Distributed Systems (CSE431) Monsoon 2016
Project Phase-I, HDFS

Due: 12th October, 11:55 PM

Problem Statement: The initial phase of the project requires to implement a distributed file system similar to HDFS.

The file system would be built on using two major components NameNode and DataNode(s) held together with a Client.

NameNode:
The component primarily responsible to handle the communication with DataNodes.
When queried, a NameNode should be able to perform all major file operations such as open, close, list and should support a block mechanism similar to HDFS. Thus, NameNode also takes care of managing block allocations and get locations of existing blocks.

DataNodes:
Should be used for performing read and write operations within a block.

Client:

- Command line interface to interact with the NameNode.
- Usage :
  - $> get (Writes a file to the local filesystem from HDFS)
  - $> put (Writes a file to HDFS from local file system)
  - $> list (Displays all files present in HDFS)
- Handles reads and writes from DataNode on receiving metadata information from NameNode about a file.

Key Points:

- Use RMI (Java) or Sun RPC (C/C++) for communication
- Server should be multithreaded for seamless communication
- No update operation is required for the purpose of simplification
- The system should persist its state even in the event of a NameNode restart.
- The DataNodes should correspondingly support this restart and get, put, list functions should be carried on ordinarily after NameNode server is up and running again
- Replication Factor: Two way cascading
- You are allowed to maintain a configuration file of your own
- Use CRC for error detection while communicating (optional but recommended)
- All the parameters and return objects should be marshalled and unmarshalled from byte arrays (Using Google Protobuf)

Reference Implementation Guide:

All APIs take a byte[ ] as an input param, and return a byte[ ] as an output. The input and output byte[ ] arrays should be built over Google protobuf

- **Open**
  - Assign an initial block.
  - Unique handle for each opened file.
- **Write**
  - Write contents to the assigned block number

    Workflow:

    ```
    openFile("filename");
    In a loop:
        Call assignBlock() using handle from openFile
        Obtain a reference to the Remote DataNode object using the
        first entry in the DataNodeLocation
        Call writeBlock() to the DataNode
    closeFile()
    ```
- **Read**
  - Get all block locations for the file
  - Read blocks in sequence

Workflow:

```
openFile("filename");
In a loop:
    getBlockLocations() using handle from openFile
    Obtain a reference to the Remote DataNode object using the
    a random entry in the DataNodeLocation
    Issue a readBlock() to the DataNode
    closeFile()
```

- **Close**
  - Close the file after read or write request.
- No update operation
- All responses should contain a status message depicting the success or failure of a request.
- Heartbeats from a DataNode should be periodically received by a NameNode to keep a check on the status of its health and on that of its BlockReport(in turn used by the NameNode to get information about all the blocks of a DataNode it is holding within).
- For each file, NN knows just the filename and list of blocks. NN does not store the locations. Once DNs send blockReports, NN knows the locations of each block and tracks this information in memory (and never writes this information to a persistent storage such as disks). NN has to maintain the list of files, and the blocks of each file in a directory (or a file) in the local OS.
- Clients and DN discover the NN from a conf file and read from a standardized location. The conf file contains the socket information of the NN.

Possible Cluster Setup Configurations:

- Each of the two members can have one virtual machine installed by using VMware or other virtualization tools. You'll then have four machines for the HDFS cluster (Two VMs and two host machines). Connecting the machines with the LAN wire with static IPs and configuring VMs in bridged mode in VMware should establish all working connections to resume actual work.
- Other than VMs, you can also use docker containers having setup for Java or C++.
- Don't expect support in establishing network and working configuration.