***Aysncio Server Herd Prototype***

Computer Science 131, Spring 2019

Professor Paul Eggert

Taylor Lee, *604790788*

Abstract

This report investigates the practicality, efficiency, and feasibility of using the asyncioPython module as a framework for a project such as the LAMP Wikimedia Architecture. After comparing Python’s type checking, memory management, multithreading, and ease of use with other frameworks such as Java’s Node.js, I recommend asyncio as a viable server herd framework.

1. Introduction

The Synchronized state is safe but slow. This state is consistently reliable and is a DRF, simply because the *synchronized* keyword does not allow for multi-threading to happen within the entire class. Multi-threading does not increase the performance or speed of this state.

2. Sample Server Herd Prototype

I ran the same test cases on two different version of Java: Java 9 and Java 11.0.2. Based on the data collected from the two different versions, it is clear that the latest version optimizes multi-threading to a higher degree than Java 9. Additionally, Java 9 resulted in more *sum mismatch* errors or infinite loops.

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.

2.1 Individual Server Design

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.

2.1.1 Inter-Server Connections

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.

2.2.Client Prototype

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.

2.3 Client Requests

2.3.1 *IAMAT*

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.

2.3.2 *WHATSAT*

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.

2.4 Special Conditions

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.

2.5 Documentation & Logging

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.

3. Evaluation of asyncio

Additionally, it should be noted that *BetterSafeState* optimization reaches greater performance at 8 threads compared to 16 threads. This is most likely due to the fact that for small sample cases, a high level of threads will not increase performance to the degree it would with a large volume of sample cases. Generally speaking, the higher the amount of swap cases, the greater the margin of performance optimization *BetterSafeState* had compared to the other classes.