CS131 Project - Report

Abstract

This report is to summarize the research of measuring the performance of using asyncio asynchronous networking library in Python when building a server herd that can synchronize data and communicate with client applications. The research on asyncio is based on python 3.8.2. Through the performance, the paper will discuss the pros and cons of asyncio, and then analyze if it is a suitable framework to implement the Wikimedia-style service with multiple servers. Moreover, it analyzes the problems regarding type checking such as memory management and multithreading, compared to a Java-based approach to this problem. What is more, the paper will compare asyncio to Node.js. Eventually, the paper comes up with a discussion.

1. Introduction

Wikipedia is based on the Wikimedia Server platform, which is based on GNU/Linux, Apache, MariaDB, and PHP+JavaScript. Our task is to design a new Wikimedia-style service using different architecture which can support faster and higher frequency to update articles. And access will be required via various protocols, not just HTTP. What is more, the clients will tend to be more mobile.

Asyncio asyncio is a library to write concurrent code using the async/await syntax [1]. It has both high-level and low-level APIs. It used an event loop to handle the scheduling of asynchronous 'tasks'.

2. Server Implementation

2.1 Prototype

The prototype consists of five servers in the proxy herd. Their server IDs: 'Hill', 'Jaquez', 'Smith', 'Campbell', 'Singleton'. They communicate with each other as the below figure shown (Note that the communications are bidirectional):

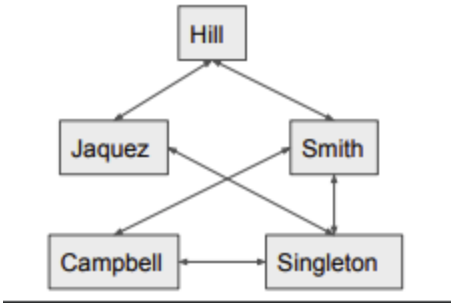

Figure 1. Server Communication (From TA's slides)

Each server should accept TCP (Transmission Control Protocol) connections from clients that emulate mobile devices with IP addresses and DNS names. One message/response pair per connection.

2.2 Commands and Messages

2.2.1 IAMAT (Client -> Server)

A client can send its location to the server by sending a message with the command name IAMAT.

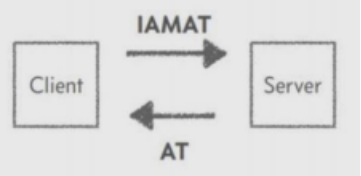

Figure 2.

2.2.2 WHATSAT

The clients can query for information about places near other clients' locations by using Google Places API.

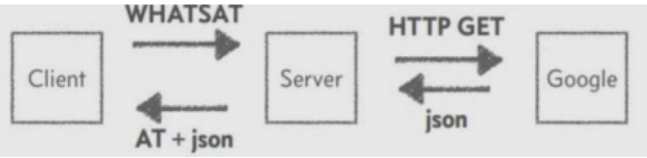

Figure 3.

2.2.3 Invalid

Any invalid command sent by the clients will be responded with a line that contains a question mark (?), a space, and then a copy of the invalid command.

3. Recommendations of Asyncio IO

3.1 Advantages

Asyncio allows the program to continue executing other instructions while waiting for specific processes to finish. It increases the efficiency of the server over a purely sequential implementation. For applications which has a large number of IO, using asyncio can bring substantial savings.

3.2 Disadvantages

The tasks do not execute in the same order they arrive. It is hard to track and avoid the race conditions.

4. Problems Met

When using the testing script for the project provided by TA, I cannot use it in my local machine (MAC). The

problem wasn't solved when I installed all the modules the project needs. To save time, I gave up running the testing in the local machine and did the testing in SEAS Net which was not so convenient.

In the project, I occasionally met a situation when I assigned two different port numbers for two servers. (i.e., two servers share two ports) It worked and the robustness score increased 4 points, but it doesn't make sense so I didn't use it in my project.

5. Comparison to Node.js

The general semantics of Asyncio and Node.js are similar in writing asynchronous code. And they both are related to the concept of asynchronous IO.

Node.js is a server-side platform for JavaScript which also uses asynchronous programming. Similarly, Node.js also uses event loops and coroutines in implementation, it uses a single threaded model with event looping. It uses a non-blocking, event-driven I/O model. Although it is usually running slower than asyncio, Node.js is best suited for asynchronous programming.

Asyncio is a Python library, it is friendlier to junior developers. It usually has better performance speed than node.js. One interesting thing is if using uvloop (a drop-in replacement of the built-in asyncio event loop), asyncio could be faster, which is at least 2x faster than node.js [5].

6. Discussion

6.1 Type Checking

Python is a dynamically typed language, the type of a variable is allowed to change over its lifetime. Java is a compiled and statically typed language.

6.2 Memory Management

Python and Java both offer automatic memory management. In Java, memory is managed by JVM. In Python, the objects are labelled and referenced by the variables. It could have unused variables but they would be garbage collected.

6.3 Multithreading

Java has better performance in multithreading. Multithreading in Python is limited by GIL, or Global Interpreter Lock. Unlike Java, Python is effectively single-threaded-it can only run on a single CPU core at a time.

5. Conclusion

Based on the analysis above, it could be concluded that Asyncio is suitable for the kind of application in this project in designing a server heard. Writing asyncio-based programs is not that easy as expected.

Reference

[1] asyncio — Asynchronous I/O. *Python Software Foundation.,* 2020.
https://docs.python.org/3/library/asyncio.html#module-asyncio

[2] Brett Cannon. *How the heck does async/await work in Python 3.5?* 11 Feb 2016.
https://snarky.ca/how-the-heck-does-async-await-work-in-python-3-5/

[3] Mark McDonnell. *Guide to Concurrency in Python with Asyncio.* Nov 30, 2019
https://www.integralist.co.uk/posts/python-asyncio/

[4] Andrei Notna. *Intro to Async Concurrency in Python vs. Node.js.* Feb 5, 2019.
https://medium.com/@interfacer/intro-to-async-concurrency-in-python-and-node-js-69315b1e3e36

[5] Michael Flaxman, *Python 3's Killer Feature: asyncio.* Jun 21, 2017
https://eng.paxos.com/python-3s-killer-feature-asyncio#:~:text=%E2%80%9Cuvloop%20makes%20asyncio%20fast.,throughput%20on%20the%20same%20hardware.