CS 131 Project: Twisted Places Proxy Herd

Harrison Liddiard
University of California, Los Angeles
Winter 2017

Abstract

This paper examines the implementation of a prototype application server herd using the Twisted Python networking library and its viability for use as a replacement for all or part of the Wikimedia platform. It analyzes Twisted's practical benefits and downsides in terms of maintenance, reliability, and scalability. A comparison to Node.js is also briefly discussed.

1. Introduction

This project prototypes the use of Twisted, a Python open-source networking engine [1] for sending messages and synchronizing data among a herd of servers arranged as a partially connected graph. The architecture is designed to handle frequently updated data such as GPS locations, storing the state of each client in RAM and propagating it to other clients via TCP. [2]

This implementation contrasts with the current Wikimedia architecture which is based on LAMP and optimized for a significantly higher frequency of reads than writes. [3] The Twisted prototype allows clients to connect to any of the five servers in the herd and transmit location data using TCP. Internally, the servers propagate this information among each other using a simple flooding algorithm.

Wikimedia's current LAMP-based architecture is not well suited for the proposed applications. Apache, unlike Twisted, spawns new threads or processes for each connection which can cause significant resource use and slowdown with many concurrent connections. [4] MySQL is advantageous in it ability to model complex relations among data and store changes to disk, but it is slower than a flat, in-memory data store which is more performant and is sufficient for an application such as device location updates. [5] PHP, in contrast to Python, has many cases of unintuitive syntax and conflicting method naming and calling conventions. [6]

While the Foundation has been able to achieve reliable and quick performance through multiple layers of caching, search-optimized data storage and retrieval with Lucene, and distributed static content service though a global CDN [3], the architecture is fundamentally not well suited for high throughput writes for the reasons listed above. As multiple clients contacting multiple servers may need to update and retrieve information in quick succession, cache invalidation would become difficult to manage across a herd of servers.

2. Twisted

Twisted is an MIT-licensed, event-driven networking engine written in Python. [1] It is a full-featured library that includes client and server implementations for a wide

variety of networking protocols including TCP, UDP, and HTTP. Twisted includes production-ready tools for configuration and deployment of its applications. [7] Like Python, Twisted works cross-platform on Linux, Windows, and macOS. [8]

Twisted is inherently asynchronous and can handle multiple simultaneous requests in a single thread, reducing its overhead in comparison to a multithreaded approach. Its implementation and usage are heavily object oriented, with server and client instantiation following a factory pattern. It makes use of class-based inheritance to add custom functionality on top of its default implementations.

Twisted is written in Python, which like Java, is strongly typed. This means trying to perform an operation on two dissimilar types raises an exception as opposed to performing an implicit type coercion. [9] However, Python is dynamically typed in contrast to Java's static typing. This means every variable is bound to an object at runtime, and a variable can be reassigned to an object of a different type throughout the course of the program. In contrast, Java has an extra step which binds a variable to a type at compile time and will raise an exception if an attempt is made to assign the variable to an object of a different type.

While robust testing and coding standards that disallow reassignment of a variable to different types can make this much less likely to be the source of bugs in practice, static type checking is also supported via the built-in typing library in Python 3 [10] and through the mypy package for Python 2. [11] Twisted supports Python 2 fully, and a significant subset supports Python 3, with increased Python 3 coverage planned. [12]

3. Prototype

This prototype consists of five servers that participate in bidirectional communication with each other. Each server communicates with a predefined subset of the other servers such that all servers communicate with one another directly or indirectly through one or more intermediary servers.

Clients can send "IAMAT" requests to any single server to indicate their current position. The contacted server responds to the client with an AT message acknowledging the successful receipt of the request.

Additionally, the server sends the AT message to all servers with which it is peers to propagate the new location data. These servers, in turn, send the message to all servers with which they are peers, excluding the origin server. The propagation continues for each server unless the server has already received the message (i.e. the location update it receives is a duplicate.).

Clients can also query for places of interest around a particular named client using the "WHATSAT" command. For this request, the servers leverage data provided by the Google Places API. The servers' responses from this command are not propagated among other servers.

4. Analysis

4.1 Maintainability

Twisted's class-based architecture provides a basic framework for code organization, but beyond its predefined classes, Twisted does not enforce a

particular file structure or code layout. This is in contrast to other commonly used Python web frameworks like Django which prescribe a fairly rigid pattern for organizing code from the start of the project to growing and connecting with other applications. [13] If the Wikimedia Foundation were to adopt a Twisted-based architecture for the entirety of its applications (or even a reasonably large subset), a paradigm for code organization should be decided to maintain cohesion.

4.2 Reliability

By examining the source code and by consulting Twisted's own documentation, it is evident that Twisted's "reactor" always runs in a single thread in an event loop. [14] Due to this architecture, events are processed one by one and queued if necessary. Operations that happen following the event loop processing, however, can be asynchronous. Certain operations like hostname lookup happen asynchronously because they are blocking. Unlike other frameworks like Node.js (discussed later), the asynchronous nature is abstracted away from the programmer. As Twisted's documentation advises, "Twisted does most things in one thread! If you're just using Twisted's own APIs, the rule for threads is simply 'don't use them'." [14]

Given that Twisted can process multiple requests at once, it is not data race free. For example, if a client were to send a location update at the same time another client issued a "WHATSAT" request referencing the same client, the query could nondeterministically return results from before or after the location update. This behavior could pose problems in instances such as two people editing a Wikipedia article at the same time (in which case some sort of locking mechanism would be necessary), but it should not pose any practical user experience issues for location updates and queries as implemented in the prototype.

4.3 Scalability

Twisted was written, in part, in response to a lack of scalable services for developing realtime networked games. [15] As such, scalable deployment seems to be an area in which Twisted excels. The library comes with robust built-in deployment utilities. Twisted's architecture allows it to share data among different services within a process which makes it straightforward to wire different protocols together. [8] Twisted is used in production by companies for high-throughput services including Apple Calendar, NASA, Twilio, and Scrapy. [16] [17] [18]

5. Node.js Comparison

Node.js is a JavaScript (ECMAScript) networking library built with the V8 JavaScript engine developed for Google Chrome. [19] Like Twisted, it is event-driven and non-blocking. Node.js has been around half as long as Twisted – 7 years versus 14 [20] [21], but quickly eclipsed Twisted in popularity. [22] Unlike Twisted, Node.js leaves much more asynchronous coding up to the developer. Callbacks are extremely common, leading to an often-derided "callback hell." [23] The async/await features in ES2017, the upcoming version of ECMAScript, mimic the syntax of synchronously executed code and should help to alleviate the majority of the developer confusion caused by multiple nested callbacks like unintuitive variable scoping. [24]

In the realm of package management, Python has pip, and JavaScript has NPM. [25] [26] Python developers typically use a virtualenv – a separate installation of Python local to the project which has its own dependencies – to avoid polluting the global environment with packages from various projects. [27] These packages are typically stored in a requirements.txt file in the project repository. NPM, JavaScript's leading package manager, is generally regarded as easier to use due to its intrinsically hierarchical method of resolving dependencies and its builtin functionality to save new packages to its own dependency file register, package.json. [28] That said, many people have reported problems managing locked dependencies for production using NPM's "shrinkwrap" command. [29] Facebook has developed their own JavaScript package manager called Yarn which attempts to simplify this process, among others. [30]

Both Twisted and Node.js follow semver, so it is possible to easily compare both libraries for how often they increment a major version, indicating breaking API changes. [31] Recently, Twisted has been releasing major versions about every year, while Node.js has been releasing major versions every few months. [20] [21] There is a tradeoff between frequent major releases and stability, but Twisted may take the edge here in being the more stable – and thus more easily maintainable – platform.

6. Conclusion

Twisted's event-driven implementation and scalable architecture, coupled with the clean, "one right way" to do anything with Python [38] and a robust package ecosystem, make it a viable and attractive candidate for implementing Wikimedia's future platform. Despite its lack of intrinsic static typing and suboptimal dependency management, its extendibility, excellent documentation, defined direction are points in its favor. Node.js seems to have learned from some of the mistakes of Python web frameworks in the realm of dependency management, but has its own issues around the idiosyncrasies in the rapidly-evolving ECMAScript language standard and quick iterations in the library itself. However given Node.js's sustained interest from the developer community, it as well probably deserves further research as a platform for the Wikimedia Foundation's future endeavors.

References

[1] "Downloads." Twisted. N.p., n.d. Web. 31 Dec. 2016, http://twistedmatrix.com/trac/.

[2] Eggert, Paul. "Project. Twisted Places Proxy Herd." Project. Twisted Places Proxy Herd. UCLA, n.d. Web. 31 Dec. 2016, http://web.cs.ucla.edu/classes/winter15/cs131/hw/pr.html.

[3] Bergsma, Mark. "Wikimedia Architecture." Wikimedia. Wikimedia Foundation, 2007. Web. 31 Dec. 2016, https://wikitech.wikimedia.org/wiki/File:Bergsma_-_Wikimedia_architecture_-_2007.pdf.

[4] "NGINX vs Apache: Advantages and Disadvantages." 1&1. 1&1 Internet Inc., 14 Oct. 2016. Web. 31 Dec. 2016, https://www.1and1.com/cloud-community/learn/web-server/nginx/nginx-vs-apache-advantages-and-disadvantages/.

[5] Sanfilippo, Salvatore. "Redis in Action." Redis Labs. Redis Labs, n.d. Web. 31 Dec.

2016, https://redislabs.com/ebook/redis-in-action/part-1-getting-started/chapter-1-getting-to-know-redis/1-1-what-is-redis/1-1-1-redis-compared-to-other-databases-and-software.
[6] Embry, Darren. "Why PHP Sucks." Why PHP Sucks. Darren Embry, n.d. Web. 31 Dec. 2016, https://webonastick.com/php.html.
[7] McKellar, Jessica, and Abe Fettig. Twisted Network Programming Essentials. Farnham: O'Reilly, 2013. Print.
[8] Fettig, Abe. "Getting Started." Safari. O'Reilly Media, Inc., Oct. 2005. Web. 31 Dec. 2016, https://www.safaribooksonline.com/library/view/twisted-network-programming/0596100329/ch01.html.
[9] Ferg, Steve. "Static vs. Dynamic Typing of Programming Languages." Python Conquers The Universe. WordPress, 08 Apr. 2012. Web. 31 Dec. 2016, https://pythonconquerstheuniverse.wordpress.com/2009/10/03/static-vs-dynamic-typing-of-programming-languages.
[10] "26.1. Typing — Support for Type Hints." Python Documentation. Python Software Foundation, 1 Jan. 2017. Web. 31 Dec. 2016, https://docs.python.org/3/library/typing.html.
[11] "Mypy - Optional Static Typing for Python." Mypy. The Mypy Project, 2014. Web. 31 Dec. 2016, http://mypy-lang.org/.
[12] "Plan/Python3 – Twisted." Twisted Matrix Labs. N.p., 31 Dec. 2016. Web. 31 Dec. 2016, https://twistedmatrix.com/trac/wiki/Plan/Python3.
[13] Reitz, Kenneth. "Structuring Your Project." Structuring Your Project — The Hitchhiker's Guide to Python. N.p., 2016. Web. 31 Dec. 2016, http://docs.python-guide.org/en/latest/writing/structure/.
[14] "Using Threads in Twisted." Twisted 16.5.0 Documentation. N.p., n.d. Web. 31 Dec. 2016, http://twistedmatrix.com/documents/current/core/howto/threading.html.
[15] McKellar, Jessica. "Twisted." The Architecture of Open Source Applications (Volume 2): Twisted. N.p., n.d. Web. 31 Dec. 2016, http://www.aosabook.org/en/twisted.html.
[16] Apple. "Apple/ccs-calendarserver." GitHub. N.p., 21 Dec. 2016. Web. 31 Dec. 2016, https://github.com/apple/ccs-calendarserver.
[17] Hakka Labs. "Tech Talk: An Introduction to the Python Framework Used by NASA, Twilio, and HipChat." YouTube. Google, 11 Apr. 2014. Web. 31 Dec. 2016, https://www.youtube.com/watch?v=qmdQttKEHBM.
[18] "Common Practices." Scrapy 1.3.0 Documentation. Scrapy Developers, n.d. Web. 31 Dec. 2016, https://doc.scrapy.org/en/latest/topics/practices.html.
[19] "Node.js." Node.js. Node.js Foundation, n.d. Web. 31 Dec. 2016, https://nodejs.org/en/.
[20] "Node – Releases." GitHub. N.p., n.d. Web. 31 Dec. 2016, https://github.com/nodejs/node/releases.
[21] "Twisted – Releases." GitHub. N.p., n.d. Web. 31 Dec. 2016, https://github.com/twisted/twisted/releases.
[22] "Compare – Google Trends." Google Trends. Google, n.d. Web. 31 Dec. 2016, https://www.google.com/trends/explore?date=all&q=node.js%2Bjavascript%2Ctwisted%2Bpython.
[23] Ogden, Max. "Callback Hell." Callback Hell. N.p., n.d. Web. 31 Dec. 2016, http://callbackhell.com/.
[24] Bevacqua, Nicolás. "Understanding JavaScript's Async Await." Pony Foo. N.p., 27 Dec. 2016. Web. 31 Dec. 2016, https://

ponyfoo.com/articles/understanding-javascript-async-await.
[25] "Installing Python Modules." Installing Python Modules. Python Software Foundation, n.d. Web. 31 Dec. 2016, https://docs.python.org/3/installing/.
[26] "Build Amazing Things." Npm. Npm, Inc., n.d. Web. 31 Dec. 2016, https://www.npmjs.com/.
[27] Reitz, Kenneth. "Virtual Environments." The Hitchhiker's Guide to Python. N.p., 2016. Web. 31 Dec. 2016, http://docs.python-guide.org/en/latest/dev/virtualenvs/.
[28] "Package.json." Npm Documentation. Npm, Inc., n.d. Web. 31 Dec. 2016, https://docs.npmjs.com/files/package.json.
[29] Reeves, Jonny. "Npm-shrinkwrap Sucks." Jonny Reeves. N.p., 03 May 2016. Web. 31 Dec. 2016, http://jonnyreeves.co.uk/2016/npm-shrinkwrap-sucks/.
[30] McKenzie, Sebastian, Christoph Pojer, and James Kyle. "Yarn: A New Package Manager for JavaScript." Facebook Code. Facebook, 11 Oct. 2016. Web. 31 Dec. 2016, https://code.facebook.com/posts/1840075619545360.
[31] "Semantic Versioning 2.0.0." Semantic Versioning. N.p., n.d. Web. 31 Dec. 2016, http://semver.org/.
[32] Peters, Tim. "PEP 20 -- The Zen of Python." Python.org. Python Software Foundation, 14 Aug. 2004. Web. 31 Dec. 2016, https://www.python.org/dev/peps/pep-0020/.