

Debugging with GDB and DDT

Ramses van Zon
SciNet HPC Consortium
University of Toronto

May 10, 2013



Outline

- ▶ Debugging Basics
- ▶ Debugging with the command line: GDB
- ▶ Debugging with DDT

Debugging basics

Debugging basics

Help, my program doesn't work!

```
$ gcc -O3 answer.c  
$ ./a.out  
Segmentation fault
```

↓
a miracle occurs
↓

My program works brilliantly!

```
$ gcc -O3 answer.c  
$ ./a.out  
42
```

- Unfortunately, “miracles” are not yet supported by SciNet.

Debugging:

Methodical process of finding and fixing flaws in software

Common symptoms

Errors at compile time

- ▶ Syntax errors: easy to fix
- ▶ Library issues
- ▶ Cross-compiling
- ▶ Compiler warnings

Always switch this on, and fix or understand them!

But just because it compiles does not mean it is correct!

Runtime errors

- ▶ Floating point exceptions
- ▶ Segmentation fault
- ▶ Aborted
- ▶ Incorrect output (nans)

Common issues

Arithmetic	corner cases (<code>sqrt(-0.0)</code>), infinities
Memory access	Index out of range, uninitialized pointers.
Logic	Infinite loop, corner cases
Misuse	wrong input, ignored error, no initialization
Syntax	wrong operators/arguments
Resource starvation	memory leak, quota overflow
Parallel	race conditions, deadlock

What is going on?

- ▶ Almost always, a condition you are sure is satisfied, is not.
- ▶ But your programs likely relies on many such assumptions.
- ▶ First order of business is finding out what goes wrong, and what assumption is not warranted.
- ▶ *Debugger*: program to help detect errors in other programs.
- ▶ **You are the real debugger.**

Ways to debug

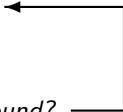
- ▶ Preemptive:
 - ▶ Turn on compiler warnings: fix or understand them!
`$ gcc/gfortran -Wall`
 - ▶ Check your assumptions (e.g. use **assert**).
- ▶ Inspect the exit code and read the error messages!
- ▶ Use a debugger
- ▶ Add print statements ← **No way to debug!**

What's wrong with using print statements?

Strategy

- ▶ Constant cycle:
 1. strategically add print statements
 2. compile
 3. run
 4. analyze output

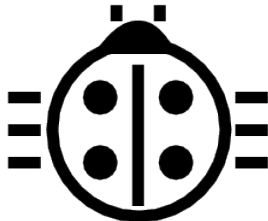
bug not found?


- ▶ Removing the extra code after the bug is fixed
- ▶ Repeat for each bug

Problems with this approach

- ▶ Time consuming
- ▶ Error prone
- ▶ Changes memory, timing... **There's a better way!**

Symbolic debuggers



Symbolic debuggers

Features

1. Crash inspection
2. Function call stack
3. Step through code
4. Automated interruption
5. Variable checking and setting

Use a graphical debugger or not?

- ▶ Local work station: graphical is convenient
- ▶ Remotely (SciNet): can be slow

In any case, graphical and text-based debuggers use the same concepts.

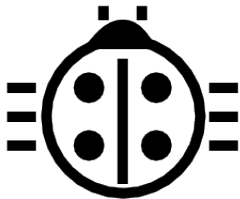
Symbolic debuggers

Preparing the executable

- ▶ Add required compilation flags:
\$ gcc/g++/gfortran -g -gstabs
\$ icc/icpc/ifort -g -debug parallel
\$ nvcc -g -G
- ▶ Optional: switch off optimization -O0

Command-line based symbolic debuggers: gdb

GDB



What is GDB?

- ▶ Free, GNU license, symbolic debugger.
- ▶ Available on many systems.
- ▶ Been around for a while, but still developed and up-to-date
- ▶ Text based, but has a '-tui' option.

```
$ module load gcc
$ gcc -g -O0 example.c -o example
$ module load gdb
$ gdb -tui example
...
(gdb)_
```

GDB basic building blocks



GDB building block #1: Inspect crashes

Inspecting core files

Core = file containing state of program after a crash

- ▶ needs max core size set (**ulimit -c <number>**)
- ▶ gdb reads with **gdb <executable> <corefile>**
- ▶ it will show you where the program crashed

No core file?

- ▶ can start gdb as **gdb <executable>**
- ▶ type **run** to start program
- ▶ gdb will show you where the program crashed if it does.

GDB building block #2: Function call stack

Interrupting program

- ▶ Press Ctrl-C while program is running in gdb
- ▶ gdb will show you where the program was.

Stack trace

- ▶ From what functions was this line reached?
- ▶ What were the arguments of those function calls?

`gdb` commands

<code>backtrace</code>	function call stack
<code>continue</code>	continue
<code>down</code>	go to called function
<code>up</code>	go to caller

GDB building block #3: Step through code

Stepping through code

- ▶ Line-by-line
- ▶ Choose to step into or over functions
- ▶ Can show surrounding lines or use **-tui**

`gdb` commands

list	list part of code
next	continue until next line
step	step into function
finish	continue until function end
until	continue until line/function

GDB building block #4: Automatic interruption

Breakpoints

- ▶ **break** [**file:**]<line>|<function>
- ▶ each breakpoint gets a number
- ▶ when run, automatically stops there
- ▶ can add conditions, temporarily remote breaks, etc.

Related gdb commands

delete	unset breakpoint
condition	break if condition met
disable	disable breakpoint
enable	enable breakpoint
info breakpoints	list breakpoints
tbreak	temporary breakpoint

GDB building block #5: Variables

Checking a variable

- ▶ Can print the value of a variable
- ▶ Can keep track of variable (print at prompt)
- ▶ Can stop the program when variable changes
- ▶ Can change a variable (“what if ...”)

`gdb` commands

<code>print</code>	print variable
<code>display</code>	print at every prompt
<code>set variable</code>	change variable
<code>watch</code>	stop if variable changes

Demonstration GDB

```
$ ssh USER@login.scinet.utoronto.ca -X
$ ssh gpc01 -X
$ qsub -l nodes=1:ppn=8,walltime=4:00:00 -I -X
$ cd $SCRATCH
$ cp -r /scinet/course/ss2013 .
$ cd ss2013/HPC106_debug/code
$ source setup
$ cd ex1
$ make dbgtest #(or dbgtestf)
$ ulimit -c 1024
$ ./dbgtest #(or dbgtestf)
Hello
Hi
You'll find that the latter does not work. Start up
$ gdb -tui dbgtest #(or dbgtestf)
```

Graphical symbolic debuggers



Graphical symbolic debuggers

Features

- ▶ Nice, more intuitive graphical user interface
- ▶ Front to command-line based tools: Same concepts
- ▶ Need graphics support: X forwarding (or VNC)

Available on SciNet: ddd and ddt

▶ ddd

```
$ module load gcc ddd
```

```
$ ddd <executable compiled with -g flag>
```

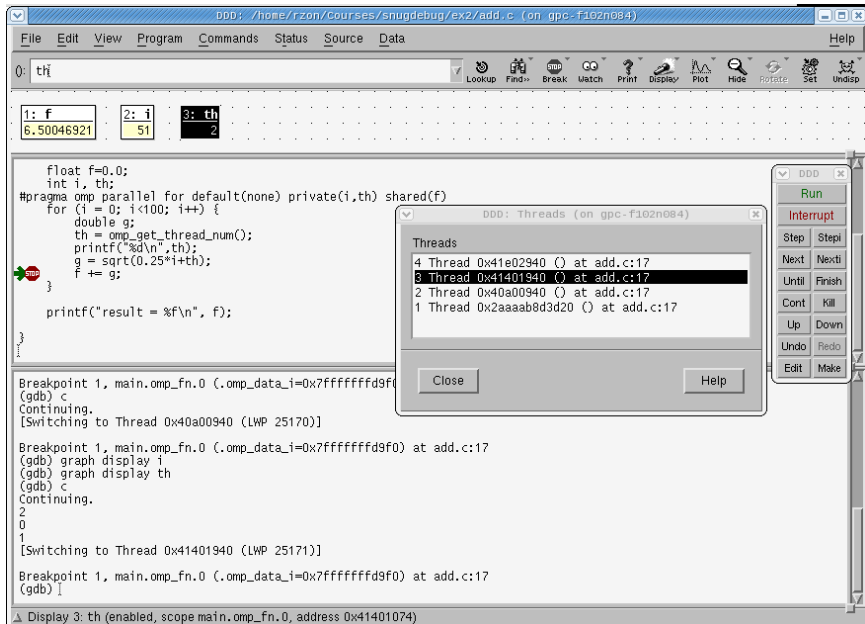
▶ ddt

```
$ module load ddt
```

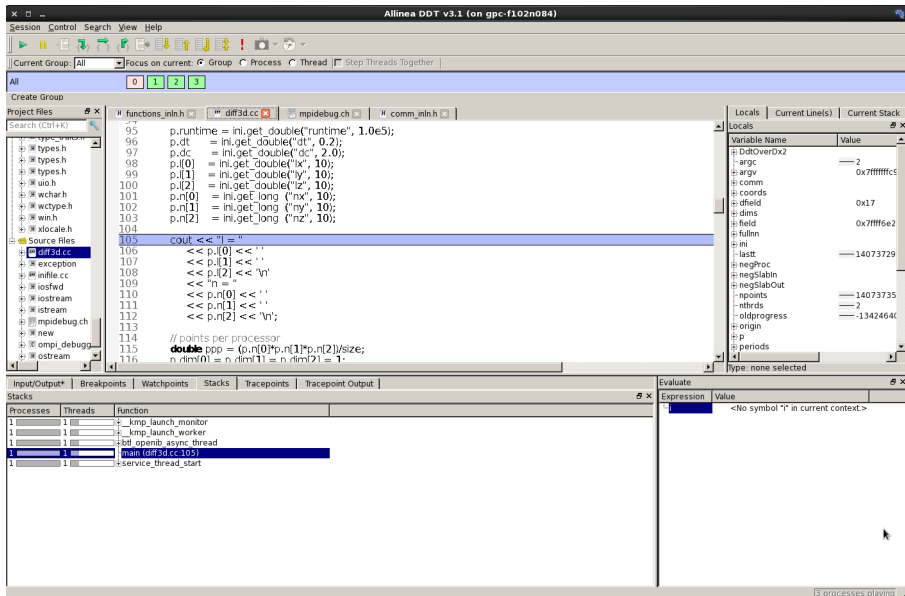
```
$ ddt <executable compiled with -g flag>
```

```
(more later)
```

Graphical symbolic debuggers - ddd



Graphical symbolic debuggers - ddt



Alinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All Focus on current: Group Process Thread Step Threads Together

Create Group

Project Files

Search (Ctrl+F)

Source Files

diff3d.cc

```

95 p.runtime = ini.get_double("runtime", 1.0e5);
96 p.dt = ini.get_double("dt", 0.2);
97 p.dc = ini.get_double("dc", 2.0);
98 p.l[0] = ini.get_double("lx", 10);
99 p.l[1] = ini.get_double("ly", 10);
100 p.l[2] = ini.get_double("lz", 10);
101 p.n[0] = ini.get_long("nx", 10);
102 p.n[1] = ini.get_long("ny", 10);
103 p.n[2] = ini.get_long("nz", 10);
104
105 cout << '=' << '\n';
106 << p.l[0] << ' ';
107 << p.l[1] << ' ';
108 << p.l[2] << '\n';
109 << 'n' << '\n';
110 << p.n[0] << ' ';
111 << p.n[1] << ' ';
112 << p.n[2] << '\n';
113
114 // points per processor
115 double ppp = (p.n[0]*p.n[1]*p.n[2])/size;
116 n.dims[0] = n.dims[1] = n.dims[2] = 1;

```

Locals

Variable Name	Value
DdtOverOx2	2
argc	0x7ffffffc
argv	0x17
comm	0x7ffff6e2
coords	14073729
dfield	14073735
dims	2
field	-1342464
fullnn	
ini	
lastt	
negProc	
negSlabin	
negSlabOut	
npoints	
nthrds	
oldprogress	
origin	
p	
periods	

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output

Stacks

Processes	Threads	Function
1	1	+ _kmp_launch_monitor
1	1	+ _kmp_launch_worker
1	1	+bb_openib_async_thread
1	1	main (diff3d.cc:105)
1	1	+service_thread_start

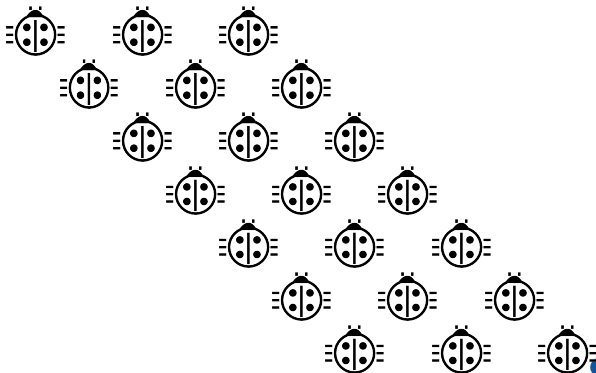
Evaluate

Expression	Value
	<No symbol '*' in current context.>

3 processes playing

compute + calcul CANADA

Parallel debugging



Parallel debugging - 1 Shared memory

Use gdb for

- ▶ Tracking each thread's execution and variables
- ▶ OpenMP serialization: `p omp_set_num_threads(1)`
- ▶ Stepping into OpenMP block: `break` at first line!
- ▶ Thread-specific breakpoint: `b <line> thread <n>`

Use helgrind for

- ▶ Finding race conditions:

```
$ module load valgrind  
$ valgrind --tool=helgrind <exe> &> out  
$ grep <source> out
```

where `<source>` is the name of the source file where you suspect race conditions (valgrind reports a lot more)

Parallel debugging - 2 Distributed memory

Multiple MPI processes

- ▶ Your code is running on different cores!
- ▶ Where to run debugger?
- ▶ Where to send debugger output?
- ▶ Much going on at same time.
- ▶ No universal free solution.

Good approach:

1. Write your code so it can run in serial: perfect that first.
2. Deal with communication, synchronization and deadlock on *smaller* number of MPI processes/threads.
3. Only then try full size.

Parallel debugging demands specialized tools: ddt

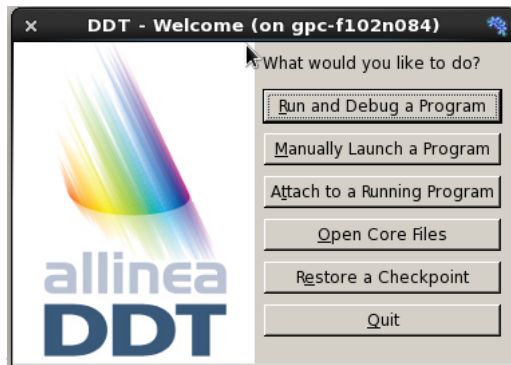
DDT



- ▶ “Distributed Debugging Tool”
- ▶ Powerful GUI-based commercial debugger by *Allinea*.
- ▶ Supports C, C++ and Fortran
- ▶ Supports MPI, OpenMP, threads, CUDA and more
- ▶ Available on all SciNet clusters (GPC, TCS, ARC, P7)
- ▶ Available on SHARCNET’s kraken, requin, orca and monk.

Launching ddt

- ▶ Load your compiler and MPI modules.
- ▶ Load the ddt module: `$ module load ddt`
- ▶ Start ddt with one of these:
 - `$ ddt`
 - `$ ddt <executable compiled with -g flag>`
 - `$ ddt <executable compiled with -g flag> <arguments>`
- ▶ First time: create config file: OpenMPI (skip other steps)
- ▶ Then gui for setting up debug session.



Run and Debug a Program (session setup)

DDT - Run (on gpc-f102n084)

Application: /home/s/scinet/rzon/Code/diff3d/diff3d

Details

Application: /home/s/scinet/rzon/Code/diff3d/diff3d

Arguments:

Input File:

Working Directory:

☒ MPI: 2 processes, OpenMPI

Details

Number of processes: 2

Implementation: OpenMPI, no queue

Change...

mpirun arguments:

☒ OpenMP: 4 threads

Details

Number of OpenMP threads: 4

☐ CUDA

Details

☒ Memory Debugging: Minimal, No guard pages, Backtraces, Preload

Details...

Environment Variables: none

Details

Plugins: none

Details

Run

Cancel

Memory Debugging Options (on gpc-f102n084)

☒ Preload the memory debugging library: Language: C++, threads

Note: Preloading only works for programs linked against shared libraries. If your program is statically linked, you must relink it against the dmalloc library manually

Heap Debugging

☒ Minimal (fewest tests, picks up invalid pointers passed to memory functions)

☐ Runtime (fast, basic tests including fence-post checking, null handling)

☐ Low (adds minimal heap checking, overwriting of allocated/freed space)

☐ Medium (adds full heap checking, always relocates block on realloc)

☐ High (adds checking for arguments to common functions)

☐ Custom:

Heap Overflow/Underflow Detection

☐ Add guard pages to detect out of bounds heap access

Guard pages: 1

Add guard pages: After

Advanced

☐ Specify heap-check interval: 100

☒ Store stack backtraces for memory allocations

☐ Only enable for these processes:

0-1

100%

Select All

x2

x0.5

1%

OK

Cancel

User interface (1)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All Focus on current: Group Process Thread Step Threads Together

All 0 1 2 3

Root 0

Workers 1 2 3

Create Group

Project Files Search (Ctrl+K)

- del_opv.cc
- del_opvnt.cc
- delete.c
- diff3d.cc
- distances.c
- divtf3.c

```
74 }
75 // MPI::COMM_WORLD.Abort(1);
76 }
77
78 const int nthrds = get_num_threads();
79 const int root = 0;
80 const int size = MPI::COMM_WORLD.Get_size();
81 int rank = MPI::COMM_WORLD.Get_rank();
82
83 cerr << "nthrds=" << nthrds << endl;
84
85 //include "mpidebug.ch"
86
87 mpiCommit<Parameters>();
88
```

Locals Current Line(s) Current Stack

Current Line(s)

Variable Name	Value
MPI::COMM_...	
rank	32767

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

Ready

User interface (2)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: A

DDT uses a tabbed-document interface.

All: 0 1 2 3

Root: 0

Workers: 1 2 3

Create Group

Project Files

Search (Ctrl+K)

del_opv.cc

del_opvnt.cc

delete.c

diff3d.cc

distances.c

divtf3.c

diff3d.cc

```
74 }
75 // MPI::COMM_WORLD.Abort(1);
76 }
77
78 const int nthrds = get_num_threads();
79 const int root = 0;
80 const int size = MPI::COMM_WORLD.Get_size();
81 int rank = MPI::COMM_WORLD.Get_rank();
82
83 cerr << "nthrds=" << nthrds << endl;
84
85 // #include "mpidebug.ch"
86
87 mpiCommit<Parameters>();
88
```

Locals

Current Line(s)

Current Stack

Current Line(s)

Variable Name	Value
MPI::COMM_...	
rank	32767

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

Ready

compute • calcul
CANADA

User interface (3)

When the session begins, DDT automatically finds source code from information compiled in the executable.

The screenshot displays the Allinea DDT v3.1 (on gpc-f102n084) interface. A text box highlights that DDT automatically finds source code from information compiled in the executable. The interface shows the 'diff3d.cc' source file with the following code snippet:

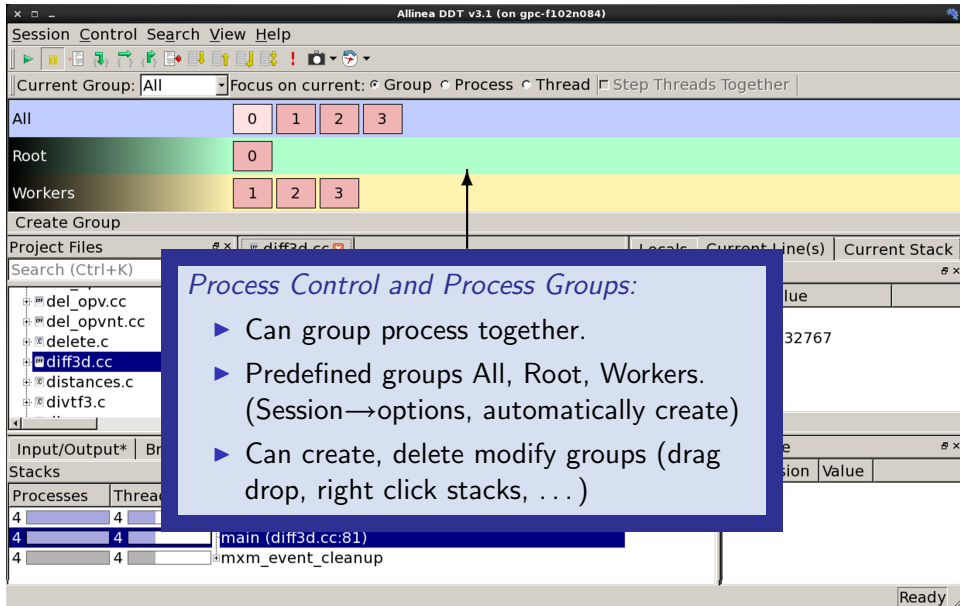
```
74 }
75 // MPI::COMM_WORLD::abort(1);
76 }
77
78 const int nthrds = get_num_threads();
79 const int root = 0;
80 const int size = MPI::COMM_WORLD::Get_size();
81 int rank = MPI::COMM_WORLD::Get_rank();
82
83 cerr << "nthrds=" << nthrds << endl;
84
85 //include "mpidebug.ch"
86
87 mpiCommit<Parameters>();
88
```

The 'Locals' panel shows the variable 'rank' with the value 32767.

The 'Stacks' panel shows the following stack:

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

User interface (4)



The screenshot shows the Allinea DDT v3.1 (on gpc-f102n084) interface. At the top, there's a menu bar with 'Session', 'Control', 'Search', 'View', and 'Help'. Below it is a toolbar with various icons. A 'Current Group' dropdown is set to 'All'. To its right are radio buttons for 'Group', 'Process', and 'Thread', with 'Group' selected. A checkbox 'Step Threads Together' is also present. The main area displays three process groups: 'All' (blue bar with sub-groups 0, 1, 2, 3), 'Root' (green bar with sub-group 0), and 'Workers' (yellow bar with sub-groups 1, 2, 3). An arrow points from the 'Workers' group towards a call stack window. The call stack window shows a list of files: 'del_opv.cc', 'del_opvnt.cc', 'delete.c', 'diff3d.cc' (highlighted), 'distances.c', and 'divtf3.c'. Below this, there's a section for 'Input/Output*' and 'Br'. The 'Stacks' section shows 'Processes' and 'Thread' columns. The 'Processes' column lists '4' for each entry. The 'Thread' column lists '4' for each entry. The 'main' thread is highlighted, showing 'main (diff3d.cc:81)'. Below it, 'mxm_event_cleanup' is listed. A blue callout box is overlaid on the interface, containing the text 'Process Control and Process Groups:' followed by three bullet points. The bottom right corner of the interface shows a 'Ready' status.

Process Control and Process Groups:

- ▶ Can group process together.
- ▶ Predefined groups All, Root, Workers. (Session→options, automatically create)
- ▶ Can create, delete modify groups (drag drop, right click stacks, ...)

User interface (5)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All

All

Root

Workers

Create Group

Project Files

Search (Ctrl+K)

diff3d.cc

del_opv.cc

del_opvnt.cc

delete.c

diff3d.cc

distances.c

divtf3.c

diff3d.cc

```
74 }
75 // MPI::COMM_WORLD.Abort(1);
76 }
77
78 const int nthrds = get_num_threads();
79 const int root = 0;
80 const int size = MPI::COMM_WORLD.Get_size();
81 int rank = MPI::COMM_WORLD.Get_rank();
82
83 cerr << "nthrds=" << nthrds << endl;
84
85 //include "mpidebug.ch"
86
87 mpiCommit<Parameters>();
88
```

Locals

Current Line(s)

Current Stack

Current Line(s)

Variable Name	Value
MPI::COMM_...	
rank	32767

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

Ready

compute • calcul
CANADA

User interface (6)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All Focus on current: Group Process Thread Step Threads Together

All 0 1 2 3

Root

Workers

Create Group

Project Files

Search (Ctrl+K)

Session Control Dialog:
Control program execution, e.g., play/continue, pause, step into, step over, step out

del_opv.cc
del_opvnt.cc
delete.c
diff3d.cc
distances.c
divtf3.c

```
77 const int nthrds = get_num_threads();  
78 const int root = 0;  
79 const int size = MPI::COMM_WORLD.Get_size();  
80 int rank = MPI::COMM_WORLD.Get_rank();  
81  
82 cerr << "nthrds=" << nthrds << endl;  
83  
84  
85 // #include "mpidebug.ch"  
86  
87 mpiCommit<Parameters>();  
88
```

Variable Name Value

MPI::COMM_...
rank 32767

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes Threads Function

4 4 gomp_thread_start (team.c:120)

4 4 main (diff3d.cc:81)

4 4 mxm_event_cleanup

Ready

compute • calcul
CANADA

User interface (7)

The screenshot displays the Allinea DDT v3.1 (on gpc-f102n084) interface. The top menu bar includes Session, Control, Search, View, and Help. Below the menu is a toolbar with various icons. The main window is divided into several panes. The top pane shows the 'Current Group' as 'All' and 'Focus on current' with radio buttons for Group, Process, and Thread. Below this are three rows of colored boxes representing different groups: 'All' (blue), 'Root' (green), and 'Workers' (yellow). The 'All' group has four boxes labeled 0, 1, 2, and 3. The 'Root' group has one box labeled 0. The 'Workers' group has three boxes labeled 1, 2, and 3. Below these is a 'Create Group' button. The middle pane shows 'Project Files' with a file named 'diff3d.cc' selected. To the right of this pane are tabs for 'Locals', 'Current Line(s)', and 'Current Stack'. The bottom pane is divided into two sections. The left section is the 'Breakpoints Tab', which is highlighted by a blue box with the text 'Breakpoints Tab' and 'Can suspend, jump to, delete, load, save'. This section contains a list of files with breakpoints: 'del_opv.cc', 'del_opvnt.cc', 'delete.c', 'diff3d.cc' (selected), 'distances.c', and 'divtf3.c'. The right section is the 'Stacks' pane, which shows a table of processes and threads. The table has columns for 'Processes', 'Threads', and 'Function'. The first row shows 'gomp_thread_start (team.c:120)'. The second row shows 'main (diff3d.cc:81)'. The third row shows 'mxm_event_cleanup'. The bottom right corner of the interface shows a 'Ready' status bar.

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All Focus on current: Group Process Thread Step Threads Together

All 0 1 2 3

Root 0

Workers 1 2 3

Create Group

Project Files diff3d.cc Locals Current Line(s) Current Stack

Search (Ctrl+K)

del_opv.cc

del_opvnt.cc

delete.c

diff3d.cc

distances.c

divtf3.c

Breakpoints Tab

Can suspend, jump to, delete, load, save

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

Ready

User interface (8)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All Focus on current: ☒ Group ☐ Process ☐ Thread ☐ Step Threads Together

All 0 1 2 3

Root 0

Workers

Create Group

Project Files

Search (Ctrl+K)

del_opv.cc
del_opvnt.cc
delete.c
diff3d.cc
distances.c
divtf3.c

```
75 // MPI::COMM_WORLD.Abort(1);  
76 )  
77  
78 const int nthreads = get_num_threads();  
79 const int root = 0;  
80 const int size = MPI::COMM_WORLD.Get_size();  
81 int rank = MPI::COMM_WORLD.Get_rank();  
82  
83 cerr << "nthreads=" << nthreads << endl;  
84  
85 // #include "mpidebug.ch"  
86  
87 mpiCommit<Parameters>();  
88
```

Current Line(s)

Variable Name	Value
MPI::COMM_...	
rank	32767

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

Ready

compute • calcul
CANADA

User interface (9)

The screenshot shows the Allinea DDT v3.1 (on gpc-f102n084) interface. A blue callout box titled "Stacks: Current and Parallel" lists the following features:

- ▶ Tree of functions (merged)
- ▶ Click on branch to see source
- ▶ Hover to see process ranks
- ▶ Use to gather processes in new groups

The interface includes a menu bar (Session, Control, Search, View, Help), a toolbar, and a status bar. The main window is divided into several panels:

- Current Group:** All
- Focus on current:** C Group, C Process, C Thread, Step, Threads Together
- Locals:** Current Line(s), Current Stack
- Current Line(s):** MPI::COMM_... rank (Value: 32767)
- Type:** none selected
- Input/Output*, Breakpoints, Watchpoints, Stacks, Tracepoints, Tracepoint Output, Evaluate**
- Stacks:** Expression, Value
- Processes:** 4, 4, 4
- Threads:** 4, 4, 4
- Function:** gomp_thread_start (team.c:120), main (diff3d.cc:81), mxm_event_cleanup

The code editor shows the following code:

```
82  
83 cerr << "nthreads=" << nthreads << endl;  
84  
85 // #include "mpidebug.ch"  
86  
87 mpiCommit<Parameters>();  
88
```

User interface (10)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All

Current line variables

All: 0 1 2 3

Root: 0

Workers: 1 2 3

Create Group

Project Files

Search (Ctrl+K)

del_opv.cc
del_opvnt.cc
delete.c
diff3d.cc
distances.c
divtf3.c

diff3d.cc

```
74 }  
75 // MPI::COMM_WORLD.Abort(1);  
76 }  
77  
78 const int nthrds = get_num_threads();  
79 const int root = 0;  
80 const int size = MPI::COMM_WORLD.Get_size();  
81 int rank = MPI::COMM_WORLD.Get_rank();  
82  
83 cerr << "nthrds=" << nthrds << endl;  
84  
85 // #include "mpidebug.ch"  
86  
87 mpiCommit<Parameters>();  
88
```

Locals Current Line(s) Current Stack

Current Line(s)

Variable Name	Value
MPI::COMM_...	
rank	32767

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

Ready

User interface (11)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All

Local variables for process

All

Root

Workers

Create Group

Project Files

Search (Ctrl+K)

diff3d.cc

```
74 }
75 // MPI::COMM_WORLD.Abort(1);
76 }
77
78 const int nthrds = get_num_threads();
79 const int root = 0;
80 const int size = MPI::COMM_WORLD.Get_size();
81 int rank = MPI::COMM_WORLD.Get_rank();
82
83 cerr << "nthrds=" << nthrds << endl;
84
85 // #include "mpidebug.ch"
86
87 mpiCommit<Parameters>();
88
```

Locals

Current Line(s)

Current Stack

Variable Name	Value
MPI::COMM_...	
rank	32767

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

Ready

compute • calcul
CANADA

User interface (12)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All

Evaluate window

All: 0 1 2 3

Root: 0

Workers: 1 2 3

Create Group

Project Files

Search (Ctrl+K)

del_opv.cc
del_opvnt.cc
delete.c
diff3d.cc
distances.c
divtf3.c

diff3d.cc

```
74 }  
75 // MPI::COMM_WORLD.Abort(1);  
76 }  
77  
78 const int nthrds = get_num_threads();  
79 const int root = 0;  
80 const int size = MPI::COMM_WORLD.Get_size();  
81 int rank = MPI::COMM_WORLD.Get_rank();  
82  
83 cerr << "nthrds=" << nthrds << endl;  
84  
85 // #include "mpidebug.ch"  
86  
87 mpiCommit<Parameters>();  
88
```

Locals Current Line(s) Current Stack

Current Line(s)

Variable Name	Value
MPI::COMM_...	
rank	32767

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Evaluate

Stacks

Processes	Threads	Function
4	4	gomp_thread_start (team.c:120)
4	4	main (diff3d.cc:81)
4	4	mxm_event_cleanup

Ready

compute • calcul
CANADA

Demonstration DDT

```
$ cd $SCRATCH/ss2013/HPC106_debug/code  
$ source setup  
$ cd ex2  
$ make  
$ ddt ex2
```

Other features of DDT (1)

- ▶ Some of the user-modified parameters and windows are saved by right-clicking and selecting a save option in the corresponding window (Groups; Evaluations)
- ▶ DDT can load and save sessions.
- ▶ *Find* and *Find in Files* in the Search menu.
- ▶ *Goto line* in Search menu (or Ctrl-G)
- ▶ Synchronize processes in group: Right-click, “Run to here”.
- ▶ View multiple source codes simultaneously: Right-click, “Split”
- ▶ Right-click power!

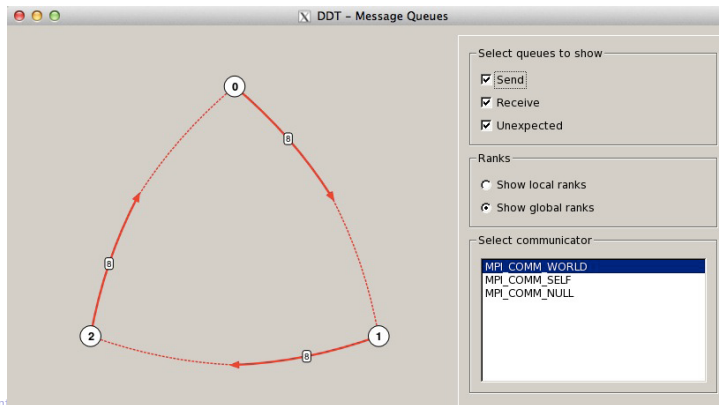
Other features of DDT (2)

- ▶ Signal handling: SEGV, FPE, PIPE,ILL
- ▶ Support for Fortran modules
- ▶ Change data values in evaluate window
- ▶ Examine pointers (vector, reference, dereference)
- ▶ Multi-dimensional arrays
- ▶ Viewer

Other features of DDT (3)

Message Queue

- ▶ View → show message queue
- ▶ produces both a graphical view and table for active communications
- ▶ Helps to find e.g. deadlocks



Demonstration DDT

```
$ cd $SCRATCH/ss2013/HPC106_debug/code  
$ source setup  
$ cd ex3  
$ make  
$ ddt ex3
```

Other features of DDT (4)

Memory debugging

- ▶ Select “memory debug” in Run window
- ▶ Stops on error (before crash or corruption)
- ▶ Check pointer (right click in evaluate)
- ▶ View, overall memory stats

Demonstration DDT

```
$ cd $SCRATCH/ss2013/HPC106_debug/code  
$ source setup  
$ cd ex4  
$ make  
$ ddt ex4
```

Useful references

- ▶ G Wilson
Software Carpentry software-carpentry.org/3_0/debugging.html
- ▶ N Matloff and PJ Salzman
The Art of Debugging with GDB, DDD and Eclipse
- ▶ *GDB*: sources.redhat.com/gdb
- ▶ *DDT*: www.allinea.com/products/ddt-support
- ▶ *SciNet Wiki*: wiki.scinethpc.ca: Tutorials & Manuals