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

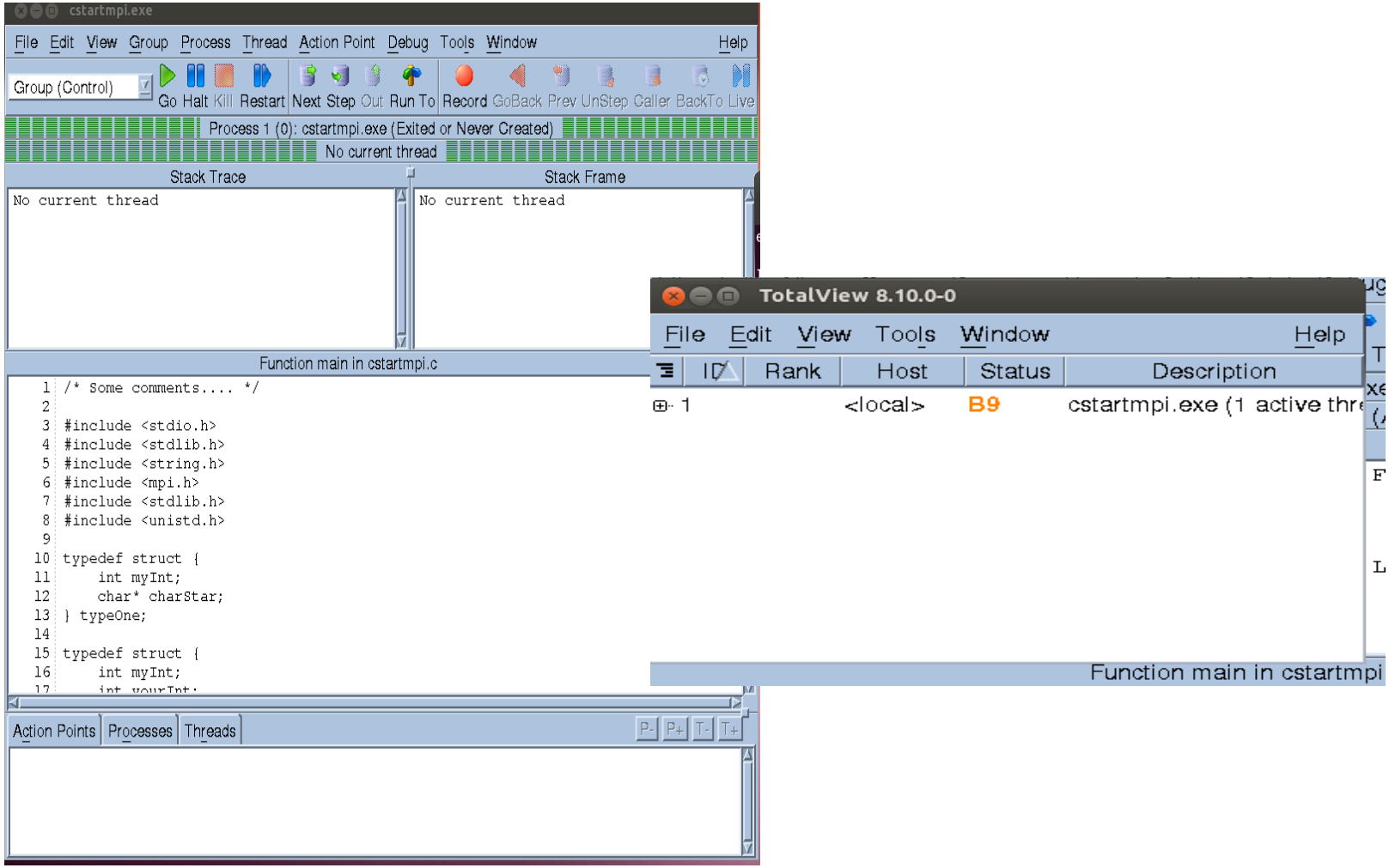
test.f –o test

- **Run Totalview**

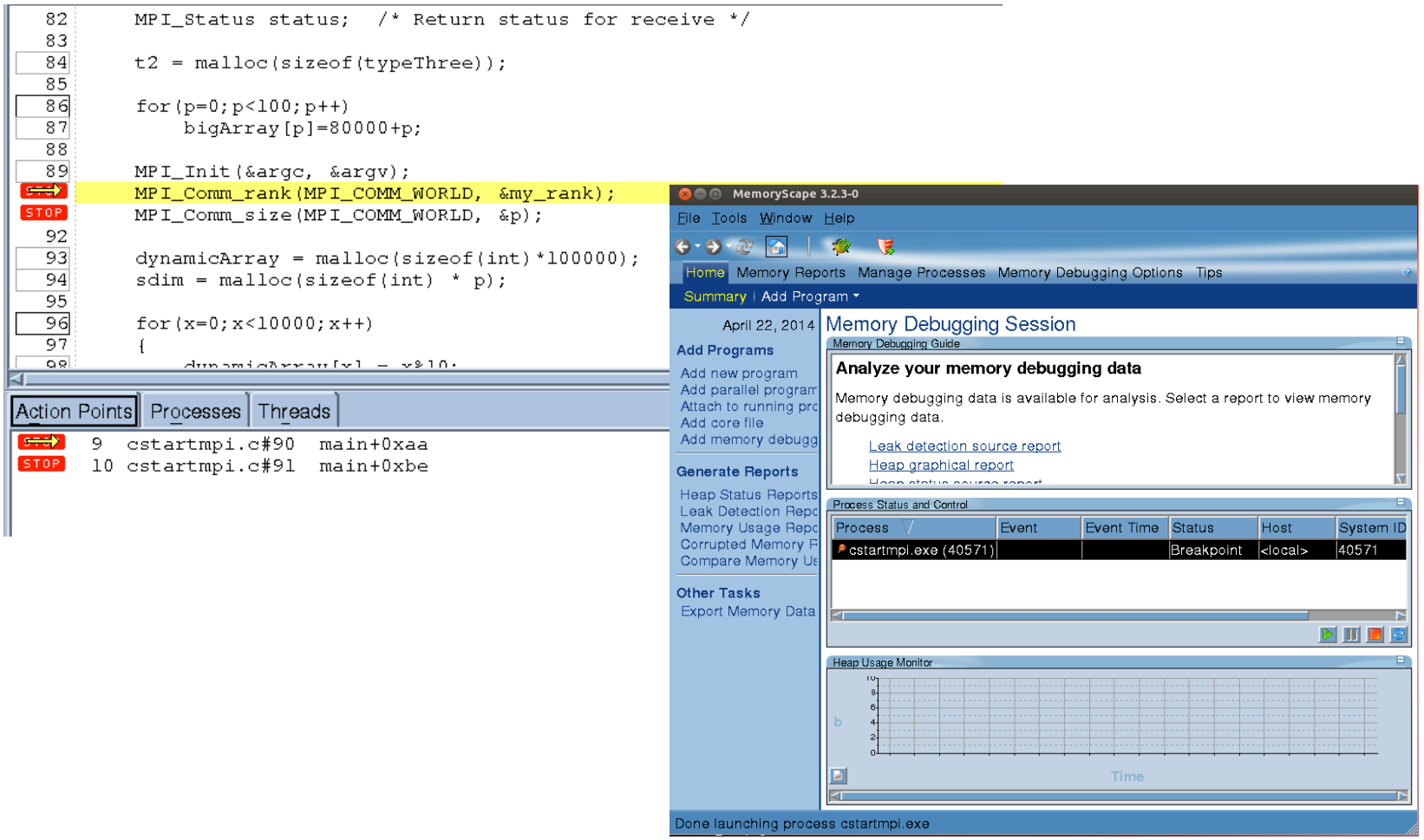
  totalview executable

  totalview executable core_file

TotalView: Window



TotalView: Window



DDT: Introduction

What is DDT?

GUI-based debugging tool

Languages supported

C, C++, Fortran 90

Parallel programming models supported

pthreads, 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:

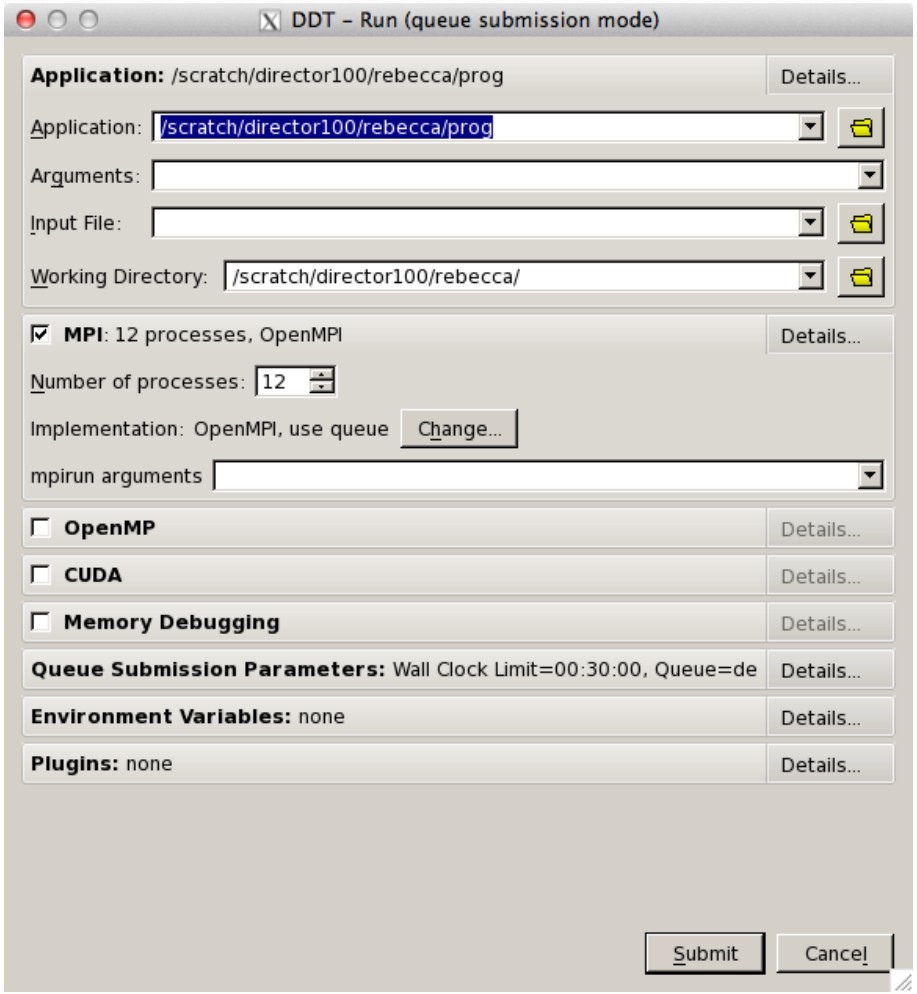
- module load ddt
- ddt &

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

DDT: Overview

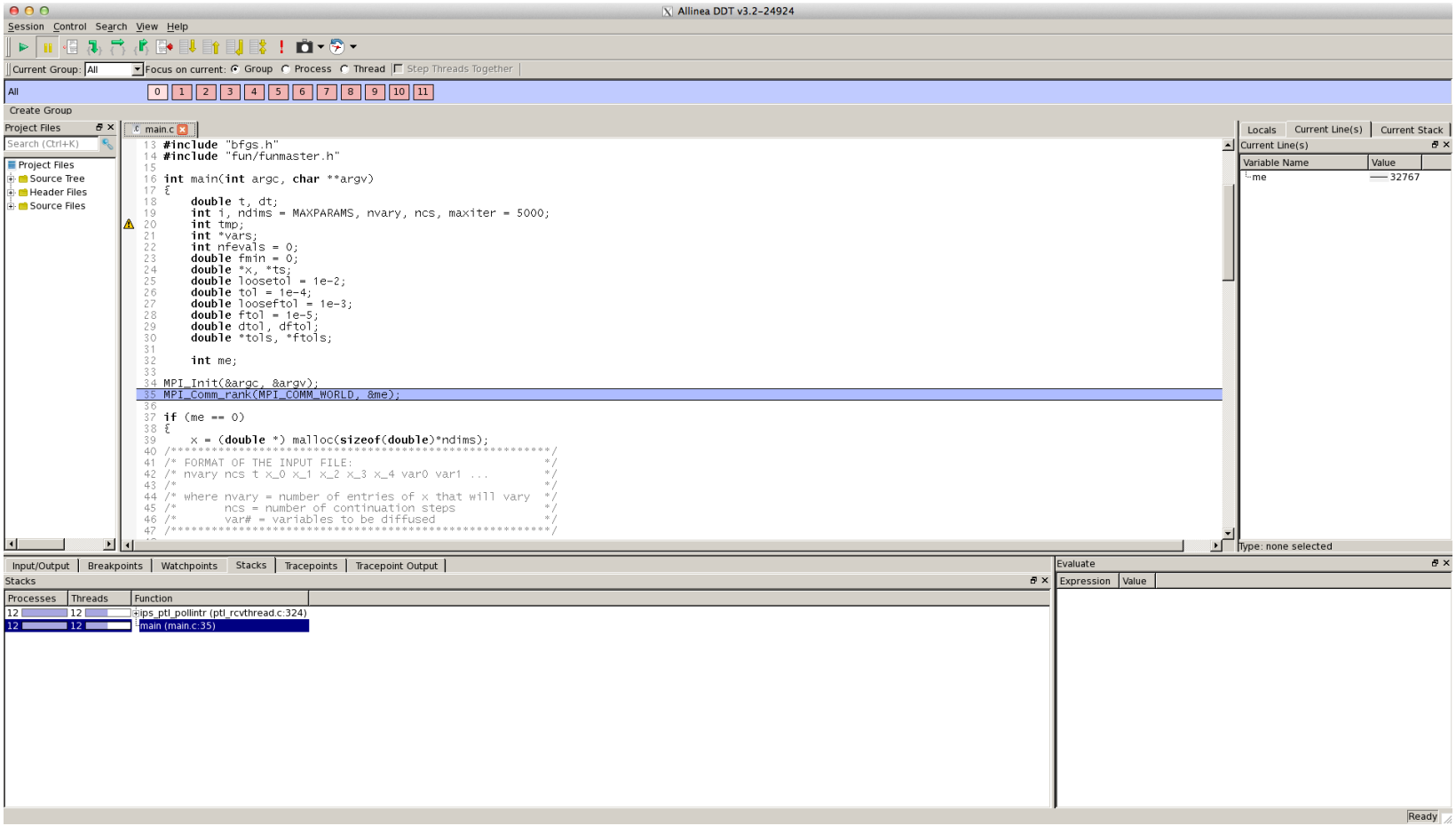
Running a job

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



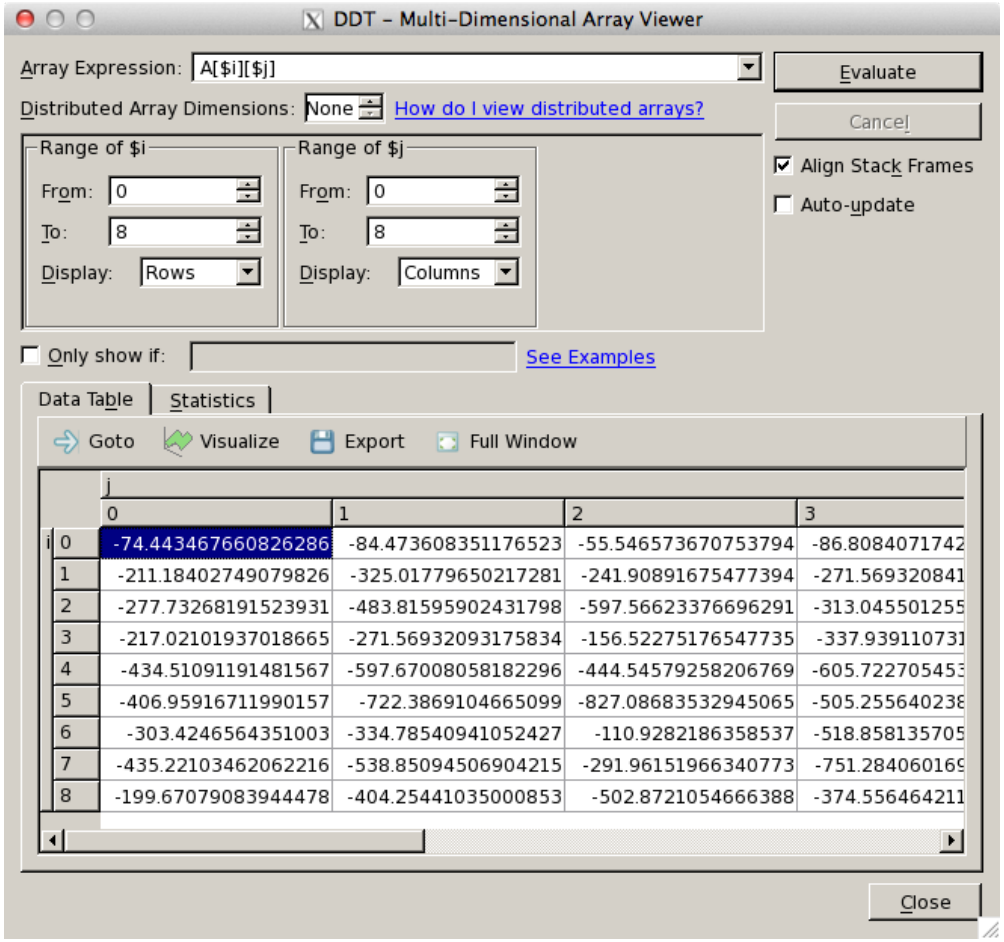
DDT: Overview

Opening Screen



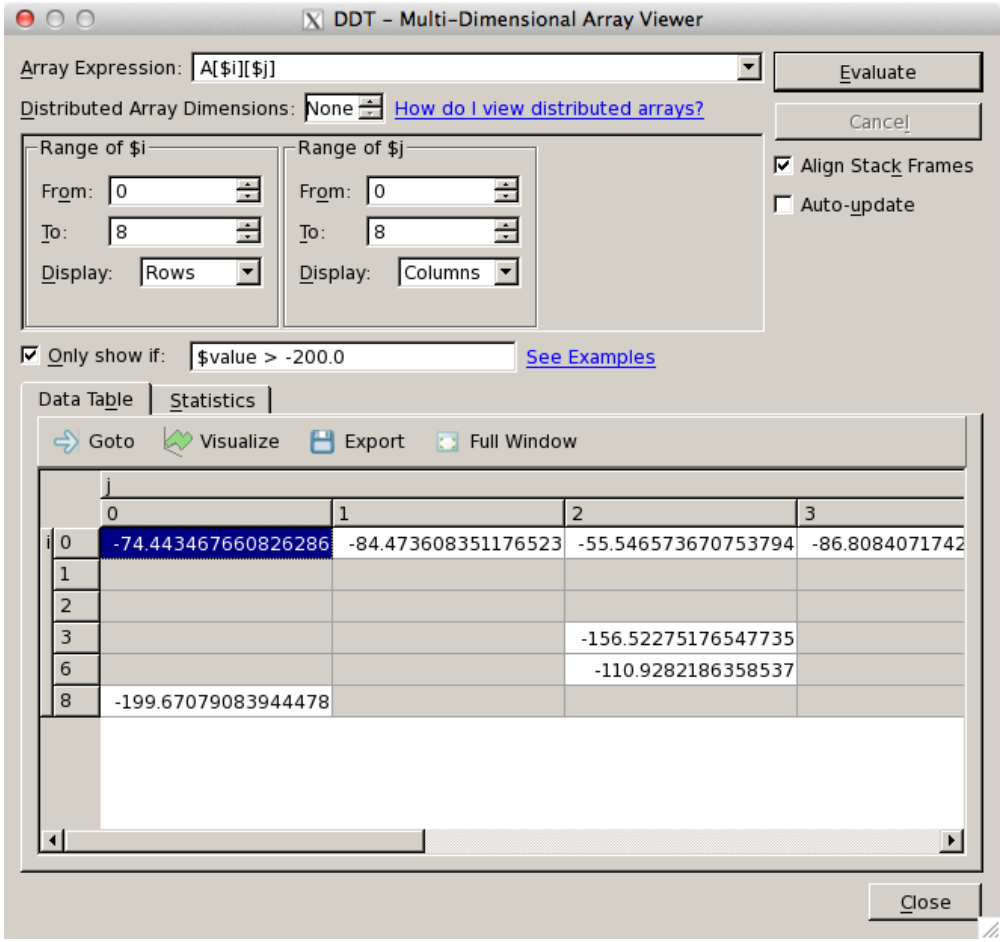
DDT: Overview

Array Viewer



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 -I -V -X -lwalltime=00:30:00 -W group_list=yourgroup -q debugq"
- -I = 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?