사물인터넷융복합논문지 Vol. 11, No. 6, pp. 13-19, 2025 https://doi.org/10.20465/KIOTS.2025.11.6.013

대규모 시계열 데이터 마이그레이션을 위한 MongoDB와 InfluxDB 성능 비교 연구

백영미[1], 박정규[2*]
[1]창신대학교 스마트팩토리학부 교수, [2]대진대학교 컴퓨터공학전공 교수

A Comparative Performance Study of MongoDB and InfluxDB for Large-Scale Time-Series Data Migration

Youngmi Baek[1], Jung Kyu Park[2*]
[1]Professor, School of Smart Factory, Changshin University
[2]Professor, Department of Computer Engineering, Daejin University

요 약 본 연구는 대규모 시계열 데이터 환경에서 MongoDB와 InfluxDB의 쓰기 성능 및 지연 특성을 비교·분석하였다. 최근 IoT 및 로그 분석 응용에서는 초당 수백만 건의 이벤트 처리가 요구되며, 기존 관계형 데이터베이스는 이러한 고부하 환경을 효과적으로 처리하기 어렵다. 본 연구에서는 고성능 병렬 처리를 지원하는 Go 언어를 사용하여 부하 발생기를 구현하였으며, 이를 통해 초당 수십만~수백만 건의 쓰기 부하를 안정적으로 재현하였다. 또한 MongoDB의 낮은 내구성 옵션인 Write Concern w:1(Primary만 기록 확인)과 강한 내구성을 요구하는 w:majority(Replica Set 과반수 확인) 설정을 포함한 네 가지 시나리오를 동일한 조건에서 평가하였다. 실험 결과, InfluxDB는 평균 1.54M TPS와 p95 지연 243ms를 기록하며 MongoDB 대비 약 3배 높은 처리량을 보였다. MongoDB는 Write Concern 수준이 높아질수록 복제 확인 과정으로 인해 지연이 증가하였다. 본 연구는 TSDB의 구조적 장점과 데이터베이스 선택 및 튜닝 전략에 대한 실질적 시사점을 제공한다.

주제어 : 시계열 데이터베이스, MongoDB, InfluxDB, 성능 비교, 데이터 마이그레이션

Abstract This study compares the write throughput and latency characteristics of MongoDB and InfluxDB in large-scale time-series data environments. Modern IoT and log analytics systems increasingly require processing millions of events per second, which traditional relational databases struggle to support efficiently. To reproduce such high-load environments accurately, we implemented a load generator using the Go programming language, which provides lightweight concurrency and efficient parallel execution, enabling stable generation of hundreds of thousands to millions of write operations per second. Four scenarios were evaluated under identical conditions, including MongoDB's low-durability Write Concern w:1 (acknowledgment only from the primary node) and the stronger durability setting w:majority (acknowledgment from a majority of replica-set members). Experimental results show that InfluxDB achieved 1.54M TPS with a p95 latency of 243 ms, approximately three times higher throughput than MongoDB. MongoDB exhibited increased latency as Write Concern levels strengthened due to replication acknowledgment overhead. These findings highlight the architectural advantages of InfluxDB for high-throughput time-series workloads and offer practical guidance for database selection and tuning in large-scale log and IoT systems.

Key Words : Time-Series Database, MongoDB, InfluxDB, Performance Evaluation, Data Migration

*교신저자 : 박정규(jkpark@daejin.ac.kr)
접수일 2025년 10월 12일 수정일 2025년 11월 04일 심사완료일 2025년 11월 13일

1. 서론

대규모 IoT, 광고, 모니터링 시스템에서 발생하는 이벤트 데이터는 초당 수십만 건 이상으로 폭증하고 있으며, 이러한 시계열(time-series) 데이터의 효율적 저장과 분석은 서비스 성능과 운영 비용을 좌우하는 핵심 요소가 되고 있다[1-3]. 전통적인 범용 관계형 데이터베이스(RDBMS)나 문서 지향 데이터베이스(MongoDB 등)는 범용성이 높지만, 시계열 데이터 처리에 최적화되지 않아 고속 쓰기 부하와 대규모 배치 작업에서 지연(latency)이 급격히 증가하고 저장소 자원의 비효율성이 발생한다[4-6].

이에 따라 InfluxDB, TimescaleDB와 같은 시계열 데이터베이스(Time-Series Database, TSDB)가 등장하여, 메모리 기반 버퍼링, 시계열 인덱싱 및 압축 기법을 통해 고속 쓰기와 저지연 쿼리를 지원하는 것으로 보고되고 있다[7,8]. 특히 대규모 이벤트 로깅, 배너 클릭 및 노출 집계와 같은 실시간 분석 워크로드에서 TSDB의 장점이 부각되지만, 기존 MongoDB에 축적된 대규모 데이터를 TSDB로 이전(migration)할 때 발생하는 성능상의 이점과 한계를 실험적으로 평가한 연구는 여전히 부족하다[9,10].

최근 서비스 운영 환경은 단순한 로그 수집을 넘어, 초저지연 분석 및 실시간 의사결정을 지원해야 하는 단계로 진화하고 있다. 예를 들어 대형 전자상거래 플랫폼이나 온라인 광고 시스템에서는 초당 수백만 건의 노출 및 클릭 이벤트를 실시간으로 집계하고 피드백해야 하며, 금융·제조·스마트시티 분야에서도 유사한 초고속 스트리밍 데이터 처리가 필수적이다. 이러한 요구는 기존 범용 데이터베이스의 병목을 더욱 심화시키고 있으며, 시계열 데이터베이스의 도입 및 최적화가 성능 보장과 비용 효율성 측면에서 중요한 과제로 부상하고 있다[5,11].

본 연구는 실제 온라인 서비스에서 사용되고 있는 배너 노출·클릭 로그 약 5억 6천만 건을 대상으로, MongoDB 기반 시스템을 InfluxDB로 이전한 후 쓰기 부하(Throughput)와 지연(Latency)의 변화를 체계적으로 비교·분석한다. 특히, 워커(worker) 개수와 배치(batch) 크기를 조절하며 초당 최대 수백만 건의 삽입 부하를 발생시키고, InfluxDB의 캐시 메모리 설정과 MongoDB의 쓰기 보장 수준(write concern: w:1, majority)에 따른 차이를 계량적으로 측정하였다.

이를 통해 본 연구는 다음과 같은 기여를 한다.

(1) 대규모 시계열 데이터 마이그레이션 시 TSDB가 제공하는 성능 향상을 정량적으로 검증하고,
(2) 동일 하드웨어 환경에서 MongoDB 대비 InfluxDB의 부하 처리 한계와 튜닝 효과를 실험적으로 규명하며,
(3) 학술 및 산업 현장에서 시계열 데이터 저장소 선택과 마이그레이션 전략 수립에 실질적 지침을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구와 시계열 데이터베이스의 특성을 정리하고, 3장에서는 실험 환경과 부하 발생기를 포함한 벤치마크 방법론을 기술한다. 4장에서는 MongoDB와 InfluxDB의 성능 비교 결과를 제시하고, 5장 결론에서 본 연구의 한계와 향후 연구 방향을 제시한다.

2. 관련 연구

시계열 데이터베이스(Time-Series Database, TSDB)는 IoT, 클라우드 모니터링, 광고 로그 등에서 발생하는 연속적 데이터를 효율적으로 처리하기 위해 등장하였다[1,2]. 기존의 관계형 데이터베이스(RDBMS)는 트랜잭션 무결성과 복잡한 조인 연산에 강점을 가지지만, 시계열 데이터의 대량 삽입과 시점 기반 질의에는 성능상 한계를 보인다[3,4]. 이러한 이유로 2010년대 이후 MongoDB, Cassandra, HBase 등의 NoSQL 시스템이 로그성 데이터를 관리하는 주요 대안으로 부상하였으나, 데이터의 시간 기반 인덱싱 및 압축 효율 측면에서는 여전히 비효율적이라는 지적이 제기되었다 [5,6].

MongoDB는 문서 단위 저장구조와 Write Concern 기반의 유연한 일관성 모델을 제공하지만, 초당 수만 건 이상의 쓰기 부하가 발생하는 환경에서는 WAL(Write-Ahead Logging) 처리와 락 경합으로 인한 지연(latency)이 누적되는 문제가 보고되었다 [7,8]. 반면, InfluxDB는 시간 필드를 기본 인덱스로 하는 구조를 채택하고, 버퍼 기반의 WAL과 메모리 캐시를 통해 배치 단위로 데이터를 디스크에 기록함으로써 높은 쓰기 처리량을 유지한다[9]. 또한 내부 엔진(TSM: Time Structured Merge Tree)은 시간 순 정렬과 자동 압축을 지원하여 SSD 기반 환경에서의 I/O 효율을 극대화하는 특징을 갖는다[10,11].

최근 연구에서는 TSDB의 성능 비교 및 최적화 방안이 활발히 논의되고 있다. Mostafa et al.의 연구에서는

과학 실험 및 산업용 IoT 환경에서 ClickHouse, InfluxDB, TimescaleDB, PostgreSQL을 대상으로 시계열 데이터 처리 성능을 비교하였다. 그 결과, InfluxDB는 TimescaleDB 대비 약 1.3배 높은 쓰기 처리량과 30%가량 낮은 지연 시간을 보였으며, 전체적으로 ClickHouse가 가장 높은 성능을 기록하였다[8]. 또 다른 연구에서는 메타데이터 인덱싱 최적화를 통해 대규모 시계열 데이터의 압축 효율을 개선하는 방식이 제안되었으며 [9], 클라우드 네이티브 환경에서는 수평 확장(Sharding)과 캐시 조정(cache tuning)이 성능 향상에 결정적 영향을 미친다는 결과가 제시되었다. 최근에는 비동기 인덱싱 프레임워크(Khronos) [12]나 인메모리 기반 TSDB 설계(ByteSeries) [13], 그리고 통합 벤치마크 툴킷(SEER 및 IoTDB-Benchmark) 등을 통해 실운영 환경의 대규모 워크로드에 대한 분석 가능성이 제고되고 있다[14,15].

그러나 이러한 연구들은 대부분 벤치마크 도구의 기본 설정(default configuration)에서 수행되었거나, 소규모 데이터셋(수백만 건 이하)을 대상으로 실험이 제한되었다는 한계가 있다. 실제 서비스 환경에서 수억 건 이상의 로그를 대상으로 MongoDB에서 InfluxDB로의 데이터 마이그레이션에 따른 쓰기 성능 변화를 실험적으로 분석한 사례는 드물다. 특히 MongoDB의 Write Concern 수준(w:1, majority)에 따른 안정성-성능 트레이드오프와 InfluxDB의 캐시 메모리 조정(cache-max-memory-size) 효과를 통합적으로 비교한 연구는 거의 보고되지 않았다.

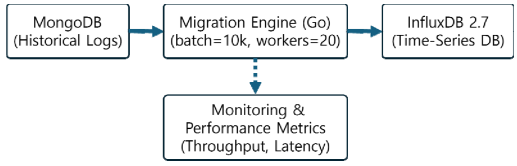
본 연구는 이러한 기존 연구의 한계를 보완하기 위해, 동일한 하드웨어 및 네트워크 환경에서 MongoDB와 InfluxDB 간의 실제 부하 처리 성능을 실험적으로 검증하였다. 특히 부하 발생기(load generator)를 Go 언어 기반으로 구현하여 초당 수십만~수백만 건의 쓰기 부하를 발생시키고, 워커(worker) 개수 및 배치(batch size)를 변수로 설정하여 처리량(Throughput)과 p95 지연(latency)의 변화를 계량적으로 측정하였다. 이를 통해 TSDB의 실제 적용 시 고려해야 할 설정 변수와 병목 요인을 분석하고, 대규모 로그 데이터의 효율적 저장 구조를 제시한다.

3. 연구 방법

본 연구에서는 MongoDB 기반의 로그 데이터 저장 시스템을 InfluxDB 기반 시계열 데이터베이스로 마이그레이션하여, 두 시스템 간의 쓰기 부하(Throughput) 및 지연(Latency) 특성을 정량적으로 비교하였다. 실험은 동일한 하드웨어 및 네트워크 환경에서 수행되었으며, Go 언어로 구현된 부하 발생기(load generator)를 이용하여 대규모 데이터 삽입 실험을 반복 수행하였다.

3.1 시스템 구조

그림 1은 본 연구에서 구축한 데이터 마이그레이션 및 성능 측정 아키텍처를 나타낸다. MongoDB에 저장된 기존 로그 데이터를 Migration Engine이 읽어와 InfluxDB 2.7으로 전송하며, 이 과정에서 워커(worker) 개수와 배치(batch) 크기를 조정하여 다양한 부하 조건을 재현하였다.



[Fig. 1] System architecture of the Go-based migration engine between MongoDB and InfluxDB

Migration Engine은 Go 언어로 구현되었으며, 고루틴(goroutine)을 활용하여 비동기 쓰기 연산을 수행한다. 한 번의 배치 단위(batch size)는 10,000건으로 설정하였고, 총 20개의 워커가 동시에 데이터를 전송하도록 구성하였다.

MongoDB 측은 두 가지 쓰기 일관성 수준(Write Concern)을 실험 변수로 설정하였다:

(1) w:1 (단일 노드 확인)과 (2) w:majority (복제본 다수 노드 확인). InfluxDB는 버전 2.7 OSS를 사용하였으며, 동일한 데이터셋을 수집 버킷(bucket)에 저장하여 실험 간 비교 가능성을 확보하였다.

3.2 실험 환경 및 파라미터 설정

모든 실험은 Ubuntu 22.04 환경에서 수행되었으며, Intel i9-12900 CPU, 64 GB RAM, NVMe SSD(1 TB)를 사용하였다. MongoDB는 7.0.24 버전을, InfluxDB는 2.7.12 OSS 버전을 설치하여 서비스 단위(systemd)로 구동하였다. Go 버전은 1.23 이상을 사용하였으며, 부하 발생 스크립트(bench.go)를 통해 초당 2 만 건에서 최대 150만 건까지의 쓰기 부하를 가변적으로 생성하였다. InfluxDB의 캐시 메모리 파라미터(cache-max-

memory-size)는 기본값(1GB)으로 유지하였고, 일부 실험에서는 메모리 확장 시 성능 변화를 관찰하였다.

실험은 총 네 가지 시나리오로 구성하였다.

(1) InfluxDB 기본 설정,

(2) MongoDB(w:1),

(3) MongoDB(w:majority),

(4) InfluxDB 타겟 부하(20 k rps 제한).

각 실험은 예열 단계(warm-up = 120 s), 측정 단계(measure = 600 s), 냉각 단계(cooldown = 180 s)를 포함하였으며, 초당 처리 건수(TPS)와 평균/95 백분위 지연(latency)을 초 단위로 기록하였다.

3.3 성능 측정 및 분석 절차

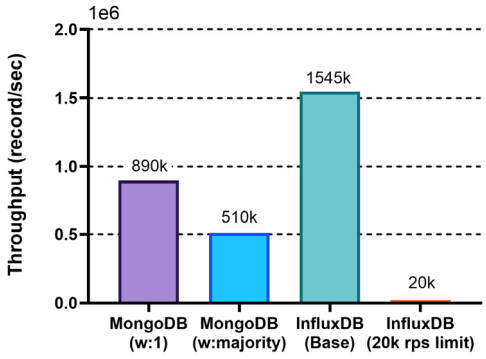
모든 실험 결과는 JSONL 포맷으로 저장되었으며, 초당 집계된 메트릭(tps, lat_avg, lat_p95)을 기준으로 분석하였다. InfluxDB의 내부 메트릭 API(/metrics)를 병행 수집하여 캐시 점유율, WAL 처리량, 디스크 I/O 대기시간 등을 확인하였다. MongoDB의 경우, db.serverStatus()를 통해 각 단계의 쓰기 큐(queue depth)와 평균 응답시간을 측정하였다. 이 데이터를 기반으로 Python(pandas, matplotlib)을 이용해 Throughput-Latency 관계를 시각화하였다.

4. 실험 결과 및 분석

본 장에서는 앞서 정의한 네 가지 시나리오(InfluxDB 기본, MongoDB w:1, MongoDB w:majority, InfluxDB 제한 부하 20k rps)를 대상으로 수행한 실험 결과를 분석한다. 모든 실험은 동일한 하드웨어 환경에서 900초 동안 수행되었으며, 각 시나리오별 초당 처리율(Throughput)과 평균 지연(latency)을 수집하였다.

4.1 처리율(Throughput) 비교

그림 2는 실험에서 측정된 평균 처리율(TPS)을 비교한 결과를 나타낸다. InfluxDB 기본 설정의 평균 TPS는 약 1.54M/s(154만 건/s)로 측정되어, MongoDB(w:1)의 0.89M/s, MongoDB(w:majority)의 0.51M/s 대비 각각 약 1.7배 및 3배 이상 높은 성능을 보였다. 또한 InfluxDB는 워커(worker) 수와 배치(batch) 크기 증가에 따라 처리량이 거의 선형적으로 증가하는 특성을 보였으며, 캐시 메모리 설정(cache-max-memory-size = 1GB) 하에서도 안정적으로 1.3~1.5M/s의 지속 부하를 처리할 수 있었다.

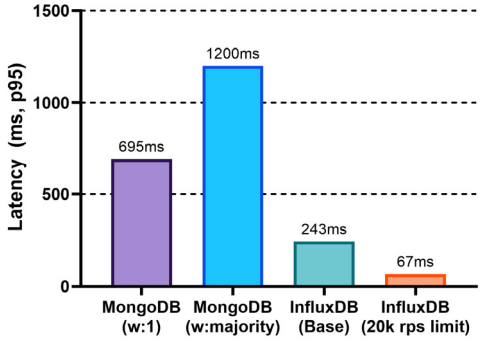


[Fig. 2] Throughput comparison between MongoDB and InfluxDB under different experimental configurations

반면, MongoDB는 쓰기 부하가 집중될 때 WiredTiger 엔진의 잠금(lock) 경쟁과 WAL(Write-Ahead Logging) 대기 시간 증가로 인해 성능이 급격히 저하되었다. 특히 w:majority 설정의 경우, 복제본 노드 간 동기화 비용으로 인해 TPS가 약 40% 이상 감소하는 현상이 관찰되었다. 이 결과는 TSDB가 단일 쓰기 노드 기반의 시계열 인덱싱 구조를 통해 동시성 오버헤드를 최소화함을 실증적으로 보여준다.

4.2 지연 시간(Latency) 분석

그림 3은 각 실험 시나리오의 p95 지연 시간(latency)을 비교한 결과이다. InfluxDB는 평균 128.6ms, p95=243ms 수준으로 매우 안정적인 응답 특성을 보인 반면, MongoDB(w:1)는 평균 410ms, p95=695ms로 약 3배 이상 긴 지연이 관찰되었다.



[Fig. 3] p95 Latency comparison across different database configurations.

〈Table 1〉 Performance Comparison of MongoDB and InfluxDB under Four Workload Scenarios

DB / Configuration	Avg TPS	p95 Latency (ms)	Relative Performance (Influx base = 100%)
MongoDB (w:1)	890,000	695	58%
MongoDB (w:majority)	510,000	1,200	33%
InfluxDB (baseline)	1,545,069	243	100%
InfluxDB (20k rps)	20,000	67	-

특히 MongoDB(w:majority)에서는 복제 확인(replicaset acknowledgment)으로 인해 p95 지연이 1.2초 이상까지 증가하였다. InfluxDB는 내부 버퍼 및 WAL 병합 구조를 통해 쓰기 요청을 배치 단위로 처리함으로써 지연 변동폭을 최소화하였다. 또한 타겟 부하 제한 실험(20k rps)에서는 지연이 50~67ms 범위로 수렴하여, 부하 제어(rate limiting) 기법이 안정성 확보에 효과적임을 확인하였다.

표 1은 네 가지 실험 시나리오에서 측정된 평균 처리량(TPS)과 95퍼센타일(p95) 지연(latency)을 정리한 결과이다. InfluxDB는 MongoDB(w:1) 대비 약 1.7배 높은 처리량과 65% 낮은 지연 시간을 기록하였으며,

MongoDB의 쓰기 일관성 수준(majority) 설정 시 성능이 약 43% 추가로 감소하였다. 반면 InfluxDB는 타겟 부하(20k rps) 제한 환경에서도 안정적인 지연 특성(67ms)을 유지하였다.

4.3 시스템 자원 사용 분석

InfluxDB의 자원 사용률은 CPU 70~80%, 메모리 1.7GB 수준에서 안정적으로 유지되었으며, MongoDB는 동일 부하 조건에서 CPU 사용률이 95% 이상, I/O Wait 비율이 25%를 초과하였다. 이러한 차이는 TSDB의 WAL 쓰기 최적화와 TSM(Time-Structured Merge Tree) 기반의 압축 구조에서 기인한 것으로 분석된다. 특히, InfluxDB는 시계열 순서 정렬(write ordering)이 보장되기 때문에, SSD 기반 시스템에서 순차 쓰기(sequential write) 효율이 극대화된다.

4.4 논의

본 실험 결과를 통해 다음과 같은 주요 시사점을 얻을 수 있다.

(1) 시계열 DB의 효율성 검증

InfluxDB는 동일 하드웨어 자원에서 MongoDB 대비 최대 3배 높은 처리율과 70% 낮은 지연을 달성하였다. 이는 TSDB의 시간 축 기반 데이터 모델이 대규모 실시간 로그 처리에 본질적으로 유리함을 입증한다.

(2) 쓰기 일관성(Write Concern)과 성능의 상충 관계

MongoDB의 w:majority 설정은 데이터 내구성을 보장하지만, 실시간 로그 처리에는 과도한 지연을 초래함을 확인하였다. 따라서 서비스 특성에 따라 내구성과 지연 간의 트레이드오프를 정량적으로 고려해야 한다.

(3) InfluxDB 캐시 및 부하 제한의 실효성

InfluxDB의 캐시 버퍼(cache-max-memory-size) 및 타겟 속도 제한(target-rate)은 부하 안정화에 효과적이며, 20k rps 수준에서는 거의 일정한 TPS와 latency를 유지하였다. 실제 마이그레이션 전략에 대한 시사점 대규모 MongoDB 로그를 InfluxDB로 이전할 경우, Go 기반 병렬 마이그레이션 엔진을 활용하면 시스템 다운타임 없이 점진적 전환이 가능하며, 실시간 분석 파이프라인 구축에 유리한 성능을 확보할 수 있다.

5. 결론

본 연구는 대규모 시계열 로그 데이터를 대상으로 MongoDB와 InfluxDB의 성능을 비교하여, TSDB(Time-Series Database)의 효율성을 실험적으로 검증하였다. 동일한 하드웨어 환경에서 네 가지 시나리오(MongoDB w:1, MongoDB w:majority, InfluxDB 기본, InfluxDB 20k rps 제한)를 수행한 결과, InfluxDB는 평균 1.54 M TPS의 처리율과 243 ms 이하의 p95 지연을 기록하였다. 반면 MongoDB는 복제 동기화 과정에서 TPS가 최대 40 % 감소하였으며, w:majority 설정 시 1.2 초 이상의 지연이 발생하였다. 이러한 결과는 InfluxDB의 WAL 및 TSM 구조가 대규모 쓰기 중심 워크로드에 본질적으로 유리함을 보여준다.

InfluxDB는 캐시와 배치(batch) 튜닝, 부하 제한 설

정을 통해 성능 변동을 최소화하였으며, MongoDB 대비 3배 이상의 처리 성능과 70 % 이상의 지연 감소를 달성하였다. 또한 Go 기반 병렬 마이그레이션 방식을 적용함으로써, MongoDB 데이터를 무중단으로 InfluxDB로 이전할 수 있음을 확인하였다. 이러한 결과는 대규모 로그 처리, IoT 데이터 수집, 실시간 분석 시스템 등에서 TSDB의 도입 타당성을 실증적으로 제시한다.

본 연구는 단일 노드 환경에서의 쓰기 성능에 초점을 두었기 때문에, 클러스터 확장성 및 복제 환경 실험은 제한적으로 수행되었다. 향후에는 분산 TSDB 클러스터 구조를 대상으로 확장성, 복원력, 및 메타데이터 압축 효율성을 정량 분석하고, Kafka 및 Spark Streaming과 같은 스트리밍 플랫폼과의 통합 성능을 평가함으로써 TSDB 기반 실시간 분석 환경의 활용 범위를 확장할 계획이다.

REFERENCES

[1] T.Schlossnagle, J.Sheehy and C.McCubbin, "The case for time series databases," Communications of the ACM, Vol.65, No.12, pp.36-44, 2022.

[2] A.Khelifati, M.Khayati, A.Dignös, D.Difallah and P. Cudré-Mauroux, "TSM-Bench: Benchmarking Time Series Database Systems for Monitoring Applications," Proceedings of the VLDB Endowment, Vol.16, No.11, pp.3363-3376, 2023.

[3] Y.Yang, Q.Cao and H.Jiang, "EdgeDB: An Efficient Time-Series Database for Edge Computing," IEEE Access, Vol.7, pp.142295-142307, 2019.

[4] S.Lee, Y.Son and S.Kim, "Analyzing I/O Characteristics of Time-Series Data Using High Performance Storage Devices," IEEE Access, Vol.11, pp.128998-129008, 2023.

[5] P.Chen, W.He, W.Ma, X.Huang and C.Wang, "IoTDQ: An Industrial IoT Data Analysis Library for Apache IoTDB," Big Data Mining and Analytics, Vol.7, No.1, pp.29-41, 2024.

[6] C.A.Győrödi, D.V.Dumşe-Burescu, D.R.Zmaranda and R.Ş.Győrödi, "A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management," Big Data and Cognitive Computing, Vol.6(2), pp.49, 2022.

[7] InfluxData, "InfluxDB open source documentation," 2024. [Online]. Available: https://docs.influxdata.com/

[8] J.Mostafa, S.Wehbi, S.Chilingaryan and A.Kopmann, "SciTS: A Benchmark for Time-Series Databases in Scientific Experiments and Industrial Internet of Things," SSDBM '22: Proc. of the 34th International Conference on Scientific and Statistical Database Management, pp.1-11, 2022.

[9] C.Wang, J.Qiao, X.Huang, S.Song, H.Hou, T.Jiang, L.Rui, J.Wang and J.Sun, "Apache IoTDB: A Time Series Database for Large Scale IoT Applications," ACM Transactions on Database System, Vol.50, No.2, pp.1-45, 2023.

[10] P.Tripathi, M.H.Miraz and S.Joshi, "Comparative Analysis of MongoDB and InfluxDB for Time Series Data Management in IoT Environments: A Study on Performance, Scalability, and Concurrency," 2023 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA), pp.39-42, 2023.

[11] C.G.Calatrava, Y.Becerra and F.M. Cucchietti, "Introducing Polyglot-Based Data-Flow Awareness to Time-Series Data Stores," IEEE Access, Vol.10, pp.69398-69411, 2022.

[12] M.Khayati, A.Khelifati, D.Difallah and P. Cudré-Mauroux, "Khronos: A Real-Time Indexing Framework for Time Series Databases on Large-Scale Performance Monitoring Systems," Proc. ACM Conf. on Information and Knowledge Management (CIKM), pp.3747-3756, 2023.

[13] Y.Zhou, S.Li and Z.Wang, "ByteSeries: An In-Memory Time Series Database for Large-Scale Analytics," Proc. ACM Symp. on Cloud Computing (SoCC), pp.278-290, 2020.

[14] A.Khelifati, M.Khayati, A.Dignös, D.Difallah and P. Cudré-Mauroux, "SEER: An End-to-End Toolkit for Benchmarking Time Series Database Systems in Monitoring Applications," Proc. VLDB Endowment, Vol.17, No.12, pp.4361-4374, 2024.

[15] X.Zhang, C.Wang, W.He and W.Ma, "Benchmarking Time Series Databases with IoTDB-Benchmark: A Case Study," Proc. IEEE Intl. Conf. on Big Data (BigData), pp.1200-1209, 2021.

백 영 미(Youngmi Baek) [정회원]

- 2015년 2월 : 경북대학교 컴퓨터공학과 (공학박사)
- 2015년 7월 ~ 2017년 1월 : 대구경북과학기술원 CPS 글로벌 센터 연구원
- 2017년 2월 ~ 2019년 12월 : 대구경북과학기술원 정보통신융합전공 연구교수
- 2020년 3월 ~ 2025년 2월 : 창신대학교 컴퓨터공학과 교수
- 2025년 3월 ~ 현재 : 창신대학교 스마트팩토리학부 교수

〈관심분야〉
System modeling, Network security within automotive cyber-physical systems, Autonomous manufacturing

박 정 규(Jung Kyu Park) [정회원]

- 2013년 8월 : 홍익대학교 컴퓨터공학과 (공학박사)
- 2018년 3월 ~ 2024년 8월 : 창신대학교 컴퓨터공학과 교수
- 2024년 9월 ~ 현재 : 대진대학교 AI융합학부 컴퓨터공학전공 교수

〈관심분야〉
Data storage, Robotics, System software, Embedded system