CS 131: Implementation of python's asyncio module for a Wikimedia server herd.

Mehar Nallamalli

Abstract— **The main goal for this project is to measure the efficiency and _____ of the asynchronous library asyncio. The Wikimedia architecture uses the LAMP platform, each of which has its own pros and cons. The implementation we will look at is where there is a central server and there are frequent read/writes to that server via different clients and protocols. In this report, I will talk about this ayncio library and its pros and cons.**

I. TESTING PLATFORM

By using the command, python -version, I was able to find out the Python version:

- `Python 3.6.6 :: Anaconda, Inc.`

Once my environment was set up, I completed developing server and client on my local machine, and then tested it using the lnxserver09.

II. INTRODUCTION

When designing a web application, there are many tech stacks and many different architectures to choose from, depending on your business requirements and which tradeoffs you are willing to sacrifice. This project is using a LAMP architecture and is taking advantage of multiple web servers for load-balancing performance.

A problem that I predicted would occur would be if the central server that is receiving the read/writes has a bottleneck, then the rest of the server herd would be limited to the performance of that central server.

To solve this problem, this project utilizes python's asyncio library and asynchronous programming to design how servers communicate with each other. This project shows that an asynchronous solution will eliminate the bottle neck that would occur for large numbers of clients.

Although asyncio has its benefits, it also has its drawbacks. Such as there is no built in TCP or secure server line so we need to use external libraries such as aiohttp.

III. Server Implementation

The server herd has five servers (Goloman, Hands, Holiday, Welsh, and Wilkes) that bi-directionally communicate with each other and they accept TCP connections from clients.
The clients will send two different types of commands

- IAMAT
- WHATSAT

The servers will respond to the IAMAT messages with AT and they respond to the WHATSAT with an AT and data from the GET request to the Places API.

A. IAMAT

IAMAT msgs have four fields, Name of command, Client ID, Lat and long, Time.

The response has 6 fields: name, server, time, client, lat long, time.

B. WHATSAT

The WHATSAT msg also has four fields.

- Client ID, radius, number of results

IV. Invalid Commands

Any message that is not IAMAT or WHATSAT is invalid. A message is invalid if it does not start with a predefined headers. When you receive an invalid command, the server will return the invalid msg to the client with a "?".

V. ASYNCIO

asyncio is used for this project because there are too many processed and would be too slow to execute synchronously, and they should be handled asynchronously to handle requests. By using asyncio, different operations can be paused and they can allow other tasks to be scheduled in the meantime to reduce idle time.

A. Advantages of asyncio

Each server has a loop and you can add a co-routine that comes to the front of the loop. This is an advantage because it allows for the server to process a lot of requests at the same time .

Another advantage is each server runs the same code, and so adding new servers is very easy, just have to run python server.py <server_name> . This makes the code very scalable and modular in design.

B. Disadvantages of asyncio

The disadvantage of asyncio is the same disadvantage in all async models. The tasks that come in are not processed in the order they arrive. This makes it harder to keep track of race conditions and critical code where there are read/writes being done. It is possible that a WHATSAT message gets processed before a IAMAT, and the server would not know the location of the client and would return an error even though client sent the messages in correct order.

VI. COMPARISON OF PYTHON VS JAVA

There are a lot of features in Java that is not present in Python and some of those features could be the the reason why asyncio is a suitable choice for this application

The first feature is Type Checking. Python is an interpreted language and is dynamically type checked which means that the variable types are assigned during run time rather than compile time. Java is compiled and then statically typed for each instance of a variable. This can actually be an advantage for our purposes because more bugs and errors will be caught and will have less type casting issues.

Another feature comparison is managing memory issues. Both Python and Java are unlike C++ in the sense that they both have garbage collections and are automatically tracked. In python, the variables are on the heap and in Java the objects are on the heap. Keeping track of reference is a bottle neck because there is more overhead, but Java has a garbage collection method that is efficient. Memory management is abstracted and I as the developer did not have to keep track of references of variables.

The final comparison between Python and Java is multithreading. We talked a lot about this above, and the whole point of this project is to learn about the ayncio library which is Python's multithreading library. Multithreading in Java is much more efficient than Python because multithreading in Python is limited by the GIL. But that does not mean asyncio is inferior compared to Java. In this project, asyncio is a good candidate.

Multithreading is good for running and executing tasks, but asynchronous framework is good for handle a lot of networking requests. That is why it doesn't matter that Python is not good at multithreading.

VII. COMPARING ASYNCIO NODE.JS

Node and asyncio are both used to write server code in either JavaScript or Python. Similar to how we compared Java and Python, We will compare node and asyncio.

If we compare performance first, we know that in a server herd model, the central server can be the bottleneck. But this is not an issue in Node.js, and that is why it is faster.

Both asyncio and node.js are single-threaded. In node, a task is evaluated in the future on the same thread, but in asyncio, the advantage is the concept of co-routines. This allows for it to be more scalable.

VIII. CONCLUSION

Asyncio is very suitable for this project in designing a server heard. Its asynchronous features and the event loops provide us with all the capabilities

required to handle multiple requests at the same
time.