

Introduction to Parallel Debugging Tools

Problems with debugging for Parallel programs:

- All problems of serial programming
- Additional problems
 1. Difficult to verify correctness of program
 2. Difficult to debug N parallel processes
 3. New parallel problems: deadlocks, race conditions, etc.

Parallel Debugging Tools:

- TotalView
- DDT

TotalView: Introduction

What is TotalView?

GUI-based debugging tool by Rogue Wave Software, Inc.
full-featured, source-level, multi-process, multi-thread graphical debugger

Languages supported

C, C++, Fortran, assembler, etc.

Parallel programming models supported

OpenMP, OpenACC, pthreads, CUDA GPU, Xeon Phi, etc.

Systems and Platforms supported

Unix, Linux, OS X, Windows

Integrated Memory Debugging

TotalView: Basic operations

Data examination

- view data in the variable windows
- change the values of variables
- modify display of the variables
- visualize data

Action points

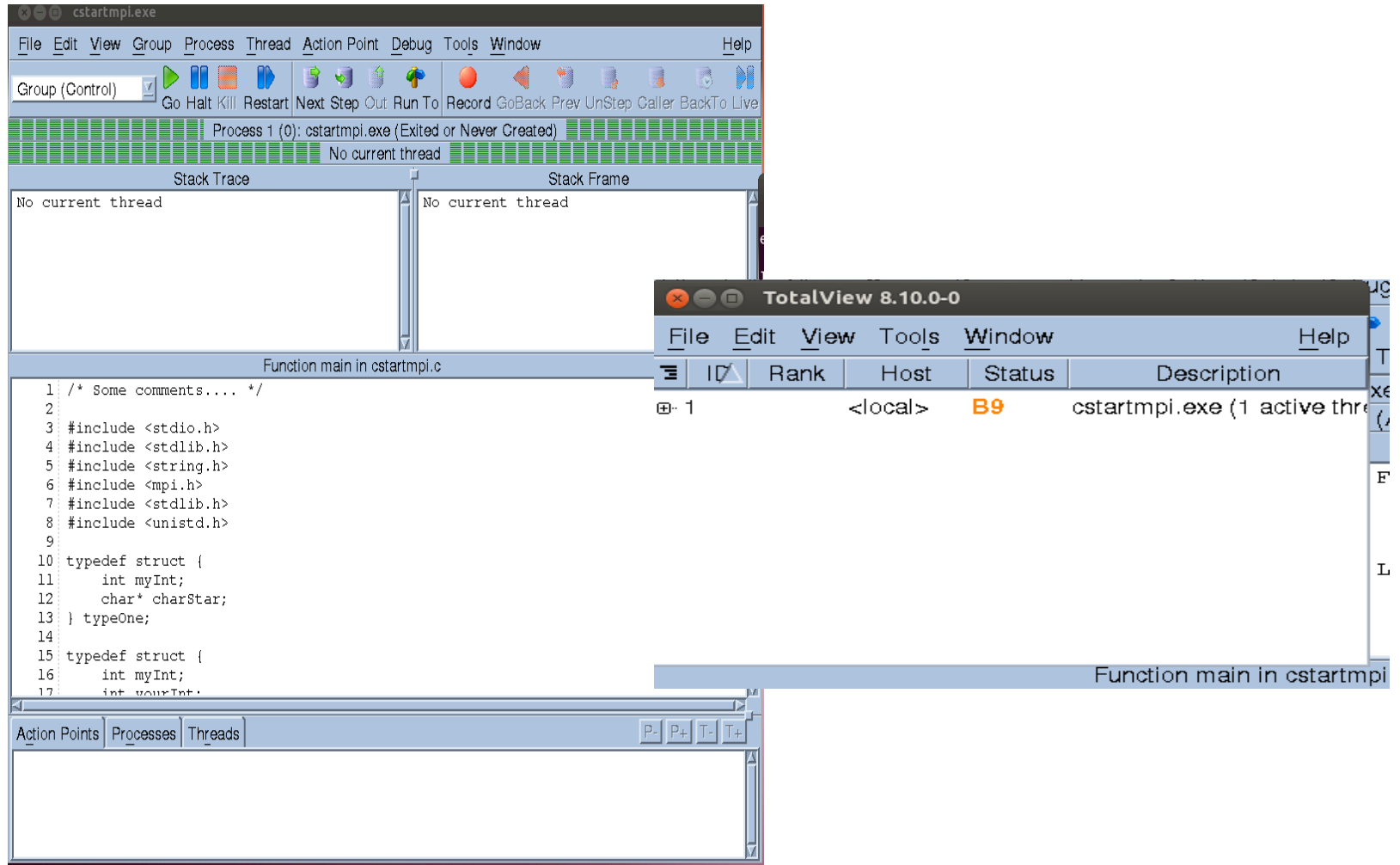
- breakpoints and barriers (static or conditional)
- watchpoints
- evaluation of expressions

TotalView: Usage

Compile binary with debugging information

- flag `-g`
`g77 -g test.f -o test`
- if use `fork()` or `execve()`, link ...
`g77 -g -L/.../totalview/linux-x86-64/lib -ldbfork`
`test.f -o test`
- **Run Totalview**
`totalview executable`
`totalview executable core_file`

TotalView: Window



TotalView: Window

```
82 MPI_Status status; /* Return status for receive */
83
84 t2 = malloc(sizeof(typeThree));
85
86 for(p=0;p<100;p++)
87     bigArray[p]=80000+p;
88
89 MPI_Init(&argc, &argv);
90 MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
91 MPI_Comm_size(MPI_COMM_WORLD, &p);
92
93 dynamicArray = malloc(sizeof(int)*100000);
94 sdim = malloc(sizeof(int) * p);
95
96 for(x=0;x<10000;x++)
97 {
98     dynamicArray[x] = x*10;
```

Action Points	Processes	Threads
9	cstartmpi.c#90	main+0xaa
STOP	10	cstartmpi.c#91 main+0xbe

MemoryScope 3.2.3-0

File Tools Window Help

Home Memory Reports Manage Processes Memory Debugging Options Tips

Summary Add Program

April 22, 2014

Memory Debugging Session

Memory Debugging Guide

Analyze your memory debugging data

Memory debugging data is available for analysis. Select a report to view memory debugging data.

[Leak detection source report](#)

[Heap graphical report](#)

[Heap status source report](#)

Process Status and Control

Process	Event	Event Time	Status	Host	System ID
cstartmpi.exe (40571)			Breakpoint	<local>	40571

Heap Usage Monitor

Graph showing Heap Usage (b) over Time.

Done launching process cstartmpi.exe

DDT: Introduction

What is DDT?

GUI-based debugging tool

Languages supported

C, C++, Fortran 90

Parallel programming models supported

pthread, MPI, OpenMP, CUDA GPU, HMPP

Systems and Platforms supported

Unix, Linux, OS X, Windows

Good Visualization

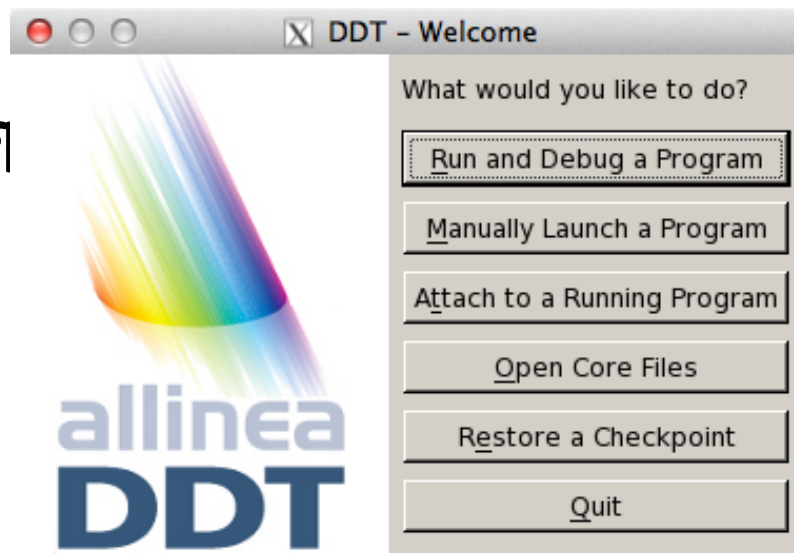
Easy to use, intuitive

DDT: Overview

Compile code with -g fl

On linux systems:

- module load ddt
- ddt &



**Can run it within interactive job, or have
DDT launch job**

DDT: Overview

Running a job

- Enter application name
- Can have DDT launch job, or run interactive job
- Set arguments as necessary

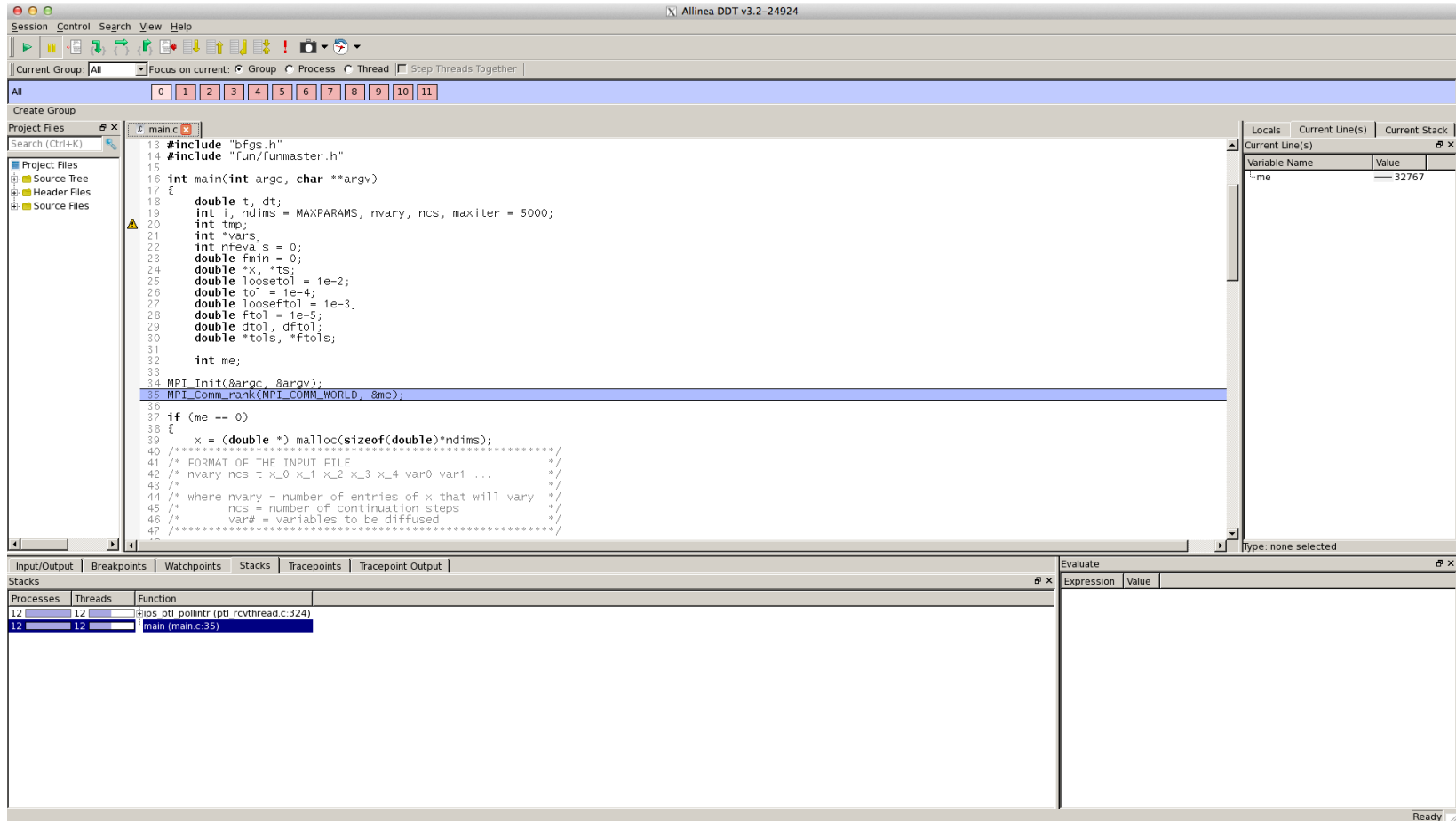
The screenshot shows the 'DDT - Run (queue submission mode)' dialog box. It contains several sections for configuring a job submission:

- Application:** /scratch/director100/rebecca/prog (with a 'Details...' button)
- Application:** /scratch/director100/rebecca/prog (with a file selection icon)
- Arguments:** (empty text field)
- Input File:** (empty text field with a file selection icon)
- Working Directory:** /scratch/director100/rebecca/ (with a file selection icon)
- MPI:** 12 processes, OpenMPI (checked) (with a 'Details...' button)
- Number of processes:** 12 (with a spin button)
- Implementation:** OpenMPI, use queue (with a 'Change...' button)
- mpirun arguments:** (empty text field)
- OpenMP:** (unchecked) (with a 'Details...' button)
- CUDA:** (unchecked) (with a 'Details...' button)
- Memory Debugging:** (unchecked) (with a 'Details...' button)
- Queue Submission Parameters:** Wall Clock Limit=00:30:00, Queue=de (with a 'Details...' button)
- Environment Variables:** none (with a 'Details...' button)
- Plugins:** none (with a 'Details...' button)

At the bottom right, there are 'Submit' and 'Cancel' buttons.

DDT: Overview

Opening Screen



DDT: Overview

Array Viewer

DDT - Multi-Dimensional Array Viewer

Array Expression:

Distributed Array Dimensions: [How do I view distributed arrays?](#)

Range of \$i: From: To: Display:

Range of \$j: From: To: Display:

☒ Align Stack Frames

☐ Auto-update

☐ Only show if: [See Examples](#)

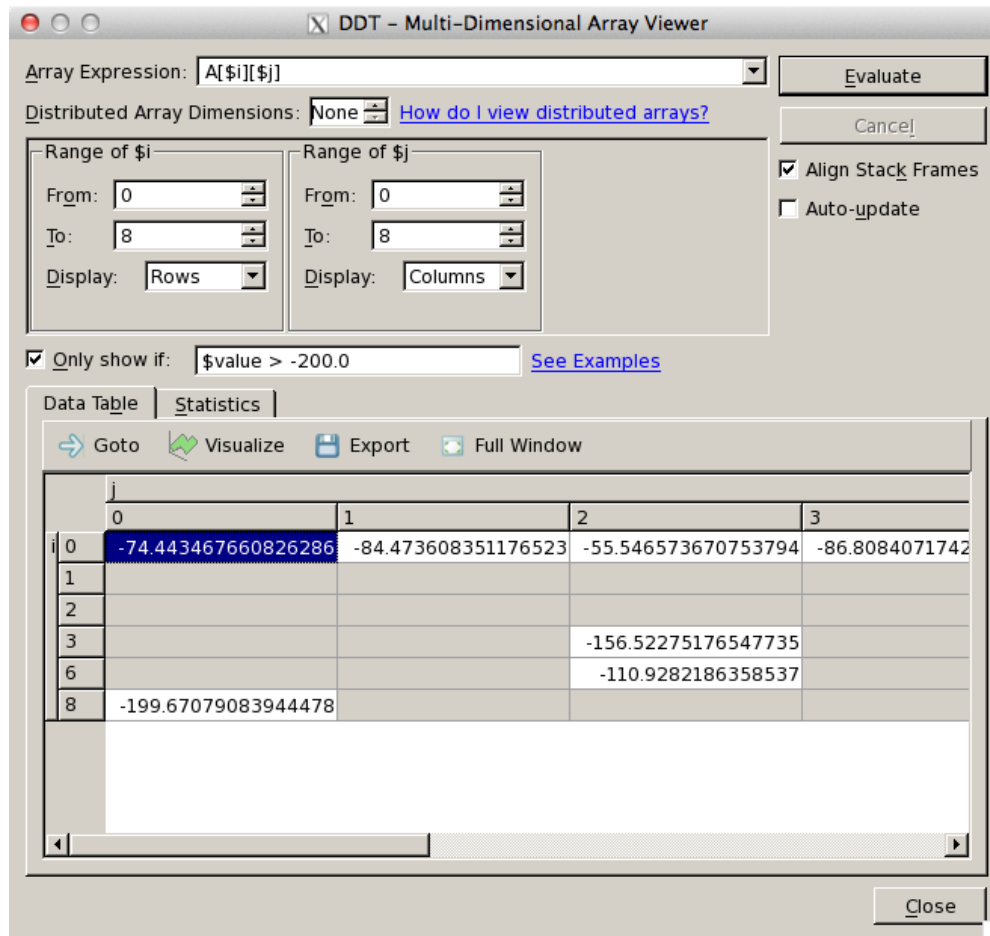
Data Table | Statistics

[Goto](#) [Visualize](#) [Export](#) [Full Window](#)

	j	0	1	2	3
i	0	-74.443467660826286	-84.473608351176523	-55.546573670753794	-86.8084071742
1	-211.18402749079826	-325.01779650217281	-241.90891675477394	-271.569320841	
2	-277.73268191523931	-483.81595902431798	-597.56623376696291	-313.045501255	
3	-217.02101937018665	-271.56932093175834	-156.52275176547735	-337.939110731	
4	-434.51091191481567	-597.67008058182296	-444.54579258206769	-605.722705453	
5	-406.95916711990157	-722.3869104665099	-827.08683532945065	-505.255640238	
6	-303.4246564351003	-334.78540941052427	-110.9282186358537	-518.858135705	
7	-435.22103462062216	-538.85094506904215	-291.96151966340773	-751.284060169	
8	-199.67079083944478	-404.25441035000853	-502.8721054666388	-374.556464211	

DDT: Overview

Array Viewer – see all $A[i][j] > -200.0$



DDT: Usage - Steps

Using DDT: Step 1 -- Compiling

- Compile your code with the usual compiler and -g flag
- If optimizations are on, line numbers may be misaligned or inexact

Using DDT: Step 2 -- Running

- You must have logged in with flags to allow X-forwarding
- `ssh -X user@epic.ivec.org (linux)`
- `ssh -Y user@epic.ivec.org (mac)`
- DDT can launch parallel interactive jobs for You
- Or, you can launch the interactive job and run DDT inside

DDT: Usage-Steps

Running from Interactive Job

- `qsub -l -V -X -lwalltime=00:30:00 -W group_list=yourgroup -q debugq"`
- `-l` = interactive
- `-V` = keep environment variables (useful if ddt module already loaded) •
 `-X` = allow X-forwarding
- Once job is running, invoke ddt: `ddt &"`

Using DDT: Step 3 -- Debugging

- Set breakpoints
- Start/Pause/restart
- Look at variables
- Look at state of program on each processor
- Run program until condition occurs (i.e., stop when `x=6`)



Thanks!

Any questions?