Python vs Java

Erdi Kidane
University of California Los Angeles

Abstract

This project explores the usage of Pythons networking library asyncio as a replacement candidate for the Wikimedia platform. To determine whether asyncio is a good fit I have setup a server architecture test bench that uses a flooding algorithm. The prototype has exactly five servers that communicate with each other keeping track of a given clients location. Queries to the servers are processed using the Google Places API. Essentially the client askes what is near them and the severs respond. Ive also looked at Pythons memory management, multithreading support and type checking against Javas systems. I have determined that Java is preferable to Python for this kind of work because Pythons dynamic type checking is a disadvantage compared to Javas static type checking. Thus Java will use less memory and execute faster. Further multithreading maybe something we should avoid but even if we decide to use it Java has support for the same event-driven execution that asyncio supports. Therefore I believe Java better meets our requirements better than Python.

1 Introduction

The Python asyncio library supports event loop driven concurrency and this can allow severs to handle asynchronous events. The server architecture has multiple nodes that are connected as a network. Clients connect to one of these nodes and can then broadcast their location and ask for who is near them. Client requests are handled by severs, essentially the servers propagate clients location information throughout the sever network. The prototype server network use the asyncio library they are designed in such a way that it allows one to easily compare it to other programming languages like Java.

2 Implementation

The main sever application is server.py. A user will setup a server to accept incoming TCP connections. It then runs an event loop until it receives an interrupt. The severs themselves contain a callback method on each new connection. The callback is passed a reader and writer that allows for the sever to communicate with each new client.

Both reader and writer are from the asyncio library, and thus must be used in any async def type syntax. In any async function we can use the await keyword on a statement to indicate that we would like it to be asynchronous and can block. If execution blocks on a statement declared await, the program switches to another task. This allows for concurrency because different connections can be served while the application waits for clients

3 Operation

Once the server is setup a client can inform the server of their location with the IAMAT command. Upon receiving a valid IAMAT input the server must update the stored location of the client if the previous locations timestamp is older than the new input timestamp. Once the server has updated the client location locally the sever must transmit this information to its neighbors. Further, any receiving server stores the ports associated with its neighbors in the server network. These are mapped to the five server names Goloman, Hands, Holiday, Welsh, and Wilkes.

The WHATSAT command is used to query a server for places around a given clients coordinates. Processing a WHATSAT command is much simpler than IAMAT command, as the server simply checks to see if it knows the given clients coordinates and then issues an HTTP request to the google places web API to retrieve the places around the client. This does not require any propagation.

The server returns a response in a JSON format.

Server to server update is done through TCP. A server first transmits updates to its neighbors then it issues a PROPAGATE command to the neighbor which contain the update information, and then closes the connection. The neighbor server will treat the incoming connection like any other client connection. It will have no information about whether a PROPGATE will occur until the command has been sent. This makes it so that no server knows about the status of any other server in the network until a propagation has occurred. If no connection can be made, the source server will simply assume that the receiving server is offline and will continue execution as normal. If a PROPGATE can be sent successfully and the receiving server has not already seen a newer update, the receiving server will try to PROPAGATE to its own neighbors, excluding the source server. This allows the update to travel through the entire server network.

4 Analysis

4.1 Memory Management

Unlike C/C++ or similar types of languages Python does not require the user to handle memory manually. The interpreter will handle this in the background automatically. This makes Python less error prone when it comes to freeing allocated memory and easier to use but it comes at the cost of performance. If programs in C for example are written correctly, they can easily outperform Python when it comes to the issue of memory.

Java is like Python in the sense that objects are allocated on the heap containing all of the programs data and so the idea of memory management in Java is abstracted away. This means that it also has some of the drawbacks that Python has when compared to C/C++. But unlike Python Java does allow for limited memory management through the keyword new and by allowing such functionality the amount of memory Java uses can vary. Still over all I would argue that Java and Python roughly preform the same when it comes to memory management. We dont have to deal with memory, but we pay for it with a performance cost.

4.2 Multithreading

The asyncio library is largely not thread safe. It was intended to be ran on a single threaded environment. Instead of using threads as the primary form of concurrency, asyncio uses the event loop. This has the advantage of performing better in situations where any connections to our server are necessary. If a sperate thread were created for each connection, we could have thousands upon thousands of threads. Then the overhead of managing the threads may be too costly, and an event loop would perform better.

In our server network design, we do not need multithreading because each server can be ran as a sperate process. They will propagate information to each other allowing for work to be distributed over many machines. This approach can use multiple CPU cores to run different processes, but it also maintains data integrity by only allowing communication through TCP. So this approach takes advantage of the primary benefit of multithreading, which is parallel execution, but avoid problems that arise from race conditions and data races. Instead of the work being split across threads in a single process, the server load is split across multiple processes.

In comparison the Java approach is to run a server by using threads. As in A typical multithreaded Java server will first listen on a socket for incoming connections. When it detects a connection, it will create a new thread that runs on its own and handles the communication with client. This approach could potentially spawn many threads that cause performance to degrade. We could implement our flooding algorithm using threads but it seems that an event driven approach is the most natural way to do what we want.

Java already has libraries like Netty that allow for event-driven programming. So this point of comparison does not solely depend on which language we choose, since we can choose a multithreaded or event-driven approach in either language. It mainly comes down to familiarity with a given API and the languages ease of use.

4.3 Type Checking

Python does not have static type checking instead it uses dynamic type checking. What this means is that client inputs must be validated and checked during runtime. One must validate all possible inputs in order to avoid undefined behavior. And so, when implementing this server, I had to validate the arguments of WHATSAT and IAMAT commands. If an input is incorrect the program catches the exception and instead executes the code that handles invalid inputs.

A situation where Pythons dynamic type checking did not help was when I had to validate coordinate inputs. I had to write my own function to verify if the argument was in valid format. I used regex to extract the longitude and latitude, then cast them to floats in order to check their ranges. This would not have been an issue in a statically typed language like Java.

In conclusion I would argue that the static type checking in Java is more preferable than the dynamic type checking of Python. While code is easier to write in Python vs Java it does force one to write code to validate inputs that wouldve been handled by static type check-

ing.

4.4 Node.js Comparison

Node.js was specifically designed for event-loop driven servers and it provides good support for HTTP requests. Node.js uses Javascript which is dynamically typed language like Python is. Thus, the concerns outlined regarding dynamically typed language applies here but unlike Python it would probably be easier to do this type of software development on Node.js since it was intended for server development. Further, Node.js syntax looks similar to C/Java in my personal opinion this is easier to read than Python. Accessing the Google Places API would be easier over HTTP, as well as parsing the JSON response. When working with asyncio library I must constantly trace through the code to make sure what a particular variable was doing due to the documentation being difficult to read. In comparison, the Node.js documentation looks promising. When considering Python vs Node.js, Python has a wide variety of libraries. Which means more different types of work can be performed on Python than Node.js but for some things that are purely web focused Node.js would be better. In the end if our work is purely web/server projects Node.js might be better if we want a Python like language that handles servers.

5 Recommendation and conclusion

I used a prototype server network that relays information across multiple nodes using a flooding algorithm in order to evaluate how good Pythons asyncio library is. I have come to the conclusion that dynamic type checking and hard to understand documentation is a major drawback of using Python. In my opinion, I would prefer and recommend Java over Python for this particular application. Static type checking while making the language somewhat verbose does cut back on the amount of validations that would have to be performed and make debugging a bit easier. Having well-defined function calls and good documentation is also a plus in Javas. Further there is an argument to make for the case of Java being too verbose and thus slowing down development, but a good IDE can check types as code is being written and the mental overhead of keeping track of types in negated. For many companies, nothing can beat the reliability that static type checking guarantees. Thats the reason why Java remains one of the most widely used languages in the software development industry.

Java has been used in the industry for a long time now it supports a vast array of libraries given its age and widespread use. Compared to something like Node.js which is newer, Java has lots of community support and helpful code examples. Although asyncio has its advantages due to its event driven nature, this type of event loop is not exclusive to Python. By using Netty, we can implement a similar server architecture in Java. Overall, we should use Java because it is simply the most practical general-purpose language for most business applications.

References

Python asyncio documentation
https://docs.python.org/3/library/asyncio.html
Python memory management documentation.
docs.python.org/3/c-api/memory.html
Java memory management documentation
https://docs.oracle.com/
Node.js documentation
http://nodejs.org/en/about