Wonderful : A Terrific Application and Fascinating Paper

Alexander Hesselgrave (904273474)
UCLA

Abstract

Twisted is an event-driven networking framework written in Python 2.7. This report examines the ease of development and pros and cons of Twisted over new and traditional frameworks.

1 Introduction

LAMP (Linux, Apache, MySQL, PHP) is a very popular web server platform that several sites and companies, such as Wikipedia, rely on. It uses multiple, redundant web servers behind a load-balancing virtual router, which works well for reliability and performance. However, Wikipedia is relatively static for its size and we want to be able to implement a new architecture that updates frequently, uses more protocols other than just HTTP, and supports more mobile clients. To beat this, we implemented an application server herd in Twisted. That is, we have a herd of servers that intercommunicate each other whenever there is an update without each one having to query the database.

2 Twisted

2.1 Overview

Twisted is an event-driven networking engine written in Python that supports multiple networking protocols beyond HTTP, such as SMTP, POP3, and DNS. Twisted is a very reliable framework for a backend for several reasons. For one, it is incredibly secure; being a high level language, it is able to overcome buffer overflow attacks that traditional C servers suffer from. Python's ease of error and exception handling allows Twisted servers to be robust and stable.

2.2 Implementation Features

Twisted is a single-threaded framework that is still able to run quickly without sacrificing speed. Since Twisted is event-based and not redundant like LAMP, there is no need to spawn more threads for more clients. The lack of multithreading also makes for much simpler, safer, and faster code to be used and developed.

Twisted uses a reactor-based event loop as its core for events. The reactor blocks until a relevant event arrives and "dispatches the event" by calling the appropriate event handler. The framework has support for several services and protocols and allows for custom callbacks. The single-threaded approach also keeps the event loop tightly knit and reliable since there is no need to worry about races or context-switching.

3 Server Herd Implementation

See `http://web.cs.ucla.edu/classes/winter16/cs131/hw/pr.html` for the specification. This section will discuss reasons for design choices as well as complications in those choices as well as the framework and language itself.

3.1 Design Overview

Most of the code is inherited from twisted.protocols.basic.LineReceiver and twisted.internet.protocol.ServerFactory. The line receiver handles sending and receiving plaintext lines on TCP, and the ServerFactory handles initiating connections, managing clients, and building the line receivers. My implementation has two factory-receiver pairs, one for client-server communication (Chatserver) and one for server herd intercommunication (Interserver). When the line receiver gets a line, it tokenizes the string and calls the appropriate handler based on the first token.

3.2 IAMAT Handler

The first of the handlers, IAMAT serves the purpose of setting up the interserver knowledge base as well as being the stepping stone for the flooding data propogation algorithm. When a server receives a valid IAMAT message, it will first get a timestamp difference between the timestamp the client sent and the current server time. It will then generate an AT message as per the spec and store that in its factory's client dictionary with the client ID as the key and the AT message as the value. It will then use the Interserver protocol to send that AT message to its neighbors that it is allowed to talk to, thus starting the flooding and communicating the client's location to all connected servers.

3.3 AT Handler

The AT handler is only called and used for interserver communication as part of the flooding algorithm. When a server receives an AT message, it will check the timestamp from the sent message. Since the interserver communication is two-way, our connection graph is cyclic and can run into an infinite loop. To solve that problem, the server checks if (1) it has a message from that client ID already, and if it does, (2) it will compare message timestamps from its cache and the new message. If the sent message isn't a new one, it will stop propogating the message.

3.4 WHATSAT Handler

The WHATSAT handler is fairly straightforward. A client can only call it after the client submits an IAMAT with proper GPS location. Once that is done, the client can query for a number of places within a variable kilometer radius of their position using the Google Places API.

3.5 Logging

The server herd uses the built-in Python logging library to log to a file. Each server has its own logfile in the format of "servername"-"file_timestamp".log. The log documents state changes and event triggers as well as exceptions and syntax errors.

3.6 Error Handling

Due to this implementation's reliance on lists and dictionaries, I have written several try-except blocks for possible KeyErrors and IndexErrors in case of out-of-bounds errors or nonexistent keys in the client dictionary.

3.7 Development Pros and Cons

Python is an incredibly simple and straightforward to develop in, and that mentality and feel is extended into the Twisted library. The API is very clear on what needs to be implemented and how TCP connections are handled in the event-based wrapper. However, one issue that gave me a hard-to-track bug for a long time is how Python 2.7 handles references from variables to packages. I accidentally changed a reference to the json package to a local variable in one of my functions, and instead of throwing an error when I tried to call a json method, my code failed silently and refused to log; it seemed like the function I called just popped off the stack. Although in hindsight this was easy to avoid, developers should be aware that reference changes can be dangerous and incredibly difficult to debug.

4 Node.js vs Twisted

Node.js is another new, event-based framework for server applications written largely in the typically frontend Javascript. Like the language it was based on, Node.js is built on an event-based, modular paradigm that makes it easy to write state and handler callbacks along with managing TCP connections. Easily complementing Node's popularity is its package manager npm. Analogous to pip for Python, npm allows developers to quickly install and manage libraries and packages to quickly and reliably extend a server implementation with a single line. Unlike Twisted, Node natively supports JSON objects which makes calculations and serialization much easier to process over the Python json library. However, Twisted and Python are able to run actual computations much faster than a Javascript engine.

5 Conclusion

Twisted is an older, tried and true framework that is incredibly simple to develop in an extend using Python's natural features and paradigms. Other popular event-based frameworks exist like Node.js that are even easier to write callbacks for and serialize data in, but at the cost of taking longer for computations. Since this server herd does not do much calculation and instead calls on other APIs for the hard work, it is very feasible to migrate the design logic to a Node framework. Ultimately, both frameworks are simple, reliable, and fast; the decision can't be made from a prototype alone. It depends on what else the server herd needs to perform, but hopefully now the reader is aware of implications and benefits of both frameworks.