Mini-DFS report

Mingkun Huang

June 24, 2018

1 System Structure

Mini-DFS is running through a process. In this process, the name server and data servers are different threads. The system structure is shown in figure 1.

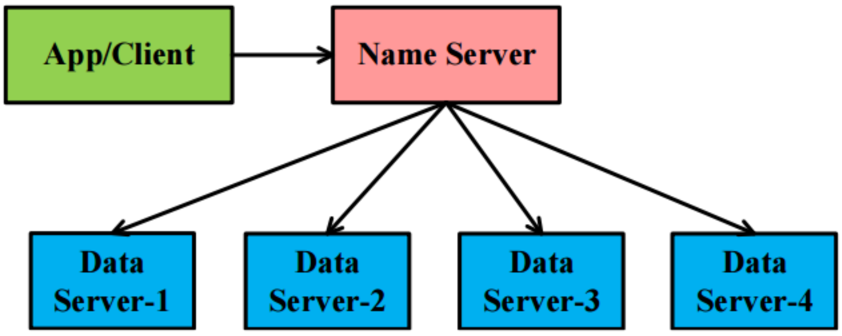

Figure 1: System structure

2 Basic Functions

- Read/write a file
  - Upload a file: upload success and return the ID of the file
  - Read the location of a file based on the file ID and the offset
- File striping
  - Slicing a file into several chunks
  - Each chunk is 2MB
  - Uniform distribution of these chunks among four data servers
- Replication
  - Each chunk has three replications
  - Replicas are distributed in different data servers

3 Details of Each Component

3.1 Name Server

- List the relationships between file and chunks
- List the relationships between replicas and data servers
- Data server management

3.2 Data Server

- Read/Write a local chunk
- Write a chunk via a local directory path

3.3 Client

- Provide read/write interfaces of a file

4 System Design

The whole system flowchart is shown in figure 2.

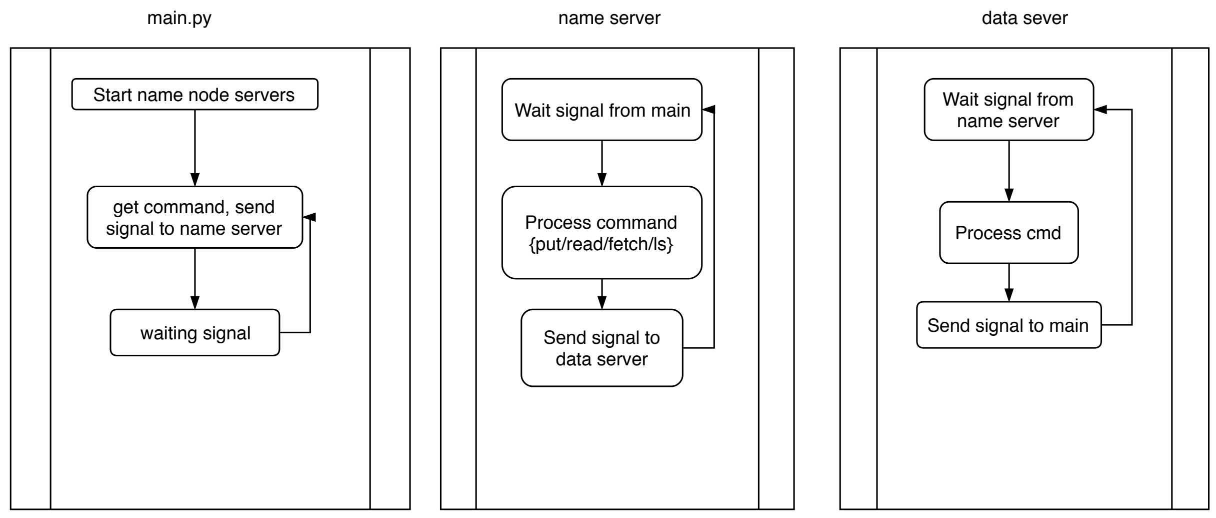
main.py
Start name node servers
get command, send signal to name server
waiting signal
name server
Wait signal from main
Process command {put/read/fetch/ls}
Send signal to data server
data sever
Wait signal from name server
Process cmd
Send signal to main

Figure 2: System flowchart

4.1 Data Structure

- Global events:
  - name_event: Name server wait for this event, send by main.py.
  - data_events: 4 data servers, 4 events. Name server send this signal to control data server.
  - main_event: wait for dataserver in main.py
  - read_event: data server send this signal to main.py once success read, and name server send this signal to main.py for some error.
  - ls_event: name server list files, then send this signal to main.py
- Name server:
  - id_chunk_map: map file id to chunks, eg. 0: ['0-part-0', '0-part1'], 1: ['1-part-0', '1-part-1']
  - id_file_map: map file id to file info, eg. 0: ('file1', size1), 1: ('file2', size2)
  - chunk_server_map: map chunk to its replications, eg. '0-part-0': [0, 1, 2, 3]
  - last_file_id: the maximum file id

4.2 Commands Effects

- Put
  1. get file id, split file size into chunks
  2. mapping file id to chunks, update id_chunk_map
  3. mapping file id to file info, update id_file_map
  4. add chunks to each data server, update server_chunk_map

- Read
  1. get file id and offset count
  2. calculate start chunk
  3. send signal to data server

5 Instructions

- ls: list all files in DFS.
- put <source_file_path>: upload local file to DFS, and return file id.
- read <file_id> <offset> <count>: read from DFS using file id.
- fetch <file_id> <save_path>: download file from DFS using file id, save file to local path.