Twisted Places Proxy Herd

Chang Liu, *University of California, Los Angeles*

Abstract

Twisted is an event-driven framework built on top of Python and in this project, I am analyzing its viability as a replacement for the Wikimedia platform if it implements an application server herd.

1. Introduction

Wikipedia and its various related sites are based on the Wikimedia Architecture, which uses the LAMP stack, consisting of Linux, Apache, MySQL and PHP. Multiple redundant web servers are set-up behind a load-balancing virtual router in this setup. However, we want to build a new platform designed around news, thus requiring access via various protocols, more servers to support clients on mobile devices, and an architecture to support frequent updates to articles.

2. Twisted Framework

2.1. Python Overview

Before detailing the Twisted framework, we can first briefly talk about Python, the language on which Twisted is built on. Python is a very popular, high-level programming language. It is an interpreted and dynamic language, like Javascript, and supports multiple programming paradigms, like object oriented code, imperative programming as well as functional programming. As of writing this, Python sits at 5th place among the most popular languages in the TIOBE index. For this project, I will be using Python 2.7.10.

2.2. Twisted Overview

Twisted is a networking engine that is written in Python. It supports numerous protocols and contains many functionalities like a web server, chat client, chat server, mail server and more.

Instead of a single-threaded or multithreaded programming model, Twisted follows an event-driven asynchronous model. In this type of program, the flow is determined by external events. The model combines aspects of both single threaded and multithreaded programming; it can handle multiple tasks in parallel while only using a single thread. The way it does this is through an event loop that uses callbacks to trigger actions when events happen. At the core of the Twisted framework lies the reactor, which handles all this. The reactor is responsible for polling for events and dispatching them to waiting event handlers.

Twisted achieves communication through transports, which represent the connection between two endpoints. Transports can write data to the physical connection in a non-blocking manner, conforming to Twisted's asynchronous manner. Twisted protocols then handle how these events are responded to and made in the first place. A new protocol is created for every connection made and terminated when the connection closes so Twisted must store its persistent configuration data in protocol factories.

3. Implementation

3.1. Setup

I started by creating a dictionary to hold all the server information — port numbers and each server's neighbors (to implement the flooding algorithm). I then separated the ProxyHerd application into a client and server class. These inherited from the ClientFactory and ServerFactory classes respectively to keep persistent data. The factories instantiate a protocol for every connection.

Meanwhile, the protocols inherit from LineReceiver, allowing me to override the linesReceived method, which triggers when a lines are sent over the connection. This way, I could parse the user input and handle IAMAT, WHATSAT, and AT commands.

3.2 handle_IAMAT

This handler first verifies that the command is a valid IAMAT command by checking the number of arguments. It then calculates the time difference and formats the response, which is then stored along with the client time in an array that's local to the factory of the current server handling the request. It then writes the response to the connection and calls the flood function to update the other servers.

The flood function basically treats the ProxyHerdServer as a ProxyHerdClient and connects to the other servers by consulting the dictionary, which has information on the reachable neighbors.

3.3 handle_WHATSAT

Like the other handler, this one also starts by validating the command. In addition, it performs a bounds check

the radius and limit parameter so it doesn't cause an error during the API call to Google Places.

Next the function retrieves the client's message cached in local storage and parses the location parameter. An API request URL is then constructed from the location and using Python's urllib2 module, we can send an HTTP request to the Google Places API. The results are parsed using Python's JSON module and limited to a certain amount if needed. The final result is then written across the transport. No flooding is required.

3.4 handle_AT

Again, this checks the number of arguments passed into the command to validate it. This function is never triggered from the client side, only in server-side to update the location of clients when the flooding algorithm runs.

To prevent an infinite loop, this function checks the timestamp if there's already an entry for the current client in the cache and returns if there is. Or else, it adds a new entry in the cache and continues flooding the rest of the servers.

3.4 Execution

To run this, a simple call to *python proxyherd.py* would do the trick. In main, the function loops through each server and starts it.

4. Twisted Pros / Cons

Given that Twisted is written in Python, the syntax is very concise and straightforward. If you look at the example codes given on the Twisted site, you can see that a simple chat client can be written in about 50 lines of code. The API documentation is also well written and easy to consult if you need to add some functionalities.

I personally have done a lot of Javascript work using Node.js so I am familiar with asynchronous programming and callbacks. The difficult part was figuring out the syntax of Twisted and which methods to call as well as how to implement the flooding algorithm.

5. Twisted vs. Node.js

Node.js is another asynchronous, event driven framework built on Google Chrome's V8 Javascript Engine. It also has non-blocking I/O, and given these properties, it is very similar to Twisted. Twisted has been around longer as a framework but Node has recently gained a lot of traction in companies who like it for its better performance (being on the V8 JS Engine) and scalability. Javascript is 8th on the TIOBE index so if you have previous knowledge of Javascript, you can pick up Node.js as well.

Being in Javascript, Node.js is already geared towards web development so HTTP is the most optimized protocol for it. While in Twisted, you must explicitly run a command to start the event loop, Node.js masks the event loop call; instead it automatically enters the event loop after the input script and stops once there are no more callbacks to perform.

As mentioned before, I've personally coded some projects using Node.js in the stack. In my experience, Node.js is slightly more wordy; it seems to me that there is less abstraction to API calls than in Twisted.

6. Conclusion

Having programmed in both frameworks, Twisted and Node.js are very similar in implementation. Both have their advantages and downsides. Twisted is a more mature framework, so it might be more stable if you're trying to implement certain functionality but being on the Chrome Engine, Node.js is said to outperform the former. However, given the purpose of this project to find a replacement for the LAMP stack to implement an application server herd, both frameworks would be very suitable to the job.

References

[1] McKellar, J. & Fettig A., "Twisted Programming Essentials," http://www.reedbushey.com/79Twisted%20Network%20Programming%20Essentials%202nd%20Edition.pdf (March 4th, 2016)

[2] Wikipedia, "Python (programming language)"https://en.wikipedia.org/wiki/Python_(programming_language) (March 4th, 2016)

[3] "About Node.js" https://nodejs.org/en/about/ (March 4th, 2016)