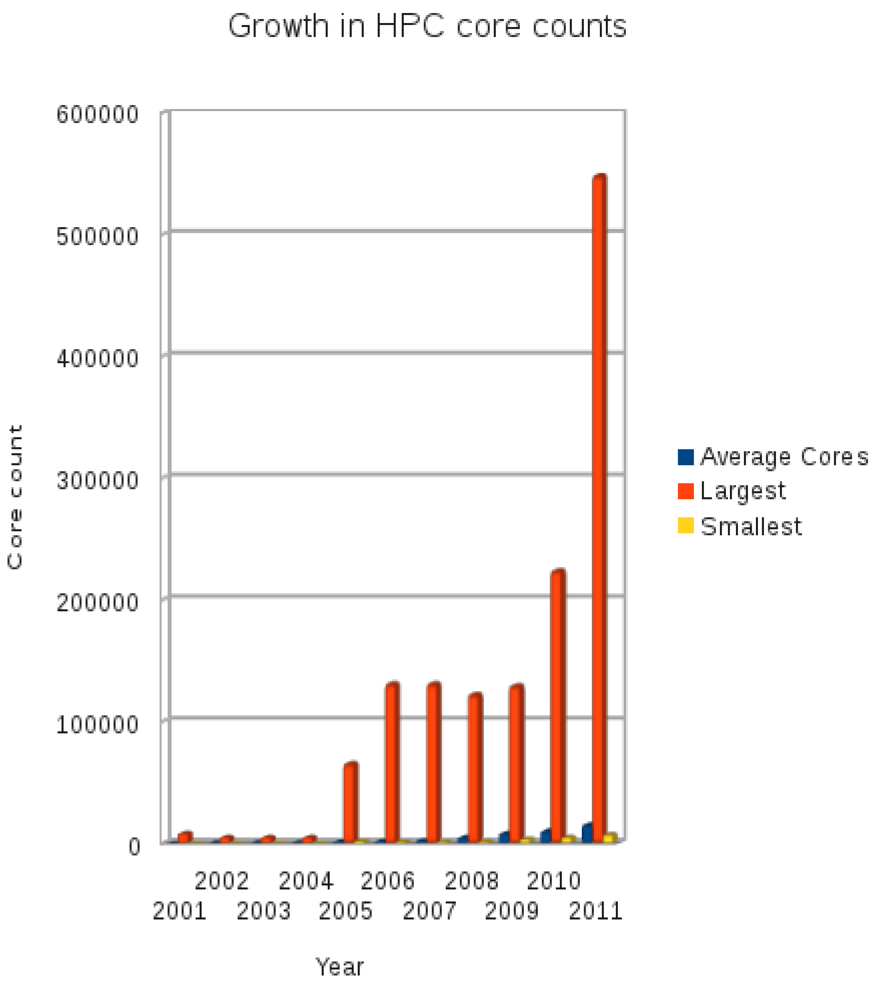
Allinea DDT
Intelligence Within

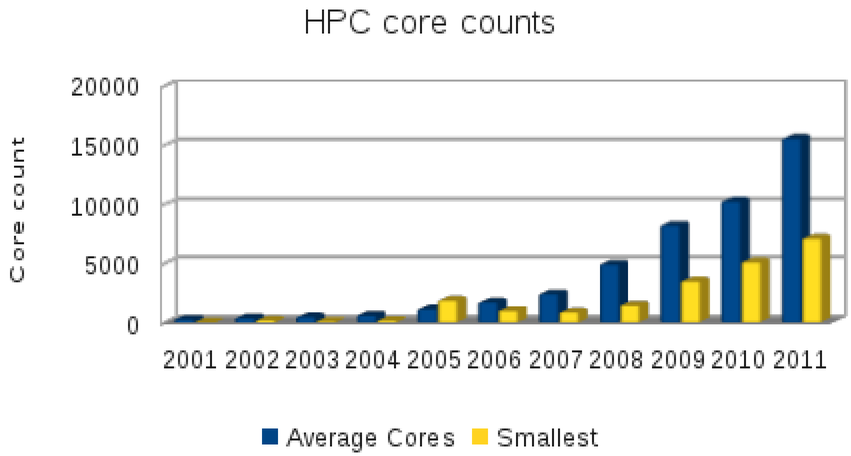
14 September 2012

Agenda

- Introduction
- Allinea DDT overview
- Conclusion

What is happening ?
Extreme Machine Size





- Scientific progress requires more CPU hours
  - Maximum machine size is exploding
  - Average machine size grows exponentially

allinea
www.allinea.com

The Company

- Development tools company
  - Leading in HPC software tools market worldwide
  - Global customer base
    - Blue-chip engineering, government and academic research
- Allinea DDT
  - The leading debugger in parallel computing
  - Production use from desktop to extreme scale
  - World's only scalable debugger
    - Record holder for debugging software on largest machines
    - First at Petascale – and first for GPUs and ARM support!
- Allinea OPT
  - Profiling tool for parallel applications

allinea
www.allinea.com

Collaboration – National Labs

OAK RIDGE National Laboratory

Partnership to develop Petascale debugger with NVIDIA support, with Cray and Caps Entreprise

Argonne NATIONAL LABORATORY

Partnership on Full Scale debugging on IBM Blue Gene /P & /Q

cea

Partnership with CEA French Atomic Energy Authority on scalable programming, CUDA and Allinea OPT

CREST

European partnership to develop techniques and solutions which address the exascale challenges

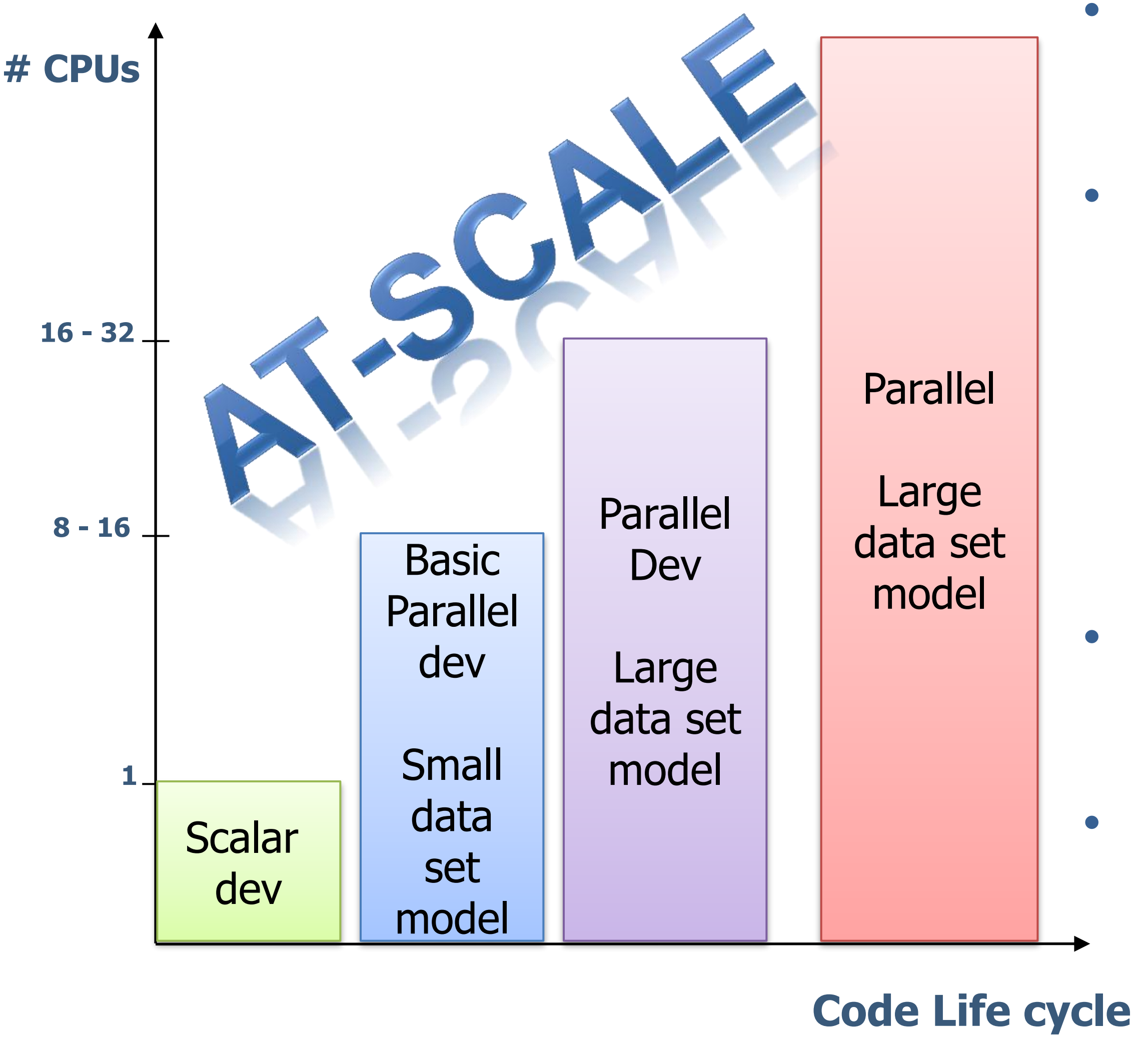
MONT BLANC

Partnership with BSC through its utilization of ARM technology to develop energy efficient HPC systems

allinea DDT

allinea
www.allinea.com

Application Development Life Cycles



- Bugs are present at any stage
- Bugs get more and more tricky
  - More threads/processes
  - More data
  - Easy bugs already found
- Time to debug increases
- Debugging at scale : this is what you mostly do !

allinea
www.allinea.com

Example

```
$> mpirun -np 4 ./cpi.exe
     Process 0 on dopey
     Process 2 on dopey
     Process 1 on dopey
     Process 3 on dopey
     pi is approximately 3.1416009869231249
     Error is 0.0000083333333318
     wall clock time = 0.012800
```

Happy !!

```
$> mpirun -np 8 ./cpi.exe
     Process 5 on dopey
     Process 1 on dopey
     Process 4 on dopey
     Process 6 on dopey
     Process 3 on dopey
     Process 0 on dopey
     Process 2 on dopey
     Process 7 on dopey

     ^Cmpirun: killing job...
```

Deadlock !!

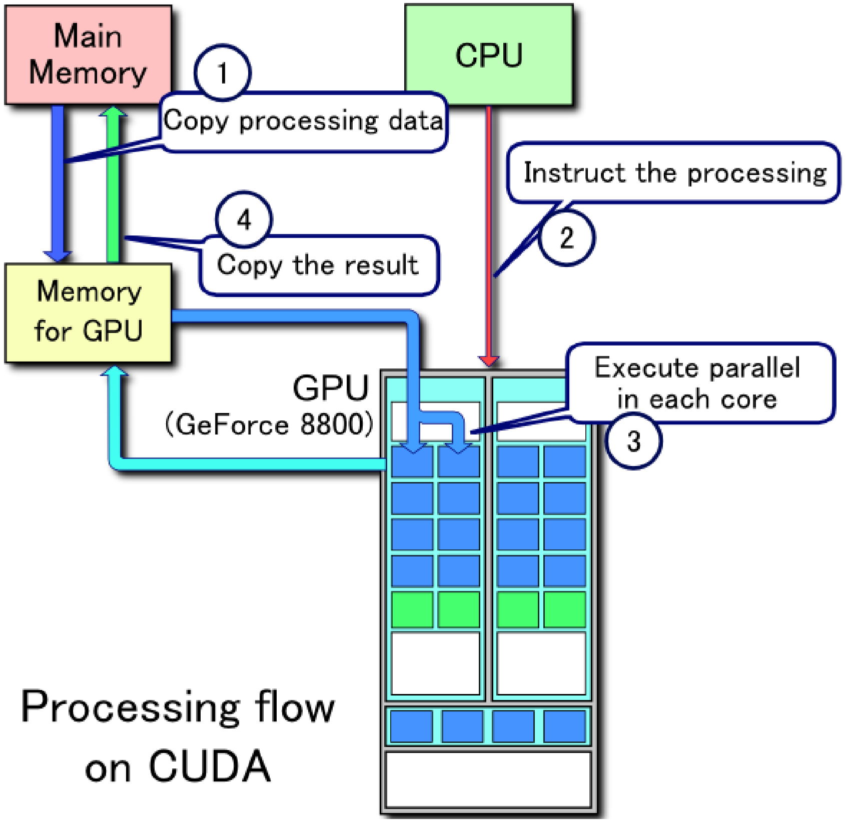
How to debug At-Scale ?

- Natural reflex : printf !
- No fully integrated solution : time consuming
  - Problems do not appear at smaller scale
  - Takes time to move the problems to a smaller size
  - Need to use multiple tools for multiple needs
  - More cores means more bugs...
  - ... And more debugging information available

What can we do... ?

HPC's current challenge

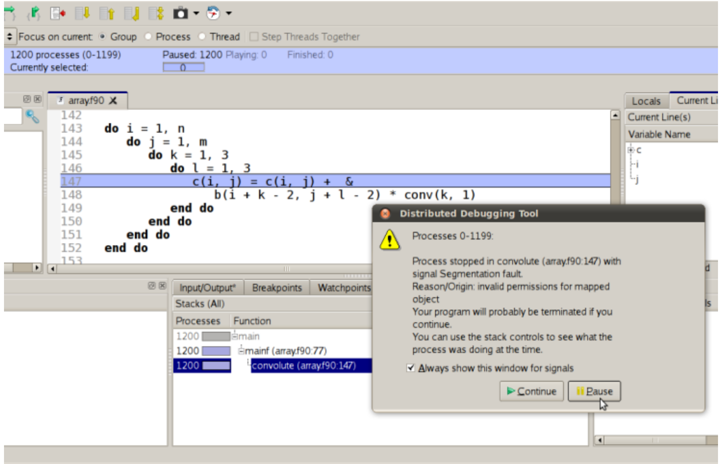
- GPUs – a rival to traditional processors
  - AMD, Intel, ARM
  - NVIDIA, OpenCL, CUDA
- A big challenge for HPC developers
  - Data transfer
  - Several memory levels
  - Grid/block layout and thread scheduling
  - Synchronization
- New languages, compilers, potential standards



Processing flow on CUDA

Allinea DDT
In a Nutshell

- Graphical debugger designed for:
  - C/C++, Fortran, UPC, CUDA
  - Multithreaded code
    - Single address space
  - Multiprocess code
    - Interdependent or independent processes
  - GPU codes
    - Hybrid software
  - Any mix of the above
- Managing concurrency
  - Emphasizing differences
  - Collective control
- Strong feature set
  - Memory debugging
  - Data analysis



```
do i = 1, n
   do j = 1, m
      do k = 1, 3
         do l = 1, 3
            c(i, j) = c(i, j) +  &
               b(i + k - 2, j + l - 2) * conv(k, 1)
         end do
      end do
   end do
end do
```

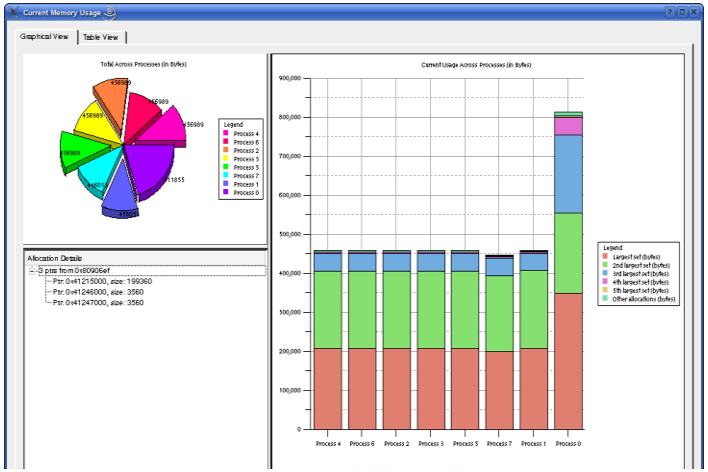
Processes	Function
1200	main
1200	mainf (array.f90:77)
1200	convolute (array.f90:147)

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

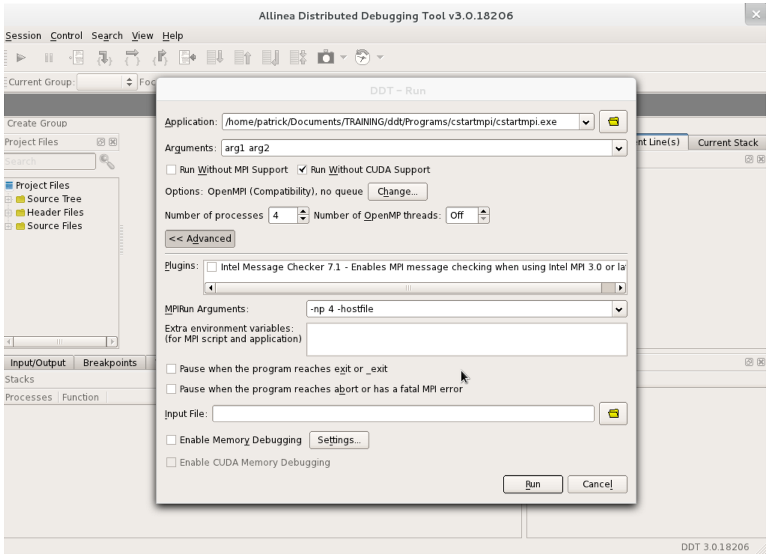
- 3 ptrs from 0x80906ef
  - Ptr: 0x41215000, size: 199360
  - Ptr: 0x41246000, size: 3560
  - Ptr: 0x41247000, size: 3560

www.allinea.com

Allinea DDT

Gather, Sort and Display information

- User and administrator friendly
- Flow control
- Data monitoring
- Many environments



Application: /home/patrick/Documents/TRAINING/ddt/Programs/cstartmpi/cstartmpi.exe

Arguments: arg1 arg2

□ Run Without MPI Support ☑ Run Without CUDA Support

Options: OpenMPI (Compatibility), no queue

Number of processes 4 Number of OpenMP threads: Off

Plugins: □ Intel Message Checker 7.1 - Enables MPI message checking when using Intel MPI 3.0 or la

MPIRun Arguments: -np 4 -hostfile

Extra environment variables: (for MPI script and application)

□ Pause when the program reaches exit or _exit

□ Pause when the program reaches abort or has a fatal MPI error

Input File:

□ Enable Memory Debugging

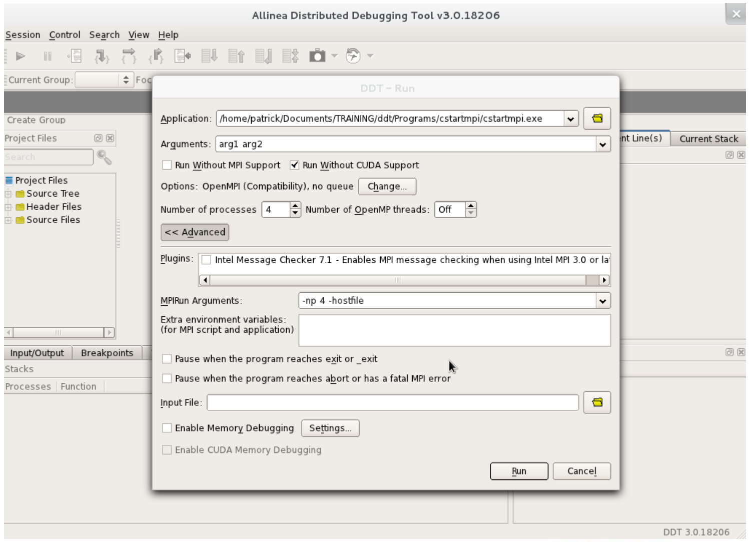
□ Enable CUDA Memory Debugging

allinea
www.allinea.com

Allinea DDT

Gather, Sort and Display information

- **User and administrator friendly**
  - Get started easily
  - Fast, reliable, simple and intuitive GUI interface
  - Offline debugging
- Flow control
- Data monitoring
- Many environments



Allinea Distributed Debugging Tool v3.0.18206

DDT – Run

Application: /home/patrick/Documents/TRAINING/ddt/Programs/cstartmpi/cstartmpi.exe

Arguments: arg1 arg2

☐ Run Without MPI Support ☑ Run Without CUDA Support

Options: OpenMPI (Compatibility), no queue

Number of processes 4 Number of OpenMP threads: Off

Plugins: ☐ Intel Message Checker 7.1 - Enables MPI message checking when using Intel MPI 3.0 or la

MPIRun Arguments: -np 4 -hostfile

Extra environment variables:
(for MPI script and application)

☐ Pause when the program reaches exit or _exit

☐ Pause when the program reaches abort or has a fatal MPI error

Input File:

☐ Enable Memory Debugging

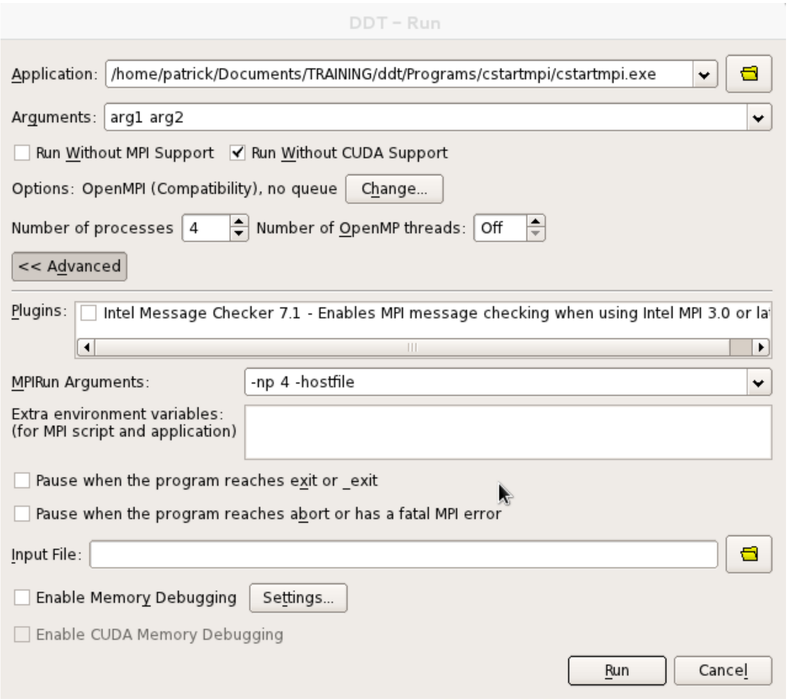
☐ Enable CUDA Memory Debugging

DDT 3.0.18206

allinea
www.allinea.com

User and administrator friendly
Getting started

- Quick creation of runs
- Well integrated in workload schedulers
  - SLURM, LSF, SGE, PBS,....



DDT – Run

Application: /home/patrick/Documents/TRAINING/ddt/Programs/cstartmpi/cstartmpi.exe

Arguments: arg1 arg2

□ Run Without MPI Support ☑ Run Without CUDA Support

Options: OpenMPI (Compatibility), no queue [Change...]

Number of processes: 4 Number of OpenMP threads: Off

[<< Advanced]

Plugins: □ Intel Message Checker 7.1 - Enables MPI message checking when using Intel MPI 3.0 or la

MPIRun Arguments: -np 4 -hostfile

Extra environment variables:
(for MPI script and application)

□ Pause when the program reaches exit or _exit

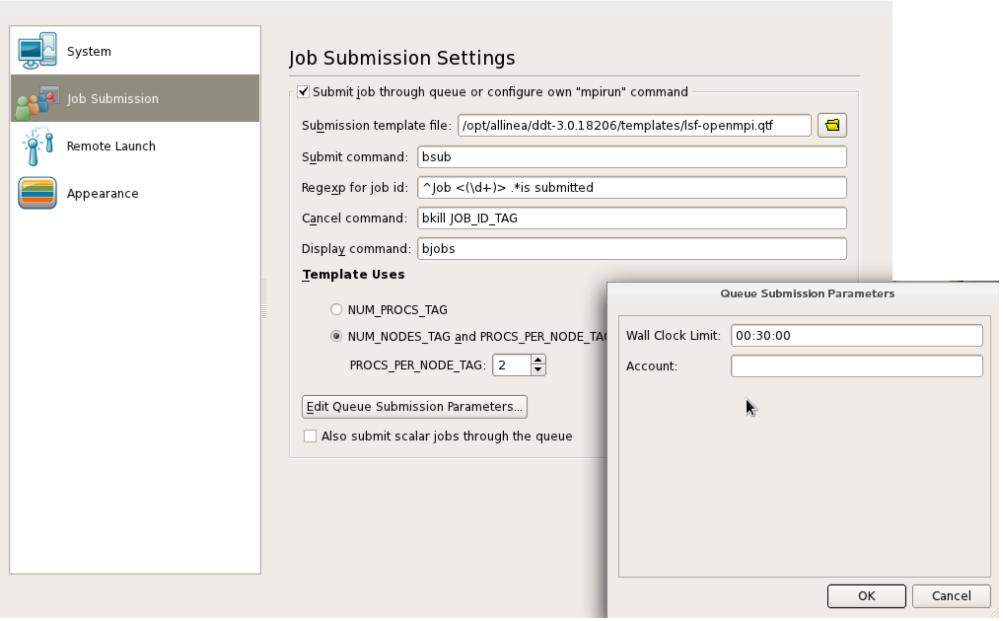
□ Pause when the program reaches abort or has a fatal MPI error

Input File:

□ Enable Memory Debugging [Settings...]

□ Enable CUDA Memory Debugging

[Run] [Cancel]



Job Submission Settings

☑ Submit job through queue or configure own "mpirun" command

Submission template file: /opt/allinea/ddt-3.0.18206/templates/lsf-openmpi.qtf

Submit command: bsub

Regexp for job id: ^Job <(\d+)> .*is submitted

Cancel command: bkill JOB_ID_TAG

Display command: bjobs

Template Uses

○ NUM_PROCS_TAG

● NUM_NODES_TAG and PROCS_PER_NODE_TA

PROCS_PER_NODE_TAG: 2

[Edit Queue Submission Parameters...]

□ Also submit scalar jobs through the queue

Queue Submission Parameters

Wall Clock Limit: 00:30:00

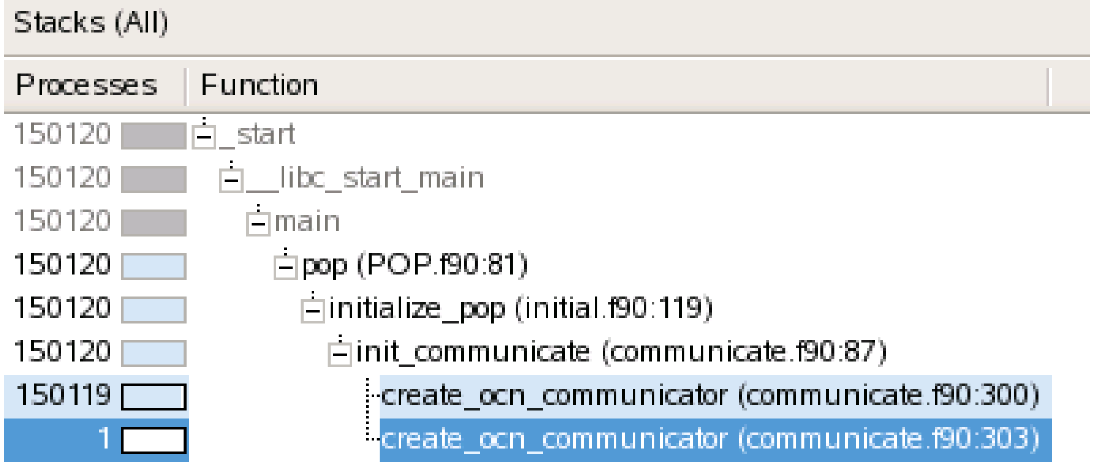
Account:

[OK] [Cancel]

www.allinea.com

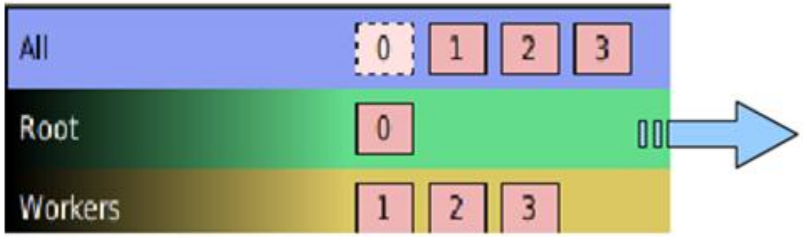
User and administrator friendly
Intuitive GUI interface

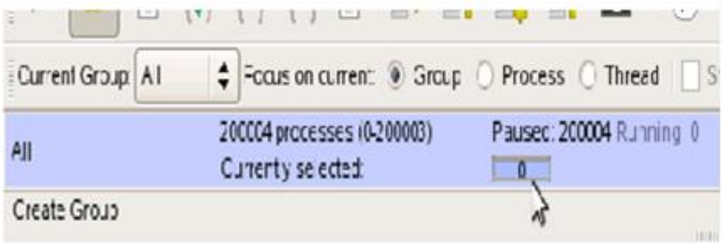
- Intelligent GUI that adapts to the environment
  - From workstation to large scale clusters



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)





Current Group: All ▲▼ Focus on current: ● Group ○ Process ○ Thread □ S

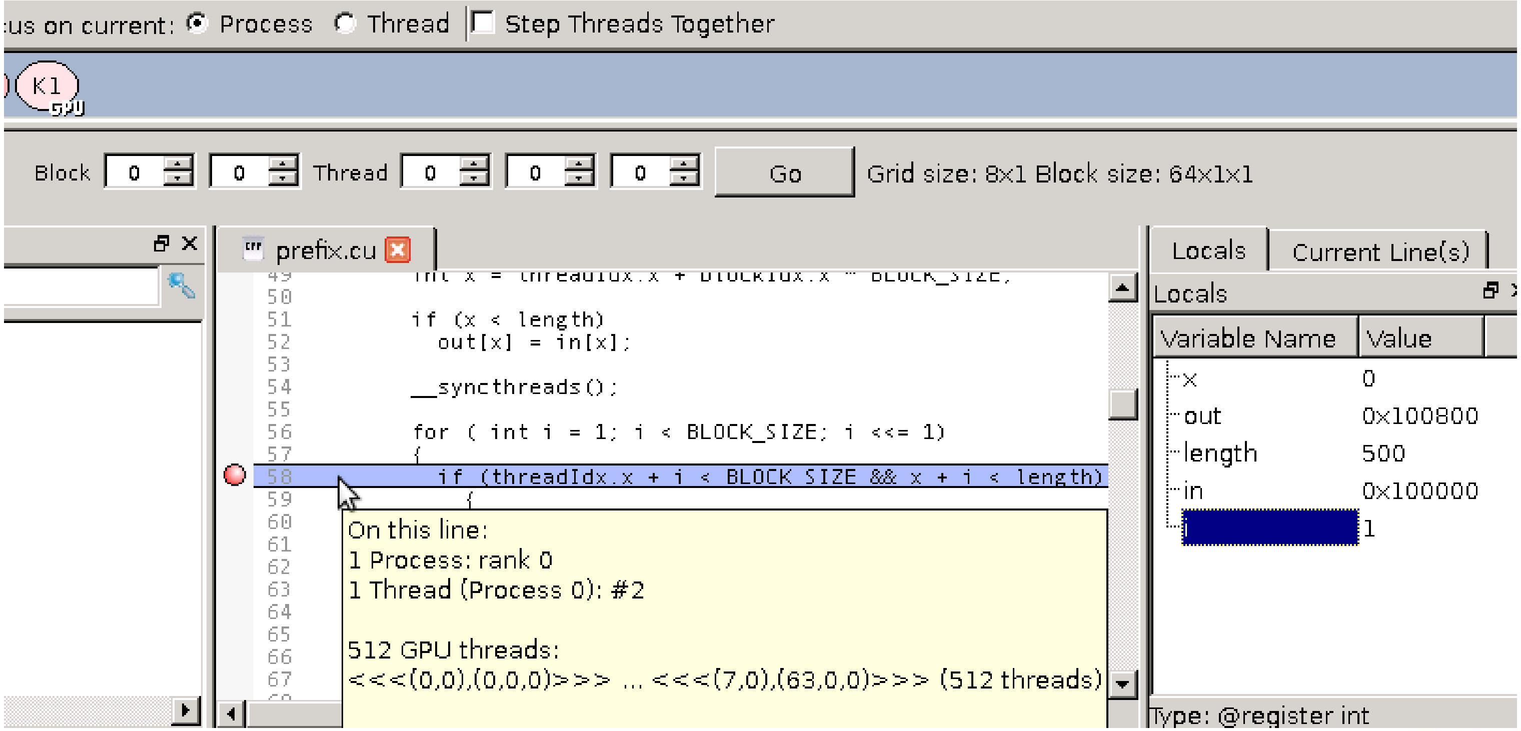
All 200004 processes (0-200003) Paused: 200004 Running: 0
Currently selected: 0

Create Group

www.allinea.com

User and administrator friendly
Intuitive GUI interface

- Intelligent GUI that adapts to the environment
  - Even in CUDA environments



Grid size: 8x1 Block size: 64x1x1

```
        int x = threadIdx.x + blockIdx.x * BLOCK_SIZE;

        if (x < length)
          out[x] = in[x];

        __syncthreads();

        for ( int i = 1; i < BLOCK_SIZE; i <<= 1)
        {
          if (threadIdx.x + i < BLOCK_SIZE && x + i < length)
            {
```

On this line:
1 Process: rank 0
1 Thread (Process 0): #2

512 GPU threads:
<<<(0,0),(0,0,0)>>> ... <<<(7,0),(63,0,0)>>> (512 threads)

Locals

Variable Name	Value
x	0
out	0x100800
length	500
in	0x100000
	1

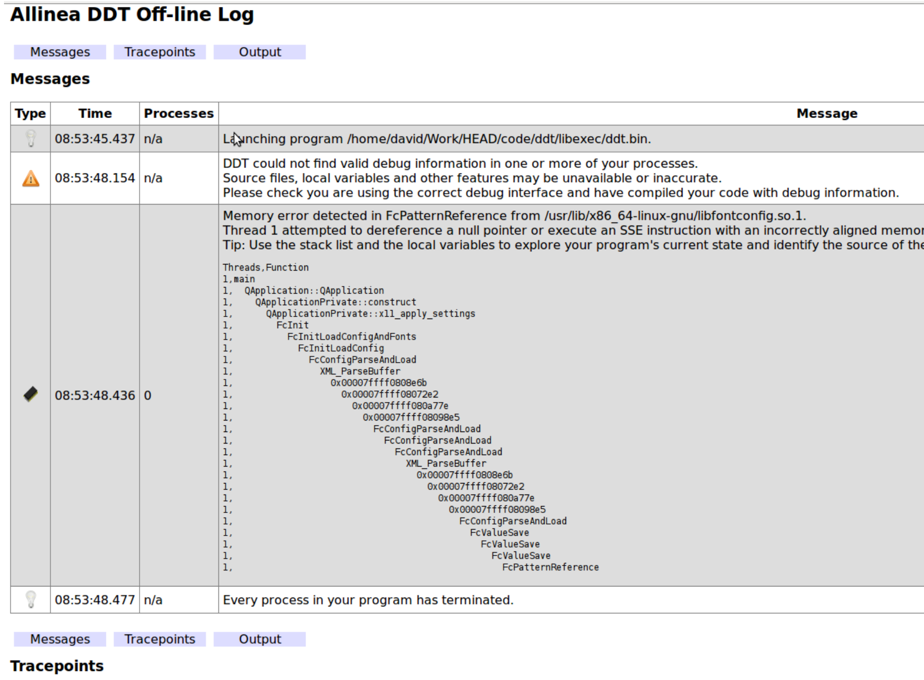
Type: @register int

User and administrator friendly Offline debugging

- Using a workload scheduler
  - Machines are available when the scheduler decides (by night ?)
  - Can be tricky to get a big cluster exactly when the developer wants it

- Offline debugging : printf replacement
  - Tracepoints and offline debugging
  - Job runs without debugger interface and record variables

- Worlds first scalable batch debugger
  - Set tracepoints, breakpoints, and run !
  - Memory debugging errors, crashes
  - Reports in HTML or plain text



Allinea DDT Off-line Log

Messages

Type	Time	Processes	Message
💡	08:53:45.437	n/a	Launching program /home/david/Work/HEAD/code/ddt/libexec/ddt.bin.
⚠	08:53:48.154	n/a	DDT could not find valid debug information in one or more of your processes. Source files, local variables and other features may be unavailable or inaccurate. Please check you are using the correct debug interface and have compiled your code with debug information.
◆	08:53:48.436	0	Memory error detected in FcPatternReference from /usr/lib/x86_64-linux-gnu/libfontconfig.so.1. Thread 1 attempted to dereference a null pointer or execute an SSE instruction with an incorrectly aligned memor Tip: Use the stack list and the local variables to explore your program's current state and identify the source of the Threads,Function 1,main 1, QApplication::QApplication 1, QApplicationPrivate::construct 1, QApplicationPrivate::x11_apply_settings 1, FcInit 1, FcInitLoadConfigAndFonts 1, FcInitLoadConfig 1, FcConfigParseAndLoad 1, XML_ParseBuffer 1, 0x00007ffff0808e6b 1, 0x00007ffff08072e2 1, 0x00007ffff080a77e 1, 0x00007ffff08098e5 1, FcConfigParseAndLoad 1, FcConfigParseAndLoad 1, FcConfigParseAndLoad 1, XML_ParseBuffer 1, 0x00007ffff0808e6b 1, 0x00007ffff08072e2 1, 0x00007ffff080a77e 1, 0x00007ffff08098e5 1, FcConfigParseAndLoad 1, FcValueSave 1, FcValueSave 1, FcValueSave 1, FcPatternReference
💡	08:53:48.477	n/a	Every process in your program has terminated.

Tracepoints

Allinea DDT and Offline Debugging Using the CLI

- General Options :

```
-offline <YYY>    : activate offline mode
-n <XXX>          : the number of MPI processes
-memory           : enable memory debugging
-ddtsession       : the session file to use (if exists)
```

- Create breakpoints : `-break-at LOCATION[,N:M:P]`

Example:

```
$> ddt –offline myjob.html –break-at main.c:22,-:2:-,x myapp
```

- Create tracepoints : `-trace-at LOCATION[,N:M:P],VAR1,VAR2...`

Example:

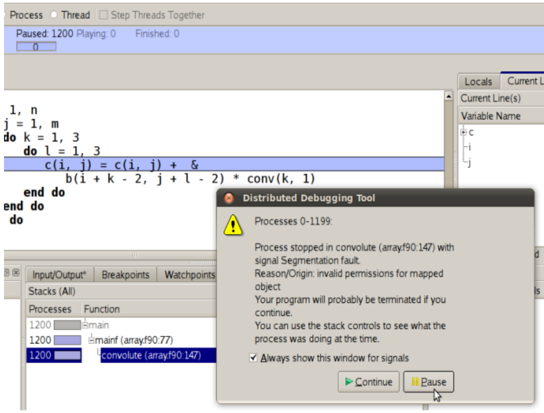
```
$> ddt –offline myjob.html –trace-at trisol.F90:14, -:-:3,x myapp
```

www.allinea.com

Allinea DDT

Gather, Sort and Display information

- User and administrator friendly
- Flow control
  - – Static analysis
  - – Control progress at scale
  - – Understand deadlocks
  - – Start investigation
- Data monitoring
- Many environments



Paused: 1200 Playing: 0 Finished: 0

```
 1, n
j = 1, m
do k = 1, 3
   do l = 1, 3
      c(i, j) = c(i, j) +  &
         b(i + k - 2, j + l - 2) * conv(k, 1)
   end do
end do
 do
```

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

Distributed Debugging Tool

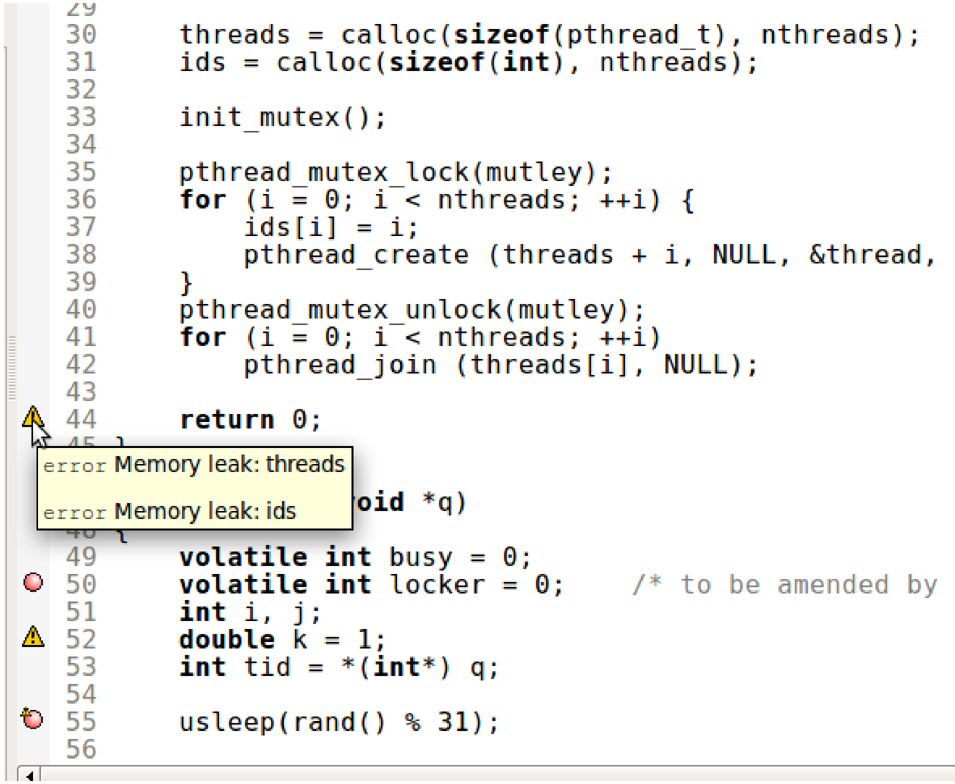
Processes 0-1199:

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.

allinea
www.allinea.com

Flow control
Static analysis

- Fix those errors before they bite !
- Static analysis
  - Integrated with cppcheck
  - Also includes ftncheck

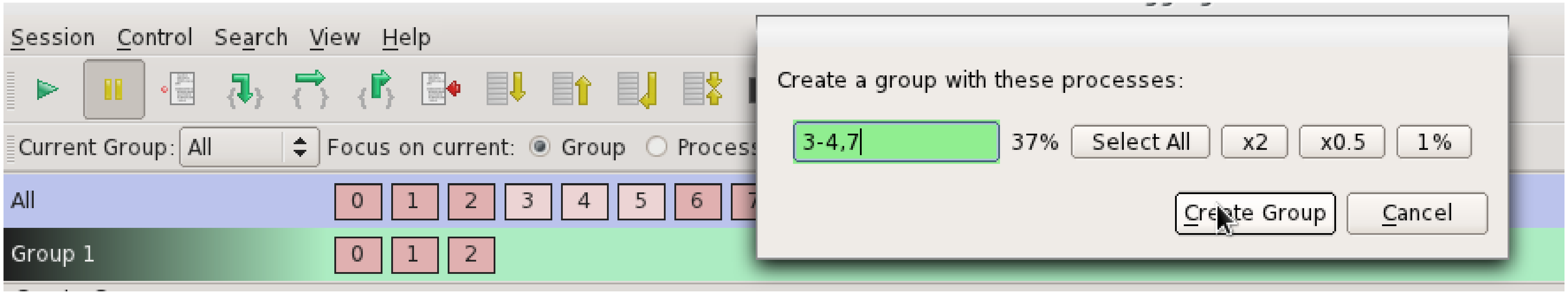


```
    threads = calloc(sizeof(pthread_t), nthreads);
    ids = calloc(sizeof(int), nthreads);

    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;
}

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);

```

```
error Memory leak: threads
error Memory leak: ids
```

Flow control
Controlling progress at scale

- Bulk control is essential for multicore debugging
  - Group processes together
  - Play, step, reach breakpoints... Based on groups
  - Change interleaving order by stepping/playing selectively



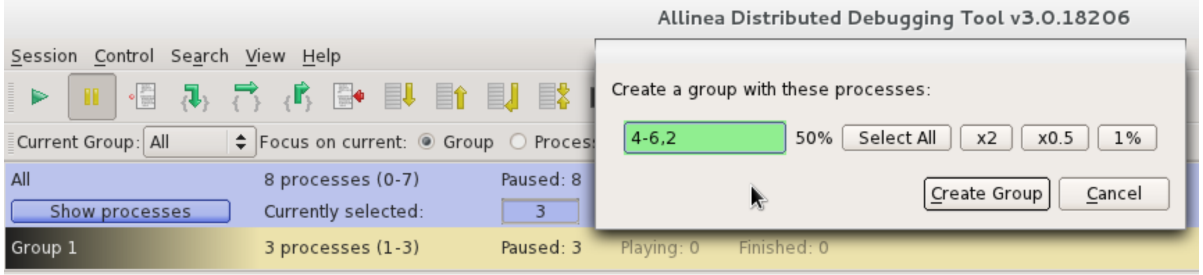
Create a group with these processes: 3-4,7 37%

Current Group: All

Focus on current: ● Group ○ Process

All 0 1 2 3 4 5 6 7

Group 1 0 1 2



Allinea Distributed Debugging Tool v3.0.18206

Create a group with these processes: 4-6,2 50%

Current Group: All

Focus on current: ● Group ○ Process

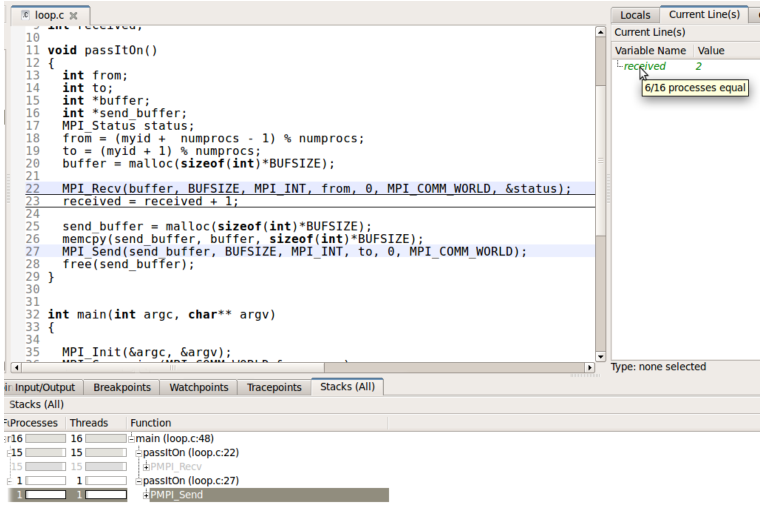
All 8 processes (0-7) Paused: 8

Currently selected: 3

Group 1 3 processes (1-3) Paused: 3 Playing: 0 Finished: 0

allinea
www.allinea.com

Flow control
Resolving MPI issues



loop.c

```
int received;

void passItOn()
{
  int from;
  int to;
  int *buffer;
  int *send_buffer;
  MPI_Status status;
  from = (myid +  numprocs - 1) % numprocs;
  to = (myid + 1) % numprocs;
  buffer = malloc(sizeof(int)*BUFSIZE);

  MPI_Recv(buffer, BUFSIZE, MPI_INT, from, 0, MPI_COMM_WORLD, &status);
  received = received + 1;

  send_buffer = malloc(sizeof(int)*BUFSIZE);
  memcpy(send_buffer, buffer, sizeof(int)*BUFSIZE);
  MPI_Send(send_buffer, BUFSIZE, MPI_INT, to, 0, MPI_COMM_WORLD);
  free(send_buffer);
}


int main(int argc, char** argv)
{

  MPI_Init(&argc, &argv);
```

Current Line(s)

Variable Name	Value
received	2

6/16 processes equal

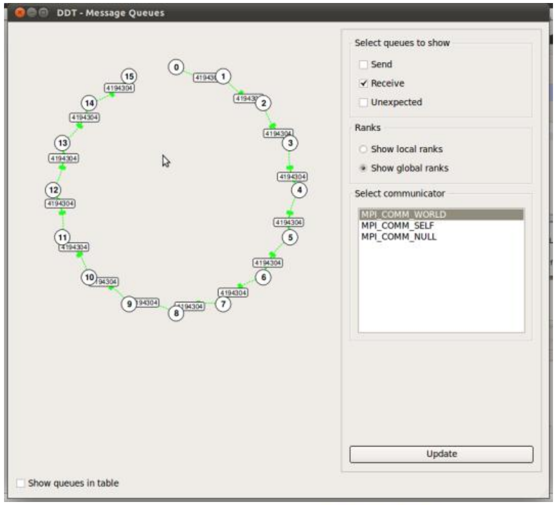
Type: none selected

Stacks (All)

Processes	Threads	Function
16	16	main (loop.c:48)
15	15	passItOn (loop.c:22)
15	15	PMPI_Recv
1	1	passItOn (loop.c:27)
1	1	PMPI_Send

- We can see messages
  - MPI standard exists for debugging message queues
- Integrates with MPI correctness tools
  - Check correctness of the messages
  - Intel MPI Checker ; Marmot

- Find cycles or blockage in message queue display
  - Parallel stacks
  - Variables
- More details than examining variables and processes alone



DDT - Message Queues

Select queues to show

☐ Send
☑ Receive
☐ Unexpected

Ranks

○ Show local ranks
● Show global ranks

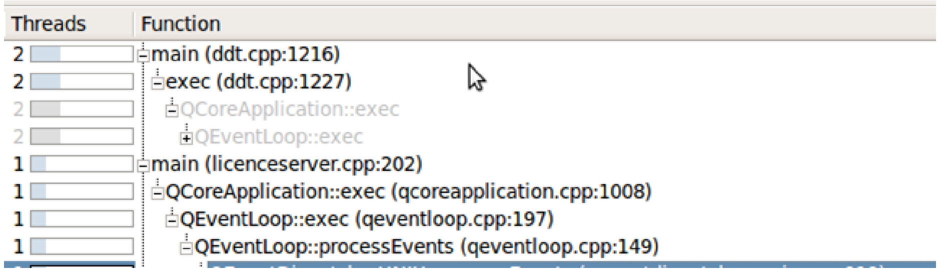
Select communicator

MPI_COMM_WORLD
MPI_COMM_SELF
MPI_COMM_NULL

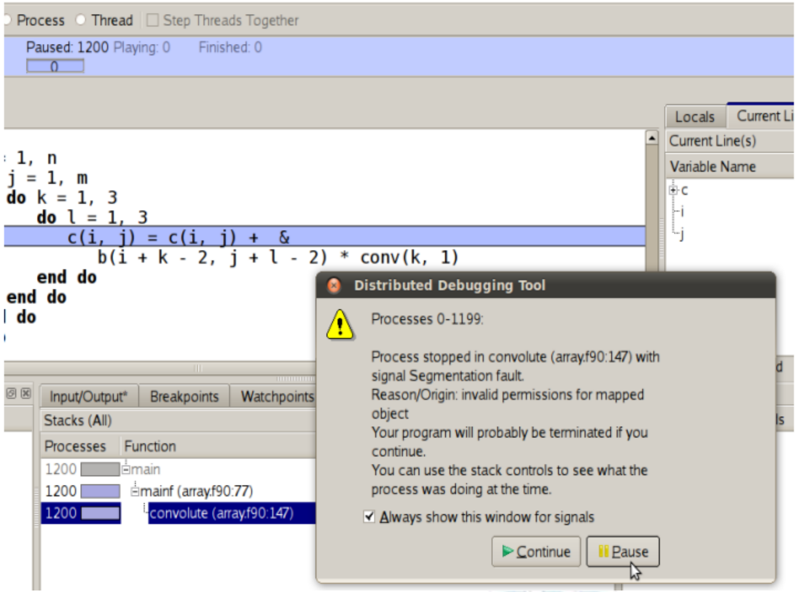
☐ Show queues in table

Flow control
Understanding what happens

- Application crashes
  - Threads/processes can be anywhere
  - Impossible to scroll through them individually
- Finding where processes crashed is essential
  - Allinea DDT merges stacks from processors and threads into a tree
  - Common faults patterns instantly evident
    - Divergence, deadlocks...
  - Information scalable without overload



Threads	Function
2	main (ddt.cpp:1216)
2	exec (ddt.cpp:1227)
2	QCoreApplication::exec
2	QEventLoop::exec
1	main (licenceserver.cpp:202)
1	QCoreApplication::exec (qcoreapplication.cpp:1008)
1	QEventLoop::exec (qeventloop.cpp:197)
1	QEventLoop::processEvents (qeventloop.cpp:149)



```
: 1, n
j = 1, m
do k = 1, 3
   do l = 1, 3
      c(i, j) = c(i, j) +  &
         b(i + k - 2, j + l - 2) * conv(k, 1)
   end do
end do
| do
```

Stacks (All)

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

Distributed Debugging Tool

Processes 0-1199:

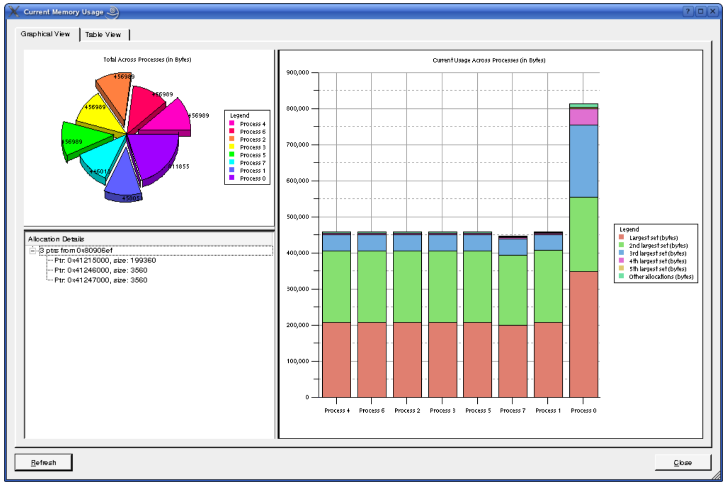
Process stopped in convolute (array.f90:147) with signal Segmentation fault.
Reason/Origin: invalid permissions for mapped object
Your program will probably be terminated if you continue.
You can use the stack controls to see what the process was doing at the time.

☑ Always show this window for signals

Allinea DDT

Gather, Sort and Display information

- User and administrator friendly
- Flow control
- Data monitoring
  - Monitor variables
  - Detect memory errors
  - Check the calculation data
- Many environments



Allocation Details

```
⊟ 3 ptrs from 0x80906ef
   Ptr: 0x41215000, size: 199360
   Ptr: 0x41246000, size: 3560
   Ptr: 0x41247000, size: 3560
```

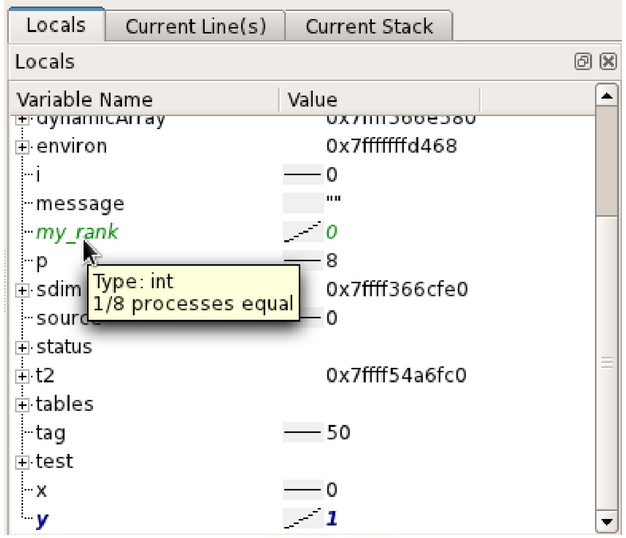
allinea
www.allinea.com

Data monitoring
Monitor variables

- Developers need to see data
  - Too many variables to trawl them manually
  - Too many tasks or thread to display them at the same time

- Intelligent data management inside the debugger
  - Automatic monitoring of the data
  - Subtle highlights differences
  - Sparklines and smart display

- Even more detailed analysis if needed
  - Cross process comparison
  - Historical values of variables



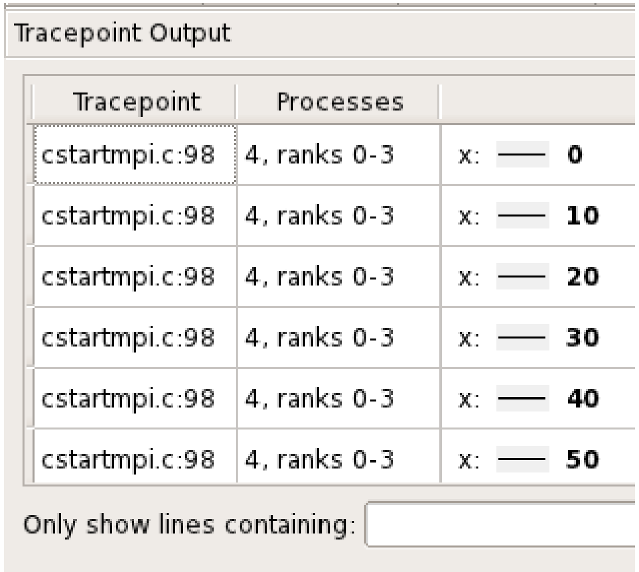
Variable Name	Value
dynamicArray	0x7ffff366e380
environ	0x7fffffffd468
i	0
message	""
my_rank	*0*
p	8
sdim	0x7ffff366cfe0
sourc	0
status	
t2	0x7ffff54a6fc0
tables	
tag	50
test	
x	0
y	***1***

Type: int
1/8 processes equal

allinea
www.allinea.com

Data monitoring
Smart displays : tracepoints

- “printf” is still mostly used but serious drawbacks:
  - Need to recompile the code
  - Information randomly printed on screen (depending on interleaving)
  - One line per process
- Scalable and advanced printf :
  - First step to reconcile printf and GUI
  - Information sorted by steps
  - Merged by groups of processes
  - No information overload
  - Possibility to filter the printed values
  - Save output for offline analysis



Tracepoint Output

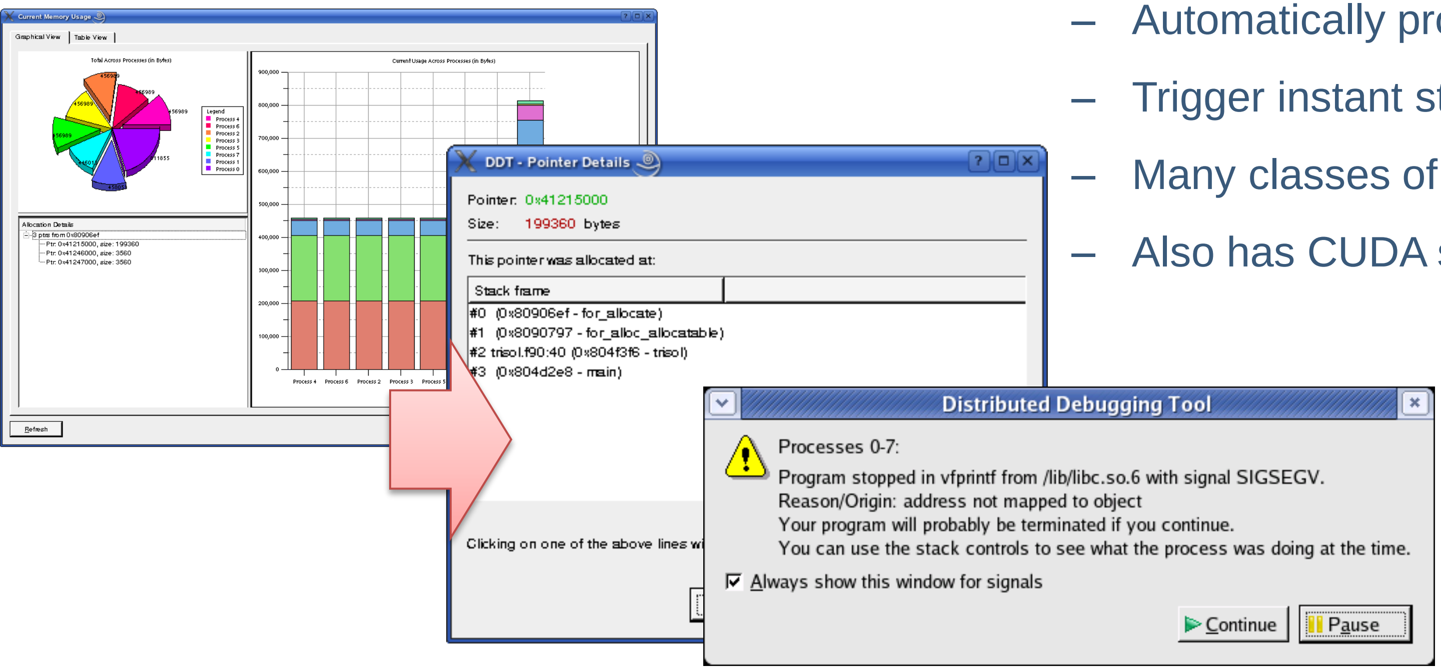
Tracepoint	Processes	
cstartmpi.c:98	4, ranks 0-3	x: — 0
cstartmpi.c:98	4, ranks 0-3	x: — 10
cstartmpi.c:98	4, ranks 0-3	x: — 20
cstartmpi.c:98	4, ranks 0-3	x: — 30
cstartmpi.c:98	4, ranks 0-3	x: — 40
cstartmpi.c:98	4, ranks 0-3	x: — 50

Only show lines containing:

Data monitoring
Memory debugging

- Random errors are sneaky
  - Can’t fix a bug that doesn’t repeat
  - Often caused by memory issues
- Memory debugging can force the bug
  - Better to happen every time, than only during product demos

- Memory debugging :
  - Places agent between memory library and user process
  - Communicates problems to the debuggers
  - Monitors usage : detect memory leaks
  - Automatically protects ends of arrays
  - Trigger instant stop on touching memory
  - Many classes of errors can be checked
  - Also has CUDA support



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

Pointer: 0x41215000

Size: 199360 bytes

This pointer was allocated at:

```
Stack frame
#0  (0x80906ef - for_allocate)
#1  (0x8090797 - for_alloc_allocatable)
#2 trisol.f90:40 (0x804f3f6 - trisol)
#3  (0x804d2e8 - main)
```

Clicking on one of the above lines wi

Processes 0-7:

Program stopped in vfprintf from /lib/libc.so.6 with signal SIGSEGV.
Reason/Origin: address not mapped to 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.

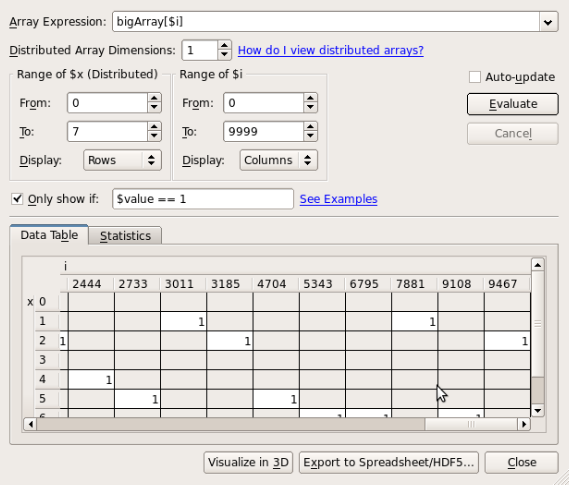
☑ Always show this window for signals

allinea
www.allinea.com

Data monitoring
Searching haystacks

- Arrays are the building blocks of HPC
  - Largest jobs accumulate TB of data
  - Usually 2GB or 4GB per core

- Integrated visualization tool
  - Search data across all tasks or threads
  - Data displayed on a picture as requested by the user
  - Export at runtime



Array Expression: bigArray[$i]

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

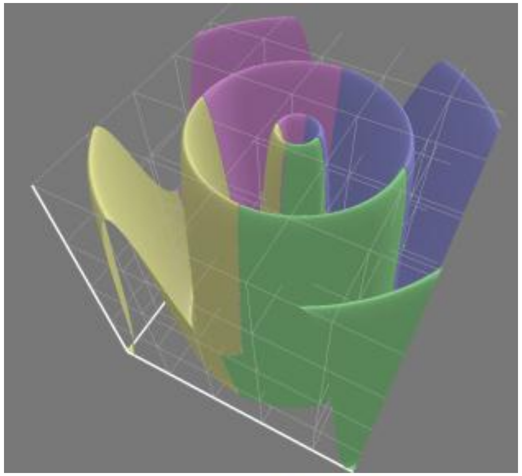
Range of $x (Distributed)
From: 0
To: 7
Display: Rows

Range of $i
From: 0
To: 9999
Display: Columns

□ Auto-update

☑ Only show if: $value == 1 See Examples

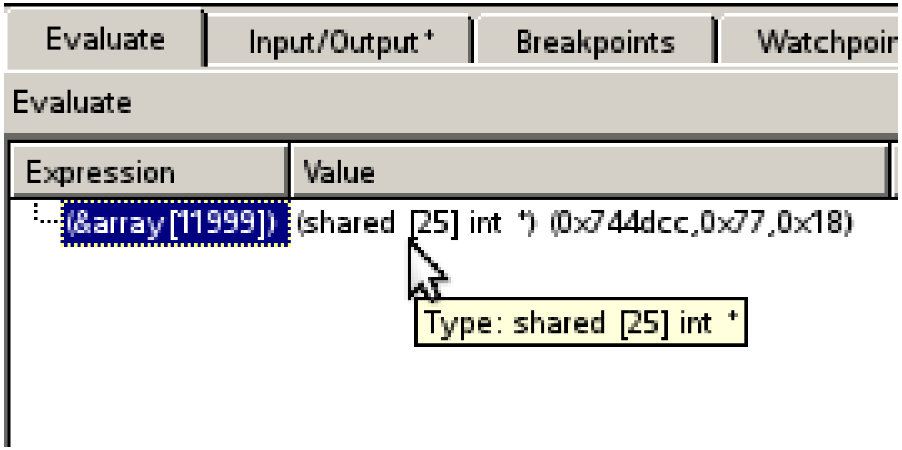
x \ i		2444	2733	3011	3185	4704	5343	6795	7881	9108	9467
0											
1				1					1		
2	1				1						1
3											
4		1									
5			1			1					
6							1	1		1	



www.allinea.com

Data Monitoring
Unified Parallel C

- New programming models
  - – Support for Cray UPC
  - – Support for Cray Co-Array Fortran



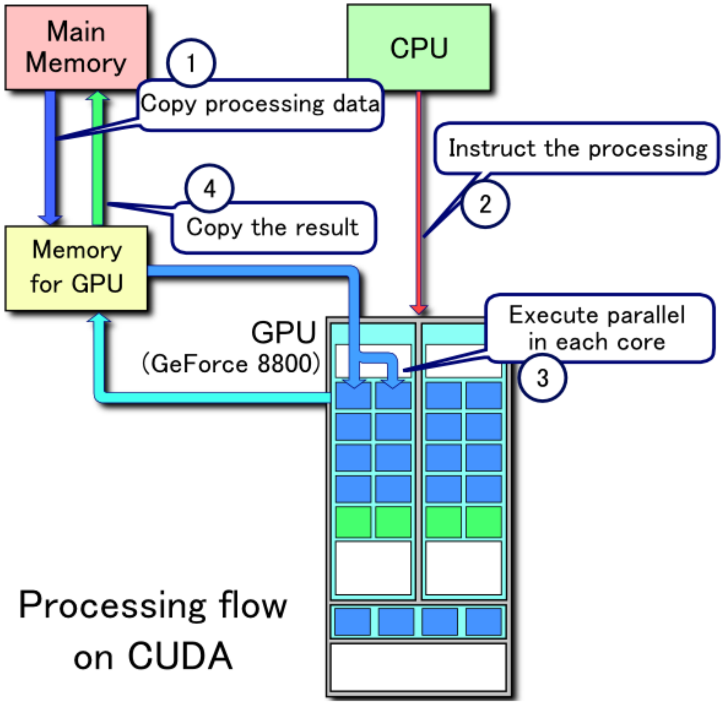
Expression	Value
(&array[11999])	(shared [25] int *) (0x744dcc,0x77,0x18)

Type: shared [25] int *

Allinea DDT

Gather, Sort and Display information

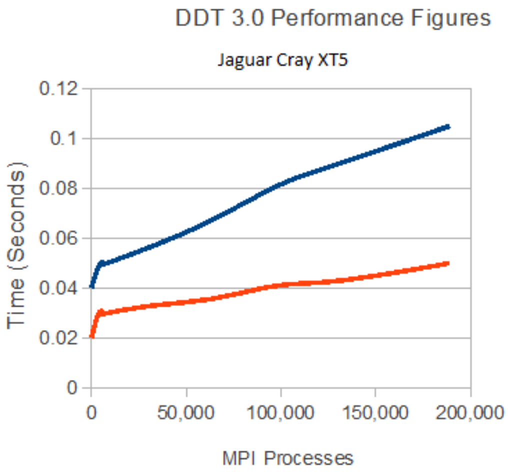
- User and administrator friendly
- Flow control
- Data monitoring
- Many environments
  - Debugging at scale
  - GPU Debugging



Processing flow on CUDA

Allinea DDT and Scalability
Simplifies debugging for everyone

- At scale : new problems appear
  - Small and medium environments : multiple tools
  - More cores means more bugs...
  - ... And more debugging information available



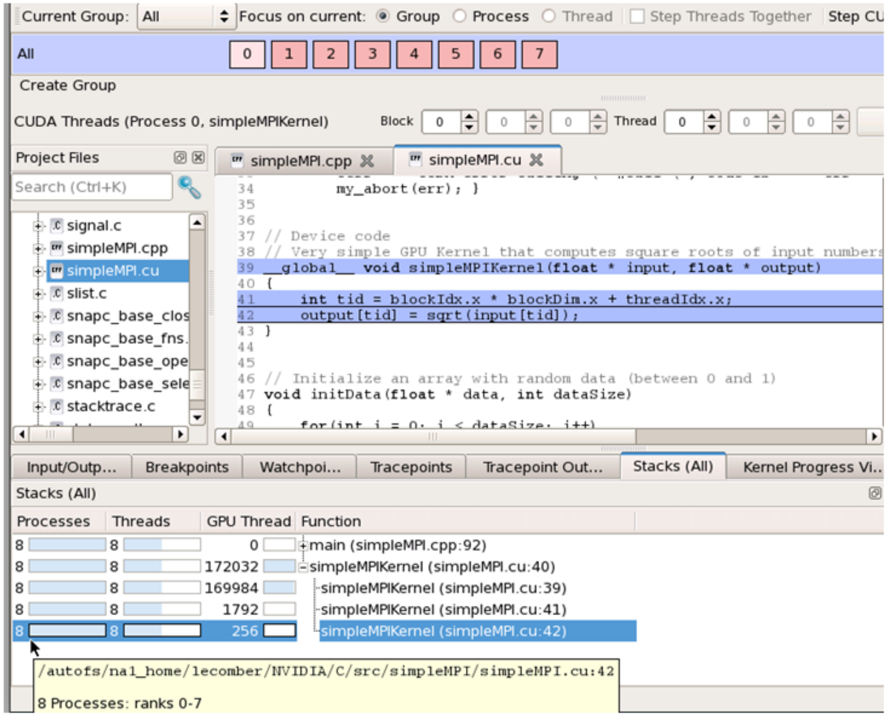
- How can the design of Allinea DDT help your debugging whatever the size of your cluster (workstation or cluster) ?

- High-end architecture (in any Allinea DDT version)
  - Data consolidation (merged and sorted)
  - Highlight differences
  - Very low footprint

allinea
www.allinea.com

Allinea DDT and GPU
Successful add-on

- Built on vendors low level efforts
  - Nvidia cuda-gdb, compiler
  - Cray compiler
- Execution model is unusual
  - GUI supports blocks and grids
  - Support 32 thread units (warps)
- Mixed GPU/CPU in one interface
  - Interaction with CPUs
  - Easy to switch between contexts (stacks, threads, data...)
  - Support multiple nodes



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

Search (Ctrl+K)

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++)
```

Stacks (All)

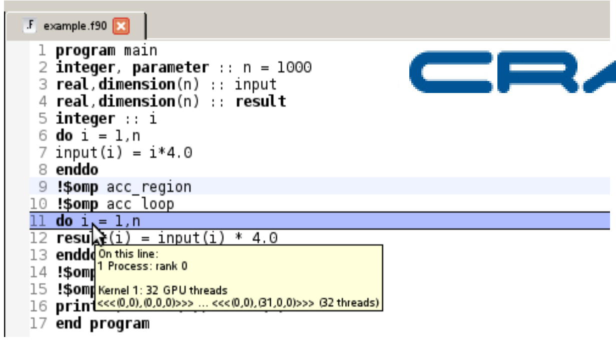
Processes	Threads	GPU Thread	Function
8	8	0	main (simpleMPI.cpp:92)
8	8	172032	simpleMPIKernel (simpleMPI.cu:40)
8	8	169984	simpleMPIKernel (simpleMPI.cu:39)
8	8	1792	simpleMPIKernel (simpleMPI.cu:41)
8	8	256	simpleMPIKernel (simpleMPI.cu:42)

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

8 Processes: ranks 0-7

allinea
www.allinea.com

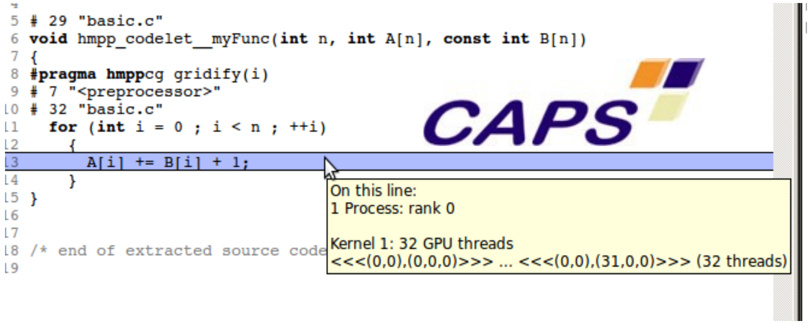
Many environments
Directives support



```
program main
integer, parameter :: n = 1000
real,dimension(n) :: input
real,dimension(n) :: result
integer :: i
do i = 1,n
input(i) = i*4.0
enddo
!$omp acc_region
!$omp acc_loop
do i = 1,n
result(i) = input(i) * 4.0
enddo
!$omp
!$omp
print
end program
```

On this line:
1 Process: rank 0

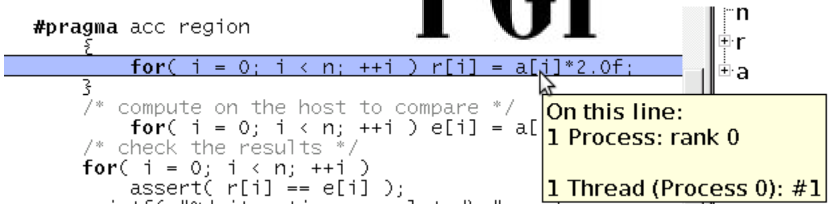
Kernel 1: 32 GPU threads
<<<(0,0),(0,0,0)>>> ... <<<(0,0),(31,0,0)>>> (32 threads)



```
# 29 "basic.c"
void hmpp_codelet__myFunc(int n, int A[n], const int B[n])
{
#pragma hmppcg gridify(i)
# 7 "<preprocessor>"
# 32 "basic.c"
  for (int i = 0 ; i < n ; ++i)
    {
      A[i] += B[i] + 1;
    }
}


/* end of extracted source code
```

On this line:
1 Process: rank 0

Kernel 1: 32 GPU threads
<<<(0,0),(0,0,0)>>> ... <<<(0,0),(31,0,0)>>> (32 threads)



```
#pragma acc region
    {
        for( i = 0; i < n; ++i ) r[i] = a[i]*2.0f;
    }
    /* compute on the host to compare */
        for( i = 0; i < n; ++i ) e[i] = a[
    /* check the results */
    for( i = 0; i < n; ++i )
        assert( r[i] == e[i] );
```

On this line:
1 Process: rank 0

1 Thread (Process 0): #1

- Wide range of partnerships
  - Support the environments you use for hybrid development
  - You swiftly benefit from the latest updates
  - Read our latest white papers available with PGI and CAPS

Many environments
Directives support - OpenACC

- OpenACC : New parallel programming standard led by
  - CAPS Enterprise, Cray, Nvidia and PGI
- Fully supported within Allinea DDT
  - CAPS and PGI : still adapting their software to the new standard
  - Same mechanisms and features for CUDA programs
  - First implementation to be supported: Cray OpenACC
- You benefit from our partnerships

Summary

- Bugs happen at all stage of the development and during the entire life of the code

- An intelligent debugger is now available to fix bugs quickly
  - Other methods have limited success and issues at scale
  - Intelligence makes debugging easier and faster for the developer

- Allinea DDT scales in both performance and interface
  - Breaking all records and making problems now manageable

Thank you

Your contacts :

- Technical Support team : support@allinea.com
- Sales team : sales@allinea.com