Design doc for ECE419 hw4

Yiming Kang (998676730)
Zexuan Wang (998851773)

Assumptions:
We designed the system based on several major assumptions

1- ZooKeeper will never die
2- Socket communication is FIFO and reliable
3- Primary and backup server will not suffer double failure

Design:
The `FileSever` and `JobTracker` are implemented using the Primary/Backup architecture as suggested. The features of P/B are abstracted in **`PBArchitechture.java`** that provides

- Connection to ZooKeeper
- Creation of an EPHEMERAL node by primary
- Watcher on that node by backup
- Transparent handling of Primary failure through Watcher

`Worker` and `ClientDriver` need to connect to a P/B based node and therefore need to handle Primary failure transparently. This feature is abstracted in **`PBArchitechtureAdaptor.java`** that provices

- Connection to ZooKeeper
- Connection to Primary with automatic failover to new Primary (backup)
- Communication with Primary (send and receive packets)

Communication between services are mostly done through zookeeper nodes where possible, each node has associated data attribute that can be accessed using `getData()` and `setData()`. However, some communications are done via java ServerSocket/Socket, we've created the serializable `MPacket` class that packages various information within it to make communication easier.

Client Driver (extends PBArchitechtureAdaptor)

- Starts with the address of ZooKeeper
- Job submission and bob status query;
    - Assemble a MPacket containing query information
    - Send to JobTracker and wait for response
    - Return response to user

JobTracker (extends PBArchitechture)

- Primary and Backup JobTracker
- Setup/Init
    - Create an EPHEMERAL znode /JobTracker
        - If KeeperException.NodeExists is thrown => this tracker is the backup
            - Watch on Primary => try to become Primary if current Primary fails
        - If no exception => this tracker is Primary
            - Create server socket and wait for client connection
- Accepting Job submission/query
    - Get md5 hash and job/status request from user
    - Job: need to create subtasks for workers
        - Create PERSISTENT znodes /Jobs/<md5>_<partition> for each of the 27 partitions (from 0 to 26). Each partition contains ~10k words from the dictionary that workers must fetch from FileServer. Each znode is created with data field set to “CREATED”
        - If the requested nodes exist, a previous job of this hash had been submitted, we can safetly return to user at this point
    - Status: need to query the status of a previously submitted job
        - For each subtask under /Jobs, check the status of the ones which match the given md5 hash. If any one of the subtasks has an answer, return the answer to user, otherwise report progress (<subtask_with_no_match> / <total_number_of_subtasks>)
- On Primary failure
    - Since Backup has a Watcher set on the EPHEMERAL znode /JobTracker, a callback handler can then process the WatchedEvent and take over as Primary

Worker (extends PBArchitechtureAdaptor)

- All workers monitor other workers for failure
- Any number of workers can work concurrently simply by spawning additional instances
- Setup
    - Worker creates an EPHEMERAL_SEQUENTIAL node under /Workers/ for itself
    - Worker uses getChildren() on /Workers/ and leaves a Watcher. This list of workers is saved for failure detection
- Get task
    - Worker finds a job under /Jobs with data “CREATED”, job data is then set to this worker’s path using atomic test-and-set provided by zookeeper through version number in setData()
    - Continue if success, otherwise look for another task
    - Worker contacts FileServer to get the dictionary partition using partitionID found in task path /Jobs/<md5>_<partitionID>
- Crack MD5 Hash
    - Brute-force check all words in dictionary against the target (md5 hash from task path)

        - Update task's data with one of "FOUND_<password>" or "NOT_FOUND" using atomic test-and-set
- On Worker failure (any worker failure)
    - Handler queries for a list of current workers, any worker missing from this list (compared to previous worker list) is considered dead
    - Worker checks every task under /Jobs and sets those with data set to dead worker's path to "CREATED" - this is to clean up unfinished tasks.

FileServer (extends PBArchitechture)

- Similar to JobTracker, creates an EPHEMERAL node under /FileServer.
- Setup:
    - The dictionary is read into memory for later processing
- Offer dictionary partition:
    - Waits for a worker to connect and retrieves partitionID
    - Returns a slice of the dictionary according to this formula:
    - `[partitionID * partitionSize, min((partitionID + 1) * partitionSize, dictionarySize)]`
- On Primary failure: similarly handled as JobTracker