Arm Forge and Arm Performance Reports Getting Started Guide

Development Solutions Group

HPC

Document number:		Version:	19.0
Date of Issue:			
Author:	F. Lebeau & P. Maxwell		
Authorised by:	P. Wohlschlegel		

Abstract

Writing and deploying efficient software is a demanding challenge. NSCC chose Arm Forge and Arm Performance Reports to help the scientific community overcome these challenges. This document details the steps involved in installing, preparing an environment and running Arm Forge and Arm Performance Reports on the NSCC High Performance Computer (HPC).

Contents

1 Introduction

Arm Forge is a graphical development toolkit that includes debugging and profiling tools. NSCC's current licenses support the debugging and profiling of programs using up to 512 processes, 64 accelerators and unlimited threads. This enables the debugging and profiling of several hundreds of cores, using MPI and OpenMP simultaneously.

Arm Performance Reports is the most effective way to characterize and understand HPC application performance. It produces single-page HTML reports that elegantly describe the behavior of applications. Arm Performance Reports inspects how much time is spent computing, communicating and performing I/O, and how efficiently these actions are completed. NSCC's current license allows you to generate reports on applications running on up to 8,192 processes, using an unlimited number of threads.

Arm Forge and Arm Performance Reports use floating licenses which are suitable for use on a laptop, workstation or on the NSCC High Performance Computer (HPC).

The Arm websites (www.arm.com and developer.arm.com) and the respective tool user guides are good resources to learn more about the advanced features of the products. The user guides (Arm Forge and Arm Performance Libraries) are available online and within the respective installation directory of your tool:

```
/app/arm/forge-X.Y/doc/userguide-forge.pdf
/app/arm/reports-X.Y/doc/userguide-reports.pdf
```

Note: `X.Y` corresponds to the version number of the tool, for example 19.0.

2 Installation

2.1 Arm Forge and Arm Performance Reports

Arm Forge and Arm Performance Reports can be downloaded from developer.arm.com and installed using either the graphical or text-based installers.

For detailed descriptions of the installation steps see the respective user guides.

2.2 Arm Forge Remote Client

The Arm Forge Remote Client allows you to debug and profile jobs on a remote server, while running the GUI on your local machine.

Any regular Linux build of the Arm tools can be used as a server or as a remote client. There are also client-only builds available for OS X and Windows.

To setup Arm Forge Remote Client:

1. Download

The first step is to download a copy of the Arm tools for your laptop or desktop. Versions for Windows, Mac OS/X and Linux are all available for download from the Arm website.

2. Install

Depending on the operating system you're using, the instructions here differ slightly:

- Linux: Extract the tarball, run installer for a GUI installer, or **textinstall.sh** for a text install.
- Mac: Open the DMG file with Finder. Then, launch the installer.
- Windows: Execute the .exe installer downloaded from the website.

Note: No license is required for the remote client. When using the remote client, the license on the remote machine or cluster will be used.

3. Connect using the remote client

To connect using the remote client, follow the steps:

1. Run the installed remote client by double-clicking on the **Arm Remote Client** icon.
2. Click **Remote Launch: Off** from the welcome screen. Next, choose **Configure**, and select **Add**.
3. Enter the details of your remote host:
   I. Username and hostname: `<username>@aspire.nscc.sg` (please refer to http://help.nscc.sg/faqs/).
   II. The full path of the arm/tools installation directory ***on the remote machine***: `/app/arm/forge-X.Y/`
   Hint: X.Y corresponds to the version number, for example 19.0.
   III. (Optional) The **`script`** parameter is optional and can usually be left blank. You can put the path of a script that you would like to **source** (execute and load the environment variables from) on the remote machine. If used, the path to this must be a path on the remote machine.
   IV. To check the settings, click **Test Remote Launch**. The Arm tools connect to the remote host using **ssh** and may prompt you for your password or, on some systems, a PASSCODE. Log in here as you normally would.
   V. To save the settings, click **Ok**. Next time you will be able to pick `<username>@nscc03` from the drop-down box on the welcome dialog directly – there will be no need to re-enter the remote host details.

The connection can now be selected from the **Remote Launch** drop down menu. Once connected there may be a slight delay. After the delay, the **Run**, **Attach** and other options will be enabled. The interface now behaves as if you were running it directly on the remote system, this means:

- All the file dialogs browse the remote filesystem.
- Settings are loaded and saved to the remote system.
- Programs will be run and debugged or profiled on the remote system.

You can now browse through existing *.map files on the remote system or use Arm DDT with Reverse Connect to debug your applications. The Arm DDT GUI will appear as if you were launching the GUI from the login node with X forwarding.

3 Preparing your environment

3.1 X forwarding

If you are not using the Arm Remote Client and intend on using interactive debugging with DDT, you must enable X forwarding in your alternative client.

Some remote clients list X forwarding as a configurable setting that can be enabled or disabled by default. Check the settings of your client and ensure it is enabled.

If there is no option in the client configuration settings, add `-X` or `-Y` to your ssh login command line.
Note: The `-Y` option often works better for Mac OS/X.

Please refer to http://help.nscc.sg/faqs/ ('How do I access NSCC') to connect to the system.

```
#Connect to NSCC system as described in the FAQ, e.g. :
$ ssh -X aspire.nscc.sg
```

3.2 Loading the Arm Forge environment

To configure your environment for Arm Forge, load the corresponding module:

```
$ module load arm/forge/X.Y
```

Note: `X.Y` corresponds to the version number of the tool, for example 19.0.

3.3 Loading the Arm Performance Reports environment

To configure your environment for Arm Performance Reports, load the corresponding module:

```
$ module load arm/reports/X.Y
```

Note: `X.Y` corresponds to the version number of the tool, for example 19.0.

There are no specific compilation requirements for Arm Performance Reports. Existing binaries can be used without recompilation.

4 Arm Forge

4.1 Prerequisites – Compiling code to run with Arm Forge

Arm Forge needs debugging symbols to show the exact lines of source code that cause problems. It also needs them to identify variables and their values or the lines corresponding to performance bottlenecks.

To enable debugging symbols, compile your program and include the `-g` flag:

```
$ mpicc -g hello.c -o hello
```

4.1.1 Arm DDT Optimization level

When debugging using Arm DDT, it is recommended that optimizations are turned off. To compile without optimizations, include the `-O0` flag on the compile line:

```
$ mpicc -g -O0 hello.c -o hello
```

Note: Turning off optimization flags is optional. Optimization flags can re-order the code in unexpected ways and make application debugging less intuitive. If a bug only occurs within an optimized binary, keep the relevant optimizations.

4.1.2 Arm MAP Optimization level

For profiling with Arm MAP, we recommend that you keep your existing optimization options enabled:

```
$ mpicc -g –O3 hello.c -o hello
```

4.2 Interactive graphical debugging with Arm DDT – Reverse Connect

Interactive debugging lets you test and explore theories in real time.

To debug or profile jobs interactively, two options are available:

- The Arm Forge Remote Client (recommended).
- X forwarding.

For details on starting a session with Arm Forge Remote Client, see Arm Forge Remote Client. For details on how to start a session with X forwarding, see Preparing your environment.

To interactively debug using Arm DDT:

1. Log in using the Arm Forge Remote Client or using an alternative client with X forwarding enabled, and configure your environment for Arm Forge.
2. Launch the GUI of the Arm DDT debugger on the login node,:

```
$ ddt &
```

The GUI waits to receive an incoming connection.

3. Either, compile your application (including the `-g` flag), or locate an appropriately pre-compiled binary.
4. Edit your submission script to run the Reverse Connect Arm DDT job. For example, `your_script.sub`, should be modified to look like the following (note the `ddt--connect` prefix to the `mpirun` command):

```
#!/bin/bash
#PBS -N test_run
#PBS -q normal
#PBS -l select=1:ncpus=24:ompthreads=1:mpiprocs=24
#PBS -l walltime=00:10:00
#PBS -j oe

module load arm/forge/X.Y

#Note : you can use different compilers or MPI libraries here
module load intelmpi intelcc

cd $PBS_O_WORKDIR

#mpirun ./your_app.exe
ddt --connect mpirun ./your_app.exe
```

Notes:

- `X.Y` corresponds to the version number of the tool, for example 19.0.
- To debug sequential applications, use:
  `ddt --connect ./your_app.exe`

5. Submit your script, using:

```
$ qsub ./your_script.sub
```

6. (Required when using X forwarding) Once the job is submitted and running, the Arm Forge GUI displays a window asking you whether you accept the incoming connection or not, click **Yes** to accept the incoming connection.
7. Use the **Run** dialog to configure advanced features such as memory debugging. Click **Run** when ready.

4.3 Non-interactive debugging with Arm DDT – Offline mode

Running interactive debugging sessions may not always be convenient, for example if your application:

- Takes hours to run.

- Cannot go through the queue immediately.
- Does not crash and does not contain any visible bug.

In these cases, running automated non-interactive debugging sessions and generating bug reports is a better route to solving your problem. In Arm DDT, this feature is called 'Offline Mode'. Offline mode runs your job as usual, but produces a detailed HTML debugging report upon program termination.

To debug using Arm DDT in offline mode, follow these steps:

1. Log in to a session and configure your environment for Arm Forge.
2. Either, compile your application (including the `-g` flag), or locate an appropriately pre-compiled binary.
3. Edit your submission script to run the offline Arm DDT job. For example, `your_script.sub` should be modified to look like the following (note the `ddt –offline --output` prefix to the `mpirun` command):

```
#!/bin/bash
#PBS -N test_run
#PBS -q normal
#PBS -l select=1:ncpus=24:ompthreads=1:mpiprocs=24
#PBS -l walltime=00:10:00
#PBS -j oe

module load arm/forge/X.Y

#Note : you can use different compilers or MPI libraries here
module load intelmpi intelcc

cd $PBS_O_WORKDIR

#mpirun ./your_app.exe
ddt --offline --output=your_report.html mpirun ./your_app.exe
```

Notes:

- `X.Y` corresponds to the version number of the tool, for example 19.0.
- To debug sequential applications, use:
  `ddt --offline --output=your_report.html ./your_app.exe`

4. Submit your script, using:

```
$ qsub ./your_script.sub
```

4.3.1 Configuring your offline job

Offline jobs can be pre-configured using options such as: `--break-at=function` or `--break-at=file:line`, `--trace-changes` and `--memdebug` to add breakpoints, tracepoints or enable memory debugging, respectively.

These configuration options should be added, prior to job submission, to one of two locations:

- The command line interface.
- An Arm DDT session file (using `--ddtsession=sessionfile`) (generate file within the GUI).

See `ddt --help` for more information.

4.4 Non-interactive profiling with Arm MAP

The simplest way to profile your application with Arm MAP is to use non-interactive profiling. Non-interactive profiling runs your job as usual but produces a detailed `.map` file, which can then be opened in the Arm MAP GUI and explored.

To profile using Arm MAP in offline mode:

1. Log in to a session and configure your environment for Arm Forge.
2. Either, compile your application (including the **`-g`** flag), or locate an appropriately pre-compiled binary.
3. Edit your submission script to run the offline Arm MAP job. For example, `your_script.sub` should be modified to look like the following (note the `map --profile` prefix to the `mpirun` command):

```
#!/bin/bash
#PBS -N test_run
#PBS -q normal
#PBS -l select=1:ncpus=24:ompthreads=1 :mpiprocs=24
#PBS -l walltime=00:10:00
#PBS -j oe

module load arm/forge/X.Y

#Note : you can use different compilers or MPI libraries here
module load intelmpi intelcc

cd $PBS_O_WORKDIR

#mpirun./your_app.exe
map --profile mpirun ./your_app.exe
```

Notes:

- `X.Y` corresponds to the version number of the tool, for example 19.0.
- To profile a sequential application, use:
  `map --profile ./your_app.exe`
- In Arm MAP 19.x+ versions, you can profile python applications (sequential and parallel using mpi4py). To profile python applications, use:
  `map --profile python ./your_app.exe`

```
map --profile mpirun python ./your_app.exe
```

4. Submit your script, using:

```
$ qsub ./your_script.sub
```

   At the end of the run, the `.map` file is generated.

5. To open the `.map` file, use:

```
$ map ./your_app_24p_1n_YYYY-MM-DD_HH-MM.map
```

   Alternatively, copy the file to another machine and use the same command to open it.

   Note: The `.map` file can be opened anywhere, no compute node allocation is needed. However, you must tell the GUI where to look for the program source files.

5 Arm Performance Reports – Generating Application Reports

5.1 Prerequisites

There are no specific compilation requirements for Arm Performance Reports. Existing binaries can be used without recompilation.

5.2 Analyzing with Arm Performance Reports

To generate a report and start analyzing your application's behavior using Arm Performance Reports:

1. Log in to a session and configure your environment for Arm Performance Reports.
2. Either, compile your application (including the `-g` flag), or locate an appropriately pre-compiled binary.
3. Edit your submission script to run the Arm Performance Reports job. For example, `your_script.sub` should be modified to look like the following (note the `perf-report` prefix to the `mpirun` command):

```
#!/bin/bash
 #PBS -N test_run
 #PBS -q normal
 #PBS -l select=1:ncpus=24:ompthreads=1:mpiprocs=24
 #PBS -l walltime=00:10:00
 #PBS -j oe

 module load arm/reports/X.Y

 #Note : you can use different compilers or MPI libraries here
 module load intelmpi intelcc
```

```
cd $PBS_O_WORKDIR

#mpirun ./your_app.exe
perf-report mpirun ./your_app.exe
```

Notes:

- `X.Y` corresponds to the version number of the tool, for example 19.0.
- To profile a sequential application, use:
  `map --profile ./your_app.exe`

4. Submit your script, using:

```
$ qsub ./your_script.sub
```

   At the end of the run, a `.html` and a `.txt` file are generated.

5. To view these results, open the file with an appropriate viewer, for example:

```
$ cat ./your_app_24p_1n_YYYY-MM-DD_HH-MM.txt
$ firefox ./your_app_24p_1n_YYYY-MM-DD_HH-MM.html
```

   Note: To use a different browser, replace the firefox command with the command to launchanother installed browser.

6 Arm tools options

All support options can be displayed by using the appropriate `--help` option:

```
$ ddt --help
$ map --help
$ perf-report --help
```

The most common options are:

`--ddtsession=<session.ddt>`
    Provides **Arm DDT** with a previously saved session file.

`--start`
    Skips the "Run" window preliminary steps and starts **Arm Forge** straight away.

`--start-after=<TIME>`
    Arm MAP starts sampling <TIME> seconds after program starts.

`--stop-after=<TIME>`

  Arm MAP stops sampling <TIME> seconds after program starts.

`--version`

  Displays the version number of the Arm tool.

7 Troubleshooting

If you are having trouble launching Arm Forge, check the following potential causes:

1. Ensure you compiled the code with the **-g** flag.
2. If possible, check if the problem persists on a more recent version of the tool.
3. Ensure you have the most recent versions of the `system.config` configuration file:

```
$ rm -rf $HOME/.allinea
```

If none of these points resolve your problem, please contact the support team. To help reduce the number of support iterations, please include in your email:

- A log file that can be generated by starting the tool and using the `--debug --log=output.log` options with the appropriate tool:

```
$ ddt --debug --log=output.log
$ map --debug --log=output.log
$ perf-report --debug --log=output.log
```

- Information about the environment (version of Arm Forge or Arm Performance Reports, compiler used, MPI library, and any other relevant tools included in the loaded environment modules).

You can contact the HPC support team at allinea-support@arm.com.

Commented [PM1]: Is this still correct?

More information about our tools can be found on our developer support website.