Project: Proxy Herd

Quentin Truong - University of California, Los Angeles

Abstract

Python's recently added asyncio provides support for asynchronous operations through an event loop and coroutines. Implementing a proxy herd using asyncio and other asynchronous libraries proved to be a quick task. The performance of the server is good when compared to other possible implementations in languages like Java and Javascript's Node.js.

1. Introduction

1.1. Wikipedia and MediaWiki

Wikipedia is a web-based encyclopedia based upon a model of openly-editable content. Wikipedia runs on the MediaWiki software, which is supported by the LAMP (Linux, Apache, MySQL, PHP) architecture and uses multiple, redundant web servers behind a load-balancing virtual router [1]. The MediaWiki software is designed to be scalable and serve open content for high-traffic websites [2].

1.2. Wikipedia-style Service for News

We are tasked with building a Wikipedia-style service for news, where updates to articles happens far more often, access will be required via various protocols, and clients will tend to be mobile. The MediaWiki software appears to be a bottleneck, and thus we explore different architectures. In particular, we discuss various implementations of an application server herd, otherwise known as a parallelizable proxy.

1.3. Proxy Herd

A proxy herd is a server architecture where multiple servers communicate directly to each other as well as a central database. This enables rapid updates to data without the performance bottleneck of a central server. We will begin by comparing the suitability of three programming languages, Python, Java, and Javascript, and their associated tools for implementing this application, and will end with a discussion of a sample implementation of a Google Place's proxy herd.

2. Python and asyncio

Python is an interpreted, high-level programming language for general-purpose programming [3].

2.1. Type Checking

Dynamic type checking in Python allows for types to be determined at runtime [4]. This has a number of productivity benefits, including more concise and readable code. Python is also strongly typed, meaning that types are enforced rather than implicitly coerced. This helps prevent common programming mistakes, such as accidentally casting an integer to string, resulting in unexpected behavior. Lastly, Python supports duck typing. Duck typing is best described by the following principle: If it walks like a duck and it quacks like a duck, then it must be a duck. Duck typing is a powerful language feature which enables object reuse in a vast variety of contexts. Again, like the other features, duck typing enables us to remove much of the boilerplate code that other languages require for object reuse. Python's type checking results in concise code and fast development times.

2.2. Memory Management

Python uses an automatic garbage collector in the form of a count-based reference garbage collector [5]. A count-based reference garbage collector removes objects whose reference count has fallen to zero. This enables the developer to implicitly allocate and free memory, resulting in more concise code, faster development times, and oftentimes, fewer bugs.

2.3. Multithreading

Python relies on the Global Interpreter Lock (GIL) for memory management reasons. Although the GIL prevents true multithreading in Python, we may still call Python functions which use multithreading outside of Python; libraries such as NumPy do this [6]. Alternatively, it is common to use multiple processes instead of multithreading in Python. This way, we may leverage the multi-core nature of many modern machines and achieve the higher performance [7].

2.4. asynio

Python's asyncio, added recently in Python3.4, provides powerful asynchronous features, including an event loop and coroutines. The asyncio library enables data to be sent over connections using the TCP protocol. Moreover, asyncio has been optimized for high performance through the use of uvloop [8]. In fact, asyncio is at least 2x as fast as Node.js according to certain benchmarks [9]. The asyncio library is a powerful tool that may prove useful in building asynchronous, scalable servers because of its asynchronous functionalities and built-in support for connections.

2.5. Other Tools

The Django framework is a free, open-source web framework built in Python. Django provides a number

of components essential to building full web applications, including administration, authentication, webpages, database features, and more. To build our new service, Django will likely be essential.

The package manager pip is very popular and has extensive support for a large variety of tasks. Reusing packages from pip will enable fast development times.

3. Java

Java is a general-purpose programming language often run in the Java Virtual Machine, enabling consistency across different platforms.

3.1. Type Checking

Java, like Python, is strongly typed; this type enforcement prevents many common programming bugs. Unlike Python, Java is statically typed, meaning that all types are known at compile time [10]. This often results in fewer runtime bugs due to all type errors being caught early on.

3.2. Memory Management

Java relies on an automatic garbage collector using a mark-and-sweep algorithm as well as a nursery [11]. The mark-and-sweep algorithm effectively takes a top-down approach, marking all objects which are reachable, and removing the rest. The nursery is where new objects go and is garbage collected more frequently than the entire program. This results in higher performance than Python's memory manager, at the cost of development time. Also, this means that Java programmers do not have to explicitly free objects, although the keyword 'new' is often needed for dynamic allocation.

3.3. Multithreading

Java fully implements multithreading, locks, and synchronization primitives [12]. Synchronized code is trivially easy to write in Java, as it simply requires the 'synchronized' keyword. This is an easy way to gain performance with very little development costs.

3.4 Servlets

Java servlets are protocol and platform independent server side components [13]. They may be used to build out many common functionalities for servers, including forwarding messages to other servers. If Java were chosen as the programming language to implement this new service, Java servlets would likely be used.

4. JavaScript's Node.js

JavaScript is a high-level, interpreted programming language useful for both client and server-side programming.

4.1. Type Checking

JavaScript uses weakly typed (some may even say, untyped) type checking [14]. This means that nearly all values are implicitly and automatically coerced as necssary. For instance, if we add a string to an integer, the integer will automatically be coerced into a string, resulting in two strings being concatenated together. JavaScript, like Python, supports duck typing. JavaScript's type checking enables fast development times, as there is little overhead about worrying about types; however, bugs from type coercion may be difficult to catch.

4.2. Memory Management

JavaScript, like Java, relies on an automatic garbage collector using a mark-and-sweep algorithm, with similar benefits as to Java [15].

4.3. Multithreading

JavaScript has no true multithreading, due to the fact that it is frequently run in browsers where concurrency between threads is not easily implemented. Alternatively, many JavaScript applications use web workers instead of multithreading, achieving similar benefits. Web workers are JavaScript scripts which run in the background and communicate through messages, enabling non-blocking execution [16]. Web workers are a likely candidate if this new application needs to run intensive tasks.

4.4. Node.js

Rather than using multithreading or webworkers, Node.js is an asynchronous, event-driven environment for servers. Node.js uses an event loop and event-driven programming with callbacks to complete tasks. This results in fast, lightweight servers. Moreover, Node.js is powered by the highly optimized Google Chrome V8 engine. This results in very performant code. Moreover, Node.js is supported by npm, a package manager, enabling easy code reuse. Also, since NodeJs uses JavaScript, both client and serverside code can be in the same language, resulting in simpler architectures.

4.5. Other Tools

JavaScript has a number of very popular web frameworks including React, Angular, and Vue. These frameworks support many of the essential features for building services, especially user interfaces. Other features necessary for building a full web application may be easily found and installed through npm.

5. Google Places Proxy Herd

The Google Places API is a service which returns information about places through HTTP requests. To

explore Python's asyncio's potential as a proxy herd server, I implemented a proxy herd for a subset of the Google Places API.

5.1. Ease of Development

Developing a proxy herd in Python was extremely easy due to the many benefits from the asyncio library and qualities of Python described earlier. Documentation is readily available for asyncio is readily available and the features of asyncio were well suited to the task. Features such as building an event loop, creating a server, connecting servers to each other, receiving and sending data across connections, and logging data were all built-in features of either asyncio or Python.

Python's type checking and memory management were strongly beneficial to the ease of development. Due to Python's strongly typed nature, many bugs were easily caught. And due to the automatic allocation and garbage collection, very little boilerplate code had to be written. Running multiple servers (necessary for a proxy herd) in Python is also very straightforward, as it only requires running multiple Python scripts.

Python's asyncio is a strong contender against Node.js due to the fast performance of asyncio (because of uvloop), and the readily available documentation, packages, and overall ease of development. A Python web application may be more maintainable and extensible than JavaScript due to the strongly typed nature. That said, Node.js also has many available packages and web frameworks.

In conclusion, I recommend the use of Python due to the extreme ease of development and readily available packages. Python's asyncio works well for implementing a proxy herd because of its asynchronous nature and event loop. Moreover, Django will be a useful tool in building the other features of this web application.

5.2. Logs

Five servers were generated. Here is a section from one sample log to demonstrate the working functionalities of the proxy herd.

```
---PlacesProxyServerLog_Goloman---
Connection from ('127.0.0.1', 46010)
Received: 'CONNECT 12438'
Connection from ('127.0.0.1', 46016)
Received: 'CONNECT 12439'
Connection from ('127.0.0.1', 54422)
Received: 'CONNECT 12441'
Lost connection of ('127.0.0.1', 54422)
Connection from ('127.0.0.1', 54448)
Received: 'CONNECT 12441'
Connection from ('127.0.0.1', 54462)
Received: 'IAMAT qt +48.8566+2.3522
1527997564.123123\r\n'
Sent: 'AT Goloman +214.401330 qt
+48.8566+2.3522 1527997564.123123\r\n'
Received: 'ATSERVER qt +48.8566 +2.3522
214.40133023262024'
Received: 'ATSERVER qt +48.8566 +2.3522
214.40133023262024'
Received: 'ATSERVER qt +48.8566 +2.3522
214.40133023262024'
```

References

[1] Bergsma, Mark. Wikimedia Architecture. 2007. <https://upload.wikimedia.org/wikipedia/labs/8/81/Bergsma_-_Wikimedia_architecture_-_2007.pdf>.

[2] What is MediaWiki? 20 January 2018. <https://www.mediawiki.org/wiki/Manual:What_is_MediaWiki%3F>.

[3] The Python Tutorial. 2018. <https://docs.python.org/3/tutorial/index.html>.

[4] Why is Python a dynamic language and also a strongly typed language. 24 February 2012. <https://wiki.python.org/moin/Why is Python a dynamic language and also a strongly typed language>.

[5] Reference Counts. 22 June 2001. <https://docs.python.org/2.0/ext/refcounts.html>.

[6] Wouters, Thomas. GlobalInterpreterLock. 2 August 2018. <https://wiki.python.org/moin/GlobalInterpreterLock>.

[7] multiprocessing. 2018. <https://docs.python.org/3/library/multiprocessing.html>.

[8] uvloop. 2018. <http://uvloop.readthedocs.io/>.

[9] Selivanov, Yury. uvloop: Blazing fast Python networking. 3 May 2016. <https://magic.io/blog/uvloop-blazing-fast-python-networking/>.

[10] Dynamic typing vs. static typing. March 2015. <https://docs.oracle.com/cd/E57471_01/bigData.100/extensions_bdd/src/cext_transform_typing.html>.

[11] Understanding Memory Management. 2016. <https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/garbage_collect.html>.

[12] Processes and Threads. 2017. <https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html>.

[13] The Java Servlet API White Paper. <http://www.oracle.com/technetwork/java/whitepaper-135196.html>.

[14] Sanders, B. William. Javascript Design: An Orientation to JavaScript. 8 Feb 2002.

<http://www.peachpit.com/articles/article.aspx?p=25275&seqNum=4>.

[15] Memory Management. 31 Decemeber 2017. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Memory_Management>.

[16] Peng, Max. Multithreading Javascript. 14 September 2017. <https://medium.com/techtrument/multithreading-javascript-46156179cf9a>.