Analysis of Twisted in Implementing a Server Herd Versus Other Scripting Languages

Kacey Ryan
604047388
University of California, Los Angeles

Abstract

This research report will analyze the use of the Python web framework, Twisted, in order to implement a service similar to Wikimedia but with more mobility and more updates. To judge the appropriateness of Python and Twisted for this Application, they will be compared to the efforts of Java and Node.js. In the long run, the modularity, simplistic code, and ease of prototyping ensure that Twisted is a viable option for the Application presented.

1 Introduction

The Twisted framework is an event driven networking framework developed for Python. Its continually supported development has made it one of the most robust and easily accessible web frameworks to work with in Python. In analysis of the framework itself, I used the sample chatserver application and altered it to develop my own prototype of a server herd. The simplicity in design behind this server herd proves how easily Twisted can be utilized in implementing larger projects.

Based on the spec given, my prototype is designed to have five servers that accept connections from clients over TCP. Clients will then issue commands to a server node of the herd and the herd will propagate the command between the other servers via a flooding algorithm. I used this functionality to allow commands to provide a clients location and proceed to query for locations nearby using Google's Places API.

The potential of Twisted will be further analyzed through the comparing its features with those of similar frameworks and scripting languages to note the competitive attributes it has to offer. Also, I will analyze my implementation of the server herd in order to justify my overall acceptance of Twisted as a plausible solution to the task presented.

2 Implementing a Server Herd

My implementation of the server herd was a growth of the basic chat server application presented by Twisted. The application itself reveals the frameworks simplistic depth as well as its robustness when put to use. It is designed with five servers all related by a graph provided. This relation is implemented through a ServerFactory, which allows for quick addition of servers and organization of nodes information. Servers are constructed through TCPServer using their own specified ports.

The neighbor relationship is used to minimize errors and failures. In order to maintain these relationships, I have a flooding algorithm that upon a node update, creates a client that forwards and updating command to the nodes neighbors. The client is then disconnected creating a clean and fluid transfer of data between servers.

I implemented my own logging scheme using the Python logging module. All logs are sent to a file called serverherd.log. The format of this log file is upon an action, the server and the potential recipient of information. An action is classified as new connections, lost connections, incoming messages, outgoing messages, and forwarded messages via flooding.

Using the Google Places API I was able to make my server herd respond with JSON messages from a google search query. When a client sends an IAMAT command, it marks that client IDs location. Then once a WHATSAT command is used, it uses that location along with a provided radius and a number of responses parameter to send a query.

The query was implemented by first formatting the url, and then requesting a json load of this page. Once the page was gathered, I checked the number of requests was a viable amount and if not I sliced the array of results at the limit. The issue I had was getting this newly created json data back into the correct format. I ended up using json dump, however the output is not exactly the same due to the issues with the separators.

Overall, this application involved a simple design while using the Twisted framework. It is a very suitable way of developing this application as it was efficient, simple, well organized, scalable, and robust.

3 Python versus Java

While Python code is very versatile, java code is very rigid and stable. The contrast of these two languages is separated primarily by Java's integrity and safety, given that it is well written, and Python's simplicity and modularity. The coding style of these two languages is very opposing as Java has a deep and specific coding style that is difficult to write, but once written is very powerful. This can require more work to get a simple application completed. Python code is merely the opposite, its dynamic typing and robust nature lead to simple efficient designs but ones that become less specialized as they scale. [1]

Pythons Typing

The first thing, which should be analyzed, is Python's use of dynamic type or duck typing. This feature has many negative aspects attributed to it such as runtime errors constantly occurring due to the lack of static checking. Also, many errors could be harder to catch because of this typing style thus slowing down production and making code harder to debug and more difficult to write. In Java, these problems don't exist due to the fact that a Java programmer can merely look at his data types that he wrote to ensure they are the correct ones to be used. [2]

The positive of duck typing make Python a very powerful and useful language. The code immediately becomes more robust as code can be easily ported. Python can easily manipulate data within the code by changing data structures, switching containers, and using the tools of whatever data types are at hand. The lack of flexibility in Java allows for safer code as static type checking saves the programmer from runtime defects. Overall the two have conflicting methods of type definition but both can lead to effectively written code with its own advantages if used correctly.

Memory Management

The concern for memory management when creating the Wikimedia style application is a very minor concern at best. Wikimedia is not a real-time system so the default memory management methods of both languages should work equally well. The two languages have similar memory issues and seem to be viable to similar leaks in memory if garbage collection goes wrong. Overall I'd say this is an unnecessary concern and that Python's garbage collection should handle just fine. [3][4]

Multithreading

Python and Java both house the support for multithreading. The proposed application will not use multithreading on Python because of the issues behind Python's multithreading. Threads are hard to clean up after while using Python's garbage collection system. Java on the other hand, has a much more extensive support for its multithreading. Considering multithreading is reserved for asynchronous functions on a multicore processor, this application could potentially be multithreaded. However, without a massive scaling of my prototype, multithreading would be generally unnoticed.

Additional Features

A few other features that should be considered can also be pointed out. Python utilizes lazy evaluation, which assists its use as a flexible and faster language. This feature can also hinder the production value of python though due to some unnecessary bugs that can be overlooked if using eager evaluation. Python is an interpreted language, which is a major highlight. Developers can never leave the interpreter as opposed to Java where the debuggers are not as versatile. An IDE is slow as opposed to Pythons REPL which allows

programmers to modify code and execute it within the interpreter. [5][1]

4 Twisted Features and Node.js Comparison

Twisted is an extremely versatile yet powerful framework that is effective in creating a server herd application. The reasons behind why Twisted is suitable in this setting will be further described but can be narrowed to its use of deferreds and its ability to run on one thread. [6]

The use of event driven programming in Twisted is very unpredictable. Certain events may be extremely simple while others can take a varied amount of time. Certain events can be isolated as asynchronous using the deferred feature. The use of deferred can allow for a particular event to be ignored until an event occurs. This can speed up a web interface extremely, especially one that is single threaded.

The single threaded interface of Twisted simplifies many issues and allows for great potential. Servers can still be run in multiple processes throughout a network. If this server herd were run on a multi-threaded system, many race conditions would have to be considered, especially dealing with the flooding algorithm. Luckily Twisted handles this issue well.

When comparing Twisted to Node.js, Javascipts wep application framework, the two are very similar in style. Both are event driven frameworks with a focus on asynchronous architecture. Twisted is has a much larger community support given its age and thus is well documented. Twisted's robust nature, simplicity in design, and scalabily all favor it in use for creating a server herd. Because of this, Python and Twisted are a great choice in implementing the Wikimedia based server herd with Places API. [7]

5 References

[1] https://wiki.python.org/moin/LanguageComparisons

[2] http://www.voidspace.org.uk/python/articles/duck_typing.shtml

[3] http://benchmarksgame.alioth.debian.org/u64q/python.php

[4] http://pythonconquerstheuniverse.wordpress.com/2009/10/03/python-java-a-side-by-side-comparison/

[5] http://programmers.stackexchange.com/questions/163985/why-is-the-concept-of-lazy-evaluation-useful

[6] http://twistedmatrix.com/trac/wiki/Documentation

[7] http://stackoverflow.com/questions/3461549/what-are-the-use-cases-of-node-js-vs-twisted