DataMPI 0.6.0 Quick Start

Purpose

The purpose of this document is to help you get a basic DataMPI installation up and running very quickly so that you can get a flavor of the DataMPI library and MPI-D specification. The main tasks include deploying DataMPI, and performing simple tests and examples.

Pre-requisites

Supported Platforms

- GNU/Linux is supported as a development and production platform. DataMPI has been well tested on Linux clusters. It is the recommended platform for getting started. You can choose any mainstream Linux distribution.

Required Software

Compliers and Build Toolchains

Compliers and build toolchains are necessary only for building from source code.

- Java development kit and runtime environment (version >= 1.7.0), preferably from Oracle.
- GNU toolchain, including gcc, make, and so on.
- CMake (version >= 2.8.0).
- Scripting language interpreters, including bash and perl.

MPI Environment

DataMPI depends on a native MPI environment. In the quick start, we use MVAPICH2 as the backend. DataMPI also supports other MPI implementations, such as MPICH2.

We take MVAPICH2 1.8.1 as an example, which can be installed as follows in a typical Ethernet environment.

```
$ cd <mvapich2_source_directory>
$ ./configure --prefix=/home/mpiuser/mvapich2 --with-device=ch3:nemesis \
    CFLAGS=-fPIC --disable-f77 --disable-fc
$ make && make install

$ echo ’export MVAPICH2_HOME=/home/mpiuser/mvapich2’ >> ~/.bashrc
$ echo ’export PATH=$MVAPICH2_HOME/bin:$PATH’ >> ~/.bashrc
$ source ~/.bashrc
```

For detailed installing configuration of MVAPICH2, you can refer to its website (http://mvapich.cse.ohio-state.edu/support/).

Installation

You can use any account on the operating system for installing DataMPI. A non-privileged account is recommended for security reasons. However if you want to install it into a globally accessible location, you should use the `root` account.

Download

The binary release of DataMPI can be downloaded from the official website (http://datampi.org). The source code can be obtained from DataMPI team by email (mpid.contact *at* gmail.com).

For the binary package, we suppose it is unzipped into `/home/mpiuser/datampi-0.6.0-install`, which is referred to as the "install" directory in the following text. You can skip the following steps of configuration and compilation, and try to run the examples directly.

For the source package, we suppose it is unzipped into `/home/mpiuser/datampi-0.6.0-src`, which is referred to as the "source" directory in the following text. You should configure and compile the code before running.

Configure Build Environment

Create a "build" directory for storing the intermediate files. We take `/home/mpiuser/datampi-0.6.0-build` as an example.

```
$ mkdir /home/mpiuser/datampi-0.6.0-build
```

Create an "install" directory as the destination location of DataMPI installation. We take `/home/mpiuser/datampi-0.6.0-install` as an example.

```
$ mkdir /home/mpiuser/datampi-0.6.0-install
```

Enter the build directory, and use the CMake tool to create Makefile scripts.

```
$ cd <build_directory>
$ cmake -D CMAKE_INSTALL_PREFIX=<install_directory> \
    -D MPI_D_BUILD_DOCS=<ON_or_OFF> -D MPI_D_BUILD_TESTS=<ON_or_OFF> \
    -D MPI_D_BUILD_EXAMPLES=<ON_or_OFF> -D MPI_D_BUILD_BENCHMARKS=<ON_or_OFF> \
    <source_directory>
```

- The `-D CMAKE_INSTALL_PREFIX` parameter refers to the install directory. If it is omitted, DataMPI will be installed into an `installed` directory within the source directory.

- The following three parameters, `-D MPI_D_BUILD_DOCS`, `-D MPI_D_BUILD_TESTS`, `-D MPI_D_BUILD_EXAMPLES` and `-D MPI_D_BUILD_BENCHMARKS`, control whether to build documents, tests, examples and benchmarks of DataMPI. It is recommended to set them `ON` for getting starting. The default value is `OFF`.

- The last parameter refers to the source directory. It cannot be omitted.

The final command looks like this.

```
$ cd /home/mpiuser/datampi-0.6.0-build
$ cmake -D CMAKE_INSTALL_PREFIX=/home/mpiuser/datampi-0.6.0-install \
    -D MPI_D_BUILD_DOCS=ON -D MPI_D_BUILD_TESTS=ON -D MPI_D_BUILD_EXAMPLES=ON \
    -D MPI_D_BUILD_BENCHMARKS=ON /home/mpiuser/datampi-0.6.0-src
```

And it will give the result as follows.

```
-- The C compiler identification is GNU 4.4.5
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
...
-- Configuring done
-- Generating done
-- Build files have been written to: /home/mpiuser/datampi-0.6.0-build
```

Compile and Install DataMPI

You can compile and install DataMPI with the following command.

```
$ make install
```

If all goes well, you will see the result as follows.

```
Scanning dependencies of target mpi
[  1%] Building Java object src/Java/CMakeFiles/mpi.dir/mpi/CartParms.class
[  1%] Building Java object src/Java/CMakeFiles/mpi.dir/mpi/Cartcomm.class
...
Scanning dependencies of target html-manual
[100%] Compiling html-manual
Install the project...
...
-- Installing: /home/mpiuser/datampi-0.6.0-install/share/datampi/doc/
    html/md-docs-style.css
-- Installing: /home/mpiuser/datampi-0.6.0-install/share/datampi/doc/
    datampi-0.6.0-quick-start.md
-- Installing: /home/mpiuser/datampi-0.6.0-install/share/datampi/doc/
    mpiJava-spec.pdf
```

Now, DataMPI is installed in the install directory. The source and build directories are no longer being accessed when using DataMPI normally. However, you should keep the build directory if you want to perform the tests, and keep the source directory if you want to study the code of examples.

Performing Tests

DataMPI tests are used to verify the newly complied DataMPI working properly. By performing the tests, you can generally locate specific problems in certain environment.

Run Tests

DataMPI tests are located in the build directory. To run all the tests, you can use the following commands.

```
$ cd /home/mpiuser/datampi-0.6.0-build
$ make test
```

You will see the result as follows if everything goes well.

```
Running tests...
Test project /home/mpiuser/datampi-0.6.0-build
  Start   1: mpid.tests.SortJavaSerializable
1/106 Test   #1: mpid.tests.SortJavaSerializable ............... Passed 1.50 sec
  Start   2: mpid.tests.ReverseSortWritableBytesComparator
2/106 Test   #2: mpid.tests.ReverseSortWritableBytesComparator . Passed 1.41 sec
...
100% tests passed, 0 tests failed out of 106
Total Test time (real) = 104.29 sec
```

Performing Examples

DataMPI examples help you to be familiar with the basic command-line operations of DataMPI, and guide you to write MPI-D programs as well.

Configure DataMPI

Before performing DataMPI examples and writing MPI-D programs, you should make sure that your DataMPI installation is well configured. The configuration files of DataMPI are located in the `conf` subdirectory of the installation directory. For performing the following simple example on a single node, you should only modify a few files (use the real values of your environment to replace the variables in the angle brackets).

`hostfile`:

```
<your_host_name>
```

`mpi-d-env.sh`:

```
# Java home to use. Required.
# export JAVA_HOME=<your_java_home>
```

If you want to perform multi-node tests or examples, you should make the install directory available on all the nodes with the same path (either employing a network file system or copying the directory to all the nodes), and list all the hostnames line by line in `hostfile`. Please refer to the DataMPI user guide for more complicated configuration.

Run Examples

The examples are grouped and located in the `share/datampi/examples` subdirectory of the installation directory. For example, the "common" group includes some examples for the common mode of MPI-D.

The `mpidrun` command in the `bin` subdirectory of install directory is the launcher of MPI-D jobs. You can use this command to run examples. We use the Sort example to show the usage of `mpidrun`.

```
$ cd /home/mpiuser/datampi-0.6.0-install
$ bin/mpidrun -mode COMMON -O 2 -A 2 \
    -jar share/datampi/examples/common/common-example.jar mpid.examples.Sort
```

- The `-mode` parameter indicates the mode of the target program. This example is written in the common mode of MPI-D, so it should be set to `COMMON`.
- The `-O` parameter refers to the number of operator tasks in MPI-D. For the Sort example, you can consider it as the number of mappers in MapReduce.
- The `-A` parameter refers to the number of aggregator tasks in MPI-D. For the Sort example, you can consider it as the number of reducers in MapReduce.
- The `-jar` parameter indicates the jar file where the program locates.
- The last parameter is the entry class name of the program.

If all goes well, you will see the result as follows.

```
The O task 1 of 2 is working...
The O task 0 of 2 is working...
The A task 0 of 2 is working...
...
In the task 0, key is [hello], value is [3]
In the task 0, key is [sort], value is [4]
In the task 0, key is [world], value is [0]
...
In the task 1, key is [templar], value is [1]
In the task 1, key is [templar], value is [8]
In the task 1, key is [valvula], value is [2]
...
13/05/09 21:41:55 INFO common.MPI_D_Runner4Common: Job job_201305092141_0054
    ended, used time 1.03 sec.
```

Congratulations! You have learned the most basic usage of DataMPI.

What’s the Next?

You can read the other documents and examples of DataMPI, in order to build a fully-distributed DataMPI environment, and study MPI-D programming.