

Earth System Modeling Framework

Part II - JMC code 1 Baseline User's Guide

Chris Hill, Stephanie Dutkiewicz, Jean-Michel Campin, Curt Heisey

Contents

1	Introduction	2
2	Downloading JMC Code L Baseline Code and Data	2
3	Directory Structure	2
4	Build	2
4.1	System Requirements	2
4.2	General	3
4.3	Building the MITgcm on the MIT myrinet-3 cluster	3
5	Execution	4
5.1	System Requirements	4
5.2	General	4
5.3	Running the MITgcm on the MIT myrinet-3 cluster	4
6	Verification	5
7	Hints and Tips	5
7.1	Setting up ssh	5

1 Introduction

This document provides a brief synopsis of the download, compilation and execution procedure for the the Earth System Modeling Framework (<http://www.esmf.ucar.edu>) JMC code L baseline. The code and its time to solution measurements are part of ESMF joint milestone E. Code L is a coupled 2.8° ocean atmosphere configuration of the MIT General Circulation Model (MITgcm). This configuration is actively in use for research into air-sea dynamical and biogeochemical couplings, especially on interannual and greater timescales. Details of the code configuration can be found under the *Applications* link at <http://www.esmf.ucar.edu>. Reference material for the MITgcm code can be found at <http://mitgcm.org> and specific information on the current JMC code L baseline setup and its future evolution over the ESMF project can be found at <http://mitgcm.org/JMClbaseline>.

The baseline configuration provides a starting reference point for a time to solution metric that will be used to evaluate ESMF framework overheads. As the ESMF project progresses, code L will evolve internally and the framework will replace the codes' native infrastructure. Two forms of code L will exist throughout the ESMF project one outside the framework and one that is fully framework compliant. These two forms will be used, all through the ESMF project, to establish that the time to solution, for an identical problem, of the framework compliant code is less than 10% longer than the time to solution for the non-framework form.

The problem used to measure time to solution is a ten-day coupled simulation starting from the saved state of a 50 year coupled spin-up. Subsequent milestones will give the time to solution for both framework and non-framework code. At this stage, prior to the framework development, only a single time to solution is calculated and reported. Section 2 describes how to download the code for L baseline. Sections 4, 5 and 6 describe the steps for compilation, execution and validation of the results. The exact procedures and pathnames described are tailored to the configuration of the NASA Goddard HP Alpha system **halem.gsfc.nasa.gov**. However, the code executes and many other platforms and many of the steps are the same for other platforms and systems.

2 Downloading JMC Code L Baseline Code and Data

The JMC Code L Baseline code and data can be downloaded from the download link under <http://mitgcm.org/JMClbaseline>, This link will fetch a compressed tar archive called `JMClbaseline.tar.gz`. After downloading this file needs to be uncompressed using the command `gzip -d JMClbaseline.tar.gz` and its contents extracted using the command `tar -xvf JMClbaseline.tar` into an appropriate workspace directory (for example a directory under `/work` on the **halem.gsfc.nasa.gov** system).

3 Directory Structure

The top-level JMC Code L Baseline directory contains `atm`, `ocn`, and `cp1` subdirectories along with scripts and files required to build (`build.sh`), execute in batch (`run.sh`, `tj`, `tj-driver`) and validate (`STDOUT.0000.ref`) the baseline code. The file `README`, also at the top-level, contains details of the contents of each of these directories, files and scripts as well as pointers to the MITgcm release1 documentation (<http://mitgcm.org/sealion>).

4 Build

4.1 System Requirements

A Fortran 90 compiler and system library and a compatible MPI library are required to compile and link this code. All routines are in Fortran except for a single timing routine that uses C. This routine requires a C compiler and a C library with the `gettimeofday()` function. Appropriate macros for calling a C procedure with with a pointer to a 64-bit float are required and are included.

Note that each MPI combination of compiler and transport layer requires its own complete MPI build. The library paths for appropriate compiler library, include files and compiler executable must be properly set when building MPI.

In addition the MPI library paths and compiler executable path must be set when compiling or running with MPI.

4.2 General

The script `build.sh` in the JMC Code L Baseline top-level directory creates three executables.

- `ocn/src/tutorial_examples/global_ocean.128x64x15/mitgcmuv` is the executable for the ocean component and is built from sources in sub-directories under `ocn/src`.
- `atm/src/tutorial_examples/aim_51.LatLon/mitgcmuv` is the executable for the atmosphere component and is built from sources in sub-directories under `atm/src`.
- `cpl/src/tutorial_examples/aim_51_LatLon+global_ocean.128x64x15/mitgcmuv` is the executable for the atmosphere ocean coupling component and is built from sources in sub-directories under `cpl/src`.

The three sets of directories `ocn/src`, `atm/src` and `cpl/src` are all organised using the MITgcm release1 scheme that is described at <http://mitgcm.org/sealion>.

The code can be cleaned with the build option `build.sh -Clean` in the JMC Code L Baseline top-level directory.

4.3 Building the MITgcm on the MIT myrinet-3 cluster

The MIT myrinet cluster consists of a gateway node `cg01`, and sets of cluster nodes, `myrinet-0-NN`, `myrinet-3-NN`. At the moment, `myrinet-0-NN` are still undergoing testing.

Log on to gateway machine `cg01`, then on to one of the compute nodes. The Build is done on any of the compute nodes, `myrinet-3-NN`. The libraries and paths for building are not set on the gateway node (each cluster may have separate libraries).

The appropriate compiler options must be set. The build requires compiler with f90 capabilities (e.g. Portland, Intel but not g77).

Modify 3 `genmake` files and `build.sh` for platform specifics

```
./atm/src/tools/genmake
./cpl/src/tools/genmake
./ocn/src/tools/genmake
```

```
case cg01+mpi:
  set LN          = ( '/bin/ln -s' )
  set CPP         = ( '/lib/cpp -traditional -P' )
  set INCLUDES   = ( '-I/usr/local/pkg/mpi/mpi-1.2.4..8a-gm-1.5/pgi/include' )
  set DEFINES    = ( '${DEFINES} -DWORDLENGTH=4' )
  set FC         = ( '/usr/local/pkg/mpi/mpi-1.2.4..8a-gm-1.5/pgi/bin/mpif77' )
  set FFLAGS     = ( '-byteswapio -r8 -Mnodclchk -Mextend' )
  set FOPTIM     = ( '-tp p6 -v -O2 -Munroll -Mvect=cachesize:512000,transform -Kieee' )
  set LINK       = ( '/usr/local/pkg/mpi/mpi-1.2.4..8a-gm-1.5/pgi/bin/mpif77' )
  breaksw
```

(Note how the path for the specific binding of MPI-CH is specified, in this example to use Portland f90 compiler and myrinet MPI transport layer).

The `build.sh` script is edited with lines
`../../../../tools/genmake -platform=cg01.`

Alternatively, the `.genmakerc` files can be edited.

Note: the `genmake` compiler options should be checked into CVS.

The MITgcm executables can be built.

```
build.sh -clean
build.sh builds code
```

5 Execution

5.1 System Requirements

An MPI implementation that provides support for MPMD multi-process execution is required.

5.2 General

The script `run.sh` in the JMC Code L Baseline top-level directory is used to run the executables created by `build.sh` (section 4). The script takes the ocean component executable and input files from `ocn/src`, `ocn/input`, the atmosphere component executable and input files from `atm/src` and `atm/input` and the ocean-atmosphere coupling component executable from `cpl/src` and assembles run directories for each component. These directories are called `ocn/run`, `atm/run` and `cpl/run` for the ocean component, atmosphere component and ocean-atmosphere coupling component respectively. After assembling the run directories the `run.sh` script submits a batch job to the **halem.gsfc.nasa.gov** LSF queue system using the **esto-big** queue. The job scripts `tj-driver` and `tj` are used for this procedure. Together, these three scripts launch an MPMD parallel processing MPI job that assigns 8 processes to the atmosphere component, 8 processes to the ocean component and a single process to the ocean-atmosphere coupler component.

5.3 Running the MITgcm on the MIT myrinet-3 cluster

Modify the the default MPI-CH distribution `mpirun` script so that different executables ran in separate directories
This has been done for Portland MPI on myrinet-3 only

The processor groupings must be specified. `SIZE.h` (2) files must be adjusted for the processor topology

Recompile the code

Create a MPI procgrop file. Here is a sample procgrop file for 5 processors.

```
cat procgrop
```

```
myrinet-3-01 0 /cluster/home/cheisey/jmcl/./cpl/run/mitgcmuv
myrinet-3-02 1 /cluster/home/cheisey/jmcl/./ocn/run/mitgcmuv
myrinet-3-03 1 /cluster/home/cheisey/jmcl/./ocn/run/mitgcmuv
myrinet-3-04 1 /cluster/home/cheisey/jmcl/./atm/run/mitgcmuv
myrinet-3-05 1 /cluster/home/cheisey/jmcl/./atm/run/mitgcmuv
```

Note: the `run.sh` script needs to be updated for the MIT cluster Note: the myrinet cluster needs a batch system to be updated

6 Verification

After the `run.sh` executes the MPMD coupled simulation it carries out a check of output and extracts timing information to determine time to solution. The output check compares the daily global mean atmospheric potential temperature with a previously stored set of values for this configuration. The previously computed set of values are in the file `STDOUT.0000.ref` in the JMC Code L Baseline top-level directory. These stored values are checked against the time series of mean potential temperature written to the file `atm/run/STDOUT.0000` by the script `run.sh`. Time to solution information is also extracted from the `atm/run/STDOUT.0000` output file. The compute time for the 17 processor, 10-day simulation used for the Code L Baseline is converted into a measure of the time required for one year of simulation. This is printed in the summary output file `output.txt`. Startup and shutdown time costs for the code are excluded to avoid skewing short benchmark results with respect to extended simulation performance.

7 Hints and Tips

7.1 Setting up ssh

The `ssh` command enables logging on to a remote machine and executing commands on the remote machine in a secure fashion. Some configurations of MPI even require `ssh`. To set up `ssh`, each user must create a pair of unique keys which are then installed in a `.ssh` folder in the user's top level directory. Create a public and private key pair by issuing the following command, where `<keyfilename>` is the filename that the keys will be written to.

```
ssh-keygen -t rsa -N "" -f <keyfilename>
```

This will create two files `<keyfilename>` and `<keyfilename>.pub` in the directory where the command is issued. Append the contents of the `<keyfilename>.pub` file to `userid/.ssh/authorized_keys`, which should be created if it does not exist. Overwrite the contents of `<keyfilename>` onto `userid/.ssh/identity`. The file permissions on `userid/.ssh/identity` should be "rw" by owner only. e.g. `chmod 600 userid/.ssh/identity`. Note that the `identity` file must be kept secret, and this procedure creates a key pair with an empty passphrase (no password), which should only be used in very controlled network environments.