Design of Twitter Proxy Herd Using Twisted in Python

Jingyuan Fan
University of California, Los Angeles

Abstract

In this paper, I examined the feasibility of building Twitter proxy herd using Twisted in Python in detail. Implementation of the server is explicitly discussed. Specifically, servers are able to keep trying to reconnect with their neighbors when the connections are lost or failed. Also, I briefly compared the overall approach of Twisted to that of Node.js, and concluded that Twisted is a suitable choice for implementing Twitter proxy herd.

1. Introduction

Wikimedia is the platform on which Wikipedia, Wiktionary, and the other seven wiki dwarfs are built on. It essentially uses a LAMP platform based on dedicated clusters of Linux servers, including database servers running MySQL, web servers running the Apache HTTP server and Squid cache servers. While this architecture works fairly well for Wikipedia, it may still encounter some problems if (1) updates to articles will happen far more often, (2) access will be required via various protocols, not just HTTP, and (3) clients will tend to be more mobile. All in all, the application server is a central bottleneck.

The possible solution to the problem is to use "application server herd" instead of LAMP, where multiple servers can communicate with each other as well as core servers and databases under certain rules in order to improve the efficiency. So rapidly-evolving data, such as GPS information about a user, can be shared among servers without talking to central database while tweets can be retrieved through database server communication. To support such method, one can use a thread-based framework, where one thread is built for each connection. However, threads may result in other problems, like race condition, deadlocks, which all need to be carefully designed and implemented.

In this project, we chose to use event-driven networking framework, Twisted, where the control flow of the program is determined by event. In this paper, I describe steps taken towards the implementation of the Twisted Twitter proxy herd.

2. Twisted

Twisted is an event-driven network programming framework written in Python, and central to the Twisted application model is the concept of Deferred. A deferred object is designed to receive a result which hasn't been computed yet, and be passed around like a regular object. Once the deferred object gets the value, it is passed to a callback chain, which is composed of several callback or errback functions, with the result of each function becoming the input of next function, and whether to execute a callback function or an errback function depends on the output of the previous function. Not like multithread-based framework, deferred object doesn't block the execution, so CPU resources can be saved.

3. Prototype

In the prototype, it consists of five servers communicating with each other with certain patterns. It should also support three types of commands received from users or other servers. I would discuss a few main functions and classes. To execute the code, you just need to run bash servername(Hill, Gasol, Young, Meeks and Farmar).bash five times to start five servers.

3.1. ChatFactory

This class is used to build servers, each of which will have a set of attributes that describe server name, the set of active users, the set of its peers, and the host and port the server uses. When starting each server, it will generate a directory using the server name to store all the log information. One log file and one error log file with the time documented will be generated each time the server starts.

3.2. Chat

This class is to handle all the information that one server can possibly receive. connectionMade and connectionLost are all overriden methods, and simply log info with the new client information (Transport layer protocol, IP address, port number) and lost reason respectively. lineReceived function takes the input to the server as an argument, parse it, and calls the event handlers according to the first non-white-space word.

3.3. handle_IAMAT

When receiving a message of format: IAMAT kiwi.cs.ucla.edu +27.5916+086.5640 1353118103.108893381, this function is triggered. It will first parse the function and checks if the argument number is correct. The current time is returned using standard library function datetime.datetime.utcnow(), and time difference is also calculated. Then the server will store this user information and reply to the user with AT as the prefix, appended with the server name, the time difference and the original message. Then the server will generate the message for other servers, and send it using a simple flooding algorithm implemented in a for loop. This message is similar to the message the server sends back to the user, but has a current server name as a suffix which is used in handle_AT function. Also, if the connection cannot be established, the server will keep trying to reconnect, and I will discuss how I implement it in later part.

3.4. handle_WHATSAT

This function is used when the server receive a message requiring tweets from users. If the required number of tweets beyond the limit, it will generate an error info. Also if the user has not previously tried to connect with the server using IAMAT message, the server would also give a warning and return. It should be noted that the Twitty Twitter is not supported by the current Twitter API, so I just hardcoded the return tweets using the example given in the assignment.

3.5. handle_AT

A server would receive AT message from other servers trying to flood user information. It first check if the server has this user information, if not, the user's information will be stored and then flooded to other servers excluding the server which the user initially contacted and the server which sends the information to the current server. When flooding the message, the server also needs to change the last argument in the message to the current server's name. If the server had heard from this user, it will check the new message timestamp with the previously message's timestamp to decide whether it's a new message, and it won't update the information if this message is outdated. Also the flooding will keep connecting its neighbor servers until the connection is made.

3.6. connecttcp

This function is for realizing reconnecting if connection hasn't been made at the first place. To implementing this function, I added a callback and an errback. If the connection is successfully made, it will call flood_to_peer function which will flood the message to servers which are determined in handle_IAMAT and handle_AT. It the connection is failed or lost, it will call the errback and recursively call connecttcp trying to establish the connection. Once the connection is made, callback function is called and then returns.

4. Twisted: pros and cons

Twisted is written in Python, so shares all its language features.

First, Python dynamically infer the types of objects instead of forcing you to define them, so this gives programmer flexibility to use a variable at will when required. In this project, this dynamical typing property makes command parse much easier and saves me from writing extra lines of code, and for information I don't need, I don't even need to care about them.

Second, Twisted model makes excellent use of a single-thread, so the time spent waiting on making connection adds up significantly, while the time could be spent doing other things. The alternative, multi-thread based frameworks is much more performant for CPU intensive tasks. Multi-thread code is also much harder to reason about. There are issues with locking, synchronization, and other fun concurrency problems. Single-thread asynchrony doesn't suffer from the same problems.

However, the method python uses for memory management puts a lot burden on users to avoid cyclic reference and reallocate memory when needed. And the reference counting method used by Python would not return memory to operating system. Instead it holds on to it in case it's used later; so eventually it may cost memory leak. Also dynamical typing makes code harder to read and understand.

5. Comparison with Node.js

Twisted and Node.js are all using event-driven frameworks, but after doing research, there're still some differences between them. Twisted supports a wide range of protocols compared to Node.js which is primarily HTTP. However, Node.js runs faster than Python, and reason is that Node.js concentrates on low-level implementation. Therefore Node.js definitely is a suitable alternative, but I still believe Twisted is easier for implementation.

Reference

[1]https://journal.paul.querna.org/articles/2011/12/18/the-switch-python-to-node-js/
[2]http://en.wikipedia.org/wiki/Wikimedia_Foundation
[3]http://stackoverflow.com/questions/3461549/what-are-the-use-cases-of-node-js-vs-twisted
[4]http://twistedmatrix.com/documents/13.0.0/core/howto/defer.html
[5]http://nodeguide.com/beginner.html
[6]http://stackoverflow.com/questions/6357737/twisted-or-celery-which-is-right-for-my-application-with-lots-of-soap-calls
[7]http://www.quora.com/What-are-the-benefits-of-developing-in-node-js-versus-Python