Twisted Places proxy herd

Ryan Trihernawan
University of California, Los Angeles
May 25, 2015

Abstract

Twisted is a networking library written in Python. It is one of the candidates for implementing the application server herd. This paper discusses how Twisted is used and whether it is suitable for this kind of application in terms of its simplicity, ease of implementation, reliability, and maintainability.

1. Introduction

Linux, Apache, MySQL, PHP (LAMP) platform is commonly used as the basic architecture for web services. Although this architecture is sufficient for most web services, it does not perform well under certain situations where there are frequent updates to the server, access via various protocols, and mobile clients who deal with unstable connections from time to time. "Application server herd" is one possible solution to such problem. This architecture involves multiple application servers that communicate with one another as well as the core database and caches to avoid the bottleneck that a single application server in LAMP causes. Clients can connect to any of the application servers and thus the servers can share their workload and decrease the latency between them and the clients. This architecture can be simulated using Twisted, which implements event-driven programming.

2. Twisted

2.1. Overview

Twisted is an event-driven networking framework. It is written in Python, which is cross-platform and well known for its simplicity and ease of use. These properties allow Python to accelerate learning and testing especially for beginners. This framework is also asynchronous and event-based. Such implementation allows for a single-threaded application that processes multiple network communications. Twisted's myriad of features further allow for quick implementation of applications and frequent experiments and tests. Programmers do not have to reinvent the wheel to implement the basic features that form as the backbone of web architecture. These are just a few of the advantages of Twisted compared to other networking frameworks.

2.2. Syntax

Twisted is written in Python and thus even programmers with no or little Python programming experience can easily and quickly learn it using a wealth of printed and online tutorials. A simple server can be implemented in less than 50 lines of code (LOC). The application server herd prototype is implemented in less than 215 LOC. Such implementation is possible due to the rich set of features that Twisted offer that lets the programmers to focus on creating a set of protocols that the servers have to obey.

3. Implementation

3.1. Overview

The code is divided into 4 different classes (ProxyServer, ProxyServerFactory, ProxyClient, and ProxyClientFactory). ProxyServerFactory creates a factory that instantiates the protocol for every connection. ProxyServer defines what the server should do in response to particular events. This protocol describes how the server parses, executes, and replies client's command and another server's location update, and communicates with Google Places using the required API.

ProxyClientFactory creates a factory that instantiates the protocol for every connection as well. ProxyClient describes how the server sends location update to each of the server it is connected to.

3.2. Protocol

LineReceiver protocol is used to deal with incoming commands. It dispatches LineReceived method for each line received. Each command is one line long and thus it is parsed into a list of tokens. The first token is expected to be the command. For each valid command, it is known beforehand the number of arguments required and thus invalid command can easily be detected by checking the first token and the length of the arguments. Each valid command is then passed to command-specific handler.

3.3. AT Handler

This handler deals with the AT command. The server receives this command from other servers it is connected to. Since this command is also a location update, the server only updates its copy of the location update if and only if the client's timestamp in the incoming location update is greater than the client's timestamp in its own copy. This mechanism prevents flooding of the location updates since updating between servers is guaranteed to terminate at some point. If this location update is acceptable as described above, the server sends its new copy of the location update to the servers it is connected to.

3.4. IAMAT Handler

This handler deals with the IAMAT command. The server receives this command from the client.. Before storing the location info to its database (in this simple program it is a Python dictionary), it calculates the time difference between the server's timestamp and the client's timestamp. Then the server replies with the AT command and sends the location info to the servers it is connected to.

3.5. WHATSAT Handler

This handler deals with the WHATSAT command. The server receives this command from the client. The server ignores the command if the client does not exist. Otherwise, it sends API request to Google Places, appends the formatted response to the AT command, and returns this new response to the client.

3.6. Logging

Python logging module is used for logging. ProxyServerFactory instantiates and configures the log file with the filename with the following format: server name, date (Y-M-D), and time. Important events such as connection setup, input, output, Google Places API request, etc. are logged.

3.7. Execution

The program server.py is supplied the server name to instantiate the server as follows:
python server.py <server name>

3.8. Testing

Two Shell scripts are used to implement the testing. createservers.sh calls the Python server.py program to create five servers (Alford, Bolden, Hamilton, Parker, Powell). testservers.sh creates five different clients by connecting each client to each one of the servers. The first three clients send IAMAT commands to the servers they are connected to. The last two clients send WHATSAT commands to the servers they are connected to. These commands simply check if the location updates from the first three clients are actually propagated to the other two servers. Lastly, the servers are killed to terminate the program. The log files describe in detail what each server does throughout its execution. They also show what each server sends to the servers it is connected to. The servers' communication footprints explain how the location updates propagate to the servers that do not receive updates directly from their clients.

4. Benefits and Drawbacks

Twisted is suitable for the kind of application tested mainly because it is asynchronous and event-based. The server can do useful work continuously without wasting its time waiting for expensive operations to complete because a callback is registered with an event loop. The callback describes what operations to do after the executing event completes. This interleaving of executions allows the program to use a single thread. This benefit allows for simplicity and ease of implementation. Moreover, since Twisted is written in Python, it also benefits from the simplicity and portability of Python, which makes it easy to learn the language and the framework quickly.

5. Node.js Comparison

Node.js is a networking framework similar to Twisted. It is written mainly in JavaScript. Like Twisted, it is asynchronous and event-based. It is not as mature as Twisted as it was initially released several years after Twisted was released. However, in the recent years it has gained popularity and growth. Running on top of the V8 JavaScript Engine allows Node.js to have a very good performance. The close relationship between JavaScript and JSON on the web gives Node.js another advantage for working with web architecture. However, it is not suitable for data-intensive computing since it runs on a single thread. The non-blocking I/O does not help in this case since the computation becomes the bottleneck. Both Twisted and Node.js provide similar functionalities needed to implement application server herd and thus the decision to choose the framework is largely determined by the programmer's experience and preference with the corresponding language.

6. Conclusion

Twisted is suitable for applications that have frequent updates to the servers because it handles the events asynchronously. Moreover, it interleaves executions to keep the utilization high and thus it increases the throughput and decreases the latency between the server and the client. Its rich documentation and mature source code let programmers quickly write and run a server. Since it is written in Python, it also inherits the benefits that Python offers. The simplicity, ease of implementation, and cost-efficiency that Twisted offers are the main reasons why it is a reasonable choice for implementing application server herd. Node.js provides a similar functionality as Twisted and thus it is up to the programmers to decide what framework to choose based on their experience and preference with the languages the frameworks are written in.

References

[1] Jessica McKellar, and Abe Fettig, *Twisted network programming essentials*, 2nd edition, O'Reilly, 2013, ISBN 9781449326111

[2] Ben Wen, *6 things you should know about Node.js*, Java World, http://www.javaworld.com/article/2079190/scripting-jvm-languages/6-things-you-should-know-about-node-js.html, (Accessed: 24 May 2015)