Implementation of Proxy Herd with Asyncio Project

1. Abstract

When it comes to create websites, majority of them based on the Wikimedia server platform which is then based on using multiple web severs behind a load-balancing virtual router and caching proxy servers for reliability and performance. In this project, we propose to look into a different architecture called an “application server herd” where the multiple application servers communicate directly to each other as well as through the core database and caches. After implementation, we have analyzed the advantages and disadvantages of using asyncio features introduced in Python 3.8.

2. Introduction

The Wikimedia infrastructure works fairly well for Wikipedia as an example. However, if we consider to build a new style of service that is aiming for three different goals, where updates to articles that will happen far more often, access will be through various protocols, not just HTTP and clients tend to be more mobile, this approach will become a central bottleneck. As a result, we propose to implement a server herd which allow clients to be able to request a list of close attractions from any of the available servers in order to communicate with each other by providing the locations. This improves the performance by speeding up the situation when both clients need to connect to different servers and updates need to be processed frequently.

In terms of analysis, we will consider the problem regarding type checking, memory management, multithreading between Python and Java. Second, we will also look into the difference between asyncio and Node.js by showing the performance implications of using asyncio.

3. Explanation of Implementation

3.1 Architecture of the Server

In this project, we are working with 5 different servers called Hill, Jaquez, Smith, Campbell, Singleton. Each process is a separate process and communicate to certain nearby servers. Hills talks with Jaquez and Smith. Singleton talks with everyone else but Hill. Smith talks with Campbell. Client can send their current locations to any server using TCP protocol via the usage of flooding algorithm. Therefore, every server receives the message, even if it is not directly connected to the original server. Each server is able to accept multiple connections and communication at the same time by processing connection and messages asynchronously. This prevent slow down of the processing by the usage of adding task into the event loop. Client can send two types of messages which are “IAMAT” and “WHATSAT" message.

3.1 IAMAT Message

IAMAT messages are mainly for client tells the server where it is so that the server saves the location and propagates it. The format of the message : IAMAT kiwi.cs.ucla.edu +34.068930-118.445127 1520023934.918963997. The first field is the name of the command. The second field is the client ID. The third field is the client coordinates while the fourth field is timestamp in POSIX.

Server will store the corresponding information into the client dictionary after receiving a correct IAMAT message. The server will store the message, the received time of the message and the server name into the client dictionary. After receiving the IAMAT message, the server will respond to client with AT message using this format :
AT Hill +0.263873386 kiwi.cs.ucla.edu +34.068930-118.445127 1520023934.918963997.
The first field is the same of the response, the second field is the ID of the server that receive the message from client, the third field is the difference between the server’s ideas of when it receive the message from the client and the client’s time stamp and the remaining fields are a copy of the IAMAT data which is client ID, client location and sent time of the message.
Server will also send a UPDATE message to every server that it actually communicate in order to update the client location at each related server.

3.2 WHATSAT Message

WHATSAT message are mainly for client to ask what is near one of the clients. Client can query for information about places near other cliens’ locations with a query. The format of the message is WHATSAT kiwi.cs.ucla.edu 10 5. The arguments to a WHATSAT message are the name of another client (kiwi.cs.ucla.edu), a radius (in km) from the client, and an upper bound on the amount of information to receive from Places data within that radius of the client (5). The radius must be at most 50km and the information must be at most 20 items based from the places API supports conveniently.

The server will respond with a AT message in the same format as before after receiving a correct WHATSAT message. The AT message will consist of the most recent location reported by the client, along with the server that it talked to and the time the server did the talking. The server will send a nearby search request to extract the locations based from client’s given location, radius and the maximum number of requests. The Google Places API will try to look up and search for places that are nearby according to the client’s location within certain radius. In order to receive the output in

JSON format, the server will try to send a asynchronous call to the Google Places API in order to extract the first certain number of the results which are specified by the client.

3.3 UPDATEMSG message
The UPDATEMSG message are intended for servers to communicate with each other in order to update the client's location according to the communication relationship between each server with the rest of the servers. The format of the message : UPDATE ClientID NewLocation SentTime ReceiveTime ServerReceiveTime. The first field is the name of the command ,the second file is the name of the client, the third field is the location of the client, the fourth field is the time when the client send out the message at that specific location, the fifth field is the original server received time, and the sixth field is the server name that received the message.

In order to prevent infinite loop, the server will compare and check both the client ID and the timestamp in the received message. In this project, I have chosen to create a dictionary to store the client information. When the server receive the UPDATEMSG message, it first check through the client dictionary to see if it is an existing client already. If it is not an existing client, it will store the client information in a new entry in the client dictionary. After storing, the server will propagate this message to all the other servers that it communicate according to the given communication relationship. If it is an existing client, then the server will compare between the message timestamp and the stored timestamp. If the message timestamp occurred at a later time, the server will update the corresponding location of the client. After the update, the server will repeat the above steps to sent out the message to all the servers that it has a legit communication relationship. If the message timestamp occurred at an earlier time, we will discard and ignore the message . It is because it is likely that the server has already received this message before or the message is really outdated and does not provide any useful information about the client.

4. Advantages and Disadvantages of using asyncio
4.1 Advantages

First, The server is able to handle multiple requests altogether because asyncio allows the server to run the requests asynchronously. This means that every single server represent as an even loop and each message, connection represent as a co-routine. When there are new messages or connections coming in, the server can just insert the new coming co-routine into the event loop and the newly added co-routine is not executed until it move forward at the front position of the event loop. As a result, the server is able to process current request and insert a new co-routine at the same time. So, it works perfectly with our server herd architecture based on our goal. Second, each unique server is able to reuse the same kind of code under the asyncio features. Whenever we want to add a new server into the existed connection, we just simply running the same command and code from the existed ones. Third, aiohttp works well with asyncio when it comes to create a HTTP request to the Google Places API in order to search and look up information. It is because aiohttp is also asynchronous which means it allows us to both process the current request and add any new incoming message or connection into the event loop, the newly added co-routine won't executed until it is their turn at the front of the event loop. Another characteristics that aiohttp works well with aiohttp is provide the ability to create a request based on the given url and retrieve the output in JSON format for the client.

4.2 Disadvantages

First, one of the disadvantage is the incoming message or connection are guarantee to be executed in the order they arrive. If the client first sends a correct IAMAT message and then sends a correct WHATSAT message. It is possible that server will choose to process the WHATSAT message first and process the IAMAT message second. The order of execution of these two commands will cause the server to deliver a error message to the client and hinting the incorrect order of sending the necessary information even the client does follow the rules. The order of execution of command problem become a much more serious issues as the more and more servers are connected together. Imagine we are currently sending task within three servers which are Hill and Campbell. If the client first sends a correct IAMAT message to Hill and the client sends a correct WHATSAT message to Campbell, it takes time to flood the IAMAT message to all the related server in terms of communication relationship. Since server Hill and server Campbell are not directly connected, it is possible that the client could get a error message from Campbell due to the fact that the IAMAT message is still the process of propagation despite of setting a period of delay. In comparison with synchronous architecture, asynchronous architecture such as ascyncio, the reliability is not as well as synchronous ones. In returns, architecture like ascyncio has better performance since it avoids the bottleneck and able to handle multiple request at the same time.

One of the bad thing it comes along is it also introduce bugs such as the ones describe from above for the programmer to catch and solve. Second, each server in this project is sharing the same piece of code, it creates certain problems when we need to modifying the code. Since all of the servers are not expected to store the current processing information before they are accidentally shutdown, each server need to be shut down and restart again in order to resend the client information that is processing previously. As we can imagine as we have more and more servers in the connection, this process need to be applied when one of them shut down unexpectedly. Therefore, it could be considered as one of the inconvenient issue.

Lastly, asyncio does not support multithreaded servers and this will definitely lead to performance issue. It is because asyncio architecture is based on single threaded which means it could only process one task each time. In terms of this project, this might not be an issue because we do not need to handle a large amount of traffic between clients and servers. However, multithread program could avoid this problem by creating a new threads for each newly incoming tasks and connection. Each of the thread could be run parallel with the rest of the existing threads. As we can imagine, as the traffic gets heavier on a specific server, it outweighs the benefit of simplicity of usage of asyncio, performance will definitely suffers due to busy traffic. Even though we can add more servers to redistribute the traffic, all the requests are specifically direct to one server intentionally, the problem is still unsolved at the end of the day.

5. Comparison between Python and Java

The following explanation will demonstration the differences between Python and Java in terms of the implementation approach on this project in three different fields which are type checking, memory management and multithreading.

5.1 Type Checking

Python is dynamic type checking which means that the types are being checked during runtime. Java is static type checking which means that types are being declared and checked during compile time. When it comes to set up a server, it is simpler to use Python to accomplish that. It is because we as programmer do not need to understand or know all the return types of from all the function calls that we end up using. As an example, it is necessary to know the types of both reader or writer when we are using these two functions which are reader.read() and writer.write(). Another example, as a programmer, we do not need to know the type that asyncio.start_server return in order to use it and make it run until the user have provided the quit command. In terms of Java implementation to this project, we will need to understand and know all the return types for the function that we are gonna use beforehand. It might require more time for the programmer who are not familiar with Java before in order to accomplish the project. However, this serves as a benefit in the long run because it makes it easier for other programmers to figure out the flow of the implementation due to the explicit specified return types. Even though Python makes it easier to start and set up the frame of the project, it makes it harder for someone who does not involve in the project to perfectly understand the underlying content and details at the first look of the project.

5.2 Memory Management

Both Python and Java have garbage collector. Python has a private heap that is reserved for the data structures inside Python libraries. Python's garbage collector is called reference count method while Java's garbage collector is called mark and sweep method.

For Java, when an object is created, the marked bit is set to 0. Therefore, we will set the mark bit of all reachable object to 1 during the mark phase. Then, it requires a graph traversal to visit all the reachable node from the current node until we have visited all the nodes. For the seep phase, it will clear the heap for all the unreachable objects whose mark bit is set to false. After the sweep, we set all the reachable objects back to false again, and rerun the process again to mark all the reachable objects to repeat if necessary.

For Python, each object has a reference count which is referring as the total number of times that the object is being copied, deleted or assign a variable. Therefore, it is safe and correct to delete and remove the object when the reference count is zero. It is because it means there are no objects or variables that need to reach to the object for some kinds of reference. The advantage is one can delete the object and reclaim the memory that the object used immediately. But, it has performance overhead because we need to update the reference count when we have any new assignment. Also, Python does not handle circular references which means a parent is pointing to a child while the child point back to the parent. This might end up consuming more memory than usual and causing the program to run slower. However, we do not need to worry about this problem in this project because the variables that are created in our project is usually being used once or twice. As a result, memory is return immediately serve as a benefit in our project indeed and it is not necessary to worry too much about the circular references because they don't appear that often and also Python have an addition garbage collector aside from the main one that periodically free the memory.

5.3 Multithreading

Python does not have multithreading which means it only allows one thread to be execute at a time despite of having a multicore processor. We still need to wait for one thread to finish its own execution. Java has multithreading which means two or more parts could be run parallel while each of the thread is processing a different task from each other in order to make optimal usage of multiple processors. This will definitely process more request and have higher throughput within the same amount of period of time. As a result, it is not the best option to implement the server in Python in terms of lack of multithreading, it will definitely hurt the performance and throughput in the long run while it could not fully utilize the available multicore processor to increase the workload.

6. Comparison between asyncio and Node.js

Asyncio involves using Python when it comes to implementation of this project while Node.js mainly involves using Javascript in terms of code. We will be discussing the differences in the following three different fields which are performance, accessibility and concurrency.

6.1 Performance

Node.js is a Javascript runtime which is build based on Chrome's V8 Javascript engine and it uses an non blocking input output model which makes it efficient and lightweight. In terms of performance, there is possibility that the Wikimedia-style service will be bottleneck. As as result, Python is slower than Node.js in performance. It become even slower when Python is required to handle memory intensive work or task. So, Node.js is faster in terms of processing speed of tasks.

6.2 Accessibility

Both Python and Javascript are pretty much easy to use in terms of their similar characteristics. It is convenient to use either one of them to set up the server or start development on any specific project. However, Javascript might be more popular in terms of web development compared with Python. As a result, it might be a better idea to use Javascript when it comes to the backend.

6.3 Concurrency

According to above, asyncio run based on a asynchronous architecture with single threaded. It turn out that Node.js also run based on asynchronous architecture with single threaded called promise. Basically, it allows asynchronous method return promise for the value at some point in the future. Overall, two of the them are pretty much the same while asyncio stop the current executing process and pick up the information again along with updated and extra content provide a bit more freedom in terms of flexibility.

7. Problems

The major problem that I have encountered in this project is I have a hard time to understand implementation of the flooding algorithm. It is because I am not quite sure what is the correct situation to flood the received message or not. Then, I start to draw out a scenario on a piece of paper and simulate what happen when there are update message coming in. After that, I understand that it depends whether the server have seen that client before or not and the action deviate from there. If the server have not seen the client before, then store it and propagate. If the server have seen the client before, then compare the timestamp and store the updated one, and the propagate, or else ignore the outdated message.

8. Conclusion

According to our detail analyze of the project implementation, asyncio works well with Python in terms of the design of the our project goals which is an application herd with multiple servers that communicate with each other back and forth despite of some of the disadvantages it brings along in the implementation. As a result, asyncio would be considered as an good option when it comes to this project implementation.

9. Sources

https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/

https://www.geeksforgeeks.org/mark-and-sweep-garbage-collection-algorithm/

https://medium.com/dev-bits/writing-neat-asynchronous-node-js-code-with-promises-32ed3a4fd098

https://docs.python.org/3.6/library/asyncio-eventloop.html#run-an-event-loop

https://realpython.com/python-memory-management/

https://www.geeksforgeeks.org/java-memory-management/

https://docs.aiohttp.org/en/stable/client_quickstart.html#json-response-content

https://docs.python.org/3/library/asyncio.html

https://hackernoon.com/python-vs-nodejs-which-programming-language-to-choose-98721d6526f2

Implementation of Proxy Herd with Asyncio Project

https://www.freecodecamp.org/news/nodejs-vs-python-choosing-the-best-technology-to-develop-back-end-of-your-web-app/

https://hackernoon.com/concurrent-programming-in-python-is-not-what-you-think-it-is-b6439c3f3e6a