Sanjay Bappudi
Spring 2013 report
CSS 490

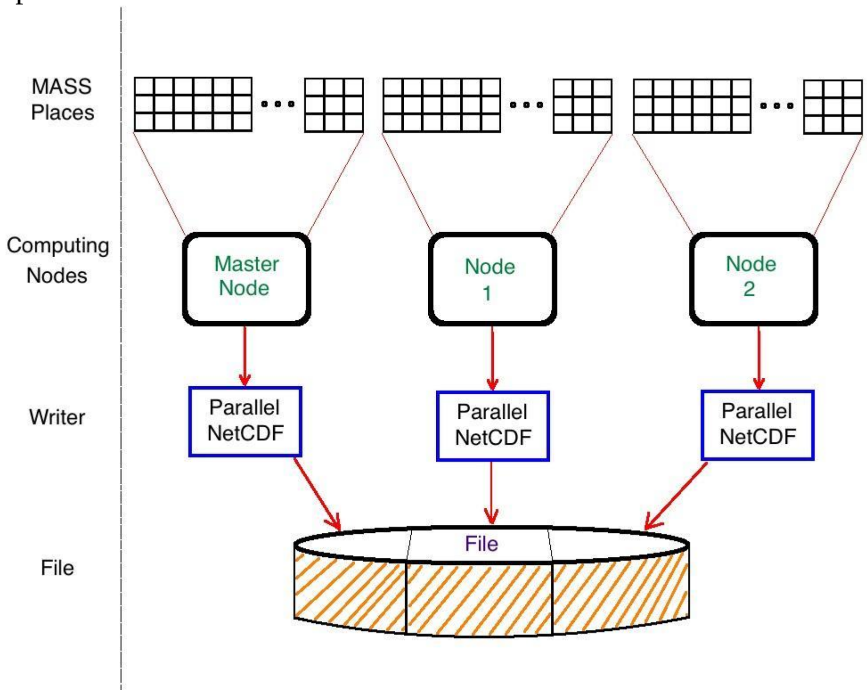
This quarter, I began to work with professor Fukuda in understanding the MASS library and to debug and test the work done by a previous student. I have not produced any individual documents or products of my own thus far, however, this quarter has provided me with a crucial understanding of the MASS library and its application possibilities.

The previous student who worked on this project made the following contributions, which I have spent much of the quarter trying to decode and understand. These include the following contributions:

Integration of MASS and NetCDF
This allows MASS applications to run in parallel with a NetCDF writer Places object that will allow the save and resume of simulation data. Saving simulation data can be usefull for a variety of situations.

It allows the user to save computational time by saving simulation state, so that we dont have to re run simulations. Also, simulation states can be used as backups incase a simulation fails to execute properly.

This integration between the MASS and NetCDF library works through the use of each computing node utilizing a parallel-ized NetCDF writer object to write data into a buffer. This buffer is then transfered and concatenated by the master node, before being saved to a file. The picture below demonstrates this:



Below are some of the contributions I have made to the MASS library during this past quarter:

Support of multiple Places objects

I have added support of multiple Places objects during my testing. Since the MASS library was never testing with multiple places objects, this bug was unknown until I ran the parallelized version of the NetCDF and MASS libraries. It was fixed by adding a HashMap of Places in Mprocesses.java which allowed it to differentiate different simulation spaces through the use of Place handles.

NetCDF file concatenation

While this was not able to be implemented completely this quarter, I have put in time on research on how to be able to concatenate NetCDF files in the master node before actually writing data to file. This is instrumental in creating good performance for the MASS + NetCDF library since we will avoid many system calls.

Multiple Thread read/write

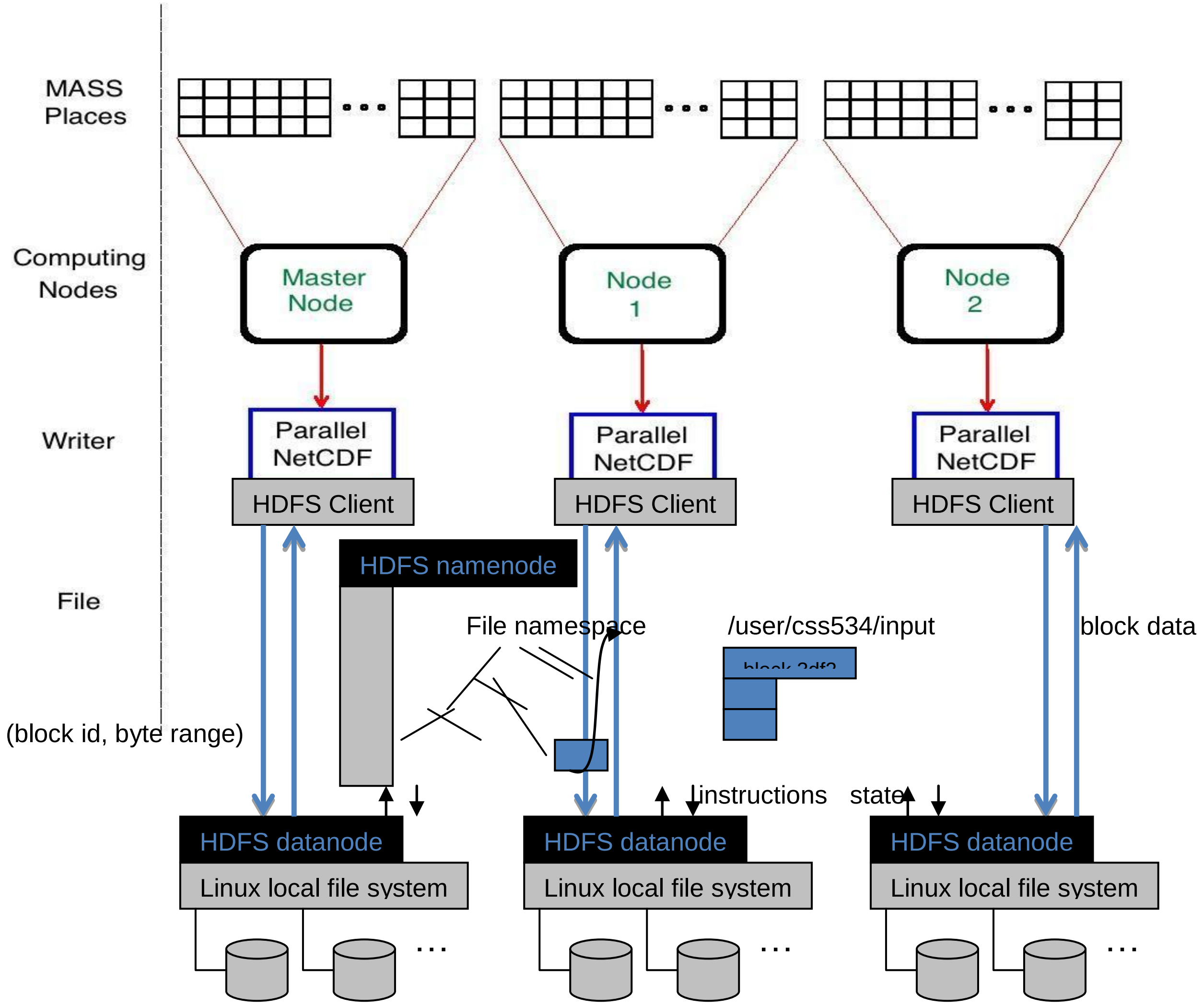
Another idea that professor Fukuda and I discussed but was unable to implement is our solution to a deadlock bug we faced. Multiple processes were trying to read/write at the same time and were deadlocked. We were able to overcome this situation by having two threads, one to write information, and another to read information continually. This prevent deadlock and allows the transfer of data to finish.

Further Work

Next quarter, I plan to implement the ideas that I was unable to this quarter. Also, I plan add to the MASS library and extend the integration of NetCDF and MASS libraries. I further documented this plan on my CSS 490 project proposal. This will include the inclusion of Hadoop in order to create a distributed file system to effectivity store large amounts of data that we can expect from computationally intensive simulations (such as weather simulations).

The way this will work, is through the use of each writer object talking to a HDFS client. The client then talks to a the local filesystem on different computing nodes, figuring out which blocks are available. It then splits the file, and stores the file on different blocks, on different disks all organized under a HDFS name node.

This is show below in the following picture:


MASS Places
Computing Nodes
Master Node
Node 1
Node 2
Writer
Parallel NetCDF
Parallel NetCDF
Parallel NetCDF
HDFS Client
HDFS Client
HDFS Client
File
HDFS namenode
File namespace
/user/css534/input
block 3df2
block data
(block id, byte range)
instructions
state
HDFS datanode
HDFS datanode
HDFS datanode
Linux local file system
Linux local file system
Linux local file system