with spring

Auto Store

상품 등록 자동화 시스템 및 서버 구현

API 서버 구현 및 Spring MVC와 Spring Webflux 성능 비교 분석

20192883 강승민

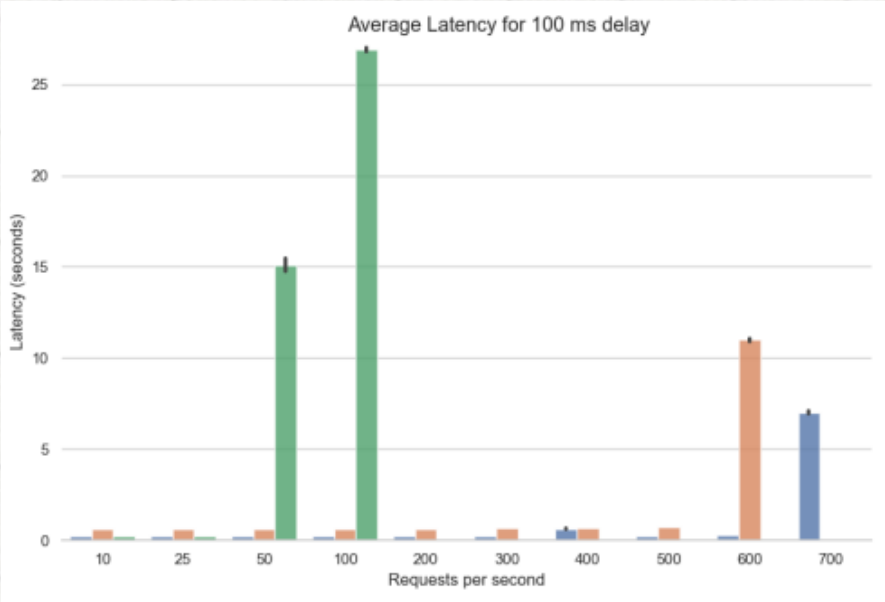
INDEX

주제 선정 배경

인터넷 사용자 및 연결된 기기의 급속한 증가는 높은 반응성과
확장 가능한 **웹 애플리케이션에 대한 수요를 증가**시켰다.

동기식 웹 개발은 높은 동시성과 처리량을 부담하기 어려워,
새로운 **Non-blocking 및 비동기적 기법을 탐색**하게 되었다.

문제 인식

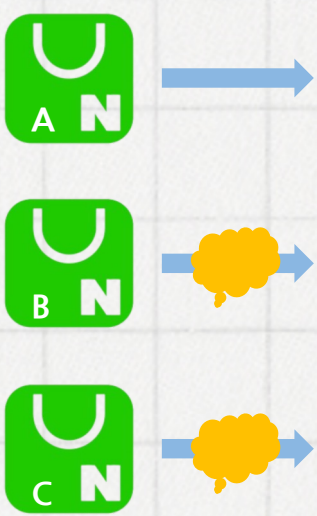


100ms 지연인 경우에서의 각 프로토타입의 평균 응답 시간을 나타내는 그래프이다.

X축은 프로토타입에 대해 수행된 요청 수를 나타내고, Y축은 평균 지연 시간을 초 단위로 나타낸다. [3]

- normal
- reactive
- virtual

문제 인식



API 서버

Spring MVC 아키텍처(비동기)를 사용한다면,

A 스토어의 요청에 대한 응답이 완료될 때까지 B, C 스토어의 요청에 대한 **응답 지연**

문제 정의

WHY

"다수의 API 요청에 대한 응답
속도 및 처리량을 향상시키기 위해"

WHAT

"기존 ~~Spring MVC~~ 방식이 아닌,
Spring Webflux 기반 API Server를 구축할 것이다"

기술 조사 1

Spring MVC

• Blocking I/O: Spring MVC는 전통적으로 블로킹 I/O를 사용한다. 여기서 들어오는 각 요청은 요청이 처리되고 응답이 다시 전송될 때까지 서버의 스레드 풀에서 스레드를 소비한다. 스레드가 I/O(ex: 데이터베이스 액세스)를 기다리면서 차단되면 다른 요청을 처리하는 데 사용할 수 없다.

• 동기 프로그래밍 모델: Spring MVC는 각 요청이 순차적으로 처리되는 동기 프로그래밍 모델을 따른다. Spring MVC는 서버의 스레드 풀 크기를 늘려 많은 수의 동시 요청을 처리할 수 있지만, 부하가 극도로 높은 경우 확장성 제한에 직면할 수 있다.

• 사용 용이성: Spring MVC는 널리 채택되었으며 전통적인 동기 프로그래밍 패러다임에 익숙한 개발자가 사용하기가 더 쉽다. [1]

Spring Webflux

• Non-Blocking I/O: Spring WebFlux는 비차단 I/O 및 이벤트 중심 아키텍처를 활용하는 반응형 프로그래밍 모델을 기반으로 한다. 스레드를 차단하는 대신 WebFlux는 소수의 스레드를 사용하여 들어오는 요청을 비동기적으로 처리한다. 이를 통해 서버는 더 적은 수의 스레드로 많은 수의 동시 연결을 처리할 수 있어 리소스 활용도와 확장성이 향상된다.

• 비동기 프로그래밍 모델: WebFlux를 사용하면 개발자는 반응형 API(ex: Flux, Mono)를 사용하여 반응형 및 비동기식 스타일로 코드를 작성한다. 이는 I/O 바인딩 작업(ex: 네트워크 요청, 데이터베이스 쿼리)과 같은 비동기 작업을 처리하는 데 유리할 수 있다.

•확장성 및 성능: Spring WebFlux는 많은 수의 동시 연결을 효율적으로 처리해야 하는 처리량이 높고 대기 시간이 짧은 애플리케이션에 적합하다. 이는 수명이 긴 연결이나 스트리밍 데이터가 있는 시나리오에서 특히 유용하다. [1]

기술 조사 2

JDBC

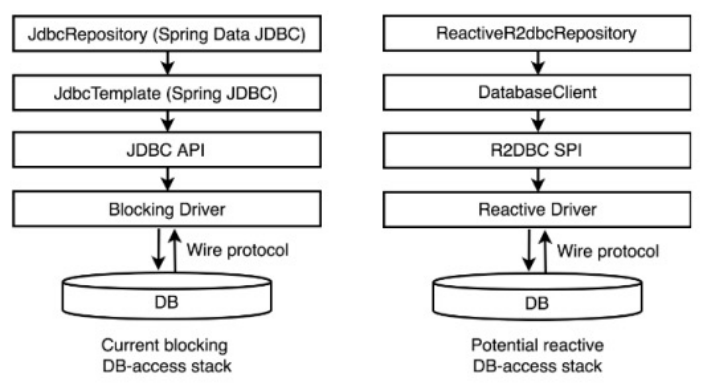
Java Database Connectivity (JDBC)의 모든 구성 요소와 기술은 JDBC API에 내장되어 있으며 구현되어 있다. 기본적으로 JDBC API는 Java 애플리케이션에서 데이터베이스와 상호 작용하는 데 사용되는 클래스와 인터페이스의 집합으로 구성된다.

JDBC API의 세 가지 주요 기능은 다음과 같다:
- Java 애플리케이션과 관계형 데이터베이스 간의 연결 설정
- SQL 문을 작성하고 실행
- 결과를 처리 [2]

R2DBC

R2DBC(R2DBC)는 관계형 데이터베이스에 대한 리액티브 프로그래밍 액세스를 제공하는 서비스 제공자 인터페이스(SPI)이다. Reactive Streams를 기반으로하여 비동기 백프레셔를 고려한 논블로킹 데이터 액세스를 가능하게 한다.

R2DBC의 목표는 다음과 같다:
- 리액티브 관계형 데이터베이스 연결 지원
- 리액티브 JVM 플랫폼과의 호환성 보장
- SQL에 집중하면서도 최소화 및 단순화 [2]



개발 환경

Spring Ecosystem

Language

IDE

Version Control

Build

Util

OS

Database

추진 일정

Mile Stone

표 | 타임라인 + 필터 정렬

API Server

완료	제목	마감일	담당자	우선순위
☑	프로젝트 주제 정하기	2024년 3월 10일	승민 경빈 고	★★★★★
☑	제안서 PPT 작성 열기	2024년 3월 16일	승민 경빈 고	★★★★★
☑	알리 익스프레스 가입하기	2024년 3월 17일	승민 경빈 고	★★★★
☑	네이버 스마트스토어 가입하기	2024년 3월 17일	승민 경빈 고	★★★★
☐	Webflux 기반 API 서버 구현	2024년 3월 31일	승민	★★★★
☐	MVC 기반 API 서버 구현	2024년 3월 31일	승민	★★★★
☐	API 서버 기능 테스트	2024년 4월 7일	승민	★★★★
☐	MVC와 Webflux 성능 테스트	2024년 4월 12일	승민	★★★★
☐	중간 결과물 제출	2024년 4월 12일	승민	★★★
☐	최종 결과물 제출	2024년 5월 24일	승민	★★★

참고 문헌

[1] https://www.theseus.fi/bitstream/handle/10024/812448/Catrina_Alexandru.pdf?sequence=2

[2] https://www.diva-portal.org/smash/get/diva2:1445480/FULLTEXT01.pdf

[3] https://www.diva-portal.org/smash/get/diva2:1763111/FULLTEXT01.pdf

Thank you

https://github.com/SoulTree-Lovers/Auto-Store