DataMPI 0.6.0 User Guide

DataMPI team

http://datampi.org

April, 2014

Contents

1 Overview

DataMPI is an efficient, flexible, and productive communication library, which provides a set of key-value pair based communication interfaces that extends MPI for Big Data. Through utilizing the efficient communication technologies in the High-Performance Computing area, DataMPI can speedup the emerging data intensive computing applications. DataMPI takes a step in bridging the two fields of HPC and Big Data.

We are working on a draft version of our proposed specification to extend MPI for Big Data. We label the specification as **MPI-D**, which is still under revision and will be published later. DataMPI is a Java binding implementation for MPI-D.

DataMPI can support multiple modes for various Big Data Computing applications, including Common, MapReduce, Streaming, and Iteration. The current version implements the functionalities and features of the Common mode, which aims to support the single program, multiple data (SPMD) applications. The remaining modes will be released in the future.

The current implementation of DataMPI is extending mpiJava. We also integrate some features from Hadoop under Apache License 2.0. The current evaluations of DataMPI use MVAPICH2 as the backend. DataMPI also supports other MPI implementations, such as MPICH2.

2 Features

Following features are included in DataMPI 0.6.0:

- High-performance implementation of the Common mode in the MPI-D specification
- Support high-speed and flexible key-value pair communication
- Provide object-oriented Java interfaces
- Easy to program data-intensive computing applications
- Support hybrid programming with MPI and MPI-D
- MPI-1.1 compliance and support the feature of dynamic process management in MPI-2
- Support performance tuning with advanced configurations
- Easy to build and config
- Tested with
  - Multiple backend MPI implementations (MVAPICH2, MPICH2, etc.)
  - Various multi-core platforms
  - Different file systems (ext3/4, HDFS, etc.)

3 Installation Instructions

3.1 Pre-requisites

3.1.1 Supported Platforms

GNU/Linux is supported as the development and production platform for DataMPI. DataMPI has been well tested on Linux clusters. Any mainstream Linux distribution can be chosen. A non-privileged account is recommended for security reasons during installation. However, if DataMPI is installed into a globally

accessible location, the `root` account is necessary.

3.1.2 Required Software

Compliers and build toolchains are necessary only for building DataMPI 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.

3.1.3 MPI Environment

DataMPI depends on a native MPI environment. We use MVAPICH2 1.8.1 as an example. MVAPICH2 1.8.1 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, please refer to its website (http://mvapich.cse.ohio-state.edu/support/).

DataMPI also supports other MPI implementations, such as MPICH2.

3.2 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, the following steps of configuration, compilation and test can be skipped and the examples can be run directly (see Section 3.6).

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, the steps below must be executed before running.

3.3 Configure Build Environment

A "build" directory should be created for storing the intermediate files. Here `/home/mpiuser/datampi-0.6.0-build` is taken as an example.

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

An "install" directory should be created as the destination location of DataMPI installation. Here `/home/mpiuser/datampi-0.6.0-install` is taken as an example.

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

CMake tool is used to created 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 four 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 start. The default value is `OFF`.
- The last parameter refers to the source directory. It **cannot** be omitted.

An example 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
```

If all goes well, the result will be printed like this.

```
-- 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
```

3.4 Compile and Install DataMPI

DataMPI can be compiled and installed with the following command.

```
$ make install
```

If all goes well, the result will be printed like this.

```
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, the "build" directory should be kept for performing the tests and the "source" directory should be kept for studying the code of examples.

3.5 Performing Tests

DataMPI tests are used to verify the newly complied DataMPI working properly. By performing the tests, specific problems in certain environment can be generally located. DataMPI tests are located in the build directory. To run all the tests, the following commands can be used.

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

If all goes well, the result will be printed like this.

```
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
```

3.6 Performing Examples

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

3.6.1 Configure DataMPI

Before performing DataMPI examples and writing MPI-D programs, the DataMPI installation must be well configured. The configuration files of DataMPI are located in the `conf` subdirectory of the install

directory. To perform the following simple example on a single node, a few files should be modified, the variables in the angle brackets should be replaced by the real values of users' environment.

`hostfile`:

```
<your_host_name>
```

`mpi-d-env.sh`:

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

To perform multi-node tests or examples, the install directory should be available on all the nodes with the same path (either employing a network file system or copying the directory to all the nodes), and all the hostnames should be listed line by line in `hostfile`.

3.6.2 Run Examples

The examples are grouped and located in the `share/datampi/examples` subdirectory of the install directory. For example, the "common" group includes some examples for the Common mode of MPI-D. The `mpidrun` command in the `bin` subdirectory is the launcher of MPI-D programs, which can be used to run examples. A `Sort` example is given 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 meanings of the parameters will be explained in Section 4

If all goes well, the result will be printed like this.

```
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_0.64
    ended, used time 1.03 sec.
```

4 Basic Usage Instructions

The most important command of DataMPI is `mpidrun`, which is used for launching a DataMPI job. Its available parameters are listed as follows.

```
$ mpidrun -f <hostfile> -O <n> -A <m> -mode <mode> -jar <jarname> <classname>
    <params ...>
```

- The `-f` parameter indicates the path of hostfile.
- The `-O` parameter refers to the number of operator tasks in MPI-D, which can be considered as the number of mappers in MapReduce.
- The `-A` parameter refers to the number of aggregator tasks in MPI-D, which can be considered as the number of reducers in MapReduce.
- The `-mode` parameter indicates the mode used by the user-defined application.
- The `-jar` parameter indicates the jar file where the program locates.
- The `classname` is the entry class name of the program.
- The `params` are the the parameters for the entry class.

In the user application code, there are two parameters of reserved keys that can be used to define data types of key-value pairs.

```
private static void initConf(HashMap<String, String> conf) {
                conf.put(MPI_D_Constants.ReservedKeys.KEY_CLASS,
                    Text.class.getName());
                conf.put(MPI_D_Constants.ReservedKeys.VALUE_CLASS,
                    Text.class.getName());
                ...
}
```

The meanings of the parameters are listed as follows.

- `MPI_D_Constants.ReservedKeys.KEY_CLASS` indicates the data type of send/receive key class, the default value of this parameter in the Common mode is `java.lang.String`.
- `MPI_D_Constants.ReservedKeys.VALUE_CLASS` indicates the data type of send/receive value class, the default value of this parameter in the Common mode is `java.lang.String`.

5 Advanced Configuration

5.1 User Application Code

There are some advanced parameters of reserved keys that can be used in users' appllications. There are some advanced parameters of reserved keys that can be used in users' appllications.

```
private static void initConf(HashMap<String, String> conf) {
                conf.put(MPI_D_Constants.ReservedKeys.COMBINER_CLASS,
                    WordCountCombiner.class.getName());
                ...
}
```

The meanings of the parameters in the `MPI_D_Constants.ReservedKeys` class are explained as follows.

- `MPI_D_Constants.ReservedKeys.PARTITIONER_CLASS` indicates the partition function class, which is used to define the distribution policy of key-value pairs from one communicator to the

other.
- `MPI_D_Constants.ReservedKeys.COMBINER_CLASS` indicates the combiner function class, which is used to tell the library how to combine the keys.
- `MPI_D_Constants.ReservedKeys.COMPARATOR_CLASS` indicates the comparator function class, which is used to compare the intermediate data for reducing the size of exchanged data.

5.2 mpi-d.properties

Parameters in `mpi-d.properties` are used to set the intermediate data cache path and tune the buffer management, these parameters are explained as follows.

- `mpid.local.dir` indicates the storage path of local intermediate data. The default value is `/tmp`.
- `mpid.io.sort.factor` indicates the number of streams to merge at once while sorting files, which determines the number of open file handles. The default value is `100`.
- `mpid.mem.used.percent` indicates the percent of maximum memory in JVM that can be used in for data transferring. The default value is `0.7`.
- `mpid.spill.percent` indicates the threshold for determining when to trigger disk spill. The default value is `0.8`.
- `mpid.block.capacity` indicates the maximum partition buffer size (KB) for each aggregator. The default value is `1024`.
- `mpid.block.metadata.percent` indicates the metainfo occupation percent in one partition buffer. The default value is `0.75`.
- `mpid.send.queue.size` indicates the length of send queue used for data caching. The default value is `3`.
- `mpid.recv.queue.size` indicates the length of receive queue used for data caching. The default value is `16`.
- `mpid.disable.mem.cache` indicates that all intermediate data spill to the disk if it is set to `true`. Otherwise, DataMPI will cache the intermediate data in memory as much as possible.

5.3 mpi-d-env.sh

The JVM memory size and hosfile file path are defined in this file.

6 Benchmarks

DataMPI provides a set of benchmarks to help users conduct performance evaluation.

6.1 Micro-Benchmarks

DataMPI Micro-Benchmark Suite is used to evaluate the performance of MPI-D implementations over various infrastructures.

DataMPI Micro-Benchmark Suite 0.6.0 mainly consists of four benchmarks:

- **WordCount** - counts the appearance times of all word in the files.

  - Input format: any document (usually in text format)
  - Output format: <word> <count>
  - Dataset: Generated through `RandomTextWriter` in Hadoop
  - Command-line execution:

```
$ ./dmb.sh wordcount <num-O-tasks> <num-A-tasks> <input-dir> <output-dir>
```

- **Sort** - sorts the input files based on the keys.
  - Input format: <key> <value>
  - Output format: <key> <value>
  - Dataset: Generated through `RandomTextWriter` in Hadoop
  - Command-line execution:

```
$ ./dmb.sh sort <num-O-tasks> <num-A-tasks> <input-dir> <output-dir>
```

- **TeraSort** - sorts 100-bytes <key, value> tuples. Each of them contains 10-bytes key and 90-bytes value.
  - Input format: <key> <value>
  - Output format: <key> <value>
  - Dataset: Generated through `TeraGen` in Hadoop
  - Command-line execution:

```
$ ./dmb.sh terasort <num-O-tasks> <num-A-tasks> <input-dir> <output-dir>
```

- **Grep** - extracts matching strings from input files and counts the appearance times of the strings.
  - Input format: <key> <value>
  - Output format: <key> <value>
  - Dataset: Generated through `RandomTextWriter` in Hadoop
  - Command-line execution:

```
$ ./dmb.sh grep <num-O-tasks> <num-A-tasks> <input-dir> <output-dir>
```

The suite also contains several primitive-level benchmarks like `Bandwidth` and `Latency` to evaluate Java-level MPI operation performance.

6.2 BigDataBench

BigDataBench (http://prof.ict.ac.cn/BigDataBench) is a big data benchmark suite abstracted from Internet services. It includes six real-world data sets, and nineteen big data workloads, covering six application scenarios: micro benchmarks, Cloud "OLTP", relational query, search engine, social networks and e-commerce. Through generating representative and various big data workloads, BigDataBench features an abstracted set of Operations and Patterns for big data processing. For the same workloads, BigDataBench provides different implementations based on various programming models, including MapReduce, MPI, MPI-D (DataMPI), Spark, etc.

The DataMPI 0.6.0 release provides its implementation of three commonly used micro-benchmarks in BigDataBench, which are listed as follows. Other benchmarks will be included in future releases gradually.

- **WordCount** - counts the appearance times of all word in the files.
- **Sort** - sorts the input files based on the keys.

- **Grep** - extracts matching strings from input files and counts the appearance times of the strings.

DataMPI-BigDataBench uses data generated by BigDataBench Text Generator, which can be downloaded from this website (http://prof.ict.ac.cn/BigDataBench/#downloads).

6.2.1 Demo

Here a demonstration is given to show how to run DataMPI-BigDataBench step by step. The major two steps include data preparation and running benchmarks (`WordCount`, `Sort`, `Grep`). Note that DataMPI-BigDataBench related scripts can be found in `$DATAMPI_HOME/benchmarks/dmbdb`.

```
# Prepare Input Data
$ cd ${BIGDATABENCH_HOME}/BigDataGeneratorSuite/Text_datagen
$ sh gen_text_data.sh lda_wiki1w 10 100 1000 gen_data
$ ${HADOOP_HOME}/bin/hadoop fs -copyFromLocal gen_data /

# Convert Text files to Sequence files
$ cd ${BIGDATABENCH_HOME}/BigDataGeneratorSuite/Text_datagen/ToSeqFile
$ ./sort-transfer.sh gen_data gen_data_seq

# Run Benchmarks
$ cd ${DATAMPI_HOME}/benchmarks/dmbdb
$ ./dmbdb.sh wordcount 4 1 /gen_data out_wc
$ ./dmbdb.sh grep 4 1 /gen_data out_gp
$ ./dmbdb.sh sort 4 4 /gen_data_seq out_st
```