JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 30 No. 7, pp. 1-10, July 2025
https://doi.org/10.9708/jksci.2025.30.07.001

Performance Comparison of RDB and NoSQL in Real-Time Data Processing Systems Based on Spring WebFlux

Minsol Kim*, Jongha Kim*, Sehoon Lee**

*Student, Dept. of Computer System Engineering, Inha Technical College, Incheon, Korea
**Professor, Dept. of Computer System Engineering, Inha Technical College, Incheon, Korea

[Abstract]

This paper compares and analyzes the real-time large-scale data processing performance of relational databases (RDB) and NoSQL databases based on Spring WebFlux and Spring MVC, providing criteria for selecting the optimal technology when designing real-time data processing systems. WebFlux, with its asynchronous non-blocking model, offers high concurrency and low response time, whereas MVC, based on a synchronous blocking I/O model, delivers stable performance in environments with relatively fewer concurrent connections. Additionally, NoSQL outperforms RDB in terms of flexibility and scalability, exhibiting less performance degradation as the number of concurrent users increases. Experimental results show that in a Spring WebFlux environment, NoSQL achieved an average response time approximately 50% shorter than RDB. Even in an MVC environment, NoSQL demonstrated about 40% faster response times compared to RDB. Notably, when the number of concurrent users exceeded 5,000, the combination of WebFlux and NoSQL yielded the best performance.

▸ **Key words**: Spring WebFlux, Spring MVC, Performance Analysis, Response time, Concurrent users

[요 약]

본 논문은 Spring WebFlux와 Spring MVC를 기반으로 관계형 데이터베이스(RDB)와 NoSQL 데이터베이스의 실시간 대규모 데이터 처리 성능을 비교·분석하여, 실시간 데이터 처리 시스템 설계 시 최적의 기술 선택 기준을 제시한다. WebFlux는 비동기 논블로킹 방식으로 높은 동시성 처리와 낮은 응답 시간을 제공하는 반면, MVC는 동기 블로킹 I/O 모델을 기반으로 비교적 적은 동시 접속 환경에서 안정적인 성능을 보인다. 또한, NoSQL은 유연성과 확장성 측면에서 RDB보다 우수하며, 동시 접속자 수가 증가할수록 성능 저하가 적은 특성을 갖는다. 실험 결과, Spring WebFlux 환경에서 NoSQL은 RDB 대비 평균 응답 시간이 약 50% 더 짧았으며, MVC 환경에서도 NoSQL이 RDB보다 약 40% 더 빠른 응답 속도를 보였다. 특히, 동시 접속자가 5000명 이상일 때 WebFlux와 NoSQL의 조합이 가장 우수한 성능을 나타냈다.

▸ **주제어**: Spring WebFlux, Spring MVC, 성능 분석, 응답 시간, 동시 접속자

• First Author: Minsol Kim, Corresponding Author: Minsol Kim, Jongha Kim
*Minsol Kim (minisol206@naver.com), Dept. of Computer System Engineering, Inha Technical College
*Jongha Kim (gbr369369@naver.com), Dept. of Computer System Engineering, Inha Technical College
**Sehoon Lee (seihoon@inhatc.ac.kr), Dept. of Computer System Engineering, Inha Technical College
• Received: 2025. 04. 11, Revised: 2025. 04. 24, Accepted: 2025. 06. 23.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

최근 실시간 데이터 처리 기술의 발전은 다양한 산업에서 데이터 스트리밍 시스템[1]의 중요성을 부각시키고 있으며, 이는 기업의 의사결정 및 서비스 품질 향상에 중요한 역할을 하고 있다. 특히, 스마트 IoT 기기[2], 금융 거래 시스템[3], 온라인 스트리밍 서비스와 같은 다양한 애플리케이션[4]에서는 대량의 실시간 데이터를 빠르게 처리하고 분석하는 것이 필수적이다. 이에 따라, 고성능 데이터 처리 프레임워크와 효율적인 데이터베이스 시스템을 선택하는 것이 점점 더 중요한 연구 주제가 되고 있다.

현재 대다수의 실시간 데이터 처리 시스템은 Spring 프레임워크 기반으로 개발되며, 대표적으로 Spring WebFlux와 Spring MVC가 많이 사용된다. Spring WebFlux는 비동기 논블로킹 방식으로 높은 동시성을 제공하는 반면, Spring MVC는 동기 블로킹 모델을 기반으로 안정적인 성능을 보인다. 이 두 프레임워크는 서로 다른 아키텍처적 특징을 가지며, 특정 환경에서의 성능 차이가 존재한다.

데이터베이스 시스템은 실시간 데이터 처리 성능에 중요한 영향을 미친다. 관계형 데이터베이스(RDB)는 데이터 정합성과 복잡한 쿼리 처리에서 강점을 가지지만, 높은 동시성을 요구하는 환경에서는 성능 저하가 발생할 수 있다. 반면, NoSQL 데이터베이스는 유연한 스키마와 뛰어난 확장성을 제공하여 대규모 실시간 데이터를 처리하는 데 유리한 특성을 가진다.

국내외 연구를 살펴보면, Spring WebFlux와 RDB의 성능을 비교한 연구는 존재하지만, Spring WebFlux 환경에서 RDB와 NoSQL의 성능을 비교 분석한 연구는 상대적으로 부족하다. 기존 연구들은 주로 Spring MVC와 Spring WebFlux의 성능 차이를 다루거나, 특정 데이터베이스의 성능 최적화[5]에 초점을 맞추는 경향이 있다. 그러나 WebFlux의 비동기/논블로킹 특성과 NoSQL의 확장성이 결합될 경우, 실시간 데이터 처리 시스템에서 어떠한 성능적 이점을 제공할 수 있는지에 대한 연구는 아직 미비하다.

본 연구는 이러한 연구 공백을 해결하고자, Spring WebFlux와 Spring MVC 환경[6]에서 RDB와 NoSQL의 성능을 비교 분석하는 것을 목표로 한다. 이를 통해 실시간 데이터 처리 시스템에서 최적의 데이터베이스 및 프레임워크 조합을 도출하고, 기업 및 개발자들이 실질적인 기술 선택 기준을 수립할 수 있도록 기여하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 Spring WebFlux, RDB, NoSQL, 그리고 실시간 데이터 처리 시스템과 관련된 선행 연구를 검토한다. 3장에서는 실험 방법론과 시스템 아키텍처를 설명하며, 4장에서는 실험 결과를 분석하고 논의한다. 마지막으로, 5장에서는 본 연구의 결론과 향후 연구 방향을 제시한다.

II. Technological Background

1. Spring WebFlux and Reactive Programming

전통적인 웹 애플리케이션은 동기식(Blocking) I/O 모델을 기반으로 하며, 요청이 들어올 때마다 새로운 스레드를 생성하여 응답을 처리하는 구조를 가진다. 이러한 방식은 일정 수의 동시 요청을 처리하는 데에는 효과적이지만, 대량의 요청이 발생할 경우 스레드 생성 및 관리의 오버헤드로 인해 성능 저하가 발생할 수 있다. 특히, I/O 대기 시간이 긴 경우(예: 데이터베이스 쿼리, API 호출)에는 불필요한 리소스 낭비가 발생할 가능성이 크다. 이러한 문제를 해결하기 위해 비동기 및 논블로킹 방식의 웹 애플리케이션 아키텍처가 등장하였으며, Spring WebFlux는 이를 기반으로 하는 대표적인 프레임워크[7]이다.

Spring WebFlux는 Spring 5에서 처음 도입된 리액티브 웹 프레임워크로, 전통적인 Spring MVC와는 달리 이벤트 기반의 논블로킹 프로그래밍 모델을 제공한다. WebFlux는 Reactive Streams 표준을 따르며, 내부적으로 Project Reactor를 활용하여 Mono 및 Flux 데이터 타입을 제공한다. Mono는 단일 요소를, Flux는 다중 요소를 비동기적으로 처리할 수 있도록 지원하는 리액티브 타입이다. 이를 통해 WebFlux는 I/O 바운드 작업을 효율적으로 수행하며, 적은 리소스로도 높은 동시성을 제공할 수 있다.

리액티브 프로그래밍은 비동기 데이터 흐름을 선언적으로 처리하는 패러다임으로, 주로 이벤트 기반 아키텍처와 결합하여 사용된다. Spring WebFlux의 핵심 특징은 다음과 같다.

첫째, 논블로킹 I/O를 지원하여 동기적 요청 처리 방식과 비교했을 때, 대량의 동시 요청을 처리하는 데 있어 성능상의 이점을 제공한다. 둘째, 이벤트 기반 아키텍처를 활용하여 요청을 수신하고 처리하는 동안 불필요한 스레드 점유를 줄일 수 있다. 셋째, Backpressure 제어를 통해 데이터 소비자가 감당할 수 있는 양만큼 데이터를 요청할 수 있도록 하여, 시스템의 부하를 효과적으로 관리할 수 있다.

WebFlux는 이러한 특성으로 인해 실시간 데이터 스트

리밍, IoT, 대규모 동시성 처리가 필요한 시스템에서 효과적으로 사용될 수 있다. 특히, 본 논문에서 다루는 실시간 데이터 처리 시스템에서 WebFlux의 논블로킹 구조는 높은 처리량과 낮은 지연 시간을 보장하는 데 중요한 역할을 한다. 본 연구에서는 이러한 Spring WebFlux의 특성을 기반으로 RDB와 NoSQL 데이터베이스의 성능을 비교하여 실시간 데이터 처리에 적합한 데이터 저장소 아키텍처를 분석하고자 한다.

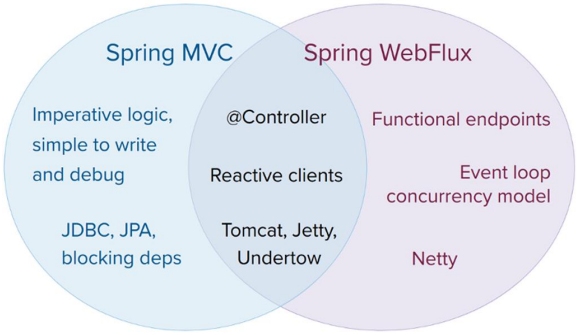

Fig. 1. Comparison of the Architectures of Spring MVC and Spring WebFlux

2. Real-Time Data Processing

실시간 데이터 처리는 지속적으로 생성되는 데이터를 즉각적으로 분석하고 처리하는 기술을 의미한다. 이는 전통적인 배치 처리 방식과 대비되는 개념으로, 데이터가 발생한 즉시 분석 및 응답이 이루어진다는 점에서 차별성을 가진다. 실시간 데이터 처리는 금융 거래 시스템, 실시간 로그 모니터링, IoT 센서 데이터 분석, 온라인 게임, 실시간 추천 시스템 등 다양한 분야에서 활용되며, 빠른 응답성과 높은 확장성이 요구된다.

실시간 데이터 처리를 위한 시스템은 일반적으로 다음과 같은 주요 요소를 포함한다.

첫째, 데이터 스트림 개념을 기반으로 한다. 데이터 스트림이란 일정한 간격으로 지속적으로 생성되는 데이터의 흐름을 의미하며, 센서 데이터, 사용자 클릭 이벤트, 주식 시장의 가격 변동 등 다양한 형태로 존재할 수 있다.

둘째, 이벤트 기반 아키텍처를 활용한다. 전통적인 요청-응답 방식과 달리, 이벤트 기반 시스템에서는 특정 이벤트가 발생할 때마다 이를 즉시 처리하는 구조를 가진다. 이는 빠른 데이터 처리를 가능하게 하며, Spring WebFlux와 같은 리액티브 프레임워크와 결합하여 강력한 성능을 발휘할 수 있다.

셋째, 낮은 지연 시간과 높은 처리량을 보장해야 한다. 실시간 데이터 처리 시스템은 지속적으로 데이터가 유입되기 때문에, 데이터가 도착한 후 이를 분석하고 응답하는 데 걸리는 시간이 최소화되어야 한다. 또한, 높은 동시 요청을 처리할 수 있도록 시스템의 확장성이 보장되어야 한다.

넷째, 확장성 및 장애 대응이 필수적이다. 실시간 데이터 처리 시스템은 초당 수십만 건 이상의 데이터를 처리해야 하는 경우가 많으며, 이를 위해 수평 확장이 가능한 아키텍처를 설계해야 한다. 또한, 장애 발생 시 데이터 손실을 방지하기 위해 데이터 복제 및 장애 복구 메커니즘이 필요하다.

실시간 데이터 처리를 구현하기 위해 다양한 기술이 활용된다. 대표적으로 메시지 브로커 기술은 데이터 스트림을 관리하는 역할을 하며, Apache Kafka, RabbitMQ와 같은 도구가 널리 사용된다. 또한, 스트리밍 처리 엔진으로는 Apache Flink, Apache Spark Streaming 등이 있으며, 대량의 데이터를 병렬로 처리하는 데 최적화되어 있다. 데이터 저장소로는 NoSQL 데이터베이스(예: MongoDB, Redis, Apache Cassandra)가 자주 활용되며, 기존의 관계형 데이터베이스(RDB)보다 높은 확장성과 처리 속도를 제공할 수 있다.

본 연구에서는 이러한 실시간 데이터 처리 시스템의 아키텍처를 기반으로, Spring WebFlux를 활용하여 관계형 데이터베이스와 NoSQL 데이터베이스의 성능을 비교할 예정이다. 특히, 두 가지 데이터베이스 유형이 실시간 데이터 처리 환경에서 얼마나 효율적으로 동작하는지를 평가하기 위해 응답시간, 처리량, 확장성 등의 성능 지표를 측정하고 분석할 것이다.

3. RDB vs NoSQL in Real-time Systems

실시간 데이터 처리 시스템에서는 지속적으로 유입되는 데이터를 빠르게 저장하고 조회할 수 있는 효율적인 데이터베이스 관리 시스템이 필수적이다. 기존의 데이터베이스는 주로 관계형 데이터베이스를 기반으로 구축되어 왔으며, 데이터의 정합성과 일관성을 유지하는 데 강점을 가지고 있다. 그러나 최근 들어 대량의 비정형 데이터가 실시간으로 생성되고 처리되는 환경에서는 NoSQL 데이터베이스가 더욱 주목받고 있다. 관계형 데이터베이스[8]와 NoSQL 데이터베이스[9]는 구조적 차이뿐만 아니라 확장성과 성능 면에서도 차이를 보이므로, 실시간 데이터 처리 시스템에서의 적절한 선택이 필요하다. 본 절에서는 두 데이터베이스 유형의 특성과 장단점을 분석하고, 실시간 데이터 처리 환경에서의 성능 비교를 위한 주요 요인을 논의하고자 한다.

3.1. Characteristics and Limitations of Relational Databases

관계형 데이터베이스는 구조화된 데이터를 저장하고 관리하기 위해 사용되며, SQL(Structured Query Language)을 통해 데이터 검색 및 조작이 이루어진다. 대표적인 관계형 데이터베이스 관리 시스템으로는 MySQL, PostgreSQL, Oracle DB, MariaDB 등이 있으며, 데이터의 무결성과 일관성을 유지하기 위해 ACID (Atomicity, Consistency, Isolation, Durability) 트랜잭션 모델을 따른다.

관계형 데이터베이스의 가장 큰 장점은 데이터 정합성과 무결성의 보장이다. 이를 위해 RDB는 스키마를 기반으로 데이터를 구조화하며, 테이블 간의 관계를 유지하기 위해 정규화 기법을 사용한다. 또한, 복잡한 쿼리를 효율적으로 처리하기 위해 인덱싱 및 조인 연산을 지원하여 다양한 데이터 간 연관 관계를 손쉽게 관리할 수 있다.

그러나, 관계형 데이터베이스는 실시간 데이터 처리 환경에서 몇 가지 한계를 가진다. 첫째, 확장성 문제가 존재한다. 관계형 데이터베이스는 일반적으로 수직 확장을 지원하며, 서버의 성능을 향상시키는 방식으로 확장된다. 하지만 데이터 증가에 따른 확장 비용이 높아질 수 있으며, 노드 간의 데이터 일관성을 유지하는 데 어려움이 발생할 수 있다. 둘째, 쓰기 및 읽기 성능 저하가 발생할 가능성이 있다. 실시간 시스템에서는 초당 수십만 건 이상의 데이터를 처리해야 하는 경우가 많지만, 관계형 데이터베이스는 트랜잭션 무결성을 유지하기 위해 동기화 작업이 필요하므로 대량의 동시 요청을 처리할 때 성능 저하가 발생할 수 있다. 마지막으로, 비정형 데이터 처리의 어려움이 존재한다. JSON, XML, 로그 데이터와 같은 반정형 또는 비정형 데이터 처리에는 관계형 데이터베이스보다 NoSQL이 더 적합한 선택이 될 수 있다.

3.2. Characteristics and Advantages of NoSQL Databases

NoSQL 데이터베이스는 관계형 데이터베이스의 한계를 극복하고, 대량의 데이터를 빠르게 처리하기 위해 설계된 데이터 저장 기술이다. 대표적인 NoSQL 데이터베이스로는 MongoDB, Cassandra, Redis, DynamoDB 등이 있으며, BASE(Basically Available, Soft state, Eventually consistent) 모델을 기반으로 높은 성능과 확장성을 제공한다.

NoSQL 데이터베이스는 스키마리스(Schema-less) 구조를 채택하고 있어 데이터 모델링이 유연하다. 이는 데이터의 구조가 사전에 고정될 필요 없이 변경될 수 있음을 의미하며, JSON, BSON, Key-Value, 컬럼, 그래프 등 다양한 데이터 모델을 지원한다. 또한, 수평 확장을 기본적으로 지원하므로, 여러 대의 노드에 데이터를 분산 저장하고 샤딩을 통해 부하를 분산시킬 수 있다. 이러한 구조는 대규모 트래픽을 처리하는 시스템에서 필수적이며, 특히 클라우드 기반 서비스에서는 NoSQL의 확장성이 중요한 요인으로 작용한다.

NoSQL 데이터베이스는 RDB와 비교했을 때 높은 쓰기 및 읽기 성능을 제공할 수 있다. 예를 들어, Cassandra는 분산 환경에서 다중 노드에 데이터를 비동기적으로 저장하여 빠른 쓰기 성능을 보장하며, Redis는 메모리 기반의 Key-Value 저장소로 빠른 읽기 속도를 제공한다. 이처럼 NoSQL은 실시간 로그 분석, IoT 센서 데이터 처리, 소셜 미디어 피드 관리 등에서 강력한 성능을 발휘할 수 있다.

하지만, NoSQL 데이터베이스는 트랜잭션 처리의 일관성 문제가 존재할 수 있다. 대부분의 NoSQL 시스템은 Eventually Consistent 모델을 따르므로, 즉각적인 데이터 일관성을 보장하지 않는 경우가 많다. 또한, 복잡한 관계형 쿼리 처리의 어려움이 있으며, 관계형 데이터 모델이 필수적인 시스템에서는 RDB보다 비효율적일 수 있다.

Table 1. Comparison between RDB and NoSQL

Feature	Relational Database (RDB)	NoSQL Database
Data Model	Structured	Flexible data model (document, key-value, column, graph)
Schema	Fixed schema (predefined structure)	Dynamic schema (flexible structure)
Query Language	SQL (Structured Query Language)	Various (e.g., MongoDB query language)
Transaction Support	ACID Compliance (Atomicity, Consistency, Isolation, Durability)	BASE Properties (Basically Available, Soft state, Eventually consistent)
Scalability	Vertical scalability (server performance scaling)	Horizontal scalability (server cluster expansion)
Data Integrity	High data integrity (for general applications)	Final consistency (in most cases)
Use Cases	General applications requiring data integrity	High scalability and massive data processing applications

3.3. Key Factors for Performance Comparison in Real-Time Data Processing Systems

실시간 데이터 처리 시스템에서 RDB와 NoSQL의 성능을 평가할 때 중요한 요소로는 응답 시간, 처리량, 확장성, 데이터 일관성, 그리고 고가용성이 있다. 먼저, 데이터 조회 및 쓰기 요청을 처리하는 데 소요되는 응답 시간은 실시간 성능을 결정하는 핵심 요소이다. 또한, 초당 처리할 수 있는 요청 수(RPS, Requests per second)를 의미하는 처리량은 대규모 트래픽[10]을 감당할 수 있는 능력을 평가하는 기준이 된다. 데이터가 증가하더라도 성능 저하 없이 확장할 수 있는지 여부도 중요한 요소이며, 이를 확장성이라고 한다. 더불어, 트랜잭션 처리에서 즉각적인 데이터 정합성이 요구되는지, 혹은 Eventually Consistent 모델이 허용될 수 있는지에 따라 데이터베이스의 적절성이 결정되므로, 데이터 일관성 역시 고려해야 한다. 마지막으로, 장애 발생 시 데이터 손실을 최소화할 수 있도록 복제 및 페일오버 기능을 지원하는지 여부를 뜻하는 고가용성도 중요한 평가 요소가 된다.

3.4. Criteria for Database Selection in Real-Time Data Processing Environments

실시간 데이터 처리 시스템에서 RDB와 NoSQL 중 적절한 데이터베이스를 선택하는 것은 시스템의 요구 사항과 성능 목표에 따라 결정된다. 데이터의 정합성이 절대적으로 중요한 경우(예: 금융 거래, 의료 데이터)에는 RDB가 더 적합할 수 있다. 반면, 대규모의 동시 요청을 처리해야 하거나 빠른 응답 시간이 필요한 경우(예: 실시간 로그 분석, IoT 데이터 처리)에는 NoSQL이 더 적절한 선택이 될 수 있다.

본 연구에서는 Spring WebFlux 기반의 실시간 데이터 처리 시스템에서 RDB와 NoSQL의 성능을 비교하고, 특정 환경에서 어느 데이터베이스가 더 적합한지를 평가하고자 한다. 이를 통해 실시간 데이터 처리 시스템을 구축할 때 고려해야 할 최적의 데이터 저장소 선택 기준을 도출하고자 한다.

III. The Proposed Scheme

1. Experimental Methodology

1.1 Experimental Environment and Scheme

본 연구는 Spring WebFlux 및 Spring MVC 기반의 실시간 데이터 처리 시스템에서 관계형 데이터베이스와 NoSQL 데이터베이스의 성능을 비교하고, 각 조합의 장단점을 체계적으로 분석하기 위해 설계되었다. 이를 위해 두 프레임워크 각각에 대해 RDB와 NoSQL 적용한 네 가지 조합을 구성하고, 동일한 요청에 대해 처리 성능을 측정하였다.

성능 비교 실험은 다음과 같이 구성되었다. 우선, 동시 접속자 수를 100, 200, 500, 1000, 5000명으로 설정하여 증가시키며 트래픽을 발생시켰다. 동시 접속은 Apache JMeter 5.6.3 도구를 활용해 시뮬레이션하였으며, 각 사용자 시나리오는 동일한 HTTP 요청을 반복적으로 발생시키도록 설정하였다. 테스트 요청은 단일 엔드포인트(API) 호출로 구성되며, 이 요청은 데이터베이스에 저장된 특정 키 값을 조회하는 단순한 GET 요청이다. 요청 도중 별도의 캐싱 또는 미들웨어 처리는 생략되었으며, 요청이 컨트롤러 → 서비스 → DAO → 데이터베이스까지 도달하고, 다시 결과를 클라이언트로 반환하기까지의 전체 흐름의 응답 시간을 측정하였다.

연구에 사용된 RDBMS는 MySQL 8.0.32, NoSQL 데이터베이스는 Redis 7.2.6이다. 두 데이터베이스는 오픈소스이며, 실무 환경에서 널리 사용되는 대표적인 기술로, 안정성과 접근성이 높아 실험 대상으로 선정되었다. 특히 MySQL은 범용적으로 사용되는 관계형 데이터베이스로, 트랜잭션 처리 및 SQL 기반 질의 최적화 기능이 풍부하여 RDB의 일반적인 특성을 잘 대표한다. Redis는 메모리 기반의 Key-Value 저장소로, NoSQL 시스템 중에서도 가장 높은 응답 성능을 보이는 구조를 가지고 있어 실시간 처리 시스템에 적합한 대표적인 NoSQL로 간주된다.

다만, 본 연구에서 선택된 MySQL과 Redis는 RDB와 NoSQL의 일반적인 성능 경향을 비교 분석하기 위한 대표 사례로, 모든 관계형/비관계형 데이터베이스의 성능 특성을 포괄하지는 않는다. 예를 들어, PostgreSQL이나 Oracle DB와 같은 RDB, 혹은 MongoDB, Cassandra, DynamoDB와 같은 NoSQL 시스템을 사용할 경우, 내부 구조, 쿼리 처리 방식, 확장성 모델의 차이로 인해 성능 결과가 일부 달라질 수 있다.

그럼에도 불구하고, MySQL과 Redis는 각각의 계열에서 실무 적용 빈도가 높고 문서화가 잘 되어 있어, 실시간 데이터 처리 환경에서의 기술 선택 기준을 비교하는 데 유효한 참조 모델로 사용될 수 있다.

응답 시간은 JMeter의 샘플링 기능을 활용하여 클라이언트 입장에서의 RTT(Round-Trip Time)를 측정하였으며, 각 실험은 약 2분간 수행 후 결과를 정리하였다. 이러한 방식은 실시간 데이터 처리 시스템에서의 트랜잭션 처리 성능을 단순하면서도 정확히 비교할 수 있는 기준을 제공한다.

Spring WebFlux는 비동기 논블로킹 기반의 처리 모델로 높은 동시성 성능을 제공하며, Spring MVC는 동기 블로킹 기반의 처리 모델로 요청을 순차적으로 처리한다. 이러한 구조적 차이가 데이터베이스 접근 시 성능에 어떤 영향을 미치는지를 분석하기 위해, 모든 실험은 네 가지조별로 동일한 조건 하에 수행되었다.

실험은 동일한 하드웨어(MacOS 14.5, Apple M2, RAM 32GB) 및 네트워크 환경에서 진행되었으며, 외부 요인의 영향을 최소화하기 위해 로컬 테스트베드 상에서 단일 네트워크 구간 내에서 구성하였다. 각 실험은 세 번 이상 반복되었고, 평균값을 산출해 이상값의 영향을 최소화하였다.

1.2 Experimental Validity and Justification

본 연구의 실험 방법은 다음과 같은 논리적 근거를 통해 타당성을 보강한다. 실제 서비스 환경과 유사한 부하 조건을 반영하였으며, 동시 접속자 수 증가에 따른 응답 시간 측정은 실시간 데이터 처리 시스템에서 빈번히 발생하는 성능 이슈를 평가하는 적절한 방식이다. 이러한 측정을 통해 서비스 규모에 따른 성능 변화를 명확히 관찰할 수 있다.

Spring WebFlux와 Spring MVC의 처리 방식 차이는 성능에 중대한 영향을 미친다. 특히, 실시간 데이터 처리 시스템에서 높은 동시성을 요구하는 경우 Spring WebFlux가 유리할 수 있으므로, 이를 데이터베이스 유형과 함께 비교하는 것은 실제 시스템 선택 시 중요한 기준을 제공한다. 동일한 조건에서 여러 번 실험을 반복 수행하여 실험 결과의 일관성을 검증하였으며, 평균값을 사용해 이상값의 영향을 최소화하였다. 마지막으로, 평균 응답 시간을 성능 비교의 주요 지표로 선정한 것은 실시간 데이터 처리 시스템에서 사용자 경험과 시스템 효율성을 동시에 측정할 수 있는 대표적인 지표이기 때문이다.

Table 2. System Configuration for Experiment

Feature	Spring MVC Server	Spring WebFlux Server	Database Server	Test Client
OS	macOs 14.5	macOs 14.5	macOs 14.5	macOs 14.5
CPU	Apple M2	Apple M2	Apple M2	Apple M2
RAM	32GB	32GB	32GB	32GB
F/W or Tools	Spring Boot 3.3.5 (Spring F/W 6.1.0)	Spring Boot 3.3.5 (Spring F/W 6.1.0)	-	Jmeter 5.6.3
WAS or DB	tomcat9	Netty 4.1.68	MySQL 8.0.32, Redis 7.2.6	-
Java	Java17	Java17	-	Java17

2. Experimental Results

2.1 Performance Testing with RDB in a Spring WebFlux Environment

Spring WebFlux와 RDB의 조합은 비동기 논블로킹 처리 방식 덕분에 낮은 응답 시간과 안정적인 성능을 보였다. 동시 접속자 수가 100명일 때 평균 응답 시간은 8.99ms였으며, 200명일 때 14.15ms, 500명일 때 35.81ms, 1000명일 때 70.61ms, 5000명일 때 291.93ms로 증가하는 경향을 보였다. 접속자 수가 많아질수록 응답 시간이 급격히 증가하였으나, 전체적으로 Spring WebFlux의 비동기 처리 방식은 상대적으로 낮은 응답 시간을 유지하였다. RDB는 데이터 일관성을 유지하는 장점이 있지만, 동시 다발적인 요청 처리 시 성능 저하가 발생하는 경향이 있었다. RDB의 특성상 데이터베이스 액세스 시 I/O 대기 시간이 길어지기 때문에 동시 접속자가 많을수록 응답 시간이 급격히 증가하는 문제가 발생하였다. 그럼에도 불구하고 WebFlux의 비동기 처리 특성은 이러한 성능 저하를 다소 완화시켰다.

Table 3. Performance Results Based on the Number of Concurrent Webflux-RDB Users

	100	200	500	1000	5000
0:00:01	8.00	14.94	40.38	75.54	368.61
0:00:05	6.78	13.70	35.84	70.48	294.78
0:00:10	6.88	14.39	36.78	70.61	281.07
0:00:15	6.96	14.41	36.54	70.35	274.52
0:00:20	6.98	14.35	35.38	71.79	296.08
0:00:25	6.99	14.18	38.41	68.54	284.59
0:00:30	7.27	14.44	35.37	68.25	285.95
0:00:35	7.11	15.61	41.20	69.47	273.96
0:00:40	7.22	13.62	35.64	81.46	309.19

2.2 Performance Testing with NoSQL(Redis) in a Spring WebFlux Environment

Spring WebFlux와 Redis의 조합은 매우 우수한 성능을 보였다. 동시 접속자 수가 100명일 때 평균 응답 시간은 3.22ms였으며, 200명일 때 6.68ms, 500명일 때 17.13ms, 1000명일 때 46.92ms, 5000명일 때 148.61ms로 증가하는 경향을 보였다. Redis는 메모리 기반 데이터베이스로 읽기/쓰기 성능이 매우 빠르며, WebFlux의 비동기 처리와 결합될 때 최고의 성능을 발휘하였다. 동시 접속자가 많아질수록 응답 시간이 증가하는 경향을 보였지만, Redis의 빠른 처리 속도 덕분에 다른 조합에 비해 여전히 상대적으로 낮은 응답 시간을 유지하였다. Redis는 비동기 처리에 매우 적합한 데이터베이스로, WebFlux

의 비동기 모델과 잘 맞아 떨어져 고성능 실시간 데이터 처리를 구현할 수 있었다. 다만, 5000명의 접속 시 148.61ms의 응답 시간이 발생한 것은 시스템 부하가 일정 부분 영향을 미쳤기 때문으로 분석된다.

Table 4. Performance Results Based on the Number of Concurrent Webflux-NoSQL Users

	100	200	500	1000	5000
0:00:01	2.85	6.97	18.04	35.57	157.11
0:00:05	3.28	6.73	17.07	35.45	144.80
0:00:10	3.24	6.82	17.51	35.11	144.48
0:00:15	3.26	6.82	17.40	35.27	144.06
0:00:20	3.25	6.73	17.30	35.62	144.75
0:00:25	3.26	6.72	17.31	35.72	147.68
0:00:30	3.31	6.74	17.42	37.24	156.85
0:00:35	3.28	6.70	17.31	35.23	142.14
0:00:40	3.23	6.63	17.37	35.36	145.08

2.3 Performance Testing with RDB in a Spring MVC Environment

Spring MVC와 RDB의 조합은 동기적 처리 모델로 인해 성능 저하가 두드러졌다. 동시 접속자 수가 100명일 때 평균 응답 시간은 13.05ms였으며, 200명일 때 22.45ms, 500명일 때 56.10ms, 1000명일 때 110.64ms, 5000명일 때 687.13ms로 급격하게 증가하였다. 동기적 처리 방식은 요청을 순차적으로 처리하므로 시스템 자원을 점유하게 되어 동시 접속자가 많을수록 성능이 급격히 떨어졌다. RDB는 데이터 일관성을 유지하는 데 유리하지만, 동기적 처리 방식에서 데이터베이스 I/O 대기 시간과 트랜잭션 처리로 인해 고부하 상태에서 성능 저하가 발생하는 특징이 있다. Spring MVC의 동기 처리 방식은 이러한 성능 저하를 더욱 부각시켜 고부하 환경에서 큰 성능 저하를 초래하였다.

Table 5. Performance Results Based on the Number of Concurrent MVC-RDB Users

	100	200	500	1000	5000
0:00:01	14.00	28.45	57.31	118.85	697.06
0:00:05	11.71	20.72	53.38	77.74	549.51
0:00:10	12.82	24.01	63.01	106.61	661.42
0:00:15	10.94	21.54	52.00	107.69	458.77
0:00:20	11.16	21.84	57.42	115.56	356.29
0:00:25	11.21	22.75	67.25	107.55	774.53
0:00:30	11.28	21.90	52.21	113.88	396.56
0:00:35	14.23	22.72	59.33	112.86	719.33
0:00:40	10.54	20.99	68.12	82.59	5575.57

2.4 Performance Testing with NoSQL(Redis) in a Spring MVC Environment

Spring MVC와 Redis의 조합은 동기적 처리 모델의 제약을 받으면서도 Redis의 빠른 성능을 일부 활용할 수 있었다. 동시 접속자 수가 100명일 때 평균 응답 시간은 5.88ms였으며, 200명일 때 12.06ms, 500명일 때 32.96ms, 1000명일 때 78.93ms, 5000명일 때 559.07ms로 증가하였다. Redis의 빠른 읽기/쓰기 성능 덕분에 Spring MVC 환경에서도 비교적 빠른 응답 속도를 보였지만, 동기 처리 모델에 의해 고부하 시 성능 저하가 심화되었다. Redis는 메모리 기반 데이터베이스로 빠른 응답 속도를 자랑하지만, Spring MVC의 동기 처리 모델에서는 이를 충분히 활용하지 못하였다. 동기 모델에서 발생하는 자원 경쟁과 I/O 대기로 인해 동시 접속자가 많을수록 응답 시간이 급격히 증가하였다.

Table 6. Performance Results Based on the Number of Concurrent MVC-NoSQL(Redis) Users

	100	200	500	1000	5000
0:00:01	6.82	10.68	38.16	79.98	575.51
0:00:05	6.34	10.16	34.63	73.14	565.70
0:00:10	5.10	12.45	32.63	69.84	584.31
0:00:15	5.77	11.58	35.73	69.34	568.68
0:00:20	5.60	12.13	29.62	74.56	563.95
0:00:25	5.17	12.23	34.67	63.94	557.11
0:00:30	6.44	12.48	32.40	68.55	567.38
0:00:35	6.32	11.87	32.80	69.34	552.99
0:00:40	5.40	13.87	30.88	69.65	584.13

2.5 Performance Comparison of RDB and Redis in Spring WebFlux and MVC Environments

Spring WebFlux 환경에서 RDB를 사용할 경우, 동시 접속자 증가에 따라 응답 시간이 선형적으로 증가하였으며, 100명일 때 8.99ms에서 5000명일 때 291.93ms로 약 32.5배 증가해 RDB의 I/O 대기 시간이 성능 저하의 주요 원인임을 보여주었다. WebFlux 환경에서 Redis를 사용할 경우, 100명일 때 3.22ms, 5000명일 때 148.61ms로 약 46배 증가했지만, 여전히 가장 낮은 응답 시간을 유지하며 가장 우수한 성능을 보였다. Spring MVC 환경에서 RDB를 사용할 경우, 동기 처리 방식으로 인해 100명일 때 13.05ms에서 5000명일 때 687.13ms로 약 52.6배 증가하며 가장 큰 성능 저하를 기록하였다. MVC 환경에서 Redis를 사용할 경우, Redis의 빠른 속도 덕분에 상대적으로 낮은 응답 시간을 유지했으나, 동기 모델의 제약으로

100명일 때 5.88ms에서 5000명일 때 559.07ms로 약 95배 증가한 결과를 얻을 수 있었다.

Table 7. Comparison of Average Response Time According to Concurrent User Count

	WebFlux RDB	WebFlux NoSQL	MVC RDB	MVC NoSQL
100	8.99	3.22	13.05	5.88
200	14.15	6.68	22.45	12.06
500	35.81	17.13	56.10	32.96
1000	70.61	46.92	110.64	78.93
5000	291.93	148.61	687.13	559.07

동시 접속자 200명 환경에서도 WebFlux-NoSQL이 가장 높은 트랜잭션 처리 성능을 보였다. MVC-RDB의 경우 응답 시간이 가장 길며 성능 저하가 점진적으로 나타났다. WebFlux-RDB는 안정적인 성능을 유지하였지만, NoSQL 대비 상대적으로 낮은 처리량을 보였다. MVC-NoSQL은 MVC-RDB보다는 높은 성능을 보였으나, WebFlux 기반 환경보다는 낮은 트랜잭션 수를 유지하였다.

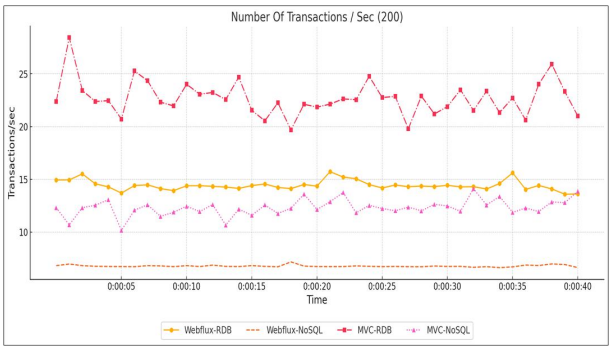

Fig. 2. Response Time Comparison with 200 Concurrent Users

동시 접속자가 500명으로 증가함에 따라 MVC-RDB의 성능 저하가 더욱 두드러졌다. WebFlux-NoSQL은 여전히 가장 높은 성능을 유지하며 트랜잭션 수가 상대적으로 안정적이었다. WebFlux-RDB는 MVC-RDB보다는 성능이 우수하지만 트랜잭션 수의 변동이 발생하였다. MVC-NoSQL은 트랜잭션 수가 일정한 수준을 유지했지만 WebFlux 기반 아키텍처보다 성능이 낮았다.

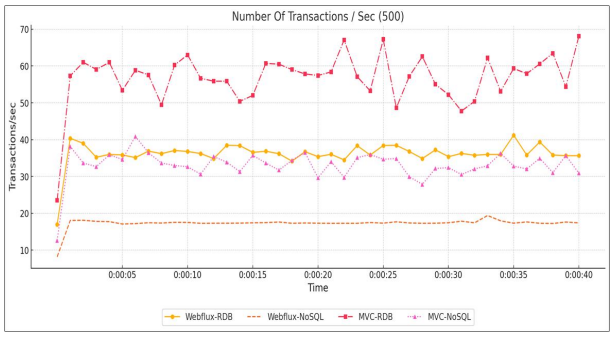

Fig. 3. Response Time Comparison with 500 Concurrent Users

5000명 이상의 동시 접속 환경에서는 MVC-RDB의 성능 저하가 극심하게 발생하였다. WebFlux-NoSQL은 여전히 가장 높은 트랜잭션 수를 기록하였지만 일부 변동이 발생하였다. WebFlux-RDB는 상대적으로 안정적인 트랜잭션 수를 유지하며 MVC-NoSQL을 초과하는 성능을 보였다. MVC-NoSQL은 MVC-RDB보다는 높은 성능을 보였으나, 트랜잭션 수의 증가폭이 제한적이었다.

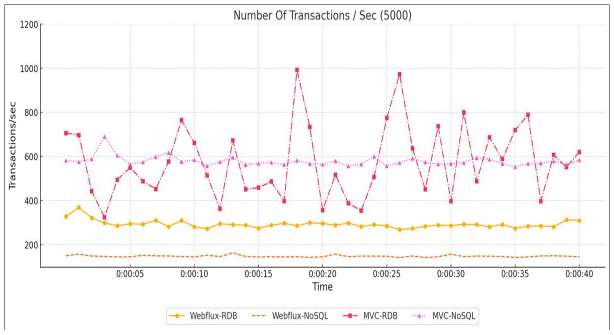

Fig. 4. Response Time Comparison with 5000 Concurrent Users

IV. Conclusions

본 연구는 실시간 데이터 처리 시스템에 최적화된 프레임워크와 데이터베이스를 선택하기 위한 평가 기준을 설정하고, 이를 기반으로 다양한 조합의 성능을 비교·분석하였다. 실시간 데이터 처리 시스템에서 중요한 요소로는 응답 시간, 처리량, 시스템 확장성, 데이터 일관성 유지 등이 있으며, 본 연구에서는 이러한 요소를 기준으로 선택된 프레임워크와 데이터베이스가 시스템 성능에 미치는 영향을 심층적으로 분석하였다.

실험 결과, Spring WebFlux와 Redis의 조합이 가장 우수한 성능을 보였으며, 특히 동시 접속 처리와 빠른 응답 속도 측면에서 뛰어난 결과를 나타냈다. WebFlux는 비동기 논블로킹 방식으로 설계되어 있어, 대규모 트래픽

을 처리하고 실시간 데이터 스트리밍을 요구하는 환경에서 높은 동시성 처리 능력을 발휘하며 실시간 데이터 처리에 최적화된 성능을 제공한다. 반면, Spring MVC는 동기 처리 모델을 채택하고 있어 고부하 상황에서 성능 저하가 발생할 가능성이 높아, 실시간 데이터 처리 시스템에는 적합하지 않음을 확인하였다.

데이터베이스 측면에서 Redis는 인메모리 기반 NoSQL 데이터베이스로, 실시간 데이터 처리에 최적화된 특성을 보인다. 뛰어난 읽기/쓰기 성능을 제공하여 빠른 응답 속도가 중요한 실시간 시스템에서 성능을 극대화하는 데 유리하며, 특히 세션 관리와 캐싱이 중요한 환경에서 Redis의 효율성이 두드러진다. 반면, RDB는 복잡한 트랜잭션 처리와 데이터 일관성 보장에 강점이 있지만, 수평 확장성에 제한이 있어 대규모 실시간 데이터 처리에는 비효율적일 수 있다. NoSQL은 확장성과 유연성 측면에서 장점을 가지지만, 복잡한 쿼리 처리와 데이터 일관성 유지에서 성능 저하가 발생할 수 있다. 이러한 이유로, Spring WebFlux와 Redis의 조합이 실시간 데이터 스트리밍 환경에서 최적의 성능을 발휘함을 확인하였다.

MySQL은 복잡한 트랜잭션 처리가 필요한 시스템에는 적합하지만, WebFlux 환경에서는 Redis와의 조합이 더 효율적임을 실험을 통해 확인하였다. 반면, Spring MVC 환경에서는 MySQL이 안정적인 선택이 될 수 있으나, 빠른 응답 속도가 중요한 경우 Redis가 더 적합한 것으로 나타났다. 따라서 애플리케이션의 요구 사항과 시스템의 특성을 고려하여 적절한 데이터베이스와 프레임워크를 신중하게 선택하는 것이 중요하다.

결론적으로, 본 연구는 Spring WebFlux와 Redis의 조합이 실시간 데이터 처리 시스템에 가장 적합한 프레임워크 및 데이터베이스 조합임을 제시한다. WebFlux의 비동기 논블로킹 특성과 Redis의 뛰어난 읽기/쓰기 성능이 결합되어, 대규모 동시 접속자와 실시간 데이터 스트리밍을 요구하는 환경에서 최상의 성능을 제공한다. 반면, Spring MVC와 RDB는 특정 요구 사항을 충족하는 데 적합할 수 있으나, 실시간 데이터 처리 환경에서는 성능적 한계를 보인다는 점을 확인할 수 있었다.

본 연구는 실시간 데이터 처리 시스템 설계에 중요한 시사점을 제공하지만, 몇 가지 한계점도 존재한다. 첫째, 연구는 특정 데이터베이스 엔진(MySQL, Redis)과 프레임워크(Spring WebFlux, Spring MVC)에 한정되어 있다. 향후 다양한 데이터베이스 엔진(DynamoDB, Cassandra 등)과 프레임워크(Netty, Akka 등)를 포함한 확장 실험이 필요하다. 둘째, 실험 환경이 단일 애플리케이션 중심으로 구성되어 있어, 다중 애플리케이션 간 상호작용이나 클러스터링 환경에서의 성능 분석이 부족하다. 셋째, 데이터 처리의 복잡성과 노드 간 통신 오버헤드가 포함된 대규모 분산 시스템 환경에 대한 실험이 충분하지 않았다.

향후 연구에서는 다양한 기술 스택과 아키텍처를 대상으로 실험을 확장하고, 실시간 데이터 처리 시스템이 직면한 새로운 문제(예: 데이터 보안, 비용 효율성 등)를 해결하는 방향으로 나아갈 필요가 있다. 또한, RDB와 NoSQL을 결합한 하이브리드 데이터베이스 접근법을 연구하여, 실시간 처리 환경에서 보다 높은 유연성과 성능을 제공할 수 있는 가능성을 모색할 것이다.

본 연구의 결과는 실시간 데이터 처리 시스템 설계 및 구현 시 최적의 기술 스택을 선택하는 데 중요한 기초 자료가 될 것이며, 시스템 성능을 극대화하고 확장성과 안정성을 보장하는 데 기여할 것이다. 나아가, 다양한 산업 환경에서 본 연구의 결과를 바탕으로 실시간 데이터 처리 시스템의 설계 및 구현을 보다 효율적으로 개선할 수 있을 것으로 기대된다.

REFERENCES

[1] Jagjeet Singh, "Real-Time Data Processing: Frameworks, Machine Learning Integration, and Traffic Analysis Using Computer Vision," International Journal of Innovative Research in Multidisciplinary Field Studies, Vol. 9, No. 7, pp. 1-6, July 2024. https://doi.org/10.37082/IJIRMPS.v12.i4.230767

[2] Syed Eqbal Alam, Mohd Abdul Ahad, "Optimizing Real-Time Data Processing: Edge and Cloud Computing Integration for Low-Latency Applications in Smart Cities," SSRN Electronic Journal, December 2023. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5121199&utm_source=chatgpt.com

[3] Dong-Hyun Kim, Sang-Won Lee, "Real-Time Processing of E-commerce User Data Based on Spark," Journal of Korean Institute of Information Technology, Vol. 18, No. 1, pp. 103-110, February 2023. https://www.kci.go.kr/kciportal/ci/sereArticleSearch/ciSereArtiView.kci?sereArticleSearchBean.artiId=ART002932349

[4] Md Arif Ul Alam, Md Mahbubur Rahman, "Edge Computing for Real-Time Data Analytics: Exploring the Use of Edge Devices for IoT Applications," The Science Brigade, Vol. 1, No. 1, pp. 1-8, 2023. https://thesciencebrigade.com/iotecj/article/view/86

[5] Elena Lupu, Adriana Olteanu, Anca Daniela Ionita, "Concurrent Access Performance Comparison Between Relational Databases and Graph NoSQL Databases for Complex Algorithms," Applied Sciences, Vol. 14, No. 21, 9867, 2024. https://www.mdpi.com/2076-3417/14/21/9867

[6] Alexandru Catrina, "A Comparative Analysis of Spring MVC and Spring WebFlux in Modern Web Development", Theseus, 2023. https://www.theseus.fi/handle/10024/812448

[7] Myunggyo Jeong, "Performance Analysis of Java-based Spring Web MVC and WebFlux," Master's Thesis, Korea University, November 2020. https://kiss.kstudy.com/Detail/Ar?key=3860348

[8] M. Sais, A. Sghir, N. Rafalia, J. Abouchabaka, "Analysis and Comparison of NoSQL Databases with Relational Database: MongoDB and HBase versus MySQL", International Journal on Technical and Physical Problems of Engineering, 2023. https://www.iotpe.com/IJTPE/IJTPE-2023/IJTPE-2023.html

[9] Wisal Khan, Teerath Kumar, Zhang Cheng, Kislay Raj, Anisha Roy, Bin Luo, "SQL and NoSQL Database Software Architecture Performance Analysis and Assessments - A Systematic Literature Review", Big Data and Cognitive Computing, 2023. https://www.mdpi.com/2504-2289/7/2/97

[10] Kyunghoon Yeom, "Distributed Server System for Large-Scale Traffic Processing in AWS Environment," Master's Thesis, Ajou University, February 2017. https://www.dbpia.co.kr/journal/detail?nodeId=T14553457

Authors

Minsol Kim received the Associate's degree in Computer Engineering from Inha Technical College, Korea, in 2023, and the B.S. degree in Computer Engineering from the same college in 2025.

Mr. Kim has been working as a software developer at AllforLand, Korea, since 2023. She received the Associate's degree in Computer Engineering from Inha Technical College, Korea, in 2023, and the B.S. degree in Computer Engineering from the same college in 2025. She is interested in web application development, real-time data processing systems, and performance optimization of relational and non-relational databases using Spring WebFlux.

Jongha Kim received the Associate's degree in Computer Engineering from Inha Technical College, Korea, in 2023, and the B.S. degree in Computer Engineering from the same college in 2025.

Mr. Kim has been working as a software developer at AllforLand, Korea, since 2023. He received the Associate's degree in Computer Engineering from Inha Technical College, Korea, in 2023, and the B.S. degree in Computer Engineering from the same college in 2025. He is interested in web application development, real-time data processing systems, and performance optimization of relational and non-relational databases using Spring WebFlux.

Sehoon Lee received the B.S. degree in Computer Science from Inha University, Korea, in 1985, and the M.S. and Ph.D. degrees in Computer Science & Engineering from the same university in 1987 and 1996, respectively. From 1987 to 1990, he was an information analyst officer at the Computing Center of the ROK Marine Corps. He joined the faculty of the Department of Computer Engineering at Inha Technical College, Incheon, Korea, in 1993. From 2001 to 2002, he was a visiting scholar at the New Jersey Institute of Technology (NJIT), USA. Currently, he is a Professor in the Department of Computer Systems & Engineering. His primary research interests are in the areas of Artificial Intelligence(AI), AI Agents, Prognostics and Health Management (PHM), and AIoT.