Documentation of Distributed Systems Project 1

Team members: Yichen Qiu, Guo Yu

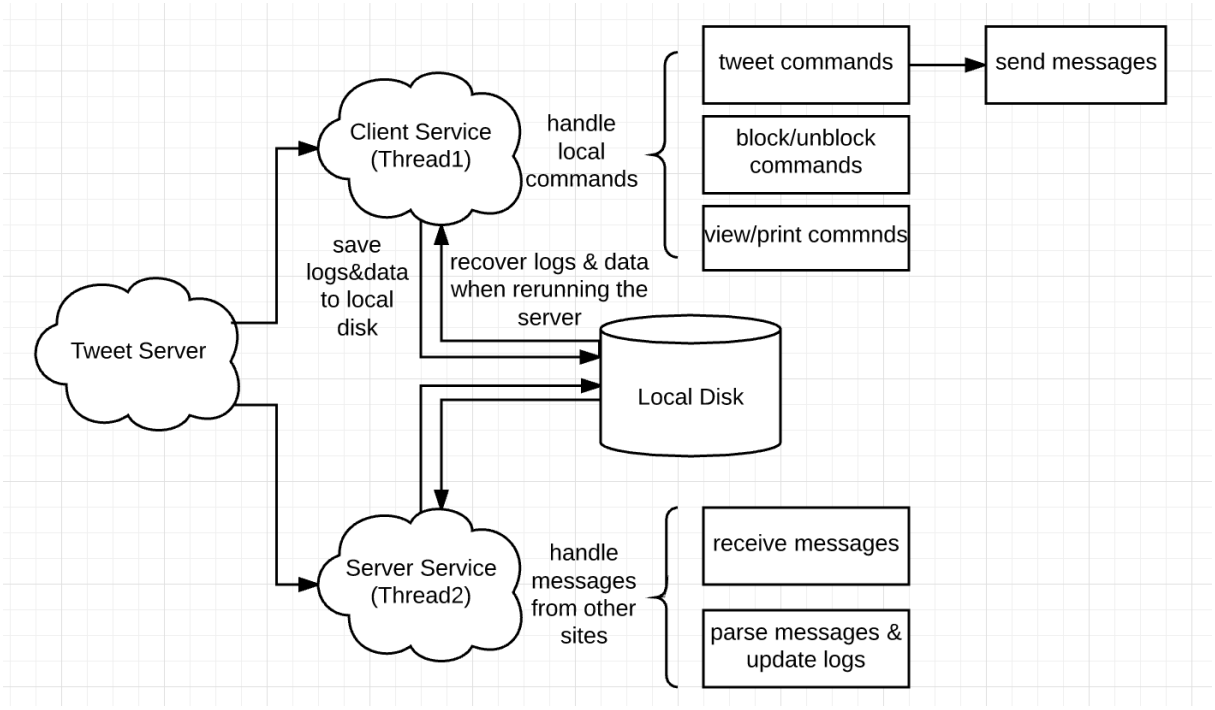
Oct. 15, 2017

- **Project Description**

This project is to implement a tweet-like server that users can post tweets, view tweets from all the other users, and block/unblock other users. Besides, this server also maintains the log and dictionary of all sites, using Wuu-Bernstein Replicated Log Algorithm. This server also survives after it crashes by storing the log and data timely to local disks.

- **System Design**

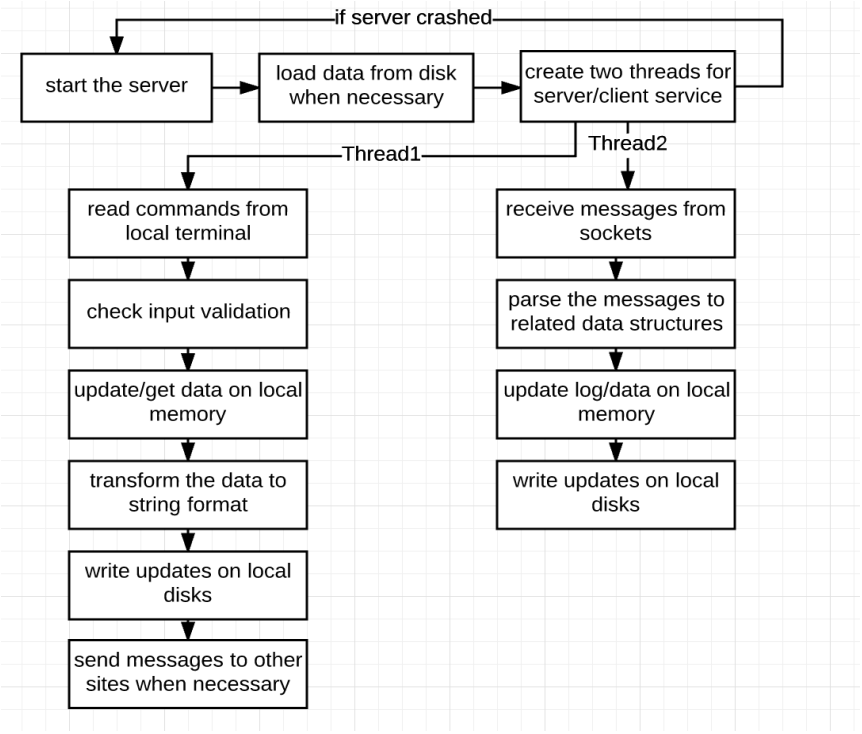
This Tweet Server is written in one Python file, and uses two threads to handle server and client services separately. These two threads update the global variables (log, virtual clock, etc.) on the local memory together, and timely write updates onto local disk after making changes on local memory. The client service is in charge of sending messages to other sites, while server service listening on the socket to receive the messages from other sites.



- **Workflow of Program**

The workflow of Tweet Server is as follows:

1. start the program
2. read the data and configuration files from disk
3. create two threads to handle server and client services
4. update the data in memory according to local commands and received messages
5. transform the data in memory to string format
6. write updates on the local disk
7. send messages to other sites when necessary



- **How to avoid duplicate inserts?**

The duplicate inserts only happen during the block/unblock commands. Since the duplicate block means the same site blocks another site twice, it is related to client service. As is shown in Workflow diagram, we will check the validation of block/unblock commands, if it blocks a site already in the block dictionary, we will output the errors on the terminal.

Besides, we maintain the block dictionary on all sites, every time the attempt to insert duplicate blocks will be forbidden. And when other sites receive the messages about the block and unblock event, it will compare the events timestamp to see if it has already receive the messages. If it has received the messages before, then no action will take, the duplicate message will be filtered out. This, from another respect, to help maintain the block dictionary and keep its consistency. So the [block user], [block user], [unblock user] series of events will function exactly like [block user], [unblock user].

- **Why the changes you made still solves the log and dictionary problem?**

One change we made on the algorithm is that we will modify the time of events from other sites in different time zones to the time the same as the time zone of local site. In this way, when users view the tweets, server will sort the tweets according to their modified timestamp, which solves the problem of sorting the timestamps in different time zones.

It still solves the log and dictionary problem, because when maintaining the log, we only use the virtual clock on every site that automatically increments 1 after every event. The actual timestamp is only used when viewing the tweets, which has no influence on the log algorithm.

- **File Recovery**

Whenever there is an update on in the partial log, the time matrix, we will dump those new changes to the local file in case there will be a crash. Initially the site will read all the initial states

from the files every time the site start up. If no such files exist, then we just use the default value for related variables.