How to use Allinea's DDT and MAP Tools

Professor Glenn Luecke
Summer 2015

Introduction

DDT is a parallel debugger from Allinea. It has a graphical user interface and can be used for debugging Fortran, C, and C++ programs that have been parallelized with MPI and/or OpenMP, as well as UPC programs. Supported languages and paradigms: C/C++, Fortran, F90, Coarray Fortran, UPC, CUDA and OpenMP. In most cases programs can be debugged faster using DDT rather than inserting print statements into the program.

MAP is a low-overhead, easy-to-use profiler from Allinea that can be used to profile and optimize serial, MPI, UPC, CUDA and OpenMP programs. To optimize a program, one should first find those portions of the program where most of the execution time is occurring. Performance can be affected by load imbalance between processors in parallel programs, cache misses and inability of compilers to optimize specific loops. To find which parts of the code should be modified to improve performance, one can use MAP from Allinea. Using MAP for performance assessment will speed up the processes of program optimization. (Note: We currently do not have a license on the student cluster to use DDT and MAP for GPUs.)

Both DDT and MAP are part of the Allinea Forge tool. The Allinea Forge User Guide can be found at http://content.allinea.com/downloads/userguide-forge.pdf . DDT and MAP are available on the student, Condo and CyEnce clusters. DDT and MAP information for each machine is available at:
http://hpcgroup.public.iastate.edu/HPC/hpc-class/DDT.debugger.html
http://hpcgroup.public.iastate.edu/HPC/CyEnce/DDT.debugger.html
http://hpcgroup.public.iastate.edu/HPC/Condo/DDT.debugger.html

Setting Up

To ensure your environment is set to use DDT and MAP, issue "which ddt" and "which map":

```
which ddt
/shared/allinea/forge/bin/ddt

which map
/shared/allinea/forge/bin/map
```

Next check whether your desktop computer accepts X connections by first logging onto Condo, CyEnce or hpc-class using ssh –X <machine name>.its.iastate.edu and then issuing:

```
xterm &
```

This will open a window if your desktop computer accepts X connections. To use DDT and MAP, your desktop computer must accept X connections. For Apple desktop machines, one will need to install XQuartz if your Mac is running 10.8 or 10.9. If you are running 10.10 or newer then X11 comes with the Mac OS utilities. For Windows install the Xming X server http://www.straightrunning.com/XmingNotes/ . See the guide for X forwarding with PuTTY & Xming at http://www.ece.unm.edu/csg/email/XServer_Putty_Windows7-ECE.pdf .

If one has a low bandwidth internet connection, then better response times will likely occur when using **Allinea's Remote Host Client** software from http://www.allinea.com/products/download-allinea-ddt-and-allinea-map. See the machine's web site for information. (Currently, July 2015, Allinea's Remote Host Client has some bugs and it is not recommended to install this until the bugs have been fixed.)

How to run DDT and MAP

To use both DDT and MAP, one must compile the program with the -g option and then start DDT and/or MAP as shown below , filling in the information requested by the Allinea window. If you select MPI then you should enter the following:

- number of (MPI) processes
- number of nodes
- MPI run arguments

If you select OpenMP then you should specify the number of threads. (Remember to compile with -qopenmp option for OpenMP programs.) To use MAP or DDT for an MPI program named "prog.f90", enter

```
mpiifort -g prog.f90    and then
map ./a.out   OR
ddt   ./a.out
```

Be sure to select "Submit to Queue" so the job will be submitted to a queue via the "qsub" command in the pop-up window and then click on Submit and the map/ddt window will appear.

Compiler optimization options can be kept when compiling, but the -g option is needed so performance information can be applied to the source code.

Configuration Information. The first time one uses DDT or MAP, one will need to provide some configuration information. See the machine's web site for information.

Sample Fortran programs can be copied from **/home/SAMPLES/GRL** on condo and share.

ISU currently has **64 license seats** that are shared between all users. For example, if one uses 64 MPI processes, then nobody else can be running DDT/MAP at that time.

Part A: Using DDT

It is recommended to reduce the problem size when debugging a program since this will provide fast response time when using DDT.

The following is a list of the important functions of DDT:

- **Stepping** though the program one statement at a time and displaying variable values. Values are displayed by clicking on "Locals" or "Current Line(s).
- **Running** the program to a selected statement where a breakpoint has been set and then displaying variable values.
- **Stepping into and out from** functions/subprograms.

- Running a program and having the program stop execution where a problem occurs. Then one knows where the program stopped execution and values of variables at that point.
- Stepping can be done with all MPI processes stepping together or with selected groups of MPI processes.

Controlling Program Execution. The following summarizes the most useful items from the toolbars at the top of the DDT window:

- ***File***: new session, restart session and quit
- ***View***: increase/decrease zoom
- ***Control****:* allows one to select many options for stepping through the program: play/continue, pause, add breakpoint, step into, step over, step out, run to line …., etc.
- ***Tools***: Multidimensional array viewer, etc..
- ***Window***: provides windows to see process groups, breakpoints, etc.
- ***Help***: allows one to view and print the DDT User Guide and displays a FAQ.
- ***Current Group***: Allows one to select All, Root, or Workers for stepping through the program.
- ***Play/Continue***: starts program execution from where it was stopped and runs to the next breakpoint or until the program stops.
- ***Pause***: pauses program execution on all MPI processes.
- ***Step Into***: execution proceeds into the function/subroutine if stopped at a function or subroutine; therwise, execution continues to the next statement.
- ***Step Over***: execution continues to the next statement and not into a function/subroutine.
- ***Step Out***: allows one to step out from a function/subroutine.
- ***Run to line:***
- ***Down stack frame:***
- ***Up stack frame:***
- ***Bottom stack frame:***
- ***Align stack frame with current:***
- ***Send a signal:***
- ***Checkpoint:*** see “checkpointing” below.
- ***Restore checkpoint:***
- ***All:*** displays all MPI processes that are to be executed under the control of DDT. One can select a specific process to examine variable values.
- ***Focus on current:*** cane select Group, Process or Thread
- ***Group – Create Group:*** Select group and then Create Group to create group for stepping.
- ***Bottom toolbar:*** Input/Output, Breakpoints, Stacks, Tracepoints, Tracepoint Output, Logbook, and Evaluate

When stepping through the program one can step using the groups All. To step using a group, first create the group. To create a group, click on “Create Group” as mentioned above.

Values of local variables are displayed in the right center window for the MPI process selected for the place where execution has stopped. Multi-dimensional array values can be viewed by selecting Multi-Dimensional Array Viewer listed under View.

Breakpoints. A breakpoint set on a line in a program is a point where the program will run to and then pause to allow one to view the values of variables at that point in the program. Breakpoints are easy to set and unset in a program by clicking to the left of the line number to set and then clicking again to remove. Breakpoints may be selected for the MPI processes listed in the All, Root, or Workers toolbar by first left clicking on one of these in the toolbar.

Checkpointing. A program's state may be saved as a checkpoint. The program may later be restored from the checkpoint and will resume execution from the recorded state. Sometimes you are not sure what information you need to diagnose a bug until it is too late to get it. For example, a program may crash because a variable has a particular unexpected value. You want to know where the variable was set to that value but it is too late to set a watch on it. However if you have an earlier checkpoint of the program you can restore the checkpoint, set the watch, and then let it fail again. Checkpoints in DDT are stored in memory. They are valid for the life-time of a session but are lost when the session is ended. To set a checkpoint, click the Checkpoint button on the tool bar. Click the Restore Checkpoint button to restore the program state to the set checkpoint. See the User Guide for additional information.

Watchpoints. One can also set a watchpoint on a variable or expression that causes DDT to pause every time it changes or is accessed. One can set a watchpoint for any group of processes. See the User Guide for instructions on how to set watchpoints.

Tracepoints. Whenever a process or thread reaches a tracepoint it will print the following to the I/O View without stopping program execution:

- the file and line number
- varable values
- expression values

The DDT Window

For the DDT window, there is the **top panel** with File, View, Control Tools, Window and Help. Click on each to see what is available under each. The **second panel** from the top contains buttons for easy use of DDT. Hold the cursor over each to find out what each one does. The **third panel** allows one to focus on a specified Group, Process or Thread. The **fourth panel** allows one to select a current group or an individual process or thread.

The middle window is the **source code viewer** window where it shows where execution is. The left window shows one where DDT is in a hierarchy of the main program and subroutines/functions. The boxes with a "+" inside immediately to the right of the line numbers allow one to view code portions. This is especially useful for programs with many lines.

The right window is the **variable window** and shows values of all variables in the current line, all local variables and the current stack. Click on arrays to view array elements.

Part B: Using MAP

When debugging a program, one usually debugs using a small problem size since program execution is short. However, when profiling a program, it is important to use a problem size that you will want to use for the optimized program since time spent in different portions of the program will normally change depending on the problem size.

The following colors are used for MAP:

- **green** = time spent in CPU activity and
- **blue** = time spent in communication.
- **grey** = idle OpenMP threads.

MAP can profile a portion of a program. See section 18.3.6, page 137-138 of the Allinea Forge 5.0.1 User Guide for details.

One can use MAP to profile serial program by simply compiling the program with the -g option and issuing “map ./a.out”.

Metric View (top panel)

- The x-axis is the time from beginning to the end of program execution showing activity of the metrics listed on the left.
- Click on the Metrics button and select what metrics you would like displayed.
- One can highlight by right clicking and dragging to the left a portion of the time to get a profiling for only that portion of time.
- Be sure to notice the information displayed both at the top and bottom of the Metric view panel.

Source Code Panel (middle panel)

- Shows profiling data for each statement, i.e. the sum of the times for all processors.
- The color blue represents time spent in communication and green in computation.

Input/Output View (bottom panel). Lists input and out put activity.

Project Files View (bottom panel). Lists files profiled.

Main Thread Stacks (bottom panel)

- Each line refers to a specific line of source code (look at the right side) and gives the amount of time spent in the listed function.
- Notice the times reported are only for the specified function/subroutine on the specified line.
- Clicking on the line in this window will cause the line of code to be “highlighted” in the source code panel.
- If you hover over a line, more information will be displayed.

Functions View (bottom panel). Lists time spent in the different functions in the profiled program.