CS 280 Presentation:

# Introduction to TotalView and DDT

NAME: XIAOPENG XU KAUST ID: 129052

# **Contents:**

1. Why we need parallel debugging – Problems with parallel debugging.
2. TotalView
3. Introduction to TotalView
4. Usage of TotalView
5. Demo
6. DDT
   1. Introduction to DDT
   2. Usage of DDT
7. Why we need parallel debugging – Problems with parallel debugging

Problems with debugging for Parallel programs:

All problems of serial programming

Additional problems

Difficult to verify correctness of program

Difficult to debug N parallel processes

New parallel problems: deadlocks, race conditions, irreproducibility

Parallel Debugging Tools:

TotalView

DDT

1. Introduction to TotalView

**What is TotalView?**

GUI-based debugging tool

full-featured, source-level, multi-process, multi-thread graphical debugger

**Languages supported**

C, C++, Fortran 77 & 90, UPC, PGI, HPF, assembler, and mixed codes

**Parallel programming models supported**

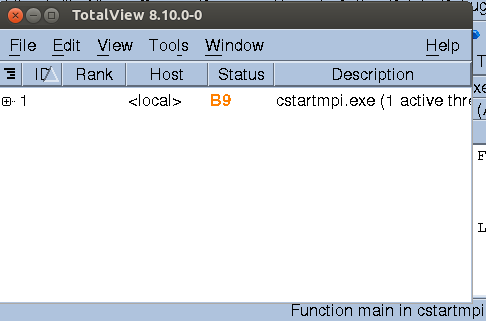
MPI, PVM, OpenMP, pthreads, SHMEM, CUDA GPU, Intel MIC, OpenACC, Intel Xeon Phi processor, and coprocessors

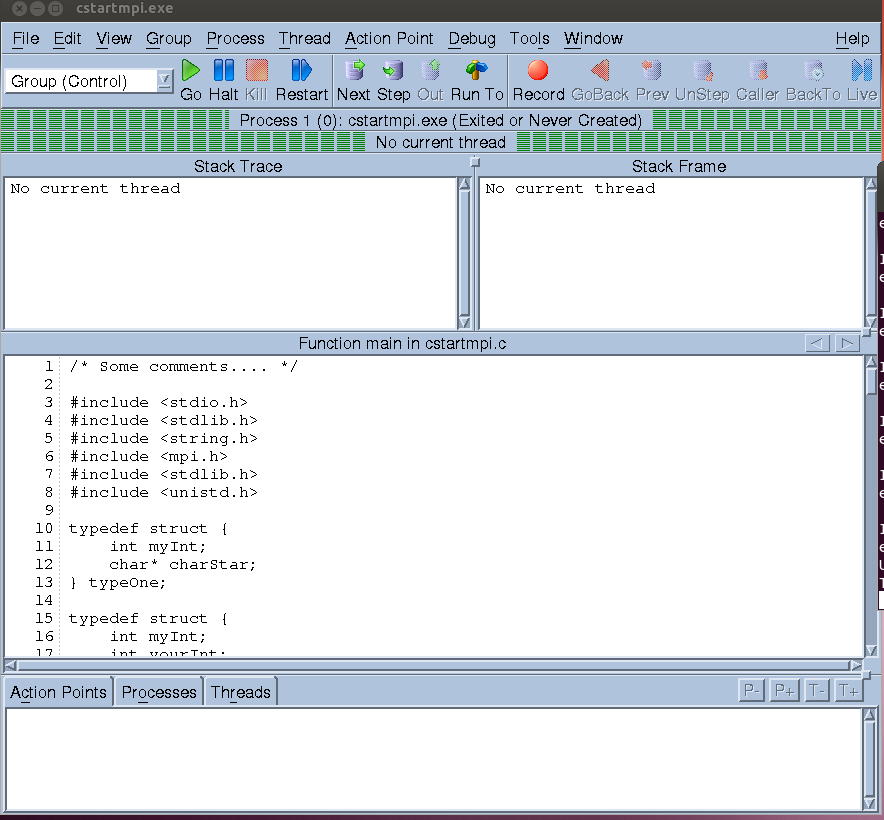
**Systems and Platforms supported**

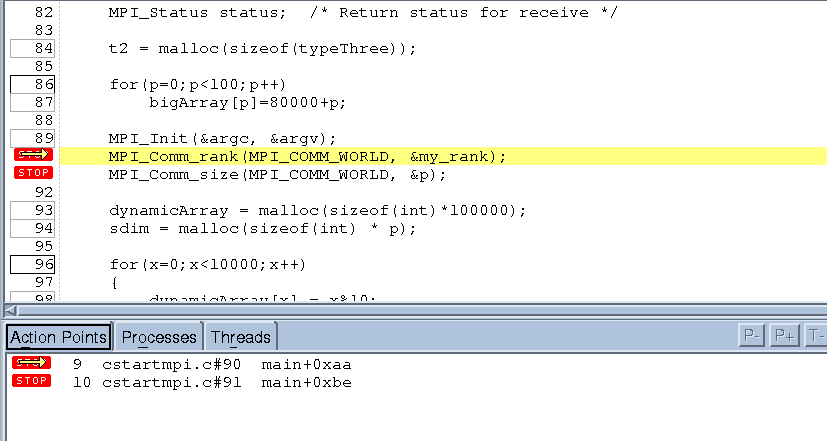
Unix, Linux, OS X, Windows

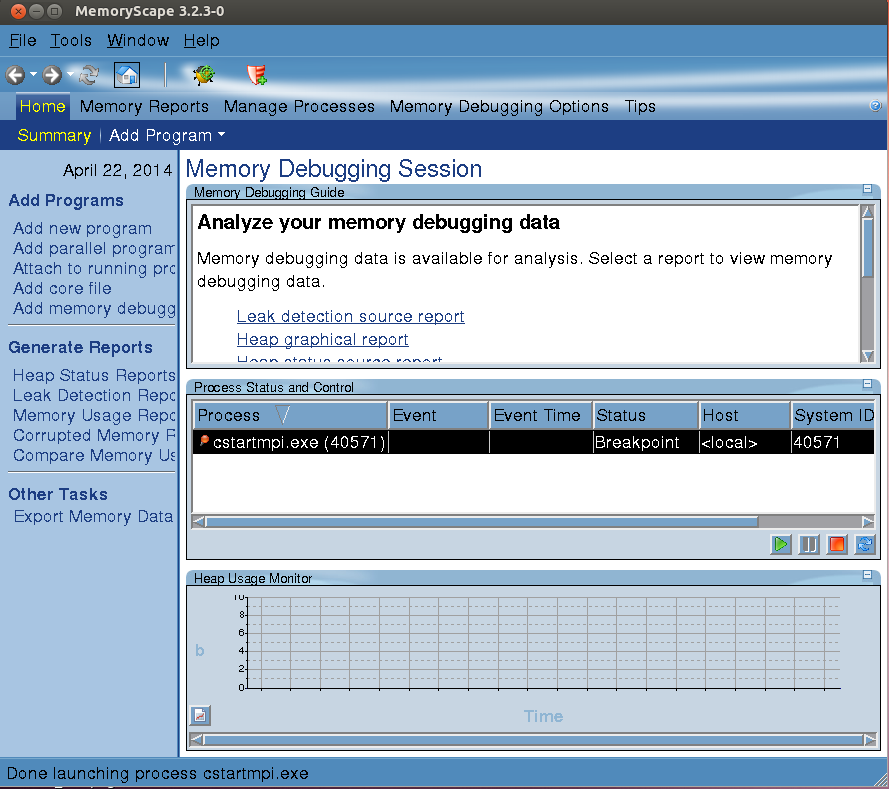
**Integrated Memory Debugging**

1. Usage of TotalView









1. Introduction to DDT

**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

**Overview**

Compile code with -g flag

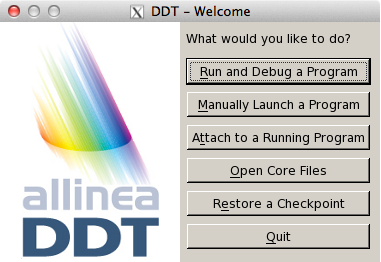
• On linux systems:

module load ddt

ddt &

• Launch DDT from scratch directory

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



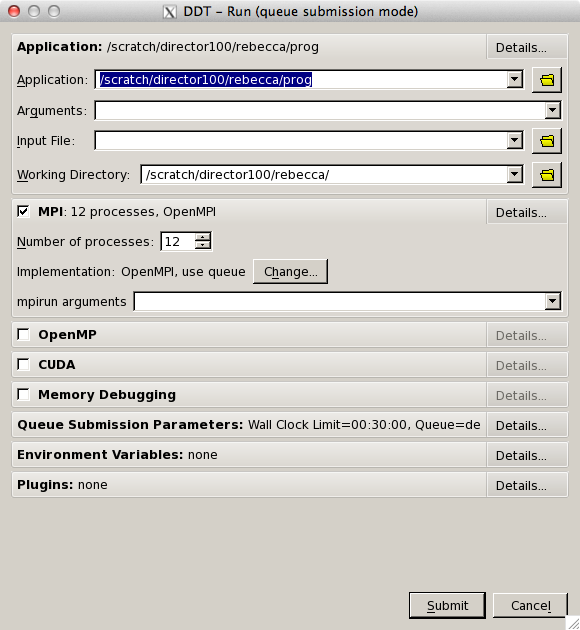
Running a job

• Enter application

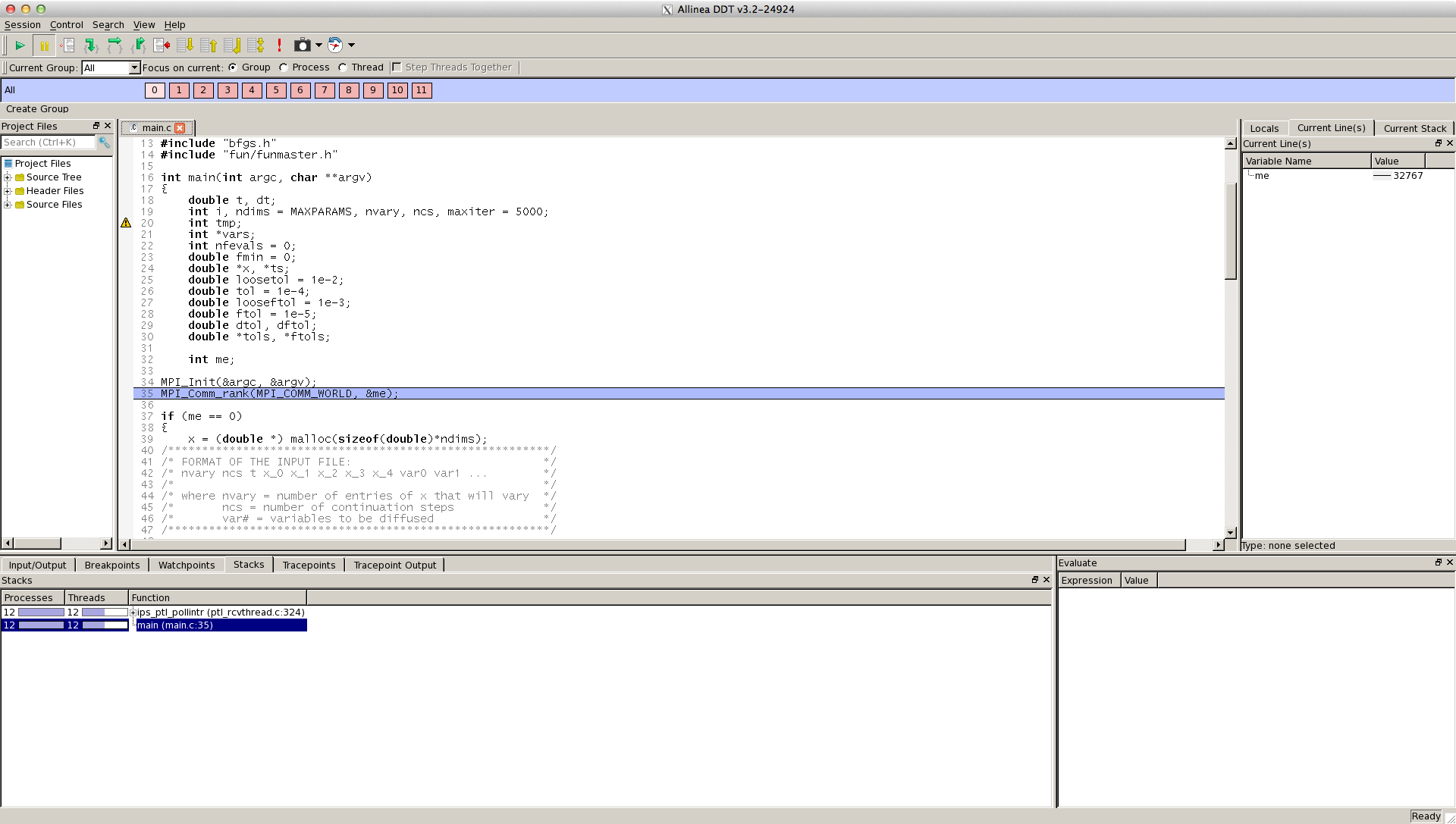
name

• Can have DDT launch job, or run interactive job

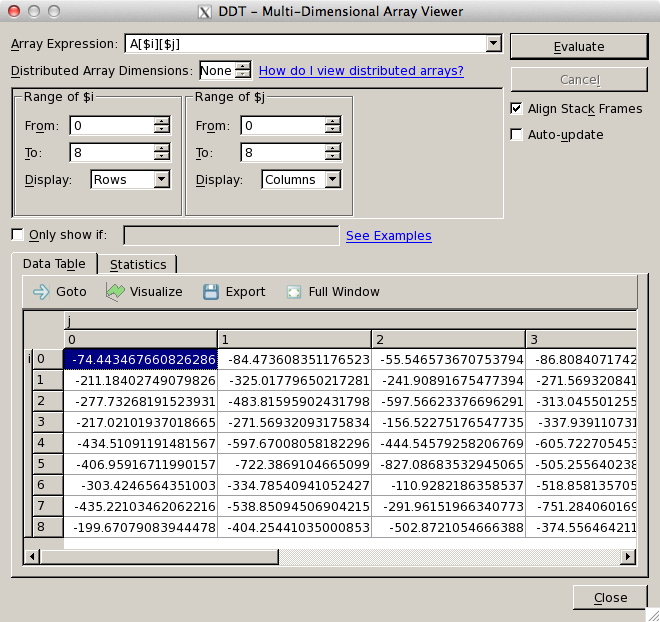
• Set arguments as necessary



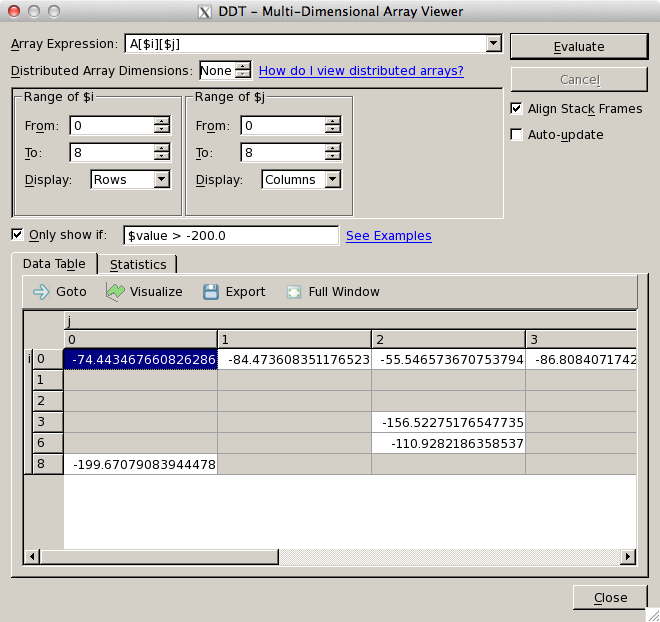
Opening Screen



Array Viewer



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



1. Usage of DDT

Using DDT: Step 1 -- Compiling

• Compile your code with the usual compiler and -g flag

• Works better if all optimizations turned off

• For some compilers, debug flag automatically turns

off optimizations

• 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)

• Verify X-forwarding by invoking xterm & – if a terminal window appears, X-forwarding works (close it and proceed)

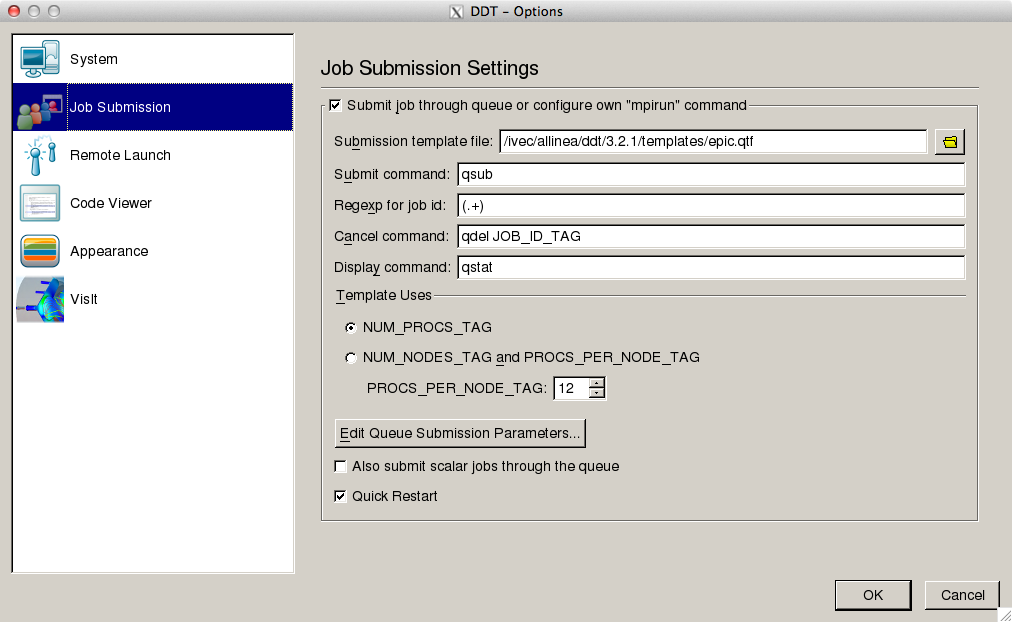
• module load ddt"

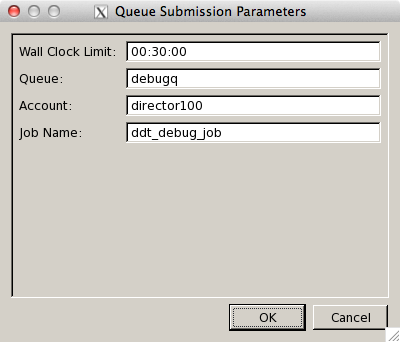
• DDT can launch parallel interactive jobs for

you

• Or, you can launch the interactive job and run DDT inside (I prefer this)

Setting Queuing Parameters





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 &"

Make sure to untick “Submit job through queue”

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)

**Reference:**

<http://www.roguewave.com/portals/0/products/totalview-family/totalview/docs/8.12/>**, TotalView online documentation.**