Building and Running OpenMPI

Edit New Page

Jump to bottom

Howard Pritchard edited this page on Aug 23, 2016 · 17 revisions

This page describes how to build and run Open MPI to test the libfabric GNI provider. It is assumed the user is building Open MPI on a Cray XC system like tiger or edision/cori, and that you have built and installed a copy of libfabric.

Building and Installing Open MPI

First, if you don't already have a clone of Open MPI

```
% git clone https://github.com/open-mpi/ompi.git
```

This is the OMPI development repository. If you want a more stable version, use https://github.com/open-mpi/ompi-release.git. Don't use the default branch (v1.10). Checkout the latest non-dev branch (e.g., v2.x). Alternatively, the official tarball release can be obtained at https://www.open-mpi.org/software/ompi/v2.0. With the tarball release, one can skip the `autogen.pl` step in the build stage.

Next, configure and build/install Open MPI. Note you will not want to try to build the Open MPI using the Cray compiler.

```
% cd ompi
% ./autogen.pl
% module load PrgEnv-gnu
% ./configure --prefix=ompi_install_dir --with-
libfabric=your_libfabric_install_dir --disable-dlopen
% make -j 8 install
```

Note if you are wanting to run MPI multi-threaded tests which use MPI_THREAD_MULTIPLE, you will need to configure Open MPI as follows

```
% ./configure --prefix=ompi_install_dir --with-
libfabric=your_libfabric_install_dir --enable-mpi-thread-multiple --disable-
dlopen
```

It is not necessary to specify `--disable-dlopen`, but for testing libfabric, it facilitates double checking that binaries linked using the Open MPI `mpiccc` script are in fact using the correct libraries.

Running Open MPI with libfabric

First you will need to build an MPI app using Open MPI's compiler wrapper:

```
% export PATH=ompi_install_dir/bin:${PATH}
% mpicc -o my_app my_app.c
```

To make sure you are using Open MPI's ofi mtl set the following:

```
% export OMPI_MCA_pml=cm
```

In addition, it is recommended to disable other portions of Open MPI that may also be using GNI. To do this, set the following environment variable:

```
% export OMPI_MCA_btl=self,vader,tcp
```

On Tiger and NERSC cori/edison, the application can be launched using srun:

```
% srun -n 2 -N 2 ./my_app
```

On systems using aprun:

```
% aprun -n 2 -N 1 /my_app
```

If you'd like to double check against the sockets provider, do the following

```
% export OMPI_MCA_mtl_ofi_provider_exclude=gni
% srun -n 2 -N 2 ./my_app
```

This will force the OFI MTL to use the sockets provider.

Special note for those packing MPI processes on to KNL nodes. You will want to disable the uGNI BTL when running in this mode. One way to do this at runtime is to set the following environment variable:

```
% export OMPI_MCA_btl=self,vader,tcp
```

Building and Testing OSU MPI benchmarks

OSU provides a relatively simple set of MPI benchmark tests which are useful for testing the GNI libfabric provider.

```
% wget http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-
benchmarks-5.0.tar.gz
% tar -zxvf osu-micro-benchmarks-5.0.tar.gz
% cd osu-micro-benchmarks-5.0
% ./configure CC=mpicc
% make
```

In the `mpi/pt2pt` and `mpi/collective` subdirectories there are a number of tests. To test, for example Open MPI send/recv message latency, `osu_latency` can be used

```
% cd mpi/pt2pt
% srun -n 2 -N 2 ./osu_latency
```

You can use the `run_osu` script (https://github.com/ofi-cray/fab-utils/blob/master/scripts/benchmarks/run_osu) to test your libfabric library using the OSU benchmarks in a few configurations (works with aprun or srun). You must pass the script the path to the OSU microbenchmarks directory and a path to your libfabric install.

NOTE: When running the one-sided tests, it is recommended to set the environment variable `OMPI_MCA_osc` to `pt2pt`.