Key Assumptions:
1. We only store files under *_/tmp/YZFS/_* path, don’t provide any directory operations for user.
2. No space in file name.

15-640 (Fall 2013) Project #3 MadReduce Facility

Yin Xu (yinxu), Zheng Kou(zhengk)

1. design

1.1 distributed file system

For the ease of implementation and the sake of communication conservation, we use master/slave architecture for the distributed system. The master server maintains all information about the file system (ie. metadata of files in the file system, metadata of part of the file into which file be broken and metadata of slave server the master server manages) but doesn't store any file. Master server is the hub of all communication in the file system and dispatch tasks to slave servers.g Slave servers are responsible for storing files for mappers and reducers to use them.

2.2 MapReduce Framework

Our framework contains one master server coordinating all slave nodes. The server program runs on the exactly the same server as our distributed file system. Every client can initiate a MapReduce job from its command line by sending NewJob message to the server. Once the master received the new job request, it splits the job into a bunch of mapper tasks based on the file replication info stored locally. All mapper tasks are put into a queue waiting to be sent out.

Once a mapper task is done, it opens a downloading port and sends back acknowledge message to inform the master server. Once the master server received such message, it will download the intermediate result generated by the mapper.

When all mapper tasks are done, all intermediate results should appear in the server. The server will start the reducer that calculates the final output. The output file is stored in the master server’s local file system.

2. implementation

2.1 distributed file system

The distributed file system is able to operate four use commands: copy from local, ls, cat and rm.

2.1.1 copy from local
the command send a message to master server telling the filename and size of the file. The master server will randomly choose which slave will store the file or part of the file (if the file is too large). After that, the slave server will fetch the file from the machine which sent the message to the master. Finally, the master server updates its global metadata about the file.
2.1.2 ls
list the name of all files from master server's global data structure that maintains the metadata of the file.
2.1.3 cat
display the content of the specific file on the standard output.
2.1.4 rm
remove a file in all slave servers and delete the metadata of that file on master server.
2.1.5 quit
quit the entire file system framework and map reduce framework.

2.2 Map Reduce API
The user who wants to utilize the map reduce framework should can write its own class containing a Map class (which must implements Mapper interface), a Reduce class (which must implements Reducer interface) and a MapReduceConf method (which takes no argument and set the Class of input/output key/value of mapper and reducer into conf object for the use of the framework).

In addition, the outputkey and outputvalue of mapper must have clone method and outputkey class must have getHashcode method for the use of merge sort.

3. Instructions (Build, Deploy and Run)

Build:
Unfortunately, no makefile provided. We built the source code using eclipse.

Deploy (Start the file system and framework):
go to ./bin directory and run ./m.sh on master server
go ahead change the first line of the s1.sh, s2.sh and s3.sh to the ip of the master host. run ./s1.sh on (n -1) slave hosts (n is the number of slave you want).

Run:
And run either ./s2.sh or ./s3.sh on the last host.
At this this time you can only run s2.sh or s3.sh once. In order to run another time, you may need to re-deploy the whole framework.

To quit the framework, use “java dfs.YZFS -yzfs quit” on any slave server.

4. System requirements

Java Virtual Machine

6. Features we haven’t implemented

- Our framework couldn’t handle the slave failure.
- Doesn’t support concurrent MapReduce jobs.
- All reducer job is done in the master. It would be better to distribute the reducer to slave nodes as well.

5. Test Examples

Example 1:

A simple WordCount program that is similar to the hadoop WordCount.

Example 2:

The Maximum example calculates the maximum integer within the input files.

To check the output result:

```
cat /tmp/YZFS/output.txt
```