|  |
| --- |
|  |
| **차량 데이터 시뮬레이터 개발을 위한 아키텍처 조사** |
| 언어별 특징을 비교하여 사용할 아키텍처 결정 |
|  |
| 인턴 이 다 솔 |
|  |
| 2018-09-17 |

|  |
| --- |
|  |

목차

**1. 개요1**

**2. 언어별 특징1**

2.1. Python1

2.2. Golang2

2.3. Java2

2.4. Node.js3

**3. 언어별 비교4**

3.1. Python vs Golang4

3.2. Java vs Node.js4

3.3. Java vs Golang5

3.4. Java vs Node.js vs Golang5

**4. 결론6**

# 1. 개요

본 문서는 차량의 샘플 데이터를 생성하는 시뮬레이터를 개발하기 위해 사용할 아키텍처를 선정하기 위해 작성되었다. 아키텍처의 선정 기준은 시뮬레이터 구동에 필요한 멀티 프로그래밍의 성능을 주로 하며, 팀원들의 언어친숙도와 라이브러리 및 자료의 접근성을 부가적으로 고려하였다. 본문에서는 Python, Golang, Java, Node.js의 4가지 언어에 대하여 언어별 특징과 멀티 프로그래밍 성능을 비교한 내용, 그리고 그에 따라 선정한 아키텍처를 서술한다.

# 2. 언어별 특징

## 2.1. Python

Python은 멀티쓰레드 방식보다 멀티프로세스 방식이 선호된다.

Python의 멀티쓰레드 프로그래밍은 GIL 때문에 성능이 저하되는 경우가 있기 때문이다.

- GIL (Global Interpreter Lock) :

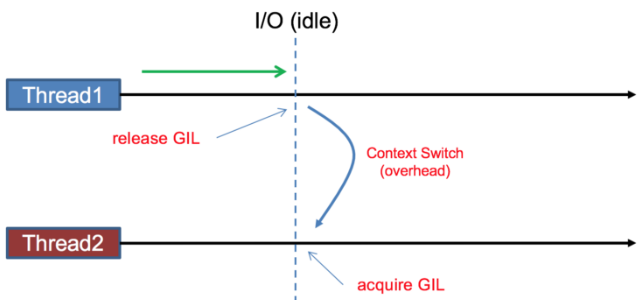
자원 배분을 한 쓰레드에 할당, 끝날 때까지 다른 쓰레드가 접근하지 못하게 하는 것이다.

- 예시)

Thread1이 자원을 사용 중인 경우, GIL을 Thread1이 잡고 있으므로 다른 쓰레드는 자원을 점유

할 수 없다. Thread1이 파일/네트워크 I/O 등의 I/O를 발생시키면 Thread1은 GIL을 release

하므로, 대기 중이었던 Thread2가 GIL을 잡고 자원을 점유하게 된다.



출처) https://medium.com/@mjhans83/python-gil-f940eac0bef9

→ 쓰레드가 원하는 시점에서 GIL을 획득하지 못해 실행이 지연될 수 있고, 그 때문에 쓰레드

개수를 늘릴수록 코드 실행 속도가 느려지는 현상이 발생하기도 한다.

위와 같은 이유로 파이썬에서는 ‘multiprocessing’ 모듈을 사용하여 멀티프로그래밍을 수행한다.

시뮬레이터 개발에서 multiprocessing 모듈을 사용하는 경우, 프로세스를 여러 머신에 뿌리기 위한 라이브러리로 ‘Parallel Python’ 을 사용할 수 있다.

하지만 멀티 ‘프로세싱’ 인만큼 멀티쓰레드 프로그래밍에 비해 무거워진다는 단점이 있다.

## 2.2. Golang

Golang은 병렬 프로그래밍에 특화되어 개발된 언어이다. 언어차원에서 동시성과 병렬성을 지원하며, 최대 프로세서 개수 설정으로 멀티 코어 환경을 지원한다.

동시성을 제공하기 위해 GoRoutine을 제공하며, 각 GoRoutine은 병렬로 동작한다.

이는 일종의 pipe인 채널을 통해 통신하며, 하나의 GoRoutine은 2KB의 스택공간을 사용한다.

이는 1MB를 사용하는 쓰레드에 비해 현저히 적은 메모리를 차지하여 메모리에 부담이 적어진다.

GoRoutine은 ‘go’ 키워드와 함께 함수를 작성하면 실행되어 간단하게 구현이 가능하다.

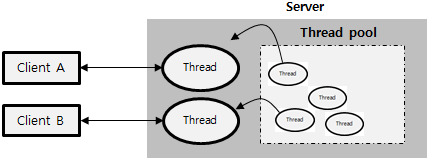
멀티 프로그래밍의 측면으로 보면 가장 좋다고 할 수 있지만, 자료나 자체 라이브러리가 부족하다는 단점이 있다.

## 2.3. Java

대중적으로 사용되는 언어인만큼 Golang과 반대로 자료나 자체 라이브러리가 풍부한 언어라는 장점이 있다.

Java를 개발 언어로 채택 시, Spring Boot를 사용하여 WAS의 멀티쓰레드 관리 방식을 사용한다.

일반적인 WAS의 동작 형태는 아래와 같다.



출처) http://bcho.tistory.com/881

- 클라이언트 측에서 요청이 오면 쓰레드 풀에서 쓰레드를 꺼내서 클라이언트의 요청을 처리한다.

요청이 끝나면 쓰레드 풀로 돌려보내고, 이를 반복한다.

- 생성 가능한 쓰레드 수에는 한계가 있고, 곧 동시에 처리할 수 있는 클라이언트 수에도 한계가

있음을 의미한다.

- 클라이언트에 할당된 쓰레드는 I/O (DB, Network, File) 작업이 있을 경우 호출만 하고, CPU를 사용하지 않는 wait 상태로 변경된다. ( → 이를 ‘block’ 이라고 한다. )

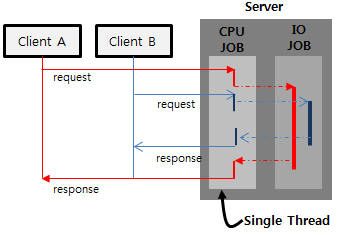
## 2.4. Node.js

Chrome V8 javascript 엔진을 기반으로 싱글 쓰레드 기반의 even loop(libuv) 가 돌면서 요청을 처리하는 언어이다. 시스템적으로 non-blocking I/O를 지원하지 않는 호출이 있는 경우, 이를 비동기 처리하기 위해 내부의 쓰레드 풀(libio) 를 별도 이용하여 처리하기도 한다.

- 비동기식 I/O 처리 : 작업 요청 시 요청이 끝나면 호출될 callback 함수를 같이 전달한다.

→ 작업 요청이 접수되면, 요청된 작업이 끝나기를 기다리지 않고 다른 작업을 진행한다.

→ 요청된 작업이 완료되면 같이 넘겼던 callback 함수를 호출하여 작업 완료를 알린다.



출처) https://wckhg89.github.io/archivers/what\_is\_node

→ block 되지 않아서I/O 작업시 기다리지 않기 때문에, 하나의 쓰레드가 다른 요청을

받아서 작업 처리가 가능하다. 즉 여러 클라이언트로부터 오는 요청을 처리 가능하다.

→ 요청을 받아서 처리하는 쓰레드를 ELP (Event Loop Thread) 라고 한다.

Node.js는 비동기식 I/O 처리를 사용하기 때문에 파일 업/다운로드와 같은 네트워크 스트리밍 서비스에 유리하고, 실시간 웹 애플리케이션(socket.io) , 단일 페이지 앱 개발에 좋다.

하지만 다른 요청 처리가 지연되기 때문에 CPU 작업이 많은 애플리케이션에는 부적절하다는 단점이 있다.

Node.js로 시뮬레이터 구현을 한다면 worker를 늘려 병렬로 동작시키는 방법을 사용해야 한다. worker를 직접적으로 컨트롤 해야하는 경우는 child\_process, 로드밸런싱 등의 기능이 필요한 경우 cluster를 사용하여 worker를 생성하는 것이 좋다.

시뮬레이터를 개발하기 위해서는 worker를 직접 컨트롤 할 경우는 없기에 좀 더 가벼운 후자의 방법을 채택하는 것이 적합하다고 판단된다.

# 3. 언어별 비교

## 3.1. Python vs Golang

python 멀티프로세싱 모듈과 Golang GoRoutine 으로 피보나치 수열을 계산하여 성능 비교

출처) <https://lincolnloop.com/blog/concurrency-python-vs-go/>

python : 275 req/sec 188MB of RAM

Go : 12,500 req/sec 120MB of RAM

위와 같은 성능 차이로 Go가 요청처리 횟수, 메모리 부분에서 Python보다 우수함을 보인다.

## 3.2. Java vs Node.js

Java와 Node.js의 동시성 처리와 성능 비교

출처) <https://www.linkedin.com/pulse/nodejs-vs-java-which-faster-apis-owen-rubel>

Java : 4,739 req/sec → 21sec 100K req 80 concurrency on 1.6GHZ machine

Node.js : 1,827 req/sec → 54sec 100,000 req 80 concurrency on 1.7GHZ machine

위와 같은 성능 차이로 Java가 요청처리 횟수 부분에서 Node.js보다 우수함을 보인다.

## 3.3. Java vs Golang

Java와 Golang의 쓰레드 생성 성능 비교 ( 5,000,000번 )

출처) <https://www.slideshare.net/lahuman1/java-go>

Java : sec - 32.81 / memory – 36,148 / CPU 36.20

Go : sec - 1.50 sec / memory – 2,360 / CPU 1.49

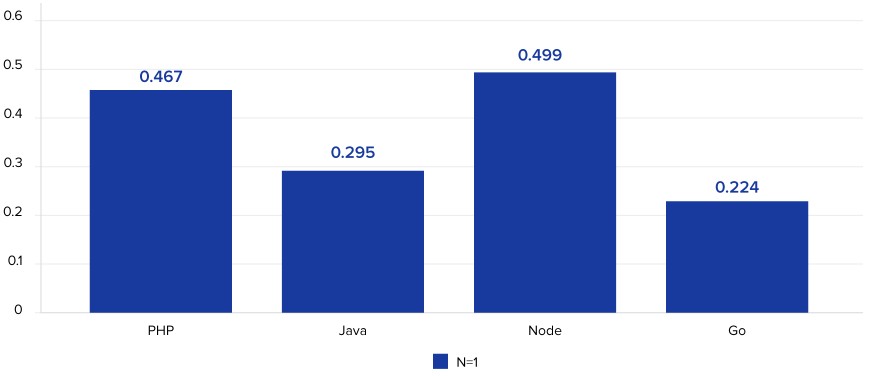
위와 같은 성능 차이로 Go가 Java보다 우수함을 보인다.

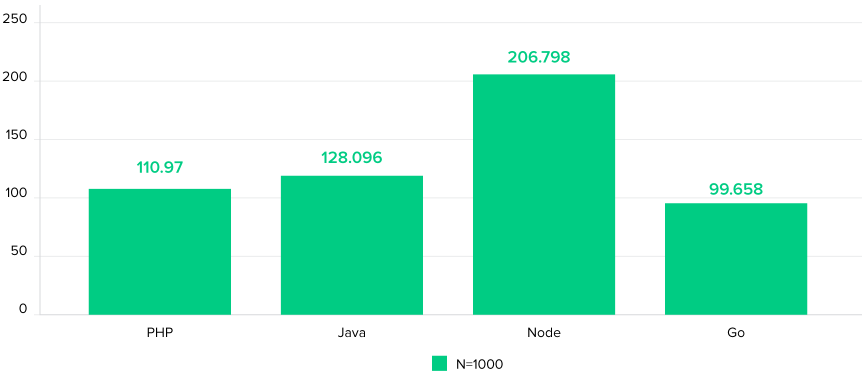
## 3.4. Java vs Node.js vs Golang

Java, Node.js, Golang 서버 환경에서 HTTP 서버 성능을 처리시간 기준으로 비교

→ 랜덤한 내용의 64KB 파일을 읽고, SHA-256으로 N번 해싱하여 결과를 출력하는 과정이다.

출처) <https://www.toptal.com/back-end/server-side-io-performance-node-php-java-go>





위와 같은 성능 차이로 처리 시간을 기준으로 Go > Java > Node.js 순으로 우수함을 보인다.

# 4. 결론

조사한 네가지 언어를 비교하여 아래와 같은 순서로 개발에 적합한 언어가 선정되었다.

1. Golang 2. Java

Golang이 병렬 처리에 뛰어나고 메모리 부담이 적다는 장점이 있고, Java는 CPU 점유율이나 메모리 소모에 있어서 Golang보다 약간 성능이 떨어진다. 특징만을 놓고 본다면 Golang을 선정하는 것이 맞겠지만, 2달의 짧은 개발 기간과 성능에 큰 차이가 없는 점을 고려하여 팀원들에게 조금 더 익숙한 Java와 Spring Boot를 함께 사용하는 것이 좋다고 판단된다.