Semantics of a Twisted Places proxy herd

Eric Sehun Oh
University of California, Los Angeles
Computer Science 131, Fall 2016

Abstract

Twisted is an event-driven networking engine written in Python, which allows for easy implementation for custom network applications. This paper will delve into implementations of an architecture called an application server herd, in which multiple application servers communicate with one another via protocols, the core database, and caches, and briefly compare event-driven programming approach with that of Node.js'.

1 Introduction

Wikipedia and related sites utilizes the Wikimedia Architecture, which uses a LAMP platform based on Linux, Apache, MySQL, and PHP all of which uses multiple, redundant web servers along with a load-balancing virtual router in order to maintain reliability and performance.

Although LAMP's platform has served Wikipedia well for its needs, the same does not apply for other types of services. When updates must occur more often, accesses must support various protocols other than HTTP, and mobility of clients must be prominent, LAMP's platform falls short as the Wikimedia application server becomes a central bottleneck consequently increasing response time to an unacceptable degree.

In order to find a more optimized and fitting architecture to meet such needs, a different architecture known as application server herd where multiple application servers communicate with another via protocols, database, and caches will be investigated in this paper.

2 Twisted

Twisted is a framework for writing asynchronous, event-driven networking framework that is implemented in Python, including support for abundant networking and communication protocols such as UDP, SSL/TLS, HTTP, TCP, and many others. For the test implementation, TCP will be used.

3 Implementation

3.1 Overview

There are 5 servers in the server herd named Alford, Ball, Hamilton, Holiday, and Welsh. Each one of these servers are instantiated with the class ServerFactory which is subclassed from `twisted.internet.protocol.Factory`. ServerFactory creates a factory that instantiates the protocol, ServerProtocol, for every connection.

3.2 Protocols

There are two protocols at work in the implementation, one each for the client and the server. They are subclasses of the LineReceiver protocol from `twisted.protocols.basic` which is a protocol that receives lines and/or raw data. For the implementation, lines are passed between servers and thus is used in line mode. In line mode, each line that's received becomes a callback to lineReceived.

ClientProtocol will send a message and then close itself. The ServerProtocol is invoked during inter-server communication. When receiving a line it will check to see the kind of command it must invoke by reading the first word of the line and will call the appropriate line handler if it is a valid message, if not it will invoke the `bad_request` function which will mark the message as an invalid command and send back a question mark.

3.3 IAMAT Handler

The IAMAT handler will parse the line received to tokens to extract information about the name, location, and time, which the server will then pass on to its neighboring servers through by a simple flood algorithm and will also reply to the sender a response to inform that the line has been received.

3.4 AT Handler

The AT handler is responsible for processing the line received from other servers and update the information according to the user. The message will be parsed to extract information about the user and the timestamp, comparing it to the stored timestamp of the receiving server to ensure that the information it currently has stored is coherent.

3.2.3 WHATSAT Handler

The WHATAT handler is responsible for processing and handling clients' request by parsing the line to extract information regarding the name and location of the client's query and will fetch the nearby geo-location information of the place specified. This is done through the Google Places API by sending an API request and will the send the response to the client.

4 Twisted, pro and cons

One big benefit of using Twisted is it has great support as it is written in Python and has been around for a very long time. For writing low-level networking services, it is very suitable, offering numerous protocols such as HTTP, POP3, TCP, SSL, SSH, etc. However, for web applications Node.js would be a much better fit.

Additionally, Twisted makes great use of a single thread, being less memory intensive than the multi-threaded alternative, which allows for better scalability.

5 Node.js Comparison

Node.js is a JavaScript runtime environment that has an event-driven architecture that supports asynchronous I/O with callbacks, a concept also in Twisted. Node.js seeks to solve the problems of concurrency in server-side programming languages and improve performance.

Node.js although much newer when compared to Twisted is significantly faster as it was developed specifically with the asynchronous nature in mind while Twisted offers async features and runs above python. Node.js is said to be easier to learn and also is preferable to those who know JavaScript.

A difference between Node.js and Python is said to be in error handling. In Python there is a try-catch notion where errors could be thrown and caught. However, in Node.js the errors are passed around in callbacks called promises and thus does not include exceptions. More modern languages, along with Go, are taking a different approach to error handling and have started using promises. Newer does not necessarily mean better, however. Both Twisted and Node.js are acceptable choices for implementation.

References

[1] Zadka, Moshe. The Twisted Network Framework.http://legacy.python.org/workshops/2002-02/papers/09/index.html. 01 Dec 2016.

[2] Twisted Matrix Labs. http://twistedmatrix.com/documents/current/core/howto/servers.html

[3] Vickery, Gavin. Why I'm Switching from Python to NodeJS. 24 Oct 2014. https://blog.geekforbrains.com/why-im-switching-from-python-to-nodejs-1fbc17dc797a#.2iq6yi4e5. 01 Dec 2016

[4] Peticolas, Dave. Twisted Introduction. http://krondo.com/an-introduction-to-asynchronous-programming-and-twisted/. 01 Dec 2016