DDT: A visual, parallel debugger on Ra

David M. Larue
dlarue@mines.edu

High Performance & Research Computing
Campus Computing, Communications, and Information Technologies
Colorado School of Mines

March, 2010
Outline

1. History
   - Ancient
   - Medieval
   - Modern

2. The Setup
   - Prerequisites
   - One-time
   - Each time

3. Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
Programs hang, give incorrect output or exhibit unexpected behavior.

Parallel computing is not deterministic.
- Things can happen in different orders or with different timings.
  - available resources can differ
  - different optimizations are applied
  - job sizes and inputs can exacerbate these issues.
- These differences can alter outcomes.

Hence programs can work some of the time, and fail some of the time.

Debugging is needed.
Outline

1. History
   - Ancient
   - Medieval
   - Modern

2. The Setup
   - Prerequisites
   - One-time
   - Each time

3. Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
Old Style

- Insertion of statements that write to a file or standard output
  - help ascertain that the program is doing what it was intended to do.
  - sometimes surprisingly effective
  - often easy to implement
- However it has serious limitations.
  - slows and interrupts your thoughts
  - requires many recompiles and restarts
  - prevents interactive use
Outline

1. History
   - Ancient
   - Medieval
   - Modern

2. The Setup
   - Prerequisites
   - One-time
   - Each time

3. Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
Command-line text-oriented serial tools
- Example: GDB (the GNU Project debugger)
- GDB was designed for serial processing
- It can be started multiple times to inspect and control multiple processes.

Many algorithms in use in a parallel environment have their beginnings as serial code
- correctness there is needed
- serial debuggers are then immediately useful
Outline

1 History
   - Ancient
   - Medieval
   - Modern

2 The Setup
   - Prerequisites
   - One-time
   - Each time

3 Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
High Style

- Debuggers with
  - Graphical User Interfaces
  - Support for Parallel Execution

- Advantages:
  - Intuitive
  - Interactive
  - Manipulate multiple processes as if one entity.

- Don’t forget Profilers
Product *du jour*: DDT

- commercial graphical interface
- built on top of GDB
- designed with debugging parallel programs in mind.
- made by Allinea.

Documentation for DDT:

- *DDT User Guide*, Version 2.5.1

Its home on Ra is

- `/lustre/home/apps/ddt/bin/ddt`
Ra’s DDT License
- 16 processes

More information on debugging on Ra

- [http://geco.mines.edu/workshop/class2/03wed/debuggers.pdf](http://geco.mines.edu/workshop/class2/03wed/debuggers.pdf)

*Debugging on Ra*
Timothy H. Kaiser, Ph.D., Revised January 2010


*Debugging Programs whilst not bugging yourself*
Timothy H. Kaiser, Ph.D., January 2010

- [http://geco.mines.edu/guide/body.shtml#h5450](http://geco.mines.edu/guide/body.shtml#h5450)

*Ra User’s Guide*, section on *Debugging*, including references to workshop resources.
Outline

1. History
   - Ancient
   - Medieval
   - Modern

2. The Setup
   - Prerequisites
   - One-time
   - Each time

3. Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
Things to have and do

- `ddt`
- tools that support debugging and `ddt`
  - compiler
  - mpi library
  - `gdb`
- access Ra with X Windows
- setup `ddt`
- compile with debugging support
Combinations of Versions

- various combinations work
- many do not
- one will be detailed here
  - OpenMPI 1.3.4
  - Intel Compilers 11.1
  - DDT 2.5.1
  - BASH
  - ssh -X
  - gdb 6.8

Non-standard combination of versions requires extra setup
**Modes**

- DDT submits a job to the queue and attaches to its processes
- DDT attaches to the processes of an already running job
- We will let DDT submit the job
  - need a PBS template script
  - configure DDT to msub
Outline

1 History
   - Ancient
   - Medieval
   - Modern

2 The Setup
   - Prerequisites
   - One-time
   - Each time

3 Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
One-Time System Setup

Append the following to the end of ~/.bashrc (then source it or logout/login).

```bash
### openmpi 1.3.4, intel 11.1 compiler wrapper settings  
### (LD_LIBRARY_PATH, PATH)
source /lustre/home/apps/mpi/db/openmpi1.3.4/intel11.1/setup.sh

### ddt 2.5.1 settings  
### (LD_LIBRARY_PATH, PATH, DMALLOCPATH)
source /lustre/home/apps/ddt/setup.sh
```

This sets the paths and environment variables that point to the executables and libraries for the specific versions of OpenMPI, the compiler, GDB and DDT.
One-Time DDT Setup, Part 1/2

- Login to Ra enabling X-Windows
  - From most Unixes, type `ssh -X ra`.
- Start DDT for the first time
  - `type ddt &`
    - this creates a `~/.ddt/` directory where settings are stored.
  - go through the *Configuration Wizard*.
  - Alternately,
    - start DDT
      - `select Session / Options... or` Session / Configuration Wizard...
Options for DDT:

System Settings
- MPI Implementation: OpenMPI (Compatibility)
- Debugger: GNU (gdb)

Job Submission Settings
- Submit job through queue: checked
- Submission template file: /lustre/home/[username]/workshop/ra.qtf
- Submit command: msub
- Regexpr for job id: (\d+)
- Cancel command: qdel JOB_ID_TAG
- Display command: qstat
- Template Uses:
  - NUM_NODES_TAG, PROCE_PER_NODE_TAG: selected
  - PROCS_PER_NODE_TAG: 8 (or less...)

(Version 11 of the Intel Compilers requires selecting the OpenMPI (Compatibility) MPI Implementation.)
A simple PBS script `prog.pbs` to submit a job on Ra reads

```bash
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l walltime=00:010:00
#PBS -N testddt
#PBS -o $PBS_JOBID.out.pbs
#PBS -e $PBS_JOBID.err.pbs
#PBS -r n
#PBS -V
#-----------------------------------------------------
    cd $PBS_O_WORKDIR
    mpirun -np 16 ./prog
```

One submits this by executing

```bash
msub prog.pbs
```
One-Time Submission template file setup, Part 2/4

To have DDT submit the job, we need a template `ra.qtf` for the PBS file with named tags that DDT will fill in.

```bash
#!/bin/bash
#PBS -l nodes=NUM_NODES_TAG:ppn=PROCS_PER_NODE_TAG
#PBS -l walltime=00:10:00
#PBS -N testddt
#PBS -o $PBS_JOBID.out.pbs
#PBS -e $PBS_JOBID.err.pbs
#PBS -r n
#PBS -V
#-----------------------------------------------------
cd $PBS_O_WORKDIR
MPIRUN_TAG \   
    AUTO_MPI_ARGUMENTS_TAG EXTRA_MPI_ARGUMENTS_TAG \   
    DDTPATH_TAG/bin/ddt-client \ 
    DDT_DEBUGGER_ARGUMENTS_TAG \ 
    PROGRAM_TAG \ 
    PROGRAM_ARGUMENTS_TAG
```
Here are what these tags might evaluate to:

PROGRAM_TAG /lustre/home/[username]/workshop/prog
PROGRAM_ARGUMENTS_TAG
NUM_NODES_TAG 2
PROCS_PER_NODE_TAG 8
DDT_DEBUGGER_ARGUMENTS_TAG
    --ddthost ra.mines.edu --ddtport 4242 --ddtsession 1
    --ddtsessionfile
        /lustre/home/[username]/.ddt/session/ra.mines.edu-1
MPIRUN_TAG mpirun
AUTO_MPI_ARGUMENTS_TAG -np 16
EXTRA_MPI_ARGUMENTS_TAG
DDTPATH_TAG /lustre/home/apps/ddt2.5.1

Other tags are pre-defined, and user-defined tags can be made.
Takeaway:

- create a submission template file
- name and place it as specified in the DDT options
- example:
  
  /lustre/home/[username]/workshop/ra.qtf
Outline

1. History
   - Ancient
   - Medieval
   - Modern

2. The Setup
   - Prerequisites
   - One-time
   - Each time

3. Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
Compile with the debug flag

Using the appropriate MPI Compiler Wrapper for the Intel Compilers

ifort  icc

compile your program with the debug flag, -g:

mpif90  -g  -o prog prog.f90
mpicc    -g  -o prog prog.c

Probably you want to compile without optimizations, as these make the connection between your source code and the running code more tenuous.

(A warning about feupdateenv may be harmless, but can be removed by adding the flag -shared-intel.)
Start DDT, Submit Job

- Login to Ra with X-Windows enabled, `ssh -X ra`.
- Change to the program directory, `cd ~/workshop`.
- Start DDT, `ddt &`.
- Prepare to run a program
  - From the Welcome screen menu, select Run and Debug a Program.
  - From the main menu, select Session / New Session... / Run...
- At the DDT - Run (queue submission mode) window:
  - Set Application to your executable, say `/lustre/home/[username]/workshop/prog`
  - Set Arguments as needed
  - Verify Number of Nodes (1 or 2, say)
- Press Submit
Wait for Queue, DDT

- DDT will submit your job to the queue
- DDT displays the result of `qstat`, highlighting your Job ID
- You wait until your job runs
- DDT connects to the processes your job has started
- Your processes have been paused after `MPI_Init`
- Sample screen with successful connection follows
Successful Submission, Run and Connection
Outline

1 History
   - Ancient
   - Medieval
   - Modern

2 The Setup
   - Prerequisites
   - One-time
   - Each time

3 Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
Serial Bugs, 1/2

Parallel programs suffer from all the same flaws as serial programs:

- out of bound memory access
- division by 0 and other improper arguments to functions
- overflow or underflow
- syntactical and semantical errors
- other algorithm flaws
All the usual debugging techniques for serial programs are available here in DDT:

- one can set break points
- examine or change the values of variables
- step over or into statements and function calls

One can do these things

- in individual processes, or
- keep all the processes in sync.
Parallel Bugs

Three often cited classes of new types of bugs in parallel code:

- *Data Race Conditions*
- *Deadlock*
- *LiveLock*
Data Race Conditions

- Shared Memory (OpenMP)
  - one process can write to memory
  - at the same time another process can read it
  - inadequate synchronization methods are present
  - the result depends on the sequence of actions

We are focusing here on message passing and MPI, this is not where we dwell, but mention that DDT has debugging support for

- OpenMP
- threads
- memory

Debuggers change timing, so these can be very difficult to locate.
Deadlock

- Shared Memory (OpenMP) and Message Passing (OpenMPI)
  - multiple blocked processes are waiting on each other
    - for release of locks
    - for messages
    - or terminating prematurely
  - circular locking:
    A waits for B waits for C waits for A, and the like.
  - protocol mismatch:
    A expecting from B,
    B not expecting to send, or sends with wrong tag
Livelock

- Shared Memory (OpenMP) and Message Passing (OpenMPI)
  - multiple non-blocked processes are repeatedly waiting on each other
  - timing coincidence keeps from resolution

This can manifest rarely and occasionally, without a blocked process, and so be hard to find.
Outline

1. History
   - Ancient
   - Medieval
   - Modern

2. The Setup
   - Prerequisites
   - One-time
   - Each time

3. Usage
   - Common classes of Parallel Programming Bugs
   - Features
   - Sample Run
Features: User Interface

- User Interface
  - GUI
  - Tabbed documents
  - Dockable windows
  - Save/Restore preferences
  - Search/Go to/Highlight current/View Multiple source code

- Control methods
  - Icons/Buttons
  - Menus
  - Function keys
  - Mouse clicks, double-clicks, right-clicks
Features: Groups

- Groups
  - Default
    - All
    - Root
    - Workers
    - Individual Processes
  - Can add/edit/delete groups
  - “Current Group” Affects
    - Which Processes you start/stop/step
    - Which Variables displayed
    - Which Code and Line, and which Stack, is visible
Features: Controlling Your Processes

- Controlling Your Processes
  - Advancing by a line (Stepping over)
  - Stepping into a function
  - Stepping out of a function
  - Following if branches
  - Pausing at breakpoints
  - Manually pausing and resuming

- Control individually, or by group
Features: Viewing and Editing Data

- Viewing and editing data
  - Current line
  - Local data
  - Keeping an expression in view: Evaluate
  - Watch: break on change
  - Editing data

- Structured data
  - Examine pointers
  - Multi-dimensional arrays
  - Visualizations, Statistics
  - View Message Queues

- Memory debugging
  - Intercept [de]-allocations
  - Statistics
Outline

1 History
   • Ancient
   • Medieval
   • Modern

2 The Setup
   • Prerequisites
   • One-time
   • Each time

3 Usage
   • Common classes of Parallel Programming Bugs
   • Features
   • Sample Run
Time and Technology permitting, let’s try a sample run
Summary

We have looked at

- Why debug?
- Different debugging technologies
- A modern product available on Ra: DDT
  - GUI
  - Parallel
  - Multiple modes

- Setup
  - One time: configure tools, template PBS script
  - Each time: compile with debugging, use X

- Bugs
  - Serial
  - Parallel: Race Conditions, Deadlock

- DDT Features
  - User Interface
  - Process Control
  - Data and Memory Control