

# Parallel Debugging with DDT

James Willis (SciNet)

March 25, 2024

# Outline

- Software Bugs
- What is Debugging?
- Symbolic Debuggers
- What is DDT?
- Setting up DDT on Teach
  - ▶ Hello-mpi Hands-on
- Matrix-Matrix Multiply Hands-on Example

# Outline

- Other Useful Features of DDT
  - ▶ Client-Server Mode
  - ▶ Attach Mode
  - ▶ Submitting Jobs to a Scheduler
  - ▶ Running Core Files

# Software Bugs

# Software Bugs

- Writing clean, efficient, error-free code is nearly impossible
- At some point you will run into situations such as:

- ▶ Compile time errors
- ▶ Segmentation faults

```
laptop:~$ gcc app.c -o app
laptop:~$ ./app
Segmentation fault
```

- ▶ Your code doesn't do what you expect
- ▶ Incorrect results
- These are all examples of what is known as a *software bug*

# Common Symptoms

- Compile time errors
  - ▶ Code syntax errors (easy to fix)
  - ▶ Linker errors when linking against libraries
  - ▶ Cross-compilation, i.e. compiling for a different computing architecture compared to the host
  - ▶ Compilation warnings

**Always turn compiler warnings on and fix or understand them before running your code! It will save you future headaches.**

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

# Common Symptoms

- Runtime errors
  - ▶ Floating point exceptions
  - ▶ Segmentation faults
  - ▶ Aborted
  - ▶ Incorrect output (e.g. NaNs, Inf)

# Error Examples

Type	Reason
Arithmetic	Corner cases e.g. <code>sqrt(-0.0)</code> , infinities
Memory Access	Index out of range, uninitialised pointers
Logic	Infinite loop, corner cases
Misuse	Wrong input, ignored error, no initialisation
Syntax	Wrong operators/arguments
Resource Starvation	Memory leak, quota exceeded
Parallel	Race conditions, deadlock



# What is going on?

- Almost always, a condition you are sure is satisfied, is not
- But your application likely relies on many such assumptions
- First order of business is finding out what is going wrong and what assumption is not warranted
- Follow the *Fundamental Principle of Confirmation*:
  - ▶ Process of confirming, one by one, that many things you **believe** to be true about the code are actually true
- Debugger: program to help detect errors in other programs
- **Debugging: Methodical process of finding and fixing flaws in software**
- **You are the real debugger**

# How to Avoid Debugging

- Write better code:
  - ▶ Simple, clear, straightforward code
  - ▶ Modular (no global variables or 10,000 line functions)
  - ▶ Avoid “cute” tricks (no obfuscated C code winners)
- Improve your code/algorithm/language/API understanding
- Don't reinvent the wheel, use existing libraries
- Write (simple) tests for each part of your code
- Use version control (GIT) so you can “roll back” your code if a bug is found

# Debugging Workflow

- As soon as you are convinced there is a real problem, create the simplest test case that reproduces the bug
- This is science: model, hypothesis, experiment, conclusion
- Try a smaller problem size, turning off physical effects with options, etc. until you have a simple, fast repeatable example of the bug
- Try to narrow it down to a particular module/function/class. For Fortran, switch on bounds checking (`-fbounds-check`)
- Now you're ready to start debugging

# Ways to Debug

- Preemptive:
  - ▶ Turn on compiler warnings: fix or understand them!  

```
laptop:~$ 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 Analyse output
- Bug not found? Repeat from 1. again
- Have to remove extra code after the bug is fixed
- Rinse and repeat for each bug

## Disadvantages

- 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 (Niagara): can be slow, but we will look at ways to improve this later

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

# Preparing Your Code for the Debugger

- Add debugging flags when compiling your code:

```
laptop:~$ gcc/g++/gfortran -g [-gstabs]
laptop:~$ icc/icpc/ifort -g [-debug parallel]
laptop:~$ nvcc -g -G
```

- Optional flag: switch off optimisation -O0 (sometimes symbol values are hidden to the debugger at higher optimisation levels e.g. -O2, -O3)



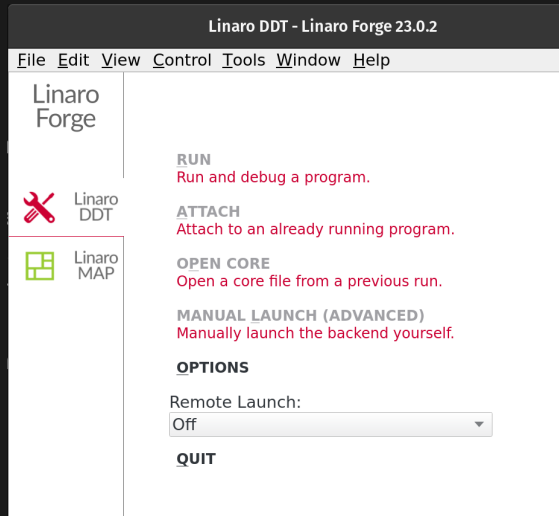
# Examples of Symbolic Debuggers

- Command-line based debuggers: GDB, LLDB
- Graphical based debuggers: **DDT**, Visual Studio, Eclipse
  - ▶ Nice, more intuitive graphical user interface
  - ▶ Front-end to command-line based tools: Same concepts
  - ▶ Need graphics support: X11 forwarding (or VNC)
- The rest of the workshop will focus on DDT

# What is DDT?

# What is DDT?

- **DDT** stands for **Distributed Debugging Tool**
- Powerful GUI-based commercial debugger by *Linaro*
- Developed for debugging parallel, multi-threaded, and distributed applications
- Widely used in high-performance computing environments
- Available on Niagara and other Alliance systems (*Note: license only allows debugging up to 64 processes*)



# DDT Features

- **Key Features:**

- ▶ Parallel and distributed debugging capabilities
- ▶ Graphical user interface for intuitive navigation
- ▶ Support for multiple programming languages (e.g. C, C++, Fortran, Python)
- ▶ Supports MPI, OpenMP, threads, CUDA, ROCm and more
- ▶ Integrated performance analysis tools - *MAP*
- ▶ Memory debugging functionalities

# Launching DDT on Teach

- Load the latest software stack with a compiler and MPI module:

```
teach-login01:~$ module load TeachEnv/2022a gcc openmpi
```

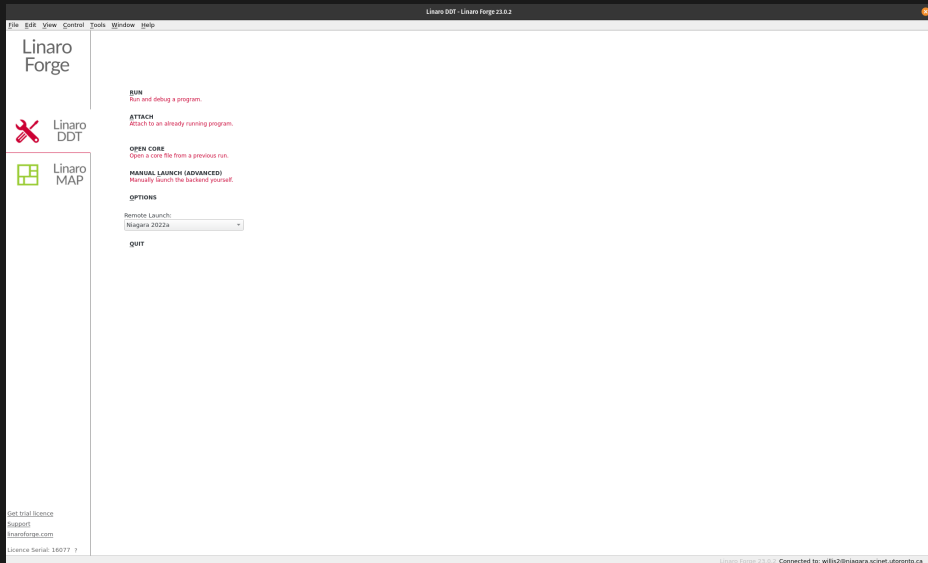
- Load DDT:

```
teach-login01:~$ module load ddt
```

- Start DDT with one of these commands:

```
teach-login01:~$ ddt
teach-login01:~$ ddt <exe compiled with -g flag>
teach-login01:~$ ddt <exe compiled with -g flag> <arguments>
teach-login01:~$ ddt -n <numprocs> <exe compiled with -g flag> <arguments>
```

# Launching DDT on Teach



# Creating a Debug Session

Run

Application: /gpfs/fs0/scratch/s/scinet/willis2/ddt-mmult/mmult\_c

Details

Application: /gpfs/fs0/scratch/s/scinet/willis2/ddt-mmult/mmult\_c

Arguments:

☐ stdin file:

Working Directory: /gpfs/fs0/scratch/s/scinet/willis2/ddt-mmult

☒ MPI: 4 processes, Open MPI (Compatibility)

Details

Number of Processes: 4

☐ Processes per Node 1

Implementation: Open MPI (Compatibility)

Change...

mpirun arguments

☐ OpenMP

Details

☐ CUDA

Details

☐ ROCm

Details...

☒ Memory Debugging: Fast, 1 guard page after, Backtraces, Pr

Details...

☐ Submit to Queue

Configure...

Parameters...

Environment Variables: none

Details

Plugins: none

Details

Help

Options

Run

Cancel

Memory Debugging Options

☒ Preload the memory debugging library

Language: Recommended

**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

Fast

Balanced

Thorough

Custom

Enabled Checks: -fence,free-protect,free-blank,alloc-blank

More Information

Heap Overflow/Underflow Detection

☒ Add guard pages to detect out of bounds heap access

Guard pages: 1

Add guard pages: After

Advanced

☐ Set node memory threshold at 90 percent

☐ Check heap consistency every 100 heap operations

☒ Store stack backtraces for memory allocations

☐ Only enable for these processes:

100%

Select All

x2

x0.5

1%

Help

OK

Cancel

# User Interface

Linaro DDT - Linaro Forge 23.0.2

File Edit View Control Tools Window 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)

Application Code

Sources

mmult.c

main(int argc, char \* argv[]): int  
mimult(int sz, double \* A): void  
mmult(int sz, int nslices, double \*,  
mwrite(int sz, double \* A, char \* fn

External Code

mmult.c

```
82  
83  
84 int main(int argc, char *argv[])  
85 {  
86     int mr, nproc, sz, slice;  
87     double *mat_a, *mat_b, *mat_c;  
88     char filename[32];  
89     int remainder;  
90     MPI_Status st;  
91  
92     MPI_Init (&argc, &argv);  
93     MPI_Comm_rank(MPI_COMM_WORLD, &mr); // my rank  
94     MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors  
95  
96     if (mr == 0)  
97     {  
98         printf(WORKED_EXAMPLE_NOTICE "\n\n");  
99     }  
100  
101     if (argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0) )  
102     {  
103         if (mr == 0)  
104         {  
105             printf("Usage: ./mmult [-h] SIZE FILENAME\n \  
106                 \t-h: display this help message\n \  
107                 \tSIZE: size of the matrix to compute (default is %d)\n \  
108                 \tFILENAME: output matrix file name (default is %s)\n", DEFAULT_SIZE, DEFAULT_FN);  
109         }  
110  
111         return 1;  
112     }  
113     else  
114     {  
115         if (argc > 1)
```

Locals

Current Line(s)

Current Stack

Locals

Name	Value
<return value>	0
argc	1
argv	0x7fffffff008
mr	0
nproc	4199357
sz	0
slice	0
mat_a	0x7fffffff000
mat_b	0x400af0
mat_c	0x401370
filename	""
remainder	0
st	

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook Evaluate

Stacks (All)

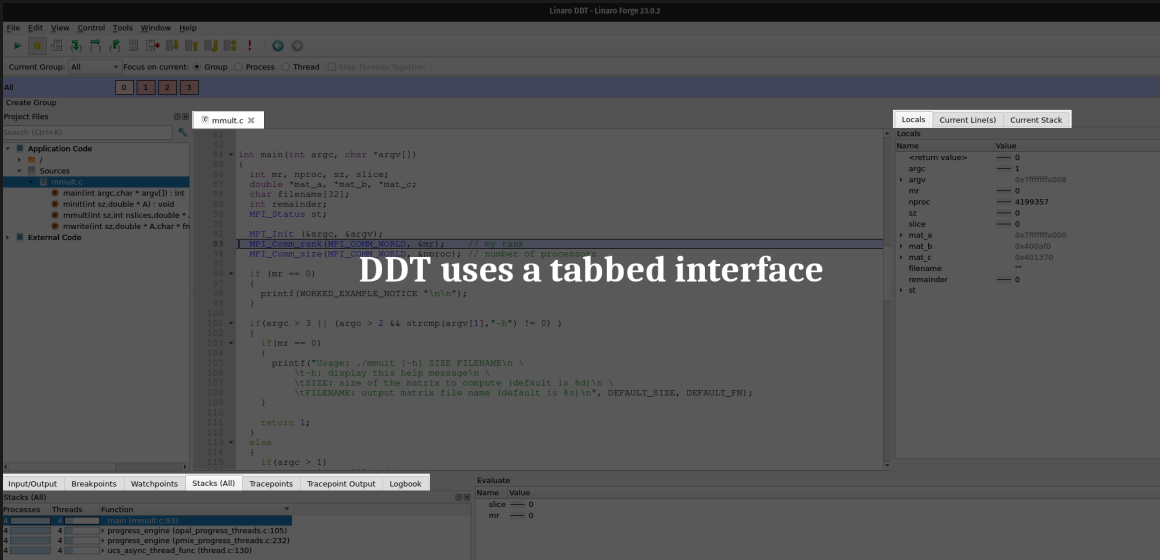
Processes	Threads	Function
4	4	main (mmult.c:93)
4	4	progress_engine (opal_progress_threads.c:105)
4	4	progress_engine (pmix_progress_threads.c:232)
4	4	ucs_async_thread_func (thread.c:130)

Evaluate

Name	Value
slice	0
mr	0



# User Interface



# User Interface

Linaro DDT - Linaro Forge 23.0.2

**DDT automatically finds the source code from the executable**

File Edit View Control Tools Window Help

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

All 0 1 2 3

Create Group

Project Files

Search (Ctrl+K)

Application Code

Sources

mmult.c

main(int argc, char \* argv[]): int

main(int argc, char \* argv[]): int

mmult(int sz, int nslices, double \* A, char \* filename): void

mmult(int sz, int nslices, double \* A, char \* filename): void

External Code

```
82
83
84 int main(int argc, char *argv[])
85 {
86     int mr, nproc, sz, slice;
87     double *mat_a, *mat_b, *mat_c;
88     char filename[32];
89     int remainder;
90     MPI_Status st;
91
92     MPI_Init (&argc, &argv);
93     MPI_Comm_rank(MPI_COMM_WORLD, &mr); // my rank
94     MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors
95
96     if (mr == 0)
97     {
98         printf(WORKED_EXAMPLE_NOTICE "\n\n");
99     }
100
101     if (argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0))
102     {
103         if (mr == 0)
104         {
105             printf("Usage: ./mmult [-h] SIZE FILENAME\n \
106                 \t-h: display this help message\n \
107                 \tSIZE: size of the matrix to compute (default is %d)\n \
108                 \tFILENAME: output matrix file name (default is %s)\n", DEFAULT_SIZE, DEFAULT_FN);
109         }
110         return 1;
111     }
112     else
113     {
114         if (argc > 1)
115         {
```

Locals

Current Line(s)

Current Stack

Locals

Name	Value
<return value>	0
argc	1
argv	0x7fffffffa008
mr	0
nproc	4199357
sz	0
slice	0
mat_a	0x7fffffffa000
mat_b	0x400af0
mat_c	0x401370
filename	""
remainder	0
st	

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook

Stacks (All)

Processes	Threads	Function
4	4	main (mmult.c:93)
4	4	progress_engine (opal_progress_threads.c:105)
4	4	progress_engine (pmix_progress_threads.c:232)
4	4	ucs_async_thread_func (thread.c:130)

Evaluate

Name	Value
slice	0
mr	0

# User Interface

Linaro DDT - Linaro Forge 23.0.2

File Edit View Control Tools Window 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)

Application Code

Sources

mmult.c

main(int argc, char \* argv[]): int

mmult(int sz, int nslices, double \*, double \*): void

mmult(int sz, int nslices, double \*, double \*): void

mmult(int sz, int nslices, double \*, double \*): void

mmult(int sz, int nslices, double \*, double \*): void

External Code

mmult.c

```
82
83
84
85
86 int mr, nproc, sz, slice;
87 double *mat_a, *mat_b, *mat_c;
88 char filename[32];
89 int remainder;
90 MPI_Status st;
91
92 MPI_Init (&argc, &argv);
93 MPI_Comm_rank(MPI_COMM_WORLD, &mr); // my rank
94 MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors
95
96 if (mr == 0)
97 {
98     printf(WORKED_EXAMPLE_NOTICE "\n\n");
99 }
100
101 if (argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0))
102 {
103     if (mr == 0)
104     {
105         printf("Usage: ./mmult [-h] SIZE FILENAME\n \
106             \t-h: display this help message\n \
107             \tSIZE: size of the matrix to compute (default is %d)\n \
108             \tFILENAME: output matrix file name (default is %s)\n", DEFAULT_SIZE, DEFAULT_FN);
109     }
110     return 1;
111 }
112 else
113 {
114     if (argc > 1)
```

Locals

Current Line(s)

Current Stack

Locals

Name	Value
<return value>	0
argc	1
argv	0x7fffffffa008
mr	0
nproc	4199357
sz	0
slice	0
mat_a	0x7fffffffa000
mat_b	0x400af0
mat_c	0x401370
filename	""
remainder	0
st	

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook

Stacks (All)

Processes	Threads	Function
4	4	main (mmult.c:93)
4	4	progress_engine (opal_progress_threads.c:105)
4	4	progress_engine (pmix_progress_threads.c:232)
4	4	ucs_async_thread_func (thread.c:130)

Evaluate

Name	Value
slice	0
mr	0

# User Interface

Linaro DDT - Linaro Forge 23.0.2

File Edit View Control Tools Window Help

Current Group: Group 1 Focus on current: Group Process Thread Step Threads Together

All 0 1 2 3

Group 1 0 1

Group 2 2 3

Create Group

Project Files

Application Code

Sources

mmult.c

main(int argc, char \*argv[]): int

main(int sz, double \*A): void

mmult(int sz, int ndices, double \*

main(int sz, double \*A, char \*f

External Code

```
1 88 int main(int argc, char *argv[])
2 89 {
3 90     int nr, nproc, sz, slice;
4 91     double *mat_a, *mat_b, *mat_c;
5 92     char filename[32];
6 93     int remainder;
7 94     MPI_Status st;
8 95
9 96     MPI_Init(&argc, &argv);
10 97     MPI_Comm_rank(MPI_COMM_WORLD, &nr); // my rank
11 98     MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors
12 99
13 100     if (nr == 0)
14 101     {
15 102         printf(WORKED_EXAMPLE_NOTICE "\n\n");
16 103     }
17 104
18 105     if (argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0))
19 106     {
20 107         if (nr == 0)
21 108         {
22 109             printf("Usage: ./mmult [-h] SIZE FILENAME\n \
23 110             \t-h: display this help message\n \
24 111             \tSIZE: size of the matrix to compute (default is %d)\n \
25 112             \tFILENAME: output matrix file name (default is %s)\n", DEFAULT_SIZE, DEFAULT_FN);
26 113         }
27 114     }
28 115 }
```

Locals

Current Line(s) Current Stack

Name	Value
argc	1
argv	0x7ffffff0
nr	0
nproc	4
sz	0
slice	0
mat_a	0x7ffffff0
mat_b	0x400a00
mat_c	0x401370
filename	""
remainder	0
st	

Input/Output Breakpoints Watchpoints Stacks (Group 1)

Processes Threads Function

1	1	main (mmult.c:98)
2	1	main (mmult.c:101)
2	2	progress_engine (opal_progress.c:1630)
2	2	opal_rbevent2022_event_base_loop (event.c:1630)
2	2	epoll_dispatch (epoll.c:407)
2	2	progress_engine (pmix_progress.c:1630)
2	2	opal_rbevent2022_event_base_loop (event.c:1630)
2	2	epoll_dispatch (epoll.c:407)
2	2	ucs_async_thread_func (thread.c:130)
2	2	ucs_event_set_wait (event_set.c:130)

- Can group processes together

- Modify group with drag and drop of processes

- Different colour coding for each group's current source code line

# User Interface

Linaro DDT - Linaro Forge 23.0.2

File Edit View Control Tools Window Help

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

0 1 2 3

Create Group

Project Files

Search (Ctrl+K)

Application Code

Sources

mmult.c

main(int argc, char \* argv[]) : int  
mini(int sz, double \* A) : void  
mmult(int sz, int nslices, double \*,  
mwrite(int sz, double \* A, char \* fn

External Code

mmult.c

```
82  
83  
84  
85  
86 int mr, nproc, sz, slice;  
87 double *mat_a, *mat_b, *mat_c;  
88  
89  
90 MPI_Status st;  
91  
92  
93  
94  
95  
96 if (mr == 0)  
97 {  
98     printf(WORKED_EXAMPLE_NOTICE "\n\n");  
99 }  
100  
101 if(argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0) )  
102 {  
103     if(mr == 0)  
104     {  
105         printf("Usage: ./mmult [-h] SIZE FILENAME\n \  
106             \t-h: display this help message\n \  
107             \tSIZE: size of the matrix to compute (default is %d)\n \  
108             \tFILENAME: output matrix file name (default is %s)\n", DEFAULT_SIZE, DEFAULT_FN);  
109     }  
110     return 1;  
111 }  
112 }  
113 else  
114 {  
115     if(argc > 1)
```

Locals

Current Line(s)

Current Stack

Locals

Name	Value
<return value>	0
argc	1
argv	0x7fffffffa008
mr	0
nproc	4199357
sz	0
slice	0
mat_a	0x7fffffffa000
mat_b	0x400af0
mat_c	0x401370
filename	""
remainder	0
st	

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook

Stacks (All)

Processes	Threads	Function
4	4	main (mmult.c:93)
4	4	progress_engine (opal_progress_threads.c:105)
4	4	progress_engine (pmix_progress_threads.c:232)
4	4	ucs_async_thread_func (thread.c:130)

Evaluate

Name	Value
slice	0
mr	0

# User Interface

Linaro DDT - Linaro Forge 23.0.2

File Edit View Control Tools Window Help

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

All 0 1 2 3

Group 1 0 1

Group 2 2 3

Create Group

Project Files

Search (Ctrl+F)

Application Code

Sources

mmult.c

main(int argc, char \* argv): int

main(int sz, double \* A): void

mmult(int sz, int ndims, double \*\*,

mmult(int sz, double \* A, char \* fn

External Code

```
1 int main(int argc, char *argv[])
2 {
3     int nr, nproc, sz, slice;
4     double *mat_a, *mat_b, *mat_c;
5     char filename[32];
6     int remainder;
7     MPI_Status st;
8
9     MPI_Init(&argc, &argv);
10    MPI_Comm_rank(MPI_COMM_WORLD, &nr); // my rank
11    MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors
12
13    if (nr == 0)
14    {
15        printf(MORDED_EXAMPLE_NOTICE "\n\n");
16    }
17
18    if (argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0))
19    {
20        if (nr == 0)
21        {
22            printf("Usage: ./mmult [-h] SIZE FILENAME\n \
23            \t-h: display this help message\n \
24            \tSIZE: size of the matrix to compute (default is %d)\n \
25            \tFILENAME: output matrix file name (default is %s)\n", DEFAULT_SIZE, DEFAULT_FN);
26        }
27        return 1;
28    }
29    ...
```

Locals Current Line(s) Current Stack

Current Line(s)

Name Value

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook

Breakpoints

Processes	Threads	File	Line	Function	Condition	Start After	Trigger Every	Stop After	Full path
<input checked="" type="checkbox"/> All	all	mmult.c	103	main		0	1	Forever	/scratch/willis2@dt-mmult/mmult.c
<input checked="" type="checkbox"/> All	all	mmult.c	118	main		0	1	Forever	/scratch/willis2@dt-mmult/mmult.c
<input checked="" type="checkbox"/> All	all	mmult.c	98	main		0	1	Forever	/scratch/willis2@dt-mmult/mmult.c

Breakpoints Tab

Can suspend, jump to, delete, load, save

Ready Connected to: willis2@niagara.scienet.utoronto.ca

SciNet

ADVANCED RESEARCH COMPUTING & AL, UNIVERSITY OF TORONTO

# User Interface

The screenshot displays the Linaro DDT - Linaro Forge 21.8.2 interface. The top menu bar includes File, Edit, View, Control, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main window is divided into several panes:

- Project Files:** A tree view on the left showing the project structure, including files like `mmult.c` and `pmix_progress_threads.c`.
- Source Code:** The central pane displays the source code of `mmult.c`, showing a loop that calculates the product of two matrices.
- Current Stack:** A pane on the right showing the current stack frame, including the function name `#1 main` and the arguments `argc=1, argv=0x7fffffa008`.
- Stacks (All):** A pane at the bottom left showing a list of all stack frames, including the current frame and its callers.
- Input/Output, Breakpoints, Watchpoints, Stacks (All), Tracepoints, Tracepoint Output, Logbook:** A row of tabs at the bottom of the interface.

Overlaid on the source code pane is a large text box with the following content:

## Stacks: Current and Parallel

- Tree of functions
- Click on branch to see source
- Hover to see process ranks

The bottom right corner of the interface shows the SciNet logo and the text "Ready. Connected to: willis2@niagara.scienet.utoronto.ca".

# User Interface

Linux DDT - Linaro Forge 23.0.2

File Edit View Control Tools Window Help

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

All 0 1 2 3

Group 1 0 1

Group 2 2 3

Create Group

Project Files

Search (Ctrl+K)

- Application Code
  - /
  - Sources
    - mmult.c
    - main(int argc, char \* argv[]) : int
    - main(int sz, double \* A) : void
    - mmult(int sz, int nlices, double \*,
    - mmult(int sz, double \* A, char \* fn
- External Code

mmult.c X

```
77 C(i*sz+) += res;
78 }
79
80 int main(int argc, char *argv[])
81 {
82     int mr, nproc, sz, slice;
83     double *mat_a, *mat_b, *mat_c;
84     char filename[32];
85     int remainder;
86     MPI_Status st;
87
88     MPI_Init(&argc, &argv);
89
90     MPI_Comm_rank(MPI_COMM_WORLD, &mr); // my rank
91     MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors
92
93     if (mr == 0)
94     {
95         printf(WORKED_EXAMPLE_NOTICE "\n\n");
96     }
97
98     if (argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0))
99     {
100         if (mr == 0)
101             printf("Usage: ./mmult [-h] SIZE FILENAME\n \n");
102         printf("\t-h: display this help message\n \n");
103     }
104 }
```

Current line variables

Locals Current Line(s) Current Stack

Name	Value
mr	0

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook

Stacks (All)

Processes	Threads	Function
2	4	main (mmult.c:3)
4	4	progress_engine (opal_progress_threads.c:103)
4	4	opal_libevent2022_event_base_loop (event.c:1630)
4	4	epoll_dispatch (epoll.c:407)
4	4	progress_engine (pmix_progress_threads.c:232)
4	4	opal_libevent2022_event_base_loop (event.c:1630)
4	4	epoll_dispatch (epoll.c:407)
4	4	ucs_async_thread_func (thread.c:130)
4	4	ucs_event_set_wait (event_set.c:198)

Evaluate

Name	Value
slice	0
mr	0

Ready. Connected to: willis2@niagara.scienet.utoronto.ca



# User Interface

Linero DDT - Linero Forge 23.0.2

File Edit View Control Tools Window Help

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

All 0 1 2 3

Group 1 0 1

Group 2 2 3

Create Group

Project Files

Search (Ctrl+F)

Application Code

mmult.c

main(int argc, char \*argv): int

main(int sz, double \*A): void

mmult(int sz, int nlices, double \*

mmult(int sz, double \*A, char \*fn

External Code

mmult.c

```
77 C[l*sz+j] += twa;
78 }
79 }
80
81 int main(int argc, char *argv[])
82 {
83     int nr, nproc, sz, slice;
84     double *mat_a, *mat_b, *mat_c;
85     char filename[32];
86     int remainder;
87     MPI_Status st;
88
89     MPI_Init(&argc, &argv);
90     MPI_Comm_rank(MPI_COMM_WORLD, &nr); // my rank
91     MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors
92
93     if (nr == 0)
94     {
95         printf(WORKED_EXAMPLE_NOTICE "\n\n");
96     }
97
98     if (argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0))
99     {
100         if (nr == 0)
101         {
102             printf("Usage: ./mmult [-h] SIZE FILENAME\n \
103                 \t-h: display this help message\n \
104                 \n");
105         }
106     }
107 }
```

Locals

Current Line(s)

Current Stack

Name Value

<return value> 0

argc 1

argv 0x7ffffff9

nr 0

nproc 4199357

sz 0

slice 0

mat\_a 0x7ffffff0

mat\_b 0x400a0

mat\_c 0x401370

filename

remainder 0

st

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook

Stacks (All)

Processes Threads Function

4 4 main (mmult.c:1)

4 4 progress\_engine (opal\_progress\_threads.c:105)

4 4 opal\_libevent2022\_event\_base\_loop (event.c:1630)

4 4 opoll\_dispatch (epoll.c:407)

4 4 progress\_engine (pmix\_progress\_threads.c:232)

4 4 opal\_libevent2022\_event\_base\_loop (event.c:1630)

4 4 opoll\_dispatch (epoll.c:407)

4 4 ucs\_async\_thread\_func (thread.c:130)

4 4 ucs\_event\_set\_wait (event\_set.c:198)

Evaluate

Name Value

slice 0

nr 0

Ready Connected to: willis2@niagara.scienet.utoronto.ca

SciNet

ADVANCED RESEARCH COMPUTING C.A. UNIVERSITY OF TORONTO

# User Interface

Linero DDT - Linaro Forge 23.0.2

File Edit View Control Tools Window Help

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

All 0 1 2 3

Group 1 0 1

Group 2 2 3

Create Group

Project Files

Search (Ctrl+F)

Application Code

Sources

mmult.c

main(int argc, char \*argv[]): int

main(int sz, double \*A) void

mmult(int sz, int slices, double \*

mmult(int sz, double \*A, char \*fn

External Code

mmult.c

```
73 C[i*sz+j] += twa;
74 }
75 }
76
77 int main(int argc, char *argv[])
78 {
79     int nr, nproc, sz, slice;
80     double *mat_a, *mat_b, *mat_c;
81     char filename[32];
82     int remainder;
83     MPI_Status st;
84
85     MPI_Init(&argc, &argv);
86     MPI_Comm_rank(MPI_COMM_WORLD, &nr); // my rank
87     MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors
88
89     if (nr == 0)
90     {
91         printf(WORKED_EXAMPLE_MESSAGE "\n");
92     }
93
94     if (argc > 3 || (argc > 2 && strcmp(argv[1], "-h") != 0))
95     {
96         if (nr == 0)
97         {
98             printf("Usage: ./mmult [-h] SIZE FILENAME\n \
99                 \t-h: display this help message\n \
100
101
102
103
104
105
106
```

Locals

Current Line(s)

Current Stack

Name Value

<return value> 0

argc 1

argv 0x7ffffff0

nr 0

nproc 4199357

sz 0

slice 0

mat\_a 0x7ffffff0

mat\_b 0x400a00

mat\_c 0x4013f0

filename

remainder 0

st

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook

Stacks (All)

Processes Threads Function

0 4 4 main (mmult.c:3)

4 4 4 progress\_engine (opal\_progress\_threads.c:105)

4 4 4 opal\_libevent2022\_event\_base\_loop (event.c:1630)

4 4 4 opoll\_dispatch (epoll.c:407)

4 4 4 progress\_engine (pmix\_progress\_threads.c:232)

4 4 4 opal\_libevent2022\_event\_base\_loop (event.c:1630)

4 4 4 opoll\_dispatch (epoll.c:407)

4 4 4 ucs\_async\_thread\_func (thread.c:130)

4 4 4 ucs\_event\_set\_wait (event\_set.c:198)

Evaluate

Name Value

slice 0

nr 0

Ready Connected to: willis2@nslagara.scienet.utoronto.ca

James Willis (SciNet)

Parallel Debugging with DDT

March 25, 2024

34 / 55

# DDT Setup Demonstration

# Hands-on hello-mpi Example

- Login to Teach

```
laptop:~$ ssh -X USERNAME@teach.scinet.utoronto.ca
```

- Load compilers, MPI library and ddt:

```
teach-login01:~$ module load TeachEnv/2022a gcc openmpi ddt
```

- Copy examples from the course directory:

```
teach-login01:~$ cp -r /home/l/lcl_uothpc245/hpc245starter/ddt-examples .
```

- Compile MPI Hello World example, hello-mpi.c:

```
teach-login01:~/ddt-examples/ddt-hello-mpi$ mpicc -g hello-mpi.c -o hello-mpi
```

- Run ddt:

```
teach-login01:~/ddt-examples/ddt-hello-mpi$ ddt -n <numprocs> hello-mpi
```

- Experiment with different features of DDT

# Memory Debugging in DDT

- Memory debugging can be turned on in the *Run* window
- Causes the code to stop on an error i.e. memory corruption/leak
- Allows you to check the pointer where the memory corruption has occurred
- Can give an overall view of the memory stats/usage
- Lets look at a real example

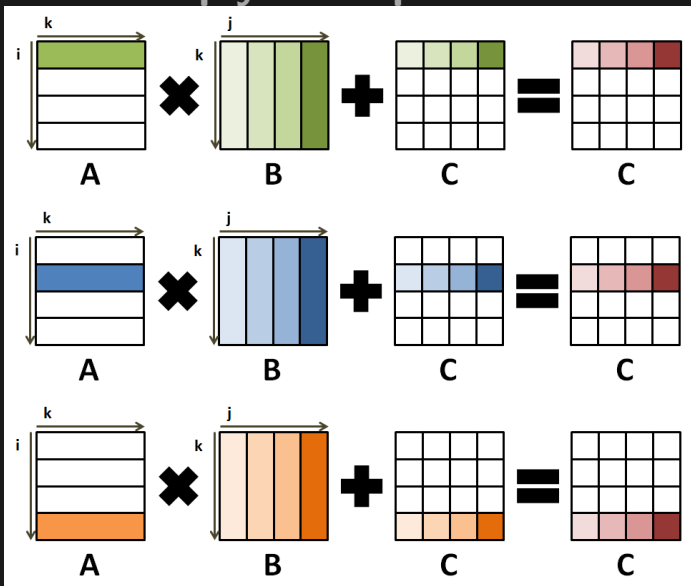
# Matrix-Matrix Multiply Example

- Imagine we want to compute the result of this Matrix equation in parallel with MPI:

$$C = A * B + C$$

- The algorithm works as follows:
  - 1 Rank 0 initialises  $A$ ,  $B$  and  $C$
  - 2 Rank 0 sends the entire matrix  $B$ , with slices of  $A$  and  $C$  to all other ranks
  - 3 Each rank performs matrix multiplication on their domain and computes a slice of  $C$
  - 4 Rank 0 collects the slices of  $C$  from each rank and forms the final matrix  $C$
  - 5 Rank 0 writes  $C$  to a file

# Matrix-Matrix Multiply Example



# Hands-on Matrix-Matrix Multiply

- Change to the ddt-mmult directory from the course examples and compile the code:

```
teach-login01:~/ddt-mmult$ make
```

- This will build C and Fortran executables with `-g -O0` named `mmult_c` and `mmult_f`
- The example can then be run with:

```
teach-login01:~/ddt-mmult$ mpirun -np 4 ./mmult_c
```

- For python, load a python module and compile the C and Fortran libraries with:

```
teach-login01:~/ddt-mmult$ module load python/3.11.5
teach-login01:~/ddt-mmult$ make -f mmult_py.makefile
```

- You will also need a python virtual environment with the `mpi4py` package. I have created one here: `/home/l/lcl_uothpc245/hpc245starter/.virtualenvs/mpi4py-TeachEnv2022-openmpi`
  - ▶ source the virtual environment before running the example:

```
source $HOME/./hpc245starter/.virtualenvs/mpi4py-TeachEnv2022-openmpi/bin/activate
mpirun -np 4 python ./mmult.py
```



# Hands-on Matrix-Matrix Multiply

- Try running the code. What output do you get?
- Run the code in DDT to find out what the error is
- Note: if running with python you will need the setup shown below

Run	
<b>Application:</b> /home/l/lcl_uothpc245/hpc245starter/.virtualenvs/mmpi4py-TeachEnv2022-openmpi/bin/python %allinea_python_debug% ./mmult.py	Details
Application: /home/l/lcl_uothpc245/hpc245starter/.virtualenvs/mmpi4py-TeachEnv2022-openmpi/bin/python	
Arguments: %allinea_python_debug% ./mmult.py	
stdin file:	
Working Directory: /gpfs/fs1/home/s/scinet/willis2/Teaching/ddt-examples/ddt-mmult	

or from the command line:

```
teach-login01:~/ddt-mmult$ ddt -n 4 python %allinea_python_debug% ./mmult.py
```

- Can you locate the error?
- Can you fix it?
- Hints: Try running with *memory debugging* enabled and make sure **Add guard pages** is enabled

<input type="checkbox"/> <b>ROCm</b>	Details...
<input checked="" type="checkbox"/> <b>Memory Debugging:</b> Balanced, 1 guard page after, Backtraces, Preload	Details...
<input type="checkbox"/> <b>Submit to Queue</b>	Configure... Parameters...

# Matrix-Matrix Multiply Demonstration

# Other Useful Features of DDT

- Client-Server mode
- Editing and recompiling code from within DDT GUI
- Attaching to a running job
- Submit SLURM jobs with DDT
- Running with core files

# Client-server Mode

- This mode can be very beneficial if you have a slow internet connection
- Keeps the bulk of the computation on Teach (*server*)
- Only sends minimal amounts of information (network traffic) to your locally running version of DDT (*client*)
- Results in a much smoother experience, avoids slow/laggy interface

# Client-server Mode Setup

## Setting up the server side

- Connect to Teach and create a startup script which will be run by the server and load the modules that your code needs:

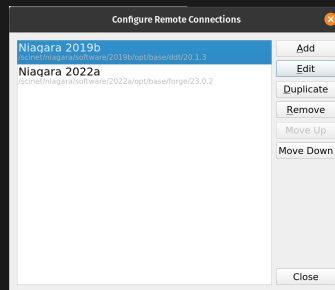
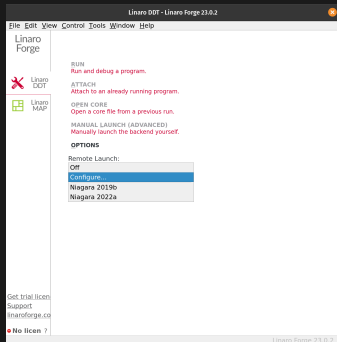
```
#!/bin/bash
module purge
module load TeachEnv/2022a
module load forge/23.0.2
module load gcc openmpi python
export ARM_TOOLS_CONFIG_DIR=${SCRATCH}/.arm
mkdir -p ${ARM_TOOLS_CONFIG_DIR}
```

- Name it `ddt_remote_setup.sh` and place it in `$SCRATCH`

# Client-server Mode Setup

## Setting up the client side

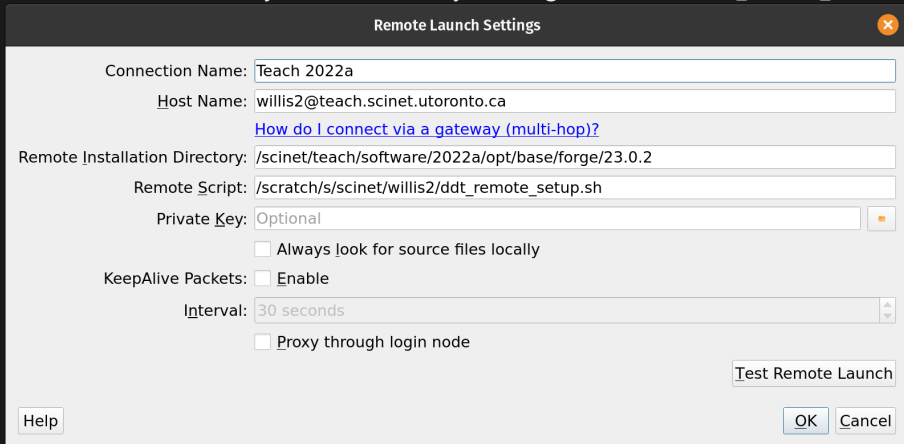
- 1 Download DDT on your local machine from Linaro and make sure the versions matches the one on Teach (23.0.2): <https://www.linaroforge.com/downloadForge/>
- 2 Launch `ddt` and select *Configure* from *Remote Connections*



# Client-server Mode Setup

- 3 Click *Add* and fill out the fields as shown below

Note: *Remote Installation Directory* can be found by running `echo $MODULE_FORGE_PREFIX` on Teach

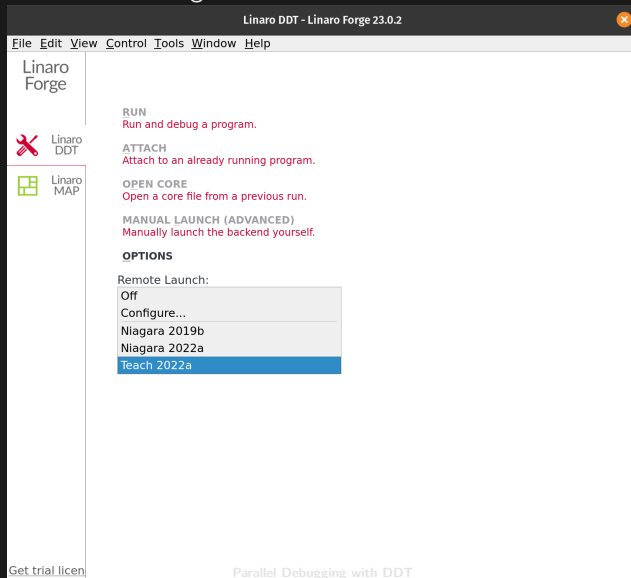


The screenshot shows a 'Remote Launch Settings' dialog box with the following fields and options:

- Connection Name: Teach 2022a
- Host Name: willis2@teach.scinet.utoronto.ca
- Remote Installation Directory: /scinet/teach/software/2022a/opt/base/forge/23.0.2
- Remote Script: /scratch/s/scinet/willis2/ddt\_remote\_setup.sh
- Private Key: Optional
- ☐ Always look for source files locally
- KeepAlive Packets: ☐ Enable
- Interval: 30 seconds
- ☐ Proxy through login node
- Test Remote Launch button
- Help button
- OK button
- Cancel button

# Client-server Mode Setup

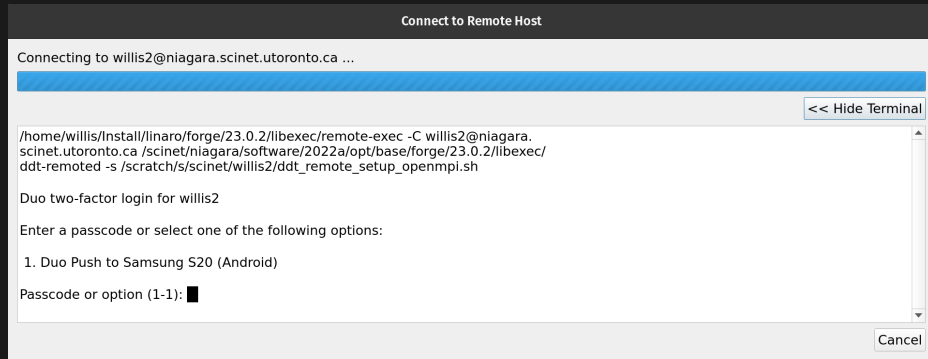
- 4 Click *OK* and now the DDT starting screen should look like this:





# Client-server Mode Setup

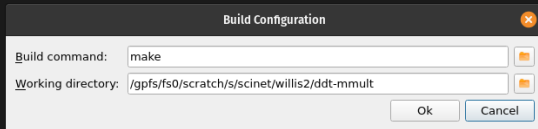
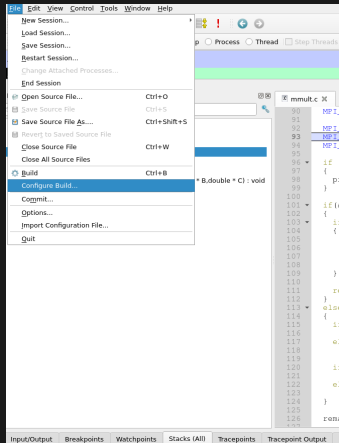
- 5 If you have MFA enabled follow the instructions outlined in the text box:



- More detailed instructions can be found here: [https://docs.linaroforge.com/23.0.2/html/forg/forg/connecting\\_to\\_a\\_remote\\_system/connecting\\_remotely.html](https://docs.linaroforge.com/23.0.2/html/forg/forg/connecting_to_a_remote_system/connecting_remotely.html)

# Editing and Compiling

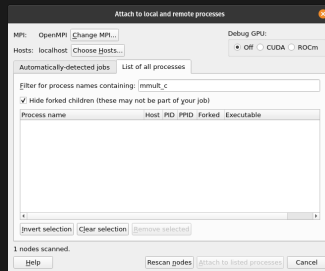
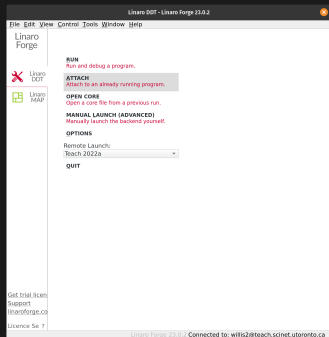
- DDT also has the ability to edit and recompile source code on-the-fly
- Making it much easier to try potential bug fixes



- Build demonstration

# Attach to a Running Job

- DDT allows you to attach to an already running job
- For example, say you have submitted a job to the scheduler on Teach and want to monitor it
- You can use the *Attach* button
- More detailed instructions can be found here: [https://docs.linaroforge.com/23.0.2/html/forge/ddt/get\\_started\\_ddt/attaching\\_to\\_running\\_programs.html#index-9](https://docs.linaroforge.com/23.0.2/html/forge/ddt/get_started_ddt/attaching_to_running_programs.html#index-9)



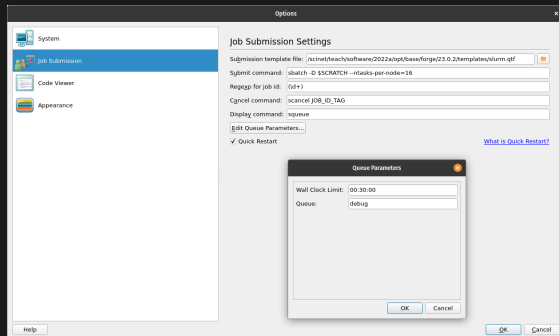
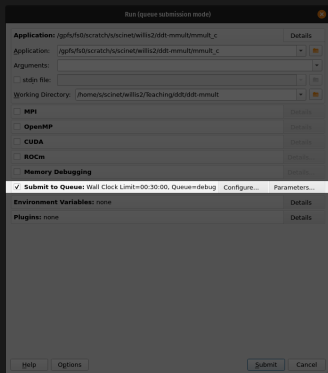
- Attach demonstration

# Submit SLURM Jobs

- DDT allows you to submit jobs directly to the SLURM scheduler on Teach
- The job will be monitored in the queue and as soon as it runs DDT will attach to the job

## Setup

- Click *Run* -> *Submit to Queue* -> *Configure*



# Running .core Files

- When your code terminates unexpectedly it will generate what is known as a *core dump*
- A core dump is a set of files ending in `.core` per process running
- Each `.core` file contains the process's address space (memory) at the time of the crash
- DDT allows you to run with the core files showing the state of the code at the time of the crash
- Can be useful if your job fails after running for a long time
- If no core dump is generated check that `ulimit -c` is set to `unlimited`, this sets the maximum size a `.core` file can be
- Demonstration

# Summary

- DDT is a powerful graphical debugger
- Supports parallel debugging in multiple languages (e.g. C, C++, Fortran, Python)
- Supports MPI, OpenMP, threads, CUDA and more
- DDT documentation: <https://docs.linaroforge.com/23.0.2/html/forgeddt/index.html>

## Support

Questions? Need help?

Don't be afraid to contact us! We are here to help.

- Email to [support@scinet.utoronto.ca](mailto:support@scinet.utoronto.ca) or to [niagara@computecanada.ca](mailto:niagara@computecanada.ca)

# References

- Slide 19: Linaro DDT
- Slide 38: Matrix-Matrix Multiply Worked Example
- Slide 44: Client-Server Mode
- Slide 51: Attach Mode
- Slide 52: Submitting to a Queue