Project: Twisted Places Proxy Herd

Brooke Wenig
University of California, Los Angeles
Computer Science 131, Spring 2015
Professor Eggert

Abstract

Twisted is an event-driven networking framework that can be utilized to implement web servers. There are numerous platforms, programming languages, and environments to choose from for implementing web servers. At the time of this paper, two of the most popular include Javascripts Node.js and Pythons Twisted. This paper will analyze implementing an application server herd in Twisted, as well as compare its relative advantages and disadvantages when compared to other languages, such as Java, and other platforms such as Node.js.

1 Introduction

Wikipedia and its related sites (based on the Wikimedia Architecture) use a LAMP platform which includes numerous redundant web servers in order to increase reliability and performance. Although this approach works well for Wikipedia, it might not be suitable for all needs. For example, when there are many updates/edits/posts that are made to a website and clients are more mobile, such as in a news site, the application server will be a bottleneck. This is because the news changes frequently, clients frequently broadcast their GPS locations, and there will be more writes than reads to the database when compared to Wikipedia. Although adding more servers would inevitably help alleviate this problem, it is not the most economically feasible approach. However, there are many other viable approaches, and this paper will focus on Twisted and its viable competitors.

Twisted is an event-driven networking framework that will reduce this aforementioned problems with the LAMP platform. Based on the architecture of Twisted, we are able to implement an application server herd in which the application servers not only communicate with the core database and cashes, but also with each other directly. This approach minimizes the amount of requests to the core database needed. For example, to query for smaller, rapidly-changing data such as GPS locations, it is better to ask other servers instead of the main database because (1) it is faster (2) by the time a GPS location arrives to the main database, it is likely already stale data. However, more stable data that is accessed not as frequently will still live in the main database.

This paper will analyze the advantages and disadvantages of tackling this problem with Twisted, note other viable solutions, as well as examine data from a Twisted program that utilizes the Google Places API.

1.1 What is Python

Before we delve into what Twisted is, we will first examine Python - the programming language that Twisted is implemented in. Python is a high-level interpreted language which was developed in the 1980s. As of the time of this paper, it is on version 3.4. According to the TIOBE Index, in May 2015, Python is the 6th most popular programming language, with Javascript a close 7th. Python is an objected-oriented language which includes support for dynamic type checking, full garbage collector, multiple inheritance, and unicode, among many other features.

1.2 What is Twisted

Twisted is an event-driven networking framework that is implemented in Python. It includes support for numerous networking and communication protocols (not just HTTP), which are exposed as method-calls on Python objects. At the time of this writing, is is on stable release 15.1.0.

1.3 Motivation for this Project

See Introduction

2 Implementation

2.1 Approach to Implementation

StartServer: This module starts and initializes all five servers using Twisteds serverFactory: Alford, Bolden, Hamilton, Parker, and Powell. It assigns each server a dictionary for its clients and an ID. It in turn invokes commandProtocol for each server, which is responsible for calling the handlers below.

evalIMAT: This handler accepts a message from the client to the server regarding the clients location. It calculates the time difference between the server's time and the time the client said it sent the message. In turn, it broadcasts this message to the other servers updating them on that clients location.

evalWHATSAT: This module queries the Google Places API. It looks up a client in its dictionary that it wants to find the location for, and then inputs that information into the proper fields of the Google Places URL to execute and evaluate.

evalAT: Only servers can return a command with AT. This is returned following a call to WHATSAT to provide information about the requested client. It provides the most recent location reported by the client, the server that it talked to, and the amount of time it spent talking with that server. The AT response is followed by a JSON object.

sendMessage: This implements a flooding algorithm that broadcasts any updates in information across the servers. This way, the servers do not need to query for the information they need, but instead receive it from servers that they can talk to.

clientProtocol: In order for the servers to communicate information with each other, I made a client that simply passes information to a server, and then drops the connection.

2.2 Ease of Implementation

Overall, the most difficult part of this project was learning how to use Twisted. I had only used Python before in CS35L at UCLA, which provides a two-week introduction to Python, which I found sufficient for this project. I have done serious Javascript programming before, which definitely helped with understanding the relationship between servers and clients, calling APIs, and handling requests. I did not find Twisted particularly difficult to use, but it did have a steep learning curve. I essentially modified the example chat client and chat server from the Twisted website in order to implement this project. However, for those that have not done serious web programming before, there may be some roadblocks. For example, understanding calling APIs, or devising clever ways to prevent the same server from broadcasting the same message twice. The application server itself has fewer than 190 lines of code (LOC). Its solution is concise, yet elegant. In summary, writing Twisted-based programs that run and exploit server herds is relatively straightforward, provided that one is willing to invest the time in learning Twisted.

2.3 Shortcomings and Performance Implications

Although Twisted is able to counteract the concerns mentioned in the introduction, there are still some performance problems. For instance, because Twisted is not truly concurrent (see discussion below comparing Java and Python), it does not take advtange of multithreading. Further, there are definitely performance problems in my implementation, but that is due to my lack of experience in Twisted.

3 Comparison with other Frameworks

3.1 Java-based approach

3.1.1 What is Java

Java is another object-oriented programming languages, and was invented in 1995 by Sun Microsystems (now part of Oracle). As of May 2015, it is the most popular programming language. Its syntax is very similar to that of C and C++, but does not expose as low-level functionalities to the programmer. For example, although the programmer must allocate dynamic memory, Java includes a garbage collector to clean it up when it is no longer needed. Unlike C and C++, Java compiles into byte code which can be run on any Java virtual machine (JVM). Although this bytecode is then interpreted, it has just-in-time compilation in case there is a segment of code that is repeatedly being called, it will convert it to machine code in order to optimize the speed of the program.

3.1.2 Comparison - Type Checking, Multithreading, and Memory Management

Python is a high-level interpreted language with dynamic type checking, whereas Java is a compiled language with static type checking (though it compiles into JVM byte code, instead of machine code). The advantage of static checking is that errors and bugs are detected sooner, and do not permit the program to run if encounters errors. However, with languages such as Java that support static type checking, It is very difficult to implement a perfect static type checker. For instance, how does one not

give too much leeway as to what programs are permitted to run, without letting buggy programs run? Or conversely, not being too strict and not allowing valid programs to run. From my research, dynamic type checking is a better approach because although it might not catch silly bugs at the beginning, it does not make assumptions about the type/value that a variable can take on until runtime, and thus is more flexible.

Although Python has multithreading, it does not support true concurrency. In CPython, there is a global interpreter lock (GIL) which according to the Python wiki site: is a mutex that prevents multiple native threads from executing Python bytecodes at once. This is because CPythons memory management is not thread safe. This is starkly contrasted with Javas multithreading, which supports true concurrency because multiple threads are actively executing at the same time, locking only if a thread needs to write to a shared variable. Thus, there is a definite speed advantage using Javas multithreading approach. Java supports asynchronous calls, but there is no asynchronous primitive in Python.

However, one of the main advantages of Python is that it is very concise. On the downside though, fewer LOC for a programmer comes at an expense: Python is slower than Java.

3.2 Node.js

3.2.1 What is Node.js

Node.js is a lightweight Javascript framework for efficiently processing network I/O in a non-blocking, event-driven manner. It uses Googles open source V8 Javascript engine to execute code, and a large percentage of its basic modules are written in JavaScript.

3.2.2 Comparison

Instead of using threading, Node.js uses an event-driven approach that relies on callbacks to signal when a task completes. Node.js clearly outperforms Twisted in various aspects, namely because it is built on the Google V8 JavaScript engine which is incredibly fast and open-source. There are many optimizations available for the V8 engine, and because it is maintained by Google, it receives lots of resources and research into further improving it. Although Node.js is singlethreaded and is based on event-driven programming like Twisted, using the V8 engine makes a large performance difference, making Node.js very scalable and efficient.

Although Twisted is admittedly more mature than Node.js (Twisted was first released in 2002, whereas Node.js was released in 2009), Node.js has a very large developer base and is selected as the server-side platform of choice for many large corporations, such as Microsoft, Walmart, and SAP. It has also had time to learn the mistakes of the other server-side options that were out there, and improve upon them. Further, because Javascript is the language for front-end web development, it would be easier for web developers to just learn one language, Javascript, instead of learning Python for the backend and Javascript for the frontend. In summary, Node.js is faster and has more potential to grow and improve in comparison to Twisted.

4 Problems Encountered

All in all, this project went rather smoothly. Although it took 30+ hours to complete the Twisted project, the majority of the time was spent learning about how to use Twisted and finding the proper Python libraries to import. The time spent learning the syntax of Python was very minimal in comparison. Probably the most difficult part of the project was getting the servers to communicate with each other via a flooding algorithm. I had not implemented a flooding algorithm before, so it was the learning curve that I struggled with again.

5 Recommendation

Overall, I would recommend Twisted over Node.js only if someone is very familiar with Python and does not want to invest the time in learning Javascript. Javascript and Python share a lot of similarities, and Javascript would not be too difficult for the average Python programmer to pick up. Apart from language familiarity, I do not see any major reasons why one should pick Twisted over Node.js. Admittedly, I have a slight bias for Node.js because I have programmed in it before, it objectively outperforms Twisted in performance, which is the key problem we were tasked to solve in this project. In summary, an application server herd in Twisted or Node.js are both liable options to effectively address the database bottleneck problems mentioned in the Introduction, with Node.js outperforming Twisted.

6 References

Eggert, Paul. "Class Lectures." UCLA. Lecture.
Eggert, Paul. "Project. Twisted Places Proxy Herd." UCLA Computer Science. N.p., n.d. Web. 27 May 2015.
"Global Interpreter Lock." Python. N.p., 13 Jan. 2015. Web. 27 May 2015.
"TIOBE Software: The Coding Standards Company." TIOBE. N.p., n.d. Web. 27 May 2015.
"Twisted 15.2.1 : Python Package Index." Python. N.p.,

n.d. Web. 27 May 2015.