Spring WebFlux를 활용한 실시간 데이터 처리 시스템의 RDB와 NoSQL 성능 비교 분석

제71차 한국컴퓨터정보학회 동계학술대회

발표자 | 김민솔 (올포랜드)

목 차

01. Introduction

BIGDATAWIRE
DATA SCIENCE • AI • ADVANCED ANALYTICS
FORMERLY DATANAMI

About Resources Subscribe

언어 선택
Translation Disclaimer
Search this site Search
Follow BigDATAwire:

HOME | COVID-19 | FEATURES | SECTORS | APPLICATIONS | TECHNOLOGIES | VENDORS | JOB BANK | EVENTS | ADVERTISE

July 12, 2023

Yes, Real-Time Streaming Data Is Still Growing

Alex Woodie

지난 8개월 동안 기술계가 거의 전적으로 ChatGPT에 집중하는 동안 재밌는 일이 발생했습니다. 그 중 하나는 실시간 스트림 데이터 처리 기술의 채택 증가로, 이에 대한 관심은 지난 몇 년 동안 여러 가지 영향력 있는 사용 사례에 대해 조용히 커져 왔습니다.

Confluent의 2023년 데이터 스트리밍 보고서 에 따르면, Confluent의 스트리밍 데이터 사용자의 거의 절반(44%)이 이 기술이 최우선 전략적 우선순위라고 말하며, 89%가 중요하다고 말했습니다.

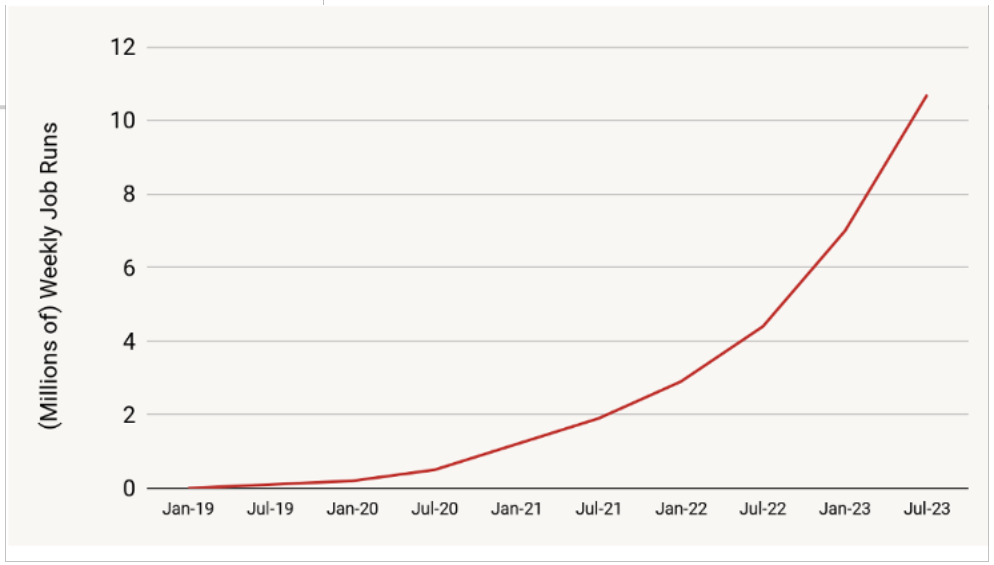


Fig. Confluent's 2023 Data Streaming Report

>> Spring WebFlux

- Spring 프레임워크의 일부로, 리액티브 프로그래밍 모델을 지원하는 웹 프레임워크
- 비동기 및 논블로킹 통신을 통해 높은 성능과 확장성을 제공
- 대규모 애플리케이션과 실시간 데이터를 처리해야 하는 시스템에서 뛰어난 성능 제공

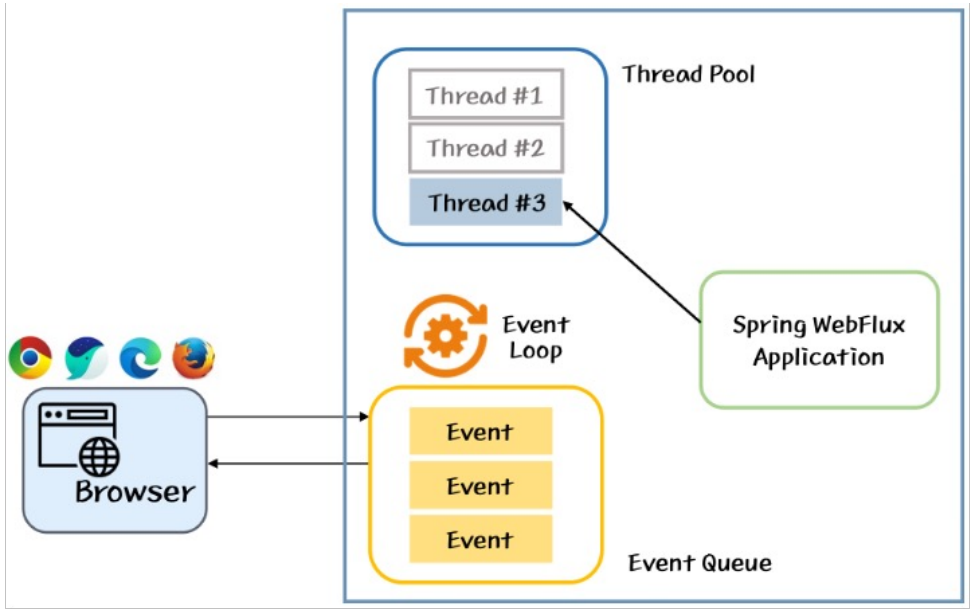


Fig. EventLoop Model

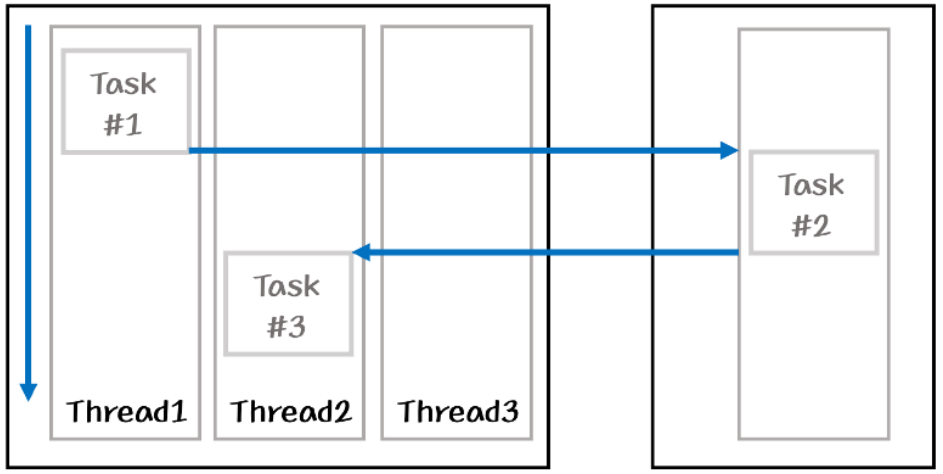


Fig. Spring WebFlux Thread Event

>> Spring MVC

- 웹 애플리케이션 개발 지원하는 모듈
- 동기 및 블로킹 통신을 통해 안정적이고 직관적인 개발
- 전통적인 웹 애플리케이션에 적합하나 고트래픽 환경에서는 성능 저하 발생

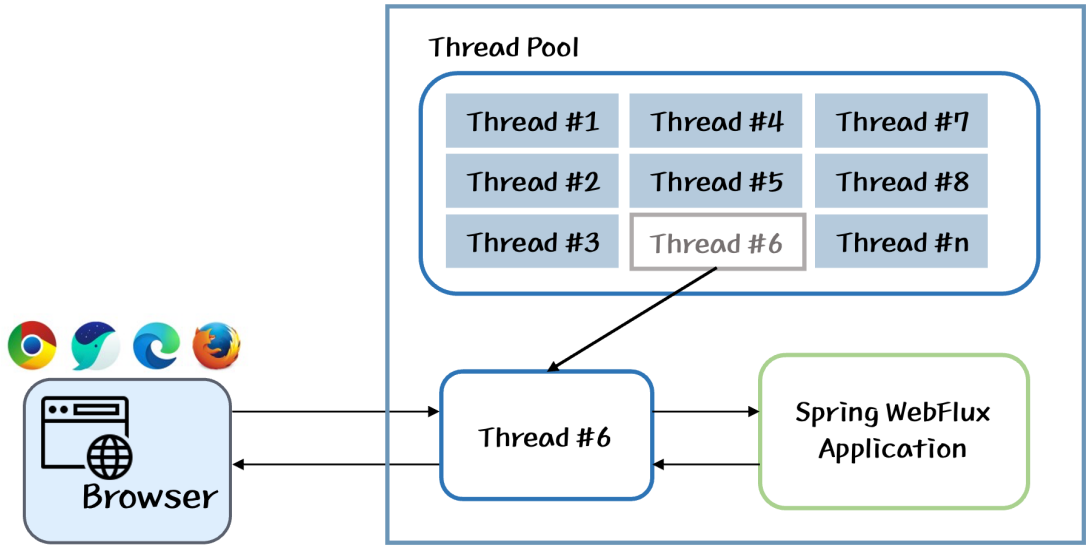


Fig. Thread Per Request Model

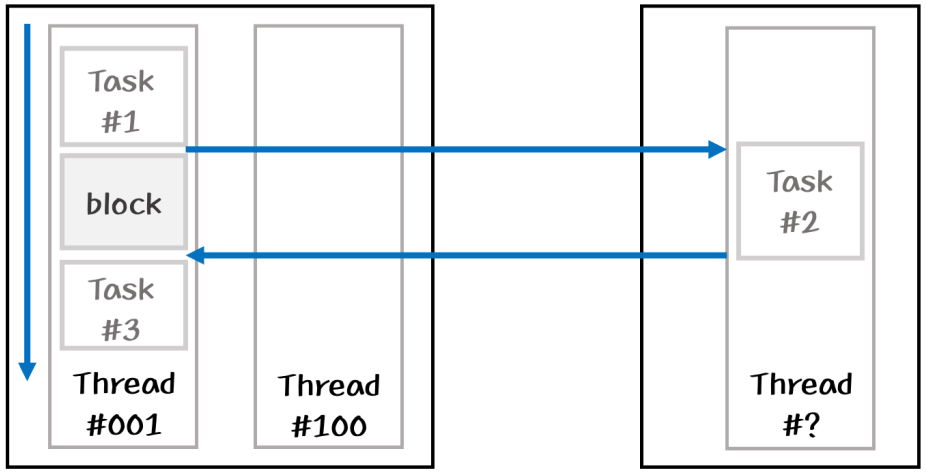


Fig. Spring MVC Thread Event

>> Spring WebFlux vs. Spring MVC

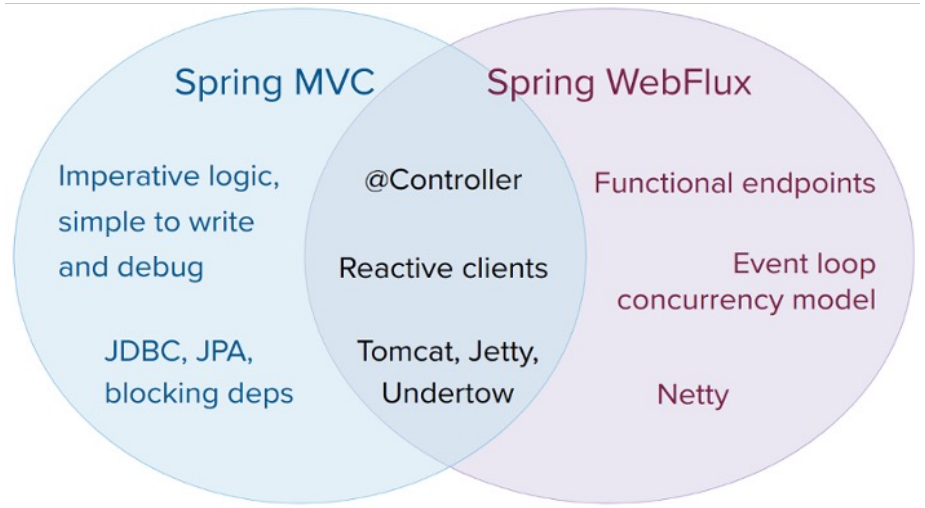


Fig. Comparison of Spring MVC and Spring WebFlux Architectures

특성	Spring WebFlux	Spring MVC
처리 방식	비동기 및 논블로킹 처리	동기 및 블로킹 처리
성능	높은 동시성 및 고성능 처리 가능	상대적으로 성능 저하 가능, 고트래픽에서 성능이 낮을 수 있음
확장성	수평적 확장성 지원	수직적 확장성 주로 사용
사용 사례	실시간 데이터 스트리밍, 웹소켓, IoT, 대규모 트래픽 처리 등	전통적인 웹 애플리케이션, 동기적 요청 처리 필요
리액티브 프로그래밍	지원 (Mono, Flux 등)	미지원
유연성	다양한 프로토콜 및 비동기 처리 가능	전통적인 요청-응답 모델 기반

Table. Comparison between Spring WebFlux and Spring MVC

>> RDBMS vs. NoSQL

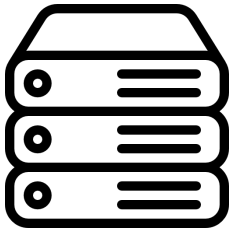
	RDBMS	NoSQL
스키마	- 명확한 데이터 구조를 보장 - 스키마로 인해 데이터가 유연하지 못함 - 스키마가 변경될 경우 번거롭고 어려움	- 스키마가 없어 유연하며 자유로운 데이터 구조를 가짐 - 언제든 저장된 데이터를 조정하고 새로운 필드를 추가 가능 - 명확한 데이터 구조를 보장X 데이터 구조 결정 어려움
확장성	- Scale-up만을 지원 (고비용)	- Saclue-up 과 Scale-out 지원(데이터 분산이 용이)
테이블 관계성	- 각 데이터를 중복없이 한번만 저장 (데이터 일관성) - 시스템이 커질 경우 JOIN문이 많은 복잡한 쿼리	- 데이터 중복이 발생 가능성 - 중복된 데이터가 변경될 경우 수정을 모든 컬렉션에서 수정 필요 - 테이블간 관계를 맺지않아 복잡한 JOIN문을 미사용
데이터 처리	- 부하 발생시, 처리가 어려움 - 데이터의 UPDATE가 빠름	- 많은양의 데이터를 처리, 저장 가능 - 데이터를 UPDATE 시 비교적 느림

Table. Comparison between RDB and NoSQL

>> Experimental Environment

Client

- OS : macOs 14.5
- CPU : Apple M2
- RAM : 32GB
- Java 17
- Jmeter 5.6.3



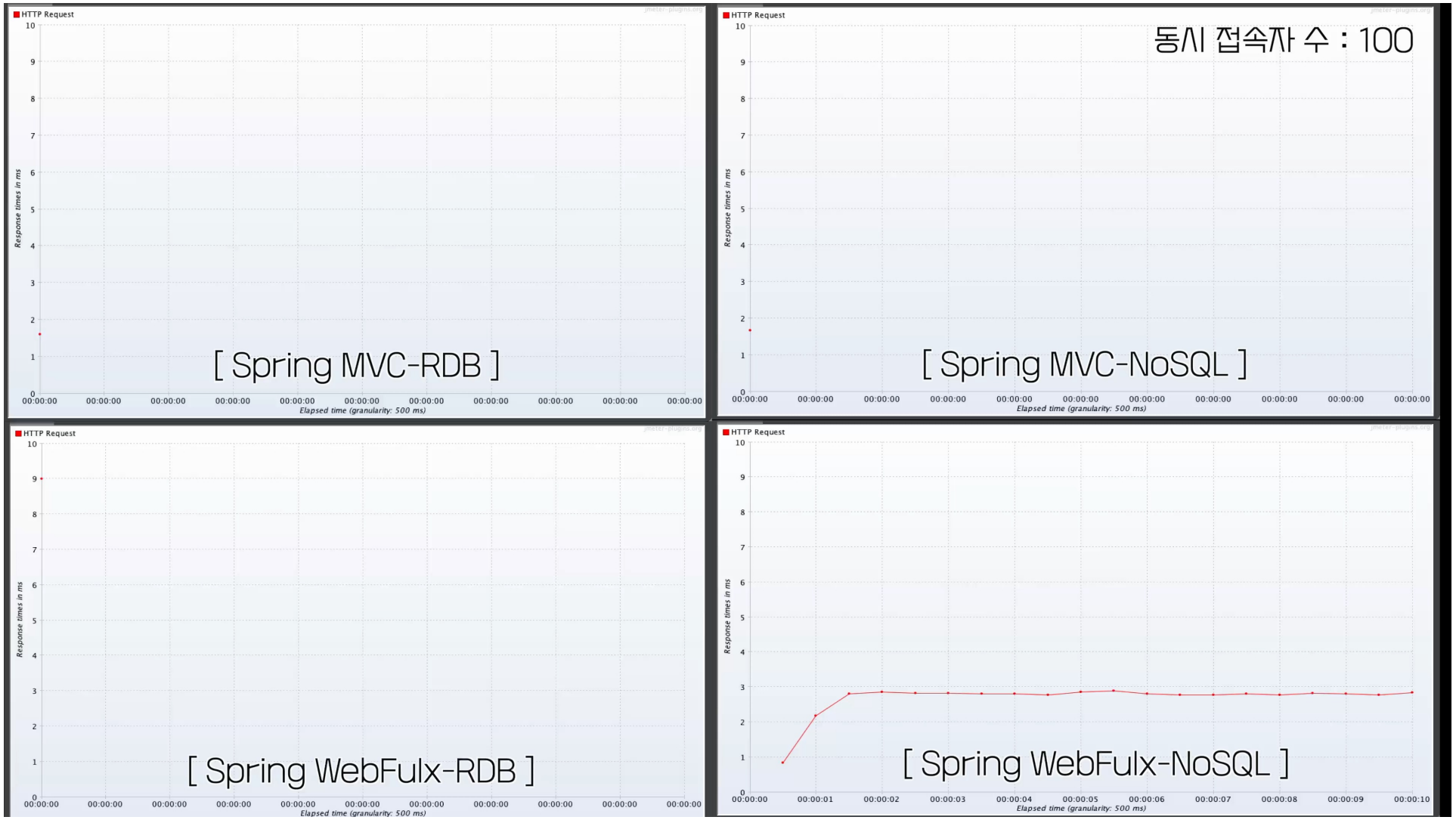
API Server

- OS : macOs 14.5
- CPU : Apple M2
- RAM : 32GB
- F/W : Spring Boot 3.3.5
- WAS : Tomcat9, Netty 4.1.68
- Java 17

Database Server

- OS : macOs 14.5
- CPU : Apple M2
- RAM : 32GB
- MySQL 8.0.32
- Redis 7.2.6
- Java 17

03. The Proposed Scheme



>> Experimental Results

	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

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

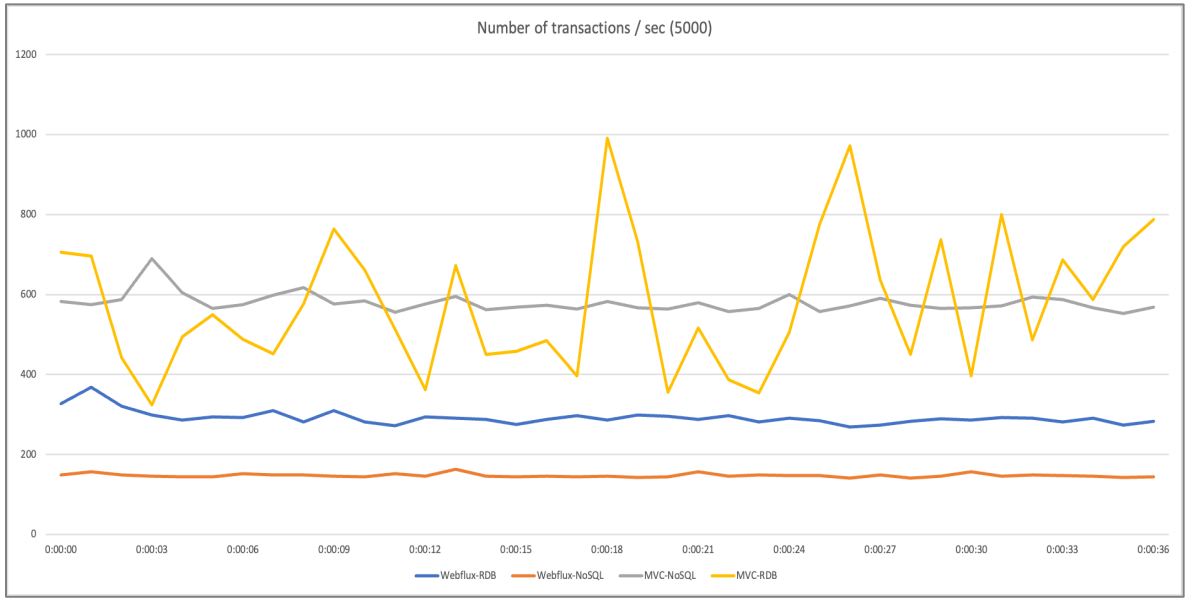


Fig. Response Time Comparison with 5000 Concurrent Users

>> 요약

적합한 기술 구성

최적 기술 스택 선택을 위한
기초 자료 제공

성능 향상

시스템 성능 극대화 및
확장성, 안정성 보장에 기여

다양한 산업 적용

다양한 산업 환경에서 효율적인 시스템
설계 및 구현 개선 기대

>> 한계점

한정된 범위

특정 데이터베이스 사용
(MySQL, Redis)

실험 환경 제약

단일 애플리케이션 중심으로 진행되어
상호작용, 클러스터링 평가 부족

대규모 환경 테스트 부족

데이터 처리 복잡성 및
통신 오버헤드 고려 미흡

>> 향후 연구

- 기술 스택 확장 : 다양한 기술 스택 및 아키텍처 대상으로 실험 확대
- 새로운 문제 해결 : 데이터 보안 및 비용 효율성 등 문제 해결
- 하이브리드 접근 : RDB와 NoSQL 통합 활용으로 유연성과 성능 향상 모색

감사합니다

Q&A