Twisted Twitter Proxy Herd

Jason Zhang

Abstract

Twisted is defined as an event-driven networking framework written in Python. As an event-driven program, Twisted behaves dependently on user actions or received messages. For the purposes of this project, it will be the latter. As a framework, Twisted's source code, which provides generic functionality, is easily modifiable to serve our purposes. This report will give a precursor to the capabilities of Twisted and evaluate its effectiveness as a real-time web service against node.js and MINA, its JavaScript and Java counterparts.

1. Introduction

In this project, we have been tasked with looking into the Twisted event-driven networking framework and determine whether or not it is a suitable replacement for the Wikimedia platform. The most attractive feature for this scenario is Twisted's ability to allow an update to be processed and forwarded rapidly to other servers which will be useful if the number of connections ranges in the thousands and upwards.

Other helpful features of Twisted include its flexibility, security, and stability. Because the Twisted library houses numerous protocols, publication of web sites and development of servers and clients can be quickly and easily accomplished. If there are features users need, Twisted's open source code allows for easy custom implementations.

Python's high-level features also allow programmers to specify program behavior with less lines of code. As a benefit from having reduced code, the program will be less prone to crashing or containing bugs, greatly increasing stability. Furthermore, less vulnerability will mean fewer flaws to exploit, preventing buffer overflows.

To put Twisted to the test, we are required to implement a prototype that can at least perform the basic functions of the service. This way, we can make firsthand observations on the effectiveness of Twisted.

2. Implementation

Our prototype consists of five servers named Blake, Bryant, Gasol, Howard, Metta. Each sever accepts TCP connections from clients in the form of message like:

IAMAT kiwi.cs.ucla.edu +27.5916+0.86.5640 1353118103.108893381

The first field, IAMAT, is the command, and subsequent fields are latitude and longitude in IOS 6709 notation and the time stamp in POSIX time respectively. The message parsing and response is handled by the MyChat class. Upon receiving a message, it determines what the first field is. If IAMAT, it calls the function processIAMAT and if WHATSAT, it calls processWHATSAT. If the message is in an unrecognizable format, the sender is notified to use the correct syntax.

IAMAT stores the different fields into a database and calculates the time difference between when the message was sent, as manually indicated by the client, and when the message received. If the processing is successful, a response is sent in the form of:

AT Blake +0.563873386 kiwi.cs.ucla.edu +27.5916+0.86.5640 1353118103.108893381

Here, AT is the name of the response, Blake is the ID of the server which received the message, and +0.563873386 is the time difference. The remaining fields are as found in IAMAT.

WHATSAT messages come in the form:

WHATSAT kiwi.cs.ucla.edu 100 2

The only notable differences are the last two fields which indicate a radius in kilometers from the client and an upper bound on the number of tweets from Twitter senders within the given radius.

Similar toIAMAT, processWHATSAT stores each field in a database and checks that the upper bound is at most 100, as indicated by our specifications. The function then calls on another custom function to query Twitter which will relay the results back to the client in a format too long to demonstrate in this report.

Lastly, we must adjust the servers to meet the following conditions: Blake talks with everybody but Gasol and Metta, Gasol talks with Bryant and with Howard, and Metta talks with Bryant.

3. Results

While learning the protocols and objects of Twisted was daunting, implementing was made easier due to its built-in libraries. The LineReceiver class did all the work of listening to the port and relaying the messages. It was also easy to overload functions and make it so that they behave the way we wanted, such as parsing messages and how to reply.

Additionally, Twisted's Factory class enables multiple clients to connect to a server using just a single port, sparing us the need to implement connections via multiple ports to a single server. Deferred objects also allowed us to handle blocking functions, eliminating the need to manage multiple threads.

4. Further Discussion

Several concerns regarding Twisted are its dynamic type-checking, memory management, and multithreading concerns. A brief examination of these areas reveal that the language upon which Twisted is built on (Python) has many safety precautions to limit or even entirely prevent the above features from raising errors.

4.1 Dynamic type-checking

Regarding dynamic type-checking, the clear advantages are that compilers may run more quickly. This is mainly attributed to the fact that types are checked during run-time. Furthermore, interpreters can dynamically load new code, reducing the need for the compiler to backtrack and consequently resulting in less code to revisit.

However, there are some caveats with dynamic type-checking. Run-time crashes may occur at the presence of an unexpected type. To ensure that this does not happen, many tests are needed to be performed and extra code may be needed to catch exceptions.

4.2 Memory Management

In Python, memory is loaded into a built-in private heap that contains all objects and data structures. All of the work is taking care of the interpreter, including allocating raw memory to ensure space in the heap for data storage. As such, user interference is typically prohibited, allowing the garbage collector to function as needed without the worry of out-out-place data.

One concern about Python's memory management is the memory overhead. After all, such safety precautions do not come without a price. Fortunately, it is virtual memory that is being used and RAM being as inexpensive as it is today, this memory overhead is no longer a serious concern.

4.3 Multithreading

Among the three concerns, multithreading might be the most serious. Because Python's memory management is not thread-safe, Python must use the Global Interpreter Lock (GIL) to prevent multiple threads from executing Python bytecodes at once. This lock prevents true multiprocessing, disabling Python multithreading to utilize multiple processors.

With Twisted, this fault can be circumvented with asynchronous programming. Instead of using multiple processors, Twisted uses a single thread to process incoming messages. While it is an effective and efficient means, using multiple cores, had it been safe, may have been the better option.

4.4 Comparison to Java

Unlike Python, Java utilizes static typing to check for types. Whereas dynamic typing occurs during runtime, static typing takes place during compile time. While this may be a slower option, it is a lot safer in ensuring that there are no runtime crashes caused by erroneous types.

Like Python, Java takes care of memory management using a true garbage collector scheme. When using either of these languages, users will no longer need to deal with reference counting at run-time. But even better on Java's end is that it handles the immediate operation stack which allows for multithreading. Unlike Python's which is impeded by the global interpreter lock, multithreading in Java can fully utilize multiple processors.

4.5 Comparison to Node.js

Node.js is a server side software system that can be used to write internet applications, most notably web servers. Like Twisted, it too utilizes event-driven, asynchronous I/O. As similar as they are, there are some advantages and disadvantages to both.

Since Node.js was developed in 2009 and Twisted in 2002, the latter will obviously have broader and more well-tested library of functions. Furthermore, Twisted supports a wide range of protocols while Node.js primarily deals with HTTP. One advantage Node.js has though is that it is a lot easier to learn due simpler syntax and better documentation.

5. Conclusion

Ultimately, Twisted is a suitable framework for setting replacing the Wikimedia platform. It has numerous advantages such as its flexibility due to its numerous protocols, and its security and stability due to Python's high-level coding. Almost every drawback of Python is in some shape or form addressed by the features of Twisted such as using asynchronous programming to resolve imperfect multithreading. As the positives outweigh the negatives, Twisted looks like the ideal candidate for our application.

6. References

[1] http://twistedmatrix.com/documents/current/core/howto/servers.html
[2] http://twistedmatrix.com/documents/current/core/howto/clients.html
[3] http://wiki.python.org/moin/GlobalInterpreterLock
[4] http://twistedmatrix.com/users/glyph/ipc10/paper.html
[5] http://stackoverflow.com/questions/3461549/what-are-the-use-cases-of-node-js-vs-twisted
[6] http://www.quora.com/What-are-the-advantages-disadvantages-of-Pythons-Twisted-vs-Node-js
[7] http://en.wikipedia.org/wiki/Twisted_(software)
[8] http://en.wikipedia.org/wiki/Node.js
[9] http://stackoverflow.com/questions/1861457/python-vs-java-which-would-you-pick-to-do-concurrent-programming-and-why
[10] http://stackoverflow.com/questions/6853449/java-python-type-checking
[11] http://en.wikipedia.org/wiki/Type_system#Static_and_dynamic_type_checking_in_practice