Parallel Debugging with DDT

Ramses van Zon, Jonathan Dursi
SciNet HPC Consortium
University of Toronto

November 28, 2012

Outline

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

SciNet compute • calcul CANADA

Debugging basics

SciNet compute • calcul CANADA

Debugging basics

SciNet compute • calcul CANADA

Debugging basics

Help, my program doesn't work!

SciNet compute • calcul CANADA

Debugging basics

Help, my program doesn't work!

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

compute • calcul CANADA

Debugging basics

Help, my program doesn't work!

↓

a miracle occurs

↓

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

SciNet compute • calcul CANADA

Debugging basics

Help, my program doesn't work!

a miracle occurs

My program works brilliantly!

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


SciNet compute • calcul CANADA

Debugging basics

Help, my program doesn't work!

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

↓

a miracle occurs

↓

My program works brilliantly!

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


SciNet compute • calcul CANADA

Debugging basics

Help, my program doesn't work!

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

↓

a miracle occurs

↓

My program works brilliantly!

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

- Unfortunately, "miracles" are not yet supported by SciNet.

SciNet compute • calcul CANADA

Debugging basics

Help, my program doesn't work!

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

↓

a miracle occurs

↓

My program works brilliantly!

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

- Unfortunately, "miracles" are not yet supported by SciNet.

Debugging:

Methodical process of finding and fixing flaws in software

SciNet compute • calcul CANADA

Common symptoms

Errors at compile time

SciNet compute • calcul CANADA

Common symptoms

Errors at compile time

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

SciNet compute • calcul CANADA

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!

SciNet compute • calcul CANADA

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!

SciNet compute • calcul CANADA

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

SciNet compute • calcul CANADA

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)

SciNet compute • calcul CANADA

Common issues

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

SciNet compute • calcul CANADA

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.
- A debugger is a program to help detect errors in other programs.
- **You are the real debugger.**

SciNet compute • calcul CANADA

Ways to debug

SciNet compute • calcul CANADA

Ways to debug

- Preemptive:
  - Turn on compiler warnings: fix or understand them!
  - Check your assumptions (e.g. use `assert`).

SciNet compute • calcul CANADA

Ways to debug

- Preemptive:
  - Turn on compiler warnings: fix or understand them!
  - Check your assumptions (e.g. use `assert`).
- Inspect the exit code and read the error messages!

SciNet compute • calcul CANADA

Ways to debug

- Preemptive:
  - Turn on compiler warnings: fix or understand them!
  - Check your assumptions (e.g. use `assert`).
- Inspect the exit code and read the error messages!
- Add print statements

SciNet compute • calcul CANADA

Ways to debug

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

SciNet compute • calcul CANADA

Ways to debug

SciNet compute • calcul CANADA

Ways to debug

- Command-line based, symbolic debuggers
  - GNU debugger: *gdb*
  - Intel debugger command-line: *idbc*

SciNet
compute • calcul
CANADA

Ways to debug

- Command-line based, symbolic debuggers
  - GNU debugger: *gdb*
  - Intel debugger command-line: *idbc*
- Symbolic debuggers with Graphical User Interface
  - GNU data display debugger: *ddd*
  - Intel debugger: *idb*
  - IDEs: Eclipse, NetBeans (neither on SciNet), *emacs/gdb*
  - Allinea DDT: *ddt*
  - Rogue Wave TotalView (not available at SciNet)

SciNet compute • calcul CANADA

What's wrong with using print statements?

Strategy

SciNet compute • calcul CANADA

What's wrong with using print statements?

Strategy

- Constant cycle:

What's wrong with using print statements?

Strategy

- Constant cycle:
  1. strategically add print statements

What's wrong with using print statements?

Strategy

- Constant cycle:
  1. strategically add print statements
  2. compile

SciNet compute • calcul CANADA

What's wrong with using print statements?

Strategy

- Constant cycle:
  1. strategically add print statements
  2. compile
  3. run

SciNet compute • calcul CANADA

What's wrong with using print statements?

Strategy

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

SciNet compute • calcul CANADA

What's wrong with using print statements?

Strategy

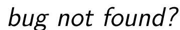
- Constant cycle:
  1. strategically add print statements
  2. compile
  3. run
  4. analyze output *bug not found?*

SciNet compute • calcul CANADA

What's wrong with using print statements?

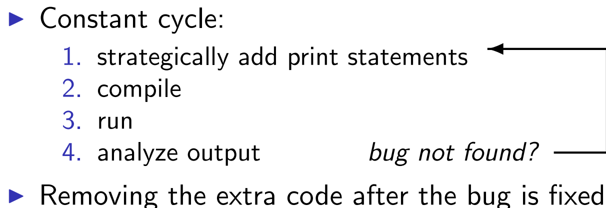
Strategy

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



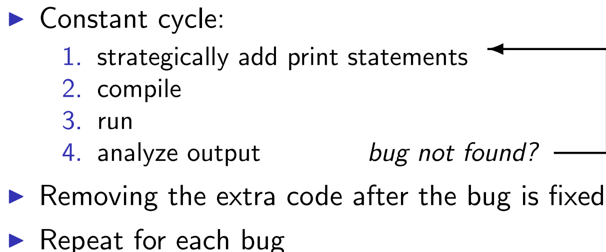
SciNet compute • calcul CANADA

What's wrong with using print statements?



What's wrong with using print statements?

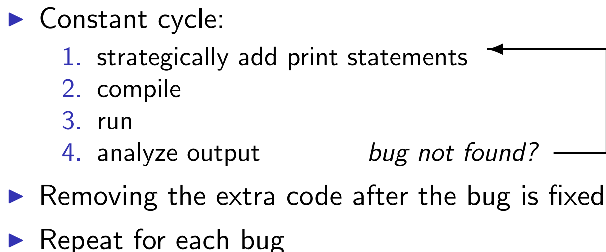
Strategy



- Constant cycle:
  1. strategically add print statements
  2. compile
  3. run
  4. analyze output *bug not found?* (return to 1)
- Removing the extra code after the bug is fixed
- Repeat for each bug

What's wrong with using print statements?

Strategy

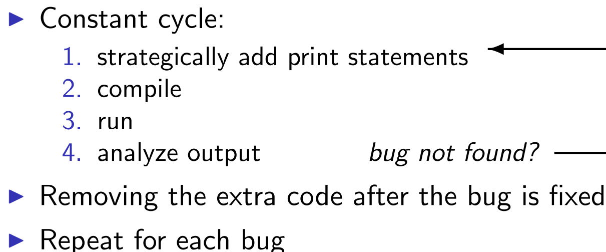


Problems with this approach

What's wrong with using print statements?

Strategy

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



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

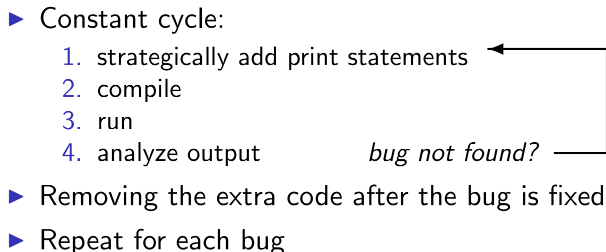
Problems with this approach

- Time consuming
- Error prone
- Changes memory, timing. . .

SciNet compute • calcul CANADA

What's wrong with using print statements?

Strategy

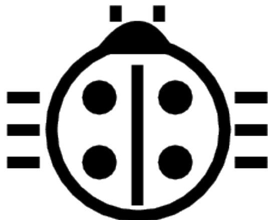


- Constant cycle:
  1. strategically add print statements
  2. compile
  3. run
  4. analyze output *bug not found?*
- Removing the extra code after the bug is fixed
- Repeat for each bug

Problems with this approach

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

Symbolic debuggers



SciNet compute • calcul CANADA

Symbolic debuggers

Features

SciNet compute • calcul CANADA

Symbolic debuggers

Features

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

SciNet compute • calcul CANADA

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?

SciNet compute • calcul CANADA

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):
  - Some graphical debuggers slow (connection)
  - Command-line based debuggers fast (esp. gdb).
  - Ddt: gui-based, with graphics light enough to work remotely.
- Graphical and text-based debuggers use the same concepts.

SciNet compute • calcul CANADA

Symbolic debuggers

Preparing the executable

- Required: compile with `-g`.
- Optional: switch off optimization `-O0`
- Same for gcc, g++, gfortran, icc, ifort, xlf, mpif90, mpicc, ...
- For nvcc (i.e. cuda), also add `-G`

SciNet compute • calcul CANADA

Symbolic debuggers

Preparing the executable

- Required: compile with `-g`.
- Optional: switch off optimization `-O0`
- Same for gcc, g++, gfortran, icc, ifort, xlf, mpif90, mpicc, . . .
- For nvcc (i.e. cuda), also add `-G`

Command-line based symbolic debuggers

- gdb

SciNet compute • calcul CANADA

Symbolic debuggers

Preparing the executable

- Required: compile with `-g`.
- Optional: switch off optimization `-O0`
- Same for gcc, g++, gfortran, icc, ifort, xlf, mpif90, mpicc, . . .
- For nvcc (i.e. cuda), also add `-G`

Command-line based symbolic debuggers

- gdb ← **Focus on this one**
- idbc

SciNet compute • calcul CANADA

Symbolic debuggers

Preparing the executable

- Required: compile with `-g`.
- Optional: switch off optimization `-O0`
- Same for gcc, g++, gfortran, icc, ifort, xlf, mpif90, mpicc, ...
- For nvcc (i.e. cuda), also add `-G`

Command-line based symbolic debuggers

- gdb ← **Focus on this one**
- idbc ← **Has gdb mode**

SciNet compute • calcul CANADA

Symbolic debuggers

Preparing the executable

- Required: compile with `-g`.
- Optional: switch off optimization `-O0`
- Same for gcc, g++, gfortran, icc, ifort, xlf, mpif90, mpicc, ...
- For nvcc (i.e. cuda), also add `-G`

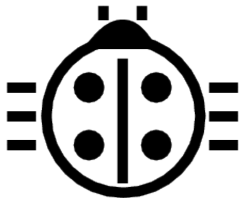
Command-line based symbolic debuggers

- gdb ← **Focus on this one**
- idbc ← **Has gdb mode**

```
$ module load intel
$ icc -g -O0 example.c -o example
$ module load gdb
$ gdb example
...
(gdb)_
```


SciNet compute • calcul CANADA

GDB



SciNet compute • calcul CANADA

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.

SciNet compute • calcul CANADA

GDB command summary

`help`	`h`	print description of command
`run`	`r`	run from the start (+args)
`backtrace/where`	`ba`	function call stack
`break`	`b`	set breakpoint
`delete`	`d`	delete breakpoint
`continue`	`c`	continue
`step`	`s`	step into function
`next`	`n`	continue until next line
`print`	`p`	print variable
`quit`	`q`	quit
`finish`	`fin`	continue until function end
`set variable`	`set var`	change variable
`down`	`do`	go to called function
`tbreak`	`tb`	set temporary breakpoint
`until`	`unt`	continue until line/function
`up`	`up`	go to caller
`watch`	`wa`	stop if variable changes
`watch`	`wa`	stop if variable changes
`quit`	`q`	quit gdb

SciNet
compute • calcul
CANADA

GDB basic building blocks

SciNet compute • calcul CANADA

GDB building blocks #1: Inspect crashes

Inspecting core files

SciNet
compute • calcul
CANADA

GDB building blocks #1: Inspect crashes

Inspecting core files

Core = file containing state of program after a crash

SciNet compute • calcul CANADA

GDB building blocks #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

SciNet compute • calcul CANADA

GDB building blocks #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?

SciNet compute • calcul CANADA

GDB building blocks #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.

SciNet compute • calcul CANADA

GDB building blocks #2: Function call stack

Interrupting program

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

SciNet compute • calcul CANADA

GDB building blocks #2: Function call stack

Interrupting program

- Press Crtl-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?

SciNet
compute • calcul
CANADA

GDB building blocks #2: Function call stack

Interrupting program

- Press Crtl-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

`backtrace`	function call stack
`continue`	continue
`down`	go to called function
`up`	go to caller

SciNet compute • calcul CANADA

GDB building blocks #3: Step through code

Stepping through code

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

SciNet compute • calcul CANADA

GDB building blocks #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

SciNet compute • calcul CANADA

GDB building blocks #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.

SciNet compute • calcul CANADA

GDB building blocks #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

SciNet compute • calcul CANADA

GDB building blocks #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 ...")

SciNet compute • calcul CANADA

GDB building blocks #5: Variables

Checking a variable

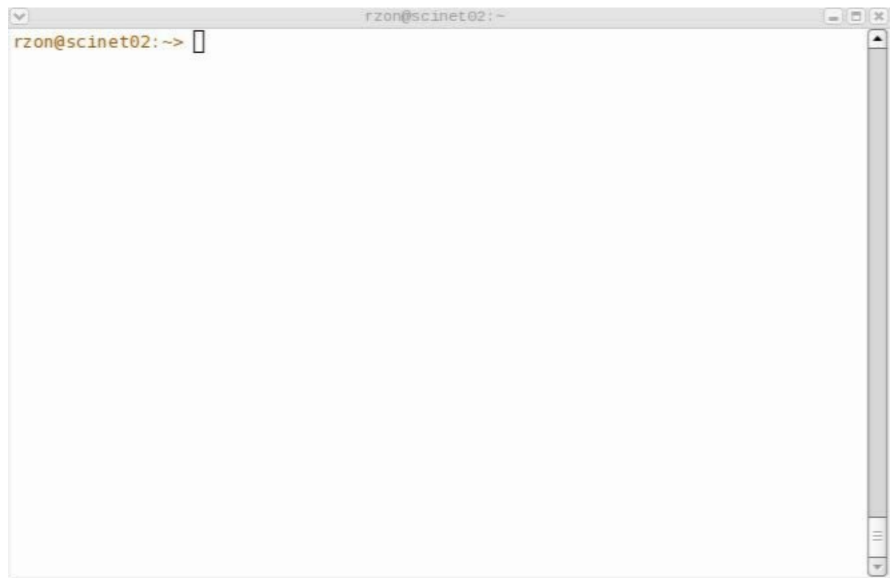
- Can print the value of a variable
- Can keep track of variable (print at prompt)
- Can stop the program when variable changes
- Can change a variable ("what if …")

gdb commands

Command	Description
`print`	print variable
`display`	print at every prompt
`set variable`	change variable
`watch`	stop if variable changes

SciNet compute • calcul CANADA

Demonstration GDB



Graphical symbolic debuggers

SciNet compute • calcul CANADA

Graphical symbolic debuggers

Features

- Nice, more intuitive graphical user interface
- Front to command-line based tools: Same concepts
- Need graphics support (`qsub -X -I ...`)

SciNet compute • calcul CANADA

Graphical symbolic debuggers

Features

- Nice, more intuitive graphical user interface
- Front to command-line based tools: Same concepts
- Need graphics support (`qsub -X -I ...`)

Available on SciNet

- ddd
  ```
  $ module load gcc ddd
  $ ddd <executable compiled with -g flag>
  ```
- idb
  ```
  $ module load intel java
  ```
  Java slow remotely
  ```
  $ idb <executable compiled with -g flag>
  ```
- ddt
  ```
  $ module load ddt
  ```
  (more later)

SciNet compute • calcul CANADA

Graphical symbolic debuggers - ddd

DDD: /home/rzon/Courses/snugdebug/ex2/add.c (on gpc-f102n084)

File Edit View Program Commands Status Source Data Help

(): th

Lookup Find» Break Watch Print Display Plot Hide Rotate Set Undisp

1: f 6.50046921 | 2: i 51 | 3: th 2

```
    float f=0.0;
    int i, th;
#pragma omp parallel for default(none) private(i,th) shared(f)
    for (i = 0; i<100; i++) {
        double g;
        th = omp_get_thread_num();
        printf("%d\n",th);
        g = sqrt(0.25*i+th);
        f += g;
    }

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

}
```

DDD: Threads (on gpc-f102n084)

Threads

```
4 Thread 0x41e02940 () at add.c:17
3 Thread 0x41401940 () at add.c:17
2 Thread 0x40a00940 () at add.c:17
1 Thread 0x2aaaab8d3d20 () at add.c:17
```

Close Help

DDD: Run, Interrupt, Step, Stepi, Next, Nexti, Until, Finish, Cont, Kill, Up, Down, Undo, Redo, Edit, Make

```
Breakpoint 1, main.omp_fn.0 (.omp_data_i=0x7fffffffd9f0
(gdb) c
Continuing.
[Switching to Thread 0x40a00940 (LWP 25170)]

Breakpoint 1, main.omp_fn.0 (.omp_data_i=0x7fffffffd9f0) at add.c:17
(gdb) graph display i
(gdb) graph display th
(gdb) c
Continuing.
2
0
1
[Switching to Thread 0x41401940 (LWP 25171)]

Breakpoint 1, main.omp_fn.0 (.omp_data_i=0x7fffffffd9f0) at add.c:17
(gdb)
```

Display 3: th (enabled, scope main.omp_fn.0, address 0x41401074)

Graphical symbolic debuggers - idb

Intel(R) Debugger

File Edit View Run Debug Parallel Options Help

$lasteventingthread

add.c

```
 9      float f=0.0;
10      int i, th;
11 #pragma omp parallel for default(none) private(i,th) shared(f)
12      for (i = 0; i<100; i++) {
13          double g;
14          th = omp_get_thread_num();
15          printf("%d\n",th);
16          g = sqrt(0.25*i+th);
17          f += g;
18      }
19
20      printf("result = %f\n", f);
```

Console | Threads

ID	Type	OS ID	Thread Library ID	Execution Attribute	Location
$allthreads					
1	nati	2606'	469387906204	thawed	void main.omp_fn.0(void) "/home/rzon/(
2	nati	2608:	1084623168	thawed	void main.omp_fn.0(void) "/home/rzon/(
3	nati	2608	1113561408	thawed	void main.omp_fn.0(void) "/home/rzon/(
4	nati	2608!	1124051264	thawed	<opaque> __write_nocancel(void)
$currentopenmpteam					

0x0000000000400887 in main.omp_fn.0 (.omp_d..."/home/rzon/Courses/snugdebug/ex2/add.c":17

Graphical symbolic debuggers - ddt

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

Create Group

Project Files

functions_inln.h diff3d.cc mpidebug.ch comm_inln.h

```
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 << "l = "
106          << p.l[0] << ' '
107          << p.l[1] << ' '
108          << p.l[2] << '\n'
109          << "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     p.dim[0] = p.dim[1] = p.dim[2] = 1;
```

Locals Current Line(s) Current Stack

Variable Name	Value
DdtOverDx2	
argc	2
argv	0x7fffffffc9
comm	
coords	
dfield	0x17
dims	
field	0x7ffff6e2
fullnn	
ini	
lastt	14073729
negProc	
negSlabIn	
negSlabOut	
npoints	14073735
nthrds	2
oldprogress	-1342464(
origin	
p	
periods	

Type: none selected

Input/Output* Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output

Stacks

Processes	Threads	Function
1	1	__kmp_launch_monitor
1	1	__kmp_launch_worker
1	1	btl_openib_async_thread
1	1	main (diff3d.cc:105)
1	1	service_thread_start

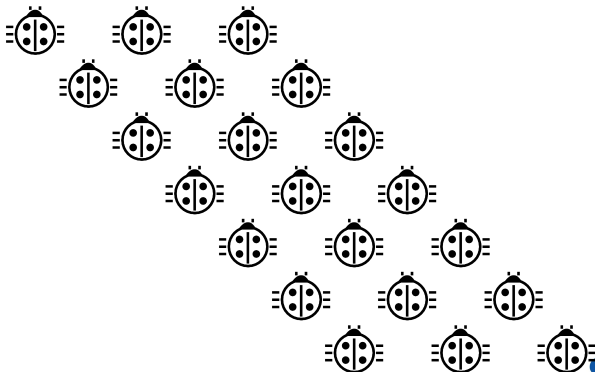
Evaluate

Expression	Value
i	<No symbol "i" in current context.>

3 processes playing

compute • calcul CANADA

Parallel debugging



Parallel debugging

- Challenge: Simultaneous execution
- Shared memory:
  OpenMP (Open Multi-Processing)
  pthreads (POSIX threads)
  - Private/shared variables

    Intel compiler extra flag: `-debug parallel`
    Later GNU compilers: `-gstabs`
  - Race conditions
- Distributed memory:
  MPI (Message Passing Interface)
  - Communication
  - Deadlock
- Hard to solve: some commercial debuggers do a good job.
  But let's see how the command-line ones handle it.

SciNet compute • calcul CANADA

Parallel debugging - 1 Shared memory

Use gdb for

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

SciNet compute • calcul CANADA

Parallel debugging - 1 Shared memory

Use gdb for

- Track each thread's execution and variables
- OpenMP serialization: `p omp_set_num_threads(1)`
- Step 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)

SciNet
compute • calcul
CANADA

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.

SciNet compute • calcul CANADA

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.

SciNet compute • calcul CANADA

Parallel debugging - 2 Distributed memory

Advanced gdb (not recommended!)

- You want #proc terminals with gdb for each process?
- Possible, but brace yourself!
- Small number of procs:
  1. Start terminals: by default X forwarding from compute nodes
  2. Submit your job on scinet
  3. Make sure its runs: checkjob -v
  4. From each terminal, ssh into the appropriate nodes
  5. Do `top` or `ps -C <exe>` to find process id (pid)
  6. Attach debugger with `gdb -pid <pid>`.
  7. This will interrupt the process.

SciNet compute • calcul CANADA

Parallel debugging - 2 Distributed memory

Advanced tricks

Wait, so the program started already?

- Yes, and that's probably not what you want.
- Instead, put infinite loop into your code:
```
int j=1;
while(j) sleep(5);
```
- Once attached, go "up" until at while loop.
- do "set var j=0"
- now you can step, continue, etc.

Now let's take a look at DDT...

SciNet compute • calcul CANADA

DDT

Allinea DDT v3.1 (on gpc-f102n0

Session Control Search View Help

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

All 0 1 2 3

Create Group

Project Files

Search (Ctrl+K)

types.h
types.h
types.h
uio.h
wchar.h
wctype.h
win.h
xlocale.h
Source Files
diff3d.cc
exception
inifile.cc

functions_inln.h | diff3d.cc | mpidebug.ch | comm_inln.h

```
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 << "l = "
106            << p.l[0] << ' '
107            << p.l[1] << ' '
108            << p.l[2] << '\n'
109            << "n = "
```

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)

SciNet compute • calcul CANADA

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.

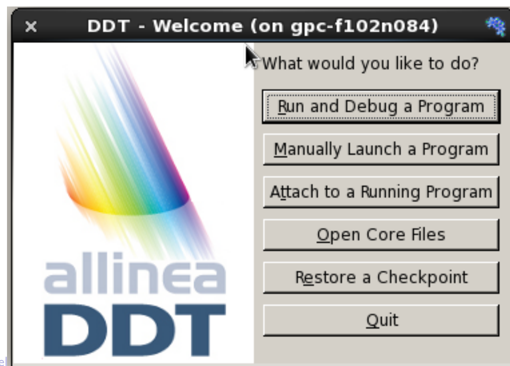
SciNet compute • calcul CANADA

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.



SciNet compute • calcul CANADA

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

SciNet compute • calcul CANADA

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 (o

Preload the memory debugging library: Lang

Note: Preloading only works for programs linked a
program is statically linked, you must relink it again

Heap Debugging

- Minimal (fewest tests, picks up invalid pointers
- Runtime (fast, basic tests including fence-post
- Low (adds minimal heap checking, overwriting
- Medium (adds full heap checking, always reloc
- High (adds checking for arguments to commor
- Custom:

Heap Overflow/Underflow Detection

Add guard pages to detect out of bounds hea

Guard pages: 1 Add guard pages: Afte

Advanced

Specify heap-check interval: 100

Store stack backtraces for memory allocations

Only enable for these processes:

0-1 100% Selec

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

Memory De

☑ Preload the memory de

Note: Preloading only wor
program is statically linked

Heap Debugging

- Minimal (fewest tests,
- Runtime (fast, basic te
- Low (adds minimal he
- Medium (adds full hea
- High (adds checking f
- Custom:

Heap Overflow/Underflow

☐ Add guard pages to c

Guard pages: 1

Advanced

☐ Specify heap-check ir

☑ Store stack backtrace

☐ Only enable for these

0-1

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

☑ Preloa

Note: Pr
program

Heap D

Mini

Runt

Low

Med

High

Cust

Heap O

☐ Add

Guard

Advanc

☐ Spe

☑ Stor

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...

iff3d

Details

ff3d

Details

Details

Details

es, Backtraces, Preload

Details...

× Memory Debugging Options

☑ Preload the memory debugging library:

Note: Preloading only works for programs link
program is statically linked, you must relink it a

Heap Debugging

◉ Minimal (fewest tests, picks up invalid poi

○ Runtime (fast, basic tests including fence-p

○ Low (adds minimal heap checking, overwri

○ Medium (adds full heap checking, always r

○ High (adds checking for arguments to com

○ Custom:

Heap Overflow/Underflow Detection

☐ Add guard pages to detect out of bounds

Guard pages: 1 Add guard pages:

Advanced

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

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

SciNet compute • calcul CANADA

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

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 << "nthreads=" << 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

Stacks

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

Evaluate

Expression	Value

Ready

compute • calcul CANADA

User interface (2)

DDT uses a tabbed-document interface.

User interface (3)

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

compute • calcul CANADA

User interface (4)

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)

Different colour coding for each group's current source line.

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All

All

Root

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 << "nthreads=" << 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

Stacks

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

Evaluate

Expression	Value

Ready

compute • calcul CANADA

User interface (6)

Session Control Dialog:

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

compute • calcul CANADA

User interface (7)

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

```
85      //#include "mpidebug.ch"
86
87      mpiCommit<Parameters>();
```

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

Expression	Value

Ready

compute • calcul CANADA

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

Focus:
Choose between Group, process or thread

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

Current Line(s) | Current Stack

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

Type: none selected

Input/Output* | Breakpoints | Watchpoints | Stacks | Tracepoints | Tracepoint Output

Stacks

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

Evaluate

Expression	Value

Ready

compute • calcul CANADA

User interface (9)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

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

Stacks: Current and Parallel

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

delete.c
diff3d.cc
distances.c
divtf3.c

```
82
83      cerr << "nthreads=" << 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

Stacks

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

Evaluate

Expression	Value

Ready

compute • calcul CANADA

User interface (9)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

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

Stacks: Current and Parallel

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

delete.c
diff3d.cc
distances.c
divtf3.c

```
82
83      cerr << "nthreads=" << 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

Stacks

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

Evaluate

Expression	Value

Ready

SciNet compute • calcul CANADA

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 << "nthreads=" << 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

Stacks

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

Evaluate

Expression	Value

Ready

SciNet compute • calcul CANADA

User interface (11)

Allinea DDT v3.1 (on gpc-f102n084)

Session Control Search View Help

Current Group: All

Local variables for process

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 << "nthreads=" << 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

Stacks

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

Evaluate

Expression	Value

Ready

compute • calcul CANADA

User interface (12)

compute • calcul CANADA

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!

SciNet compute • calcul CANADA

Other features of DDT (2)

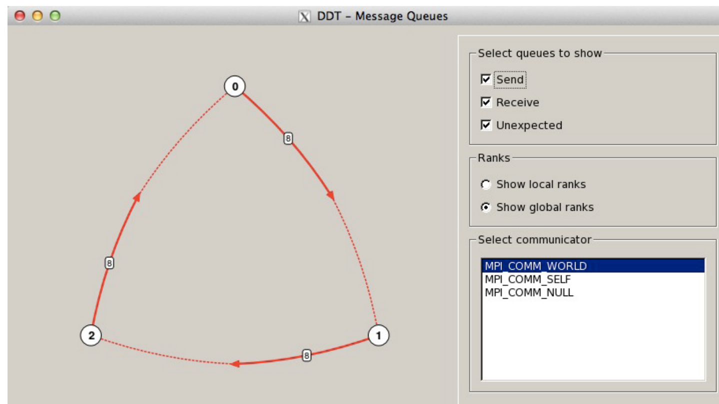
- Signal handling: SEGV, FPE, PIPE,ILL
- Support for Fortran modules
- Change data values in evaluate window
- Examine pointers (vector, reference, dereference)
- Multi-dimensional arrays
- Viewer

SciNet compute • calcul CANADA

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



SciNet compute • calcul CANADA

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

SciNet compute • calcul CANADA

DDT Hands-on. . .

SciNet
compute • calcul
CANADA

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

SciNet compute • calcul CANADA

Example code 1

```
#include <stdlib.h>
void print_scrambled(char * msg);
int main() {
  char * bad_msg;
  bad_msg=NULL;
  char * good_msg="Hello world";
  print_scrambled(good_msg);
  print_scrambled(bad_msg);
  return 0;
}
```

```
#include <stdio.h>
void print_scrambled(char * msg) {
  int i=3;
  do {
    printf("%c", msg[i]);
    i++;
  } while (*++msg);
  printf("\n");
}
```


SciNet compute • calcul CANADA

Example code 2

```
#include <stdio.h>
int main(int argc, char ** argv)
{
  unsigned int count = 999;
  unsigned int step = 2;
  do {
    printf("countdown at %d\n", count);
    count -= step;
  } while ( count >= 0 );
  printf("lift-off!\n");
  return 0;
}
```


SciNet compute • calcul CANADA

Example code 3

```
#include <mpi.h>
int main(int argc, char ** argv)
{
  MPI_Init(&argc,&argv);
  int procs, rank;
  MPI_Comm_size(MPI_COMM_WORLD, &procs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  int left = (rank+procs-1)%procs;
  int right = (rank+1)%procs;
  doubled = rank*1.1;
  MPI_Ssend(&d, 1, MPI_DOUBLE, right, 17, MPI_COMM_WORLD);
  MPI_Recv(&d, 1, MPI_DOUBLE, left, 17, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  MPI_Finalize();
}
```


SciNet compute • calcul CANADA

Example code 4

```c
#include <stdlib.h>
#include <stdio.h>
double* add_vector(double* a,double* b)
{
  double* c = malloc(3*sizeof(double));
  for (int i=0;i<3;i++)
    c[i] = a[i]+b[i];
  return c;
}
int main(int argc, char ** argv)
{
  double to_sum[10][3] = {
    {1,0,0}, {1,1,1}, {1,2,0}, {1,3,1}, {1,4,0},
    {2,0,1}, {2,1,0}, {2,2,1}, {2,3,0}, {2,4,1}
  };
  double answer[3] = {0,0,0};
  for (int i=0;i<10;i++)
    answer = add_vector(answer,to_sum[i]);
  printf("answer = %lf %lf %lf\n", answer[0], answer[1], answer[2]);
  return 0;
}
```


SciNet compute • calcul CANADA

Example code 5

```
#include <stdlib.h>
#include <stdio.h>
double ndot(int n, double*x, double*y){
  double tot=0;
  #pragma omp parallel for shared(x,y,n,tot)
  for (int i=0; i<n; i++)
    tot += x[i] * y[i];
  return tot;
}
int main(int argc, char** argv){
  int n = 10000000;
  double*x = malloc(n*sizeof(double));
  double*y = malloc(n*sizeof(double));
  for (int i=0; i<n; i++) {
    x[i] = i;
    y[i] = i;
  }
  double nn=n-1;
  double ans=nn*(nn+1)*(2*nn+1)/6.0;
  double dot=ndot(n,x,y);
  return 0;
}
```