Debugging your code with DDT

Sergey Mashchenko, SHARCNET
syam@sharcnet.ca

Demo code: https://git.sharcnet.ca/syam/Debugging_exercises

Outline

- Overview
- DDT on Graham
- If you need help with DDT
- Setup
- Code examples
- Live demo

Overview

- ARM (originally Allinea) product DDT is a powerful serial / parallel / hybrid code debugger, with a graphical interface.
- Parallelism supported:
  - Serial codes
  - MPI codes
  - Multi-threaded (OpenMP etc) codes
  - GPU codes
  - Hybrid codes (any combinations of the above)
- Programming languages supported:
  - C, C++, Fortran
  - Limited support for Python (CPython 2.7)
  - Parallel languages/models including MPI, UPC, and Fortran 2008 Co-arrays
  - GPU languages such as HMPP, OpenMP Accelerators, CUDA and CUDA Fortran

DDT on Graham

- DDT is installed on our cluster Graham
  - We have a large - 512 cpu-cores - CPU license, and a smaller - 8 GPUs - GPU license.
  - The CPU license can be used for any codes (serial, parallel, hybrid) which do not use GPUs. To use it, simply run the following command on graham:

    ```
    $ module load ddt-cpu
    ```

  - The GPU license (can be used for any debugging involving a GPU) is availed by executing the following command:

    ```
    $ module load ddt-gpu
    ```

  - Make sure you are using the new (StdEnv/2020) environment module.
  - Niagara cluster has a smaller (CPU only) license for ddt. Their instructions may differ.

If you need help with DDT

- Detailed DDT user guide
  - https://developer.arm.com/documentation/101136/latest/ (google for “DDT ARM guide”)
  - On Graham, the PDF user guide can be found inside `$EBROOTALLINEA/doc` directory (after loading the DDT module).
- Documentation on our Docs wiki:
  - https://docs.computecanada.ca/wiki/ARM_software
  - https://docs.computecanada.ca/wiki/Parallel_Debugging_with_DDT
- We have a few DDT webinars on our youtube channel:
  http://youtube.sharcnet.ca
- Send an email to support@computecanada.ca (mention my name)

Setup

- What makes using DDT slightly more difficult than most of other Compute Canada packages is that
  - It is interactive, and
  - It uses GUI (Graphical User Interface).

- I will describe a few setups for using DDT on Graham.

X11 forwarding (1)

- X11 forwarding is the easiest way to run DDT, so if it works for you, just use it
- Instructions depend on your device
  - For **Windows**, the easiest approach is to install the free app MobaXterm.
    - You get SSH terminal, SCP secure file copying, and X11 forwarding - enabled by default
  - For **Mac**, you need to install the free app XQuartz which enables the X11 forwarding functionality.
  - For **Linux**, everything you need is already included.
- Once you installed the required soft, connect to Graham using this command:

  $ ssh **-Y** user_name@graham.computecanada.ca

- Test your X11 setup by executing the xterm command - it should open in a separate window.

X11 forwarding (2)

- For a quick test (say, a few minutes; CPU codes only), you can now load the ddt-cpu module, and run the ddt command (more about that later), right on the login node.
- For more serious debugging work, first allocate interactive compute node(s) using the salloc command, e.g.

```
$ salloc  --x11  -t 0-01:00  -N 1  -c 4  --mem=32G   -A def-user
```

  - Once inside a compute node, test your X11 connection by executing xterm command.
  - If it fails ("X11 connection rejected because of wrong authentication"), one likely reason is that your home directory has too lax permissions. Fix them with

```
$ chmod og-rwX /home/$USER
```

VNC approach

- If you find the X11 approach results in a too laggy experience, the better (but also more involved) approach is to use a VNC connection.
- The first step is to install TigerVNC Viewer (not Server!) on your device.
  - Windows, Mac, and Linux binaries available.
- Next, launch the app, and enter the following remote server address: gra-vdi.computecanada.ca
- Once connected, you will get a Linux desktop window.
  - Enter you Compute Canada credentials there.
  - Open a text terminal, and load our standard environment via `module load CcEnv StdEnv` command.
  - Now you can load the `ddt-cpu` module, and run the `ddt` command
  - You can only debug CPU codes on `gra-vdi`. The node is shared, so can be rather slow.
  - It is also possible to use VNC to connect to a compute node; check these instructions: https://docs.computecanada.ca/wiki/VNC

Preparing code for debugging

- Any CPU-based code: add "-g -Ox" during compiling and linking.
  - Use the appropriate "x" - normally it should be 0 (zero), but if the bug is optimization level specific, replace "x" with the appropriate optimization level.

- CUDA code, or a hybrid code with CUDA: add both "-g -Ox" (for the CPU part) and "-G" (for the GPU part) during compiling and linking.

- To use ddt with your code, load the appropriate module (ddt-cpu or ddt-gpu), and then execute (same for serial, MPI, OpenMP, CUDA):

```
$ ddt  /path/to/your/code  [optional code arguments]
```

Code examples

- I have some simple debugging code examples on Graham, inside /home/syam/Debugging_exercises folder
  - It will not be accessible during the webinar, as I have to restrict my home permissions, to make X11 forwarding work.
  - You can also get it from our Gitlab: https://git.sharcnet.ca/syam/Debugging_exercises
- The following code categories are present:
  - Serial
  - OpenMP
  - MPI
  - CUDA
    - Also has one hybrid (CUDA + MPI) example

Live demo

Questions?

Email to syam@sharcnet.ca