15640 Project3 Report

Building a Map-Reduce Facility with specialized DFS

Name1: Jian Wang

AndrewID:jianw3

Name2: Yifan Li

AndrewID:yifanl

Part I: Design and Implementation

1. Design on MapReduce
   1. File I/O
   2. Master-Worker Framework
   3. Scheduling

Splitting

Map Tasks dispatching

Reduce Task dispatching

* 1. Management Tool
  2. Failure and Recovery

1. Design for DFS
   1. DFS upload client
   2. DFS Name Node and Data Node framework

In order to reduce the implementation effort, we design the DFS as master-slave mode. The name node works as the pivotal of the DFS and keep the whole picture of the system. Data node works as slave node and listen to the command from name node. Every data node will launch a server and accept the download file request from other data node. This is used when the file gets chunk and replicated and also used when the reducer need to get all the output files from different mapper.

Additionally, the name node will run with the Mapreduce master on the same machine in the same process. They will run as different thread when system boot up. Data node and Mapreduce worker node will run in the same process and run as different thread. This design will reduce our effort to achieve the communication between Mapreduce and DFS.

* 1. DFS abstract file system

The name node will keep a virtual file structure on it’s own. In order to achieve this, name node will keep a file chunk map for every DFS file. The map will keep the file chunk to the data node address, port and local path.

* 1. DFS file reading and writing

When never the data node want to create a file on the DFS, it will send request to name node and name node will create a map for this. Whenever there is a need to read a file chunk, the name node will send download request to the data node with the target address and port information. The receiving data node will send a download request to the target data node server to get the file chunk.

* 1. DFS file replication

All the files on the DFS will be replicated on a different node. Whenever there is DFS file chunk created, the name node will pick up a data node to replicate this file chunk. In order to achieve this, the name node will send a download request to the replica data node with the target file address. Name node will keep this replica information in the file meta data. When node failure occur, name node will use the replica file and replicate the replica file again.

* 1. DFS data node failure and recovery

Part II: System Deployment and Configuration

Build

We do not provide make files. You need to use eclipse to build the project. The code we submitted already has the project configuration.

Deploy

In order to run the system, you need to run the master on one machine and run the worker at several different machines. There are three configuration files as:

* masterConfig.properties: this configuration file contains the port number

JobSubmissionPort: the socket port to receive the job submission

WorkerServerPort: the conmmunication socket number for every worker

DataNodeServerPort: the communication port number btween nameNode and dataNode

* workerConfig.properties: this configuration file contains the master address and data node local port

MasterAddress: master IP address

MasterPort: master port

DataNodeServerPort: the same as the one in masterConfig.properties

LocalPort: the download server port for the dataNode

* clientConfig.properties: this configuration file contains the master address and reducer number. It’s used by the client program

MasterAddress: master IP address

MasterPort: master port

ReducerNum: the reducer number

When you deploy the system, you need to copy the all the files to the master and worker machine. Change the MasterAddress and MasterPort in the workerConfig.properties and clientConfig.properties files to your master’s IP address. Then set the ReducerNum in clientConfig.properties. We set it default to 2. Mostly you do not need to change other items.

Run

First, you need to cd to bin directory, then

1. on the master machine

java mapreduce.master.Master

1. on worker machine

java mapreduce.WorkerNode

1. on the client machine

Part III: User Library and Simple Usage

We provide five class in the user lib which are: Mapper,Reducer,Job, FileInputFormat and FileOutputFormat.

User need to implement and inherit these classes when submit their job.

Part IV: Features We haven’t implemented

Part V: Test Case

1. WordCount

This example will count word occurrence in a file. For every word in the file, there will be a value account to the number of occurrence in the file.

1. Maximum

This example will find the maximum number in a file.