Allinea Forge (DDT,MAP)

Allinea Forge consist of two tools - debugger DDT and profiler MAP.

Allinea DDT, is a commercial debugger primarily for debugging parallel MPI or OpenMP programs. It also has a support for GPU (CUDA) and Intel Xeon Phi accelerators. DDT provides all the standard debugging features (stack trace, breakpoints, watches, view variables, threads etc.) for every thread running as part of your program, or for every process - even if these processes are distributed across a cluster using an MPI implementation.

Allinea MAP is a profiler for C/C++/Fortran HPC codes. It is designed for profiling parallel code, which uses pthreads, OpenMP or MPI.

License and Limitations for the clusters Users

On the clusters users can debug OpenMP or MPI code that runs up to 64 parallel processes. In case of debugging GPU or Xeon Phi accelerated codes the limit is 8 accelerators. These limitation means that:

- 1 user can debug up 64 processes, or
- 32 users can debug 2 processes, etc.

In case of debugging on accelerators:

- 1 user can debug on up to 8 accelerators, or
- 8 users can debug on single accelerator.

Compiling Code to run with Forge

Modules

Load all necessary modules to compile the code. For example:

```
$ module load intel
$ module load impi   ... or ... module load OpenMPI
```

Load the Allinea DDT module:

```
$ module load Forge
```

Compile the code:

‘ $ mpicc -g -O0 -o test_debug test.c

$ mpif90 -g -O0 -o test_debug test.f ‘

Compiler flags

Before debugging, you need to compile your code with theses flags:

-g** : Generates extra debugging information usable by GDB. -g3** includes even more debugging information. This option is available for GNU and INTEL C/C++ and Fortran compilers.

-O0** : Suppress all optimizations.**

Direct starting a Job with Forge

Be sure to log in with an X window forwarding enabled. This could mean using the -X in the ssh:

```
$ ssh -X username@clustername.it4i.cz
```

Other options is to access login node using VNC. Please see the detailed information on how to use graphic user interface on the clusters .

From the login node an interactive session **with X windows forwarding** (-X option) can be started by following command:

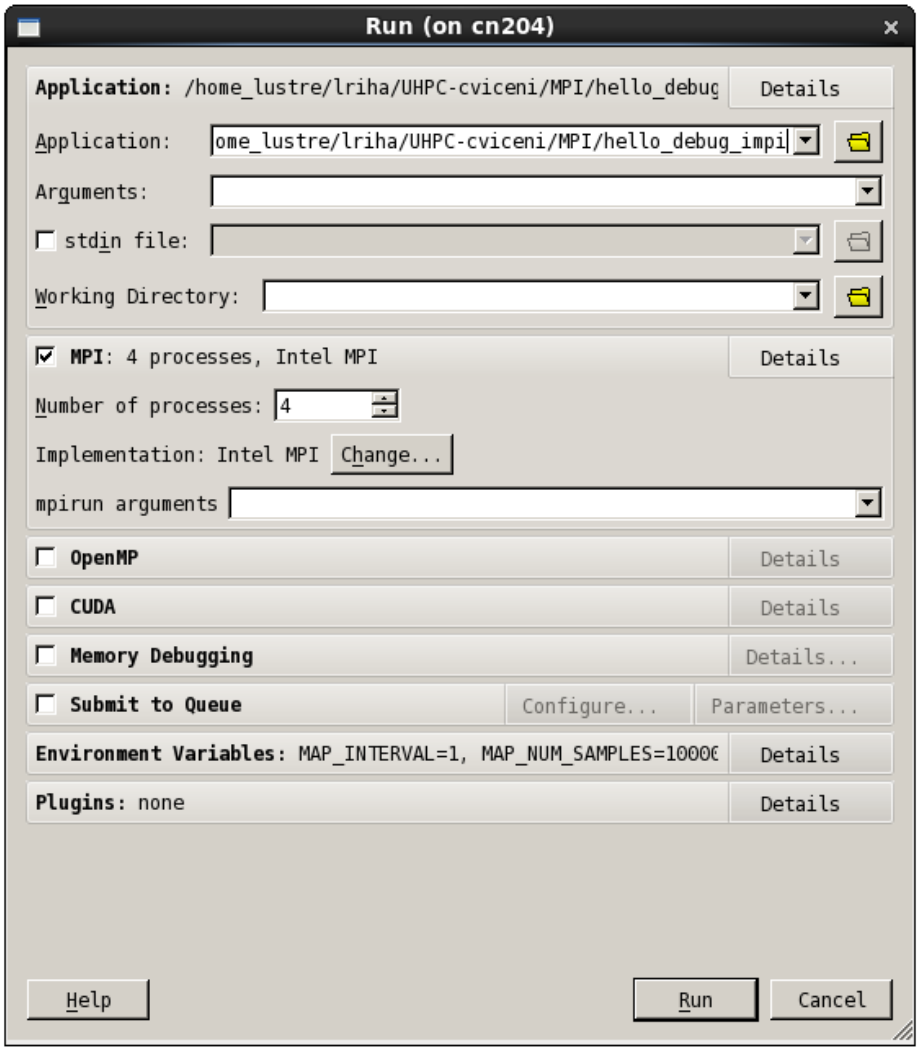
```
$ qsub -I -X -A NONE-0-0 -q qexp -lselect=1:ncpus=24:mpiprocs=24,walltime=01:00:00
```

Then launch the debugger with the ddt command followed by the name of the executable to debug:

```
$ ddt test_debug
```

Forge now has common GUI for both DDT and MAP. In interactive mode, you can launch Forge using forge, ddt or map, the latter two will just launch forge and swith to the respective tab in the common GUI.

A submission window that appears have a prefilled path to the executable to debug. You can select the number of MPI processors and/or OpenMP threads on which to run and press run. Command line arguments to a program can be entered to the "Arguments" box.



To start the debugging directly without the submission window, user can specify the debugging and execution parameters from the command line. For example the number of MPI processes is set by option "-np 4". Skipping the dialog is done by "-start" option. To see the list of the "ddt" command line parameters, run "ddt –help".

```
ddt -start -np 4 ./hello_debug_impi
```

All of the above text also applies for MAP, just replace ddt command with map.

Reverse connect

Forge now provides a new convenient mode of operation, called Reverse connect. Instead of launching a job from the GUI, the process is reserved - DDT/MAP is launched as a server in the job which then connects to a running instance of your GUI.

To use Reverse connect, use a jobscript that you would normally use to launch your application, just prepend ddt/map –connect to your application:

```
map --connect mpirun -np 24 ./mpi-test
ddt --connect mpirun -np 24 ./mpi-test
```

Launch Forge GUI on login node and submit the job using qsub. When the job starts running, Forge will ask you to accept the connection:

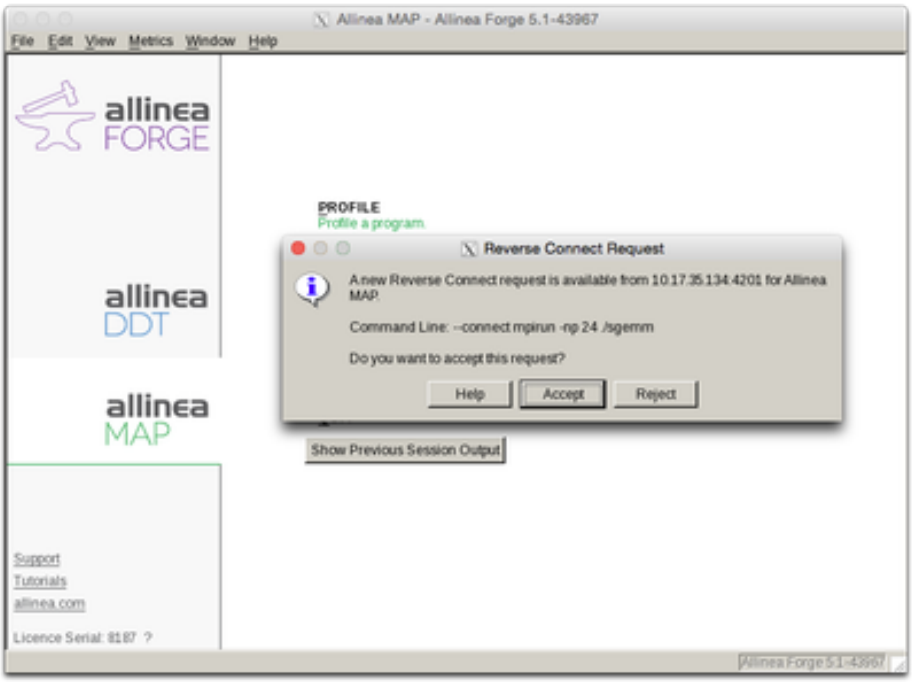


Figure 1:

After accepting the request, you can start remote profiling/debugging.

Xeon Phi

Forge allows debugging and profiling of both offload and native mode Xeon Phi programs.

Offload mode

It is recommended to set the following environment values on the offload host:

```
export MYO_WATCHDOG_MONITOR=-1      # To make sure the host process isn't killed when we enter a
export AMPLXE_COI_DEBUG_SUPPORT=true  # To make sure that debugging symbols are accessible on th
unset OFFLOAD_MAIN               # To make sure allinea DDT can attach to offloaded codes
```

Then use one of the above mentioned methods to launch Forge. (Reverse connect also works.)

Native mode

Native mode programs can be profiled/debugged using the remote launch feature. First, you need to create a script that will setup the environment on the Phi card. An example:

```
#!/bin/bash
# adjust PATH and LD_LIBRARY_PATH according to the toolchain/libraries your app is using.
export PATH=/apps/all/impi/5.0.3.048-iccifort-2015.3.187/mic/bin:$PATH
export LD_LIBRARY_PATH=/apps/all/impi/5.0.3.048-iccifort-2015.3.187/mic/lib:/apps/all/ifort/
export MIC_OMP_NUM_THREADS=60
export MYO_WATCHDOG_MONITOR=-1
export AMPLXE_COI_DEBUG_SUPPORT=true
unset OFFLOAD_MAIN
export I_MPI_MIC=1
```

Save the script in eg. ~/remote-mic.sh. Now, start an interactive graphical session on a node with accelerator:

```
$ qsub -IX -q qexp -l select=1:ncpus=24:accelerator=True
```

Launch Forge :

```
$ module load Forge
$ forge&
```

Now click on the remote launch drop-down list, select “Configure…” and Add a new remote connection with the following parameters:

Connection name: mic0

Hostname: mic0

Remote Installation Directory: /apps/all/Forge/5.1-43967/

Remote script: ~/remote-mic.sh

You can click Test Remote Launch to verify the configuration. After you save the remote launch configuration and select it in the dropdown list, you can use the Run button in the main windows to remotely launch your program on mic0.

Documentation

Users can find original User Guide after loading the Forge module:

`$EBROOTFORGE/doc/userguide-forge.pdf`

[1] Discipline, Magic, Inspiration and Science: Best Practice Debugging with Allinea DDT, Workshop conducted at LLNL by Allinea on May 10, 2013, link