https://doi.org/10.5392/JKCA.2021.21.09.198

ArangoDB기반 벤치마킹 시스템 설계 및 구현

Design and Implementation of a Benchmarking System Based on ArangoDB

최도진[*], 백연희[**], 이소민[*], 김윤아[**], 김남영[**], 최재용[*], 이현병[*], 임종태[*], 복경수[***], 송석일[****], 유재수[*]
충북대학교 정보통신공학과[*], 충북대학교 빅데이터학과[**], 원광대학교 SW융합학과[***],
한국교통대학교 컴퓨터공학과[****]

**Do-Jin Choi(mycdj91@cbnu.ac.kr)[*], Yeon-Hee Baek(yh100@cbnu.ac.kr)[**],
So-Min Lee(somin@cbnu.ac.kr)[*], Yun-A Kim(rud5356@naver.com)[**],
Nam-Young Kim(minstrel68@naver.com)[**], Jae-Young Choi(headmeat@naver.com)[*],
Hyeon-Byeong Lee(lhb@cbnu.ac.kr)[*], Jong-Tae Lim(jtlim@cbnu.ac.kr)[*],
Kyoung-Soo Bok(ksbok@wku.ac.kr)[***], Seok-Il Song(sisong@ut.ac.kr)[****],
Jae-Soo Yoo(yjs@cbnu.ac.kr)[*]**

요약

ArangoDB는 대용량 데이터 저장을 위해 많은 응용에서 활용되고 있는 NoSQL 데이터베이스 시스템이다. ArangoDB와 같은 새로운 NoSQL 데이터베이스 시스템을 실제 환경에 적용하기 위해서 성능을 평가해 줄 수 있는 벤치마킹 시스템이 필요하다. 본 논문에서는 응용 계층뿐만 아니라 커널 계층에서의 성능이 측정 가능한 ArangoDB 기반 벤치마킹 시스템을 설계하고 구현한다. 클러스터 환경에서의 NoSQL 데이터베이스 시스템 성능을 측정하기 위해서 YCSB를 일부 수정한다. 또한, 기존 자료 분석을 통해 실세계에서 발생하는 세 가지 워크로드 유형을 정의한다. 세 가지 워크로드 유형의 벤치마킹을 통해 ArangoDB에서 활용 가능한 워크로드를 도출하였고, 응용 계층뿐만 아니라 커널 계층의 성능이 가시화될 수 있음을 입증하였다. 기존 데이터베이스에서 ArangoDB로 데이터 이전 작업이 필요한 환경에서는 본 시스템의 벤치마킹을 통해 적용 가능성과 리스크 검토가 가능할 것으로 기대된다.

■ 중심어 : | ArangoDB | 데이터베이스 | 벤치마킹 | 워크로드 | NoSQL |

Abstract

ArangoDB is a NoSQL database system that has been popularly utilized in many applications for storing large amounts of data. In order to apply a new NoSQL database system such as ArangoDB, to real work environments we need a benchmarking system that can evaluate its performance. In this paper, we design and implement a ArangoDB based benchmarking system that measures a kernel level performance well as an application level performance. We partially modify YCSB to measure the performance of a NoSQL database system in the cluster environment. We also define three real-world workload types by analyzing the existing materials. We prove the feasibility of the proposed system through the benchmarking of three workload types. We derive available workloads in ArangoDB and show that performance at the kernel layer as well as the application layer can be visualized through benchmarking of three workload types. It is expected that applicability and risk reviews will be possible through benchmarking of this system in environments that need to transfer data from the existing database engine to ArangoDB.

■ keyword : | ArangoDB | Database | Benchmarking | Workload | NoSQL |

* 이 (성과)는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원과(No. 2019R1A2C2084257) 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원과(No.2014-3-00123, 실시간 대규모 영상 데이터 이해·예측을 위한 고성능 비주얼 디스커버리 플랫폼 개발) 농촌진흥청 연구사업 (세부과제번호: PJ01624701)의 지원과 과학기술정보통신부 및 정보통신기획평가원의 지역지능화혁신인재양성(Grand ICT연구센터) 사업의 연구결과로 수행되었음" (IITP-2021-2020-0-01462)

접수일자 : 2021년 05월 06일
수정일자 : 2021년 05월 27일
심사완료일 : 2021년 05월 27일
교신저자 : 유재수, e-mail : yjs@cbnu.ac.kr

I. 서 론

관계형 데이터베이스는 빅데이터의 등장으로 인해 데이터를 처리하는 비용의 증가가 발생하기 시작하였다. 이에 따라 일관성을 일부 포기하고 확장성 및 비정형 데이터를 효율적으로 처리하는 NoSQL 데이터베이스 시스템이 등장하였다. NoSQL 데이터베이스 시스템은 데이터를 저장하는 형태에 따라 키-값, wide-column, 그래프, 문서(document), 객체, 시계열 등 다양한 형태의 데이터베이스 유형이 존재한다.

ArangoDB는 ArangoDB GmbH가 개발한 무료 오픈 소스 네이티브 다중 모델 NoSQL 데이터베이스 시스템이다[1]. ArangoDB는 데이터 모델 중 키-값, 문서, 그래프 데이터 모델 세 가지를 지원한다. ArangoDB에서 개발한 질의 언어 (AQL : ArangoDB Query Language)를 통해 서로 다른 데이터 모델에서 동일한 질의로 데이터에 접근할 수 있다. ArangoDB가 다중 데이터 모델을 지원함에 따라 데이터 모델링의 유연성이 향상되어 다양한 응용에서 활용 가능하다. ArangoDB는 현재까지도 활발하게 개발되고 있는 NoSQL 데이터베이스 시스템이다.

ArangoDB와 같은 다중 모델 데이터베이스 시스템은 다양한 응용에 활용할 수 있기 때문에 응용에 따라 데이터베이스의 효율성 및 적합성을 검증할 도구가 필요하다[2-10]. 적합성이란 해당 데이터베이스를 적용하고자 하는 응용에서 생성되는 데이터를 적절하게 관리할 수 있는지를 검증 하는 것을 의미한다. 추가적으로 사용자의 질의 요청에 대한 응답 시간을 지정한 시간 내에 얻을 수 있는지를 같이 검증해야한다.

데이터베이스에 대한 효율성 및 적합성 검증을 보조하기 위해서 기존에는 YCSB(Yahoo! Cloud Serving Benchmark)와 같은 벤치마킹 도구가 존재한다[11-14]. 벤치마킹이란 기정의된 워크로드 혹은 사용자가 정의한 워크로드에 대해서 벤치마킹 대상의 데이터베이스가 어느 정도의 성능(응답시간, 응답률)을 나타내는지를 측정하는 것을 의미한다. 벤치마킹을 통해 정의한 워크로드에서 데이터베이스가 실제 운용 가능할지를 미리 판단해 볼 수 있다. 기존 벤치마킹 도구는 입력되는 데이터나 질의 형태를 워크로드로 정의한다. 정의된 워크로드를 통해 반복 작업을 수행하여 데이터베이스 시스템의 성능을 측정한다. 사용자들은 워크로드를 스스로 정의할 수 있으며 벤치마킹 도구에서 데이터베이스 시스템의 인터페이스를 제공하기 때문에 손쉽게 벤치마킹을 수행할 수 있다. 그러나 기존 벤치마킹 도구는 범용적인 목적으로 설계되었기 때문에 워크로드가 정해진 항목의 값만을 설정할 수 있으며 데이터베이스 시스템이 클러스터링 환경에서의 테스팅을 지원하지 않는 경우가 많다. 또한, 사용자가 지정한 데이터베이스 시스템에 대해 CRUD(Create/Read/Update/Delete) 연산을 수행하고 연산에 대한 응답 시간만을 측정하기 때문에 데이터베이스 시스템에서의 어떠한 연산이 병목 현상을 일으키는지 제대로 파악하기가 매우 힘들다. ArangoDB는 클러스터링 환경을 제공하기 때문에 클러스터 내의 노드별 성능 측정이 필요하고 CRUD 연산의 병목 현상을 추적하기 위한 방안이 필요하다.

본 논문에서는 ArangoDB가 실제 환경과 유사하게 구축된 환경에서 벤치마킹을 수행할 수 있는 벤치마킹 시스템을 설계하고 구현한다. 제안하는 시스템은 클러스터 환경에서 벤치마킹을 수행하기 위해서 기존 벤치마킹 도구인 YCSB를 기반으로 한다. 제안하는 시스템은 보조 기억 장치에 대한 Block I/O 추적 기술을 활용하여 응용 계층의 성능뿐만 아니라 커널 계층에 대한 성능까지 측정할 수 있다. 실험 결과 분석을 통해 제안하는 기법의 타당성 및 실효성을 입증한다.

본 논문의 구성은 다음과 같다. II 장에서는 관련 연구를 설명한다. III 장에서는 제안하는 시스템의 특징과 제안하는 시스템에서 벤치마킹하는 방법을 설명하고, IV 장에서는 제안하는 기법의 타당성 및 실효성을 입증하기 위해 성능 평가를 수행한다. 마지막으로 V장에서는 본 논문의 결론과 향후 연구를 제시한다.

II. 관련 연구

TPC (Transaction Processing Performance Council)는 OLTP(Online Transaction Processing) 시스템의 처리 성능을 측정하기 위한 벤치마킹 도구이며, 벤치마킹하고자 하는 유형에 따라 TPC-C, TPC-E, TPC-H로 세 가지로 유형이 제공된다[11]. TPC는

SQL문을 통한 벤치마킹을 수행하기 때문에 ArangoDB와 같은 NoSQL 데이터베이스 시스템에 대한 테스팅을 수행하기 위해서는 SQL 문을 AQL문으로 번역하는 중간 도구가 필요하다. TPC는 전통적인 DBMS(Database Management System)에 적합한 벤치마킹 도구로 여겨지고 있다.

YCSB는 야후에서 개발한 클라우드 서비스의 성능을 측정하기 위한 벤치마킹 도구이다[12]. TPC와 달리 RDBMS뿐만 아니라 NoSQL에 대한 벤치마킹까지 가능한 도구이다. 실생활에서 발생하는 워크로드 유형을 분석하여 기본 워크로드 유형 파일로 제공하고 있으며, GitHub을 통해 공개하여 사용자가 원하는 형태의 벤치마킹 도구로 변경이 가능한 벤치마킹 도구이다. 또한 데이터베이스 시스템의 인터페이스와 CRUD 연산에 대한 인터페이스를 제공해주기 때문에 YCSB에서 지원하지 않는 데이터베이스 시스템이더라도 사용자가 직접 구현하여 추가할 수 있다. 현재 HBase, Cassandra, MongoDB, Redis, RocksDB, ArangoDB등 많은 데이터베이스 시스템을 지원한다. 범용성을 목적으로 만들어진 YCSB는 벤치마킹 대상의 데이터베이스 시스템의 클러스터 환경을 지원하지 못하는 상황이 대부분이다. 또한, 가상의 I/O 연산을 생성하여 요청에 대한 응답 시간만을 측정하기 때문에, 실제 데이터베이스 시스템에서의 응답 지연과 관련된 성능을 측정하기는 어렵다. 이러한 YCSB의 문제점들을 개선하기 위한 연구들도 활발히 연구 중이다[13][14].

기존 YCSB는 그래프 데이터베이스 시스템을 벤치마킹하지 못한다. 그래프 데이터는 데이터 간의 연결 정보를 담고 있고, 그래프 데이터베이스에서는 워크로드가 응용 마다 매우 다양하게 정의될 수 있기 때문에 기존에 YCSB에서 정의한 워크로드만으로는 이러한 요구사항을 모두 만족하기 힘들다. LDBC-SNB(Linked Data Benchmark Council-Social Network Benchmark)는 소셜 네트워크 환경의 데이터를 기반으로 그래프 데이터베이스 시스템을 벤치마킹하고 가상의 그래프 데이터를 생성하는 도구이다[15]. 실시간으로 처리되는 상호작용 워크로드(interactive workload)와 분석용 워크로드로(business intelligence workload) 분류가 된다. 그래프 데이터베이스 시스템 전용 벤치마킹 도구로써 가장 많이 활용되고 있다. LDBC 또한 범용성을 목적으로 두고 있기 때문에 데이터베이스에 대한 실제 구성 환경과 연관된 성능 테스트를 수행하지 못한다.

blktrace는 블록 계층의 IO 요청을 추적하여, 요청 큐(request queue)에 대한 자세한 정보를 제공하는 Block I/O 추적 도구이다[16]. 커널에서 제공하는 TRACE_EVENT를 이용하여 블록 계층의 I/O를 추적하고 추적 정보를 분석하여 사람이 읽을 수 있는 형태로 변환한다. 사용자는 커널 버퍼에 저장된 추적 정보를 릴레이 채널(relay-channel)을 통해 파일 형태로 취득 할 수 있다.

III. ArangoDB 기반 벤치마킹 시스템 설계 및 구현

1. 주요 특징

실제 환경과 유사하게 구축된 환경에서의 벤치마킹을 수행하기 위해서는 새로운 벤치마킹 도구가 필요하다. 본 논문에서는 다중 모델을 지원하는 ArangoDB를 실제 운영 환경과 유사하게 구성하였을 때 성능을 측정하는 벤치마킹 시스템을 제안한다. 클러스터 시스템의 응용 계층 성능을 측정하기 위해서 기존 NoSQL 데이터베이스 시스템 벤치마킹 도구로 자주 활용되는 YCSB를 기반으로 한다. 또한, 커널 계층의 성능을 추적하기 위해서 blktrace[16]와 유사한 Block I/O 추적기를 제안한다. 최종 벤치마킹 결과는 시각화 모듈을 통해 그래프 형태로 제공한다.

[그림 1]은 제안하는 벤치마킹 시스템 구조도를 나타낸다. 사용자는 원하는 형태의 CRUD를 생성하기 위해서 YCSB에서 제공하는 형태의 워크로드 파일을 생성한다. 생성된 워크로드는 수정된 YCSB를 통해 벤치마킹이 수행된다. 이와 동시에 클러스터링의 각 노드의 Block I/O를 추적하기 위해서 I/O tracer가 실행된다. YCSB가 생성한 응용 수준의 벤치마킹 결과와 I/O tracer가 생성한 커널 수준의 벤치마킹 결과는 CSV(Comma Separated Value) 변환기를 통해 CSV 포맷으로 수정된다. 최종적으로 가시화 모듈을 통해 벤치마킹 결과를 그래프 형태로 제공한다.

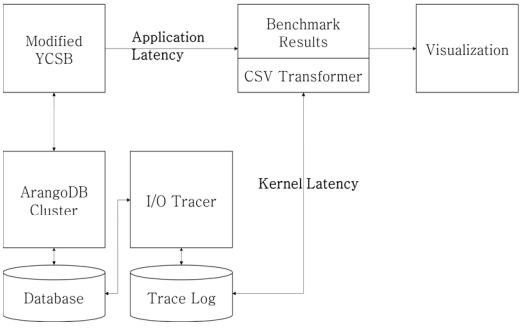


그림 1. 제안하는 벤치마킹 시스템 구조도

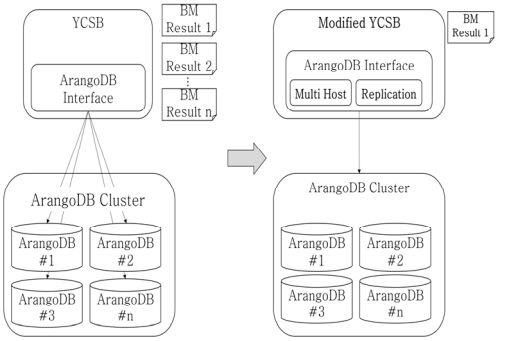


그림 2. 수정된 YCSB

2. YCSB 수정

YCSB는 기존의 다양한 데이터베이스 시스템에 대한 벤치마킹을 지원하는 벤치마킹 도구이다. 그뿐만 아니라 인터페이스 정의를 통해 YCSB에서 구현되어있지 않은 다른 데이터베이스 시스템 또한 사용자가 추가 구현 가능하게 설계되었다. 세션 저장, 사용자 프로필 캐싱, 사용자 상태 업데이트, 대화 목록 조회, 프로필 업데이트 등 실제 응용에서 발생하는 워크로드를 정의하고 이를 벤치마킹에 활용한다. 정의된 워크로드를 기반으로 데이터베이스 시스템의 성능을 측정하고 측정된 결과는 연산(CRUD) 별로 통계 정보를 제공한다. 워크로드 또한 파일로 정의되기 때문에 사용자가 손쉽게 수정 및 활용할 수 있다. 그러나 YCSB에서는 클러스터 구성 환경에서 ArangoDB의 벤치마킹을 지원하지 않기 때문에 기존 YCSB를 수정해야만 한다.

[그림 2]는 클러스터 구성 환경에서 ArangoDB를 벤치마킹하기 위한 수정된 YCSB의 구성도를 나타낸다. 기존 YCSB에서는 단일 데이터베이스에 대해서만 벤치마킹이 가능하기 때문에 실제로 ArangoDB가 클러스터 환경으로 구성되었다 하더라도 하나의 클러스터 노드에 대해서만 벤치마킹을 수행하고 각 클러스터 노드별로 벤차미킹 결과(BM result)가 생성되는 구조로 동작한다. 제안하는 수정된 YCSB는 ArangoDB 클러스터를 지원하기 위해서 YCSB의 ArangoDB 인터페이스에 호스트 목록을 추가하는 다중 호스트(multi host)기능과 복제 계수(replication factor)를 설정하는 복제 기능을 추가하였다.

다중 호스트는 클러스터로 구성된 서버의 목록을 추가하는 기능이다. 각 서버는 고유의 IP와 포트 번호를 가지고 있으며 이를 YCSB 실행 명령어를 통해 입력받도록 수정하였다. 호스트 목록은 사용자가 유연하게 구성할 수 있고, 사용자가 입력한 호스트 목록에 대해서만 Block I/O가 추적 가능하다.

복제 계수란 클러스터 환경에서 하나의 데이터베이스에 대해서 얼마나 많은 사본을 구성할 것인지를 의미하는 수치이다. 기본 값은 1로 제공되는데, 이는 사본을 구성하지 않는다는 의미이다. 복제 계수를 1로 설정한 상황에서 클러스터 노드 중 1대가 장애가 발생하면 해당 노드가 가지고 있는 정보를 모두 소실하는 상황이 발생한다. 클러스터 환경에서는 복제 계수를 통상적으로 3을 두고 사용하는데 YCSB에서 이를 지원하기 위해서 해당 기능을 추가하였다. 이를 통해 기본적인 쓰기 연산이 3대의 노드에 대해 수행되며 이에 대한 성능 측정을 수행해야만 실제 환경에서의 벤치마킹이 가능하다고 할 수 있다. 두 기능을 추가하면 ArangoDB 클러스터에 대한 성능 평가를 정상적으로 수행할 수 있다. 수정된 YCSB를 통해서 클러스터의 노드만큼 생성된 벤치마킹 결과 파일이 단일 파일로 줄어든다.

3. I/O tracer

기존 벤치마킹 도구들이 커널 계층의 성능 측정이 불가능한 문제점을 해결하기 위해서 커널 계층의 성능 측정을 수행하는 I/O tracer를 제안한다. 제안하는 I/O tracer는 BPF(Berkeley Packet Filters) 프로그램 작

성을 쉽게 만들어주는 도구인 BCC(BPF Compiler Collection)를 활용한다. BCC는 커널 함수 추적과 조작 프로그램을 효율적으로 생성하는 도구이며, 이를 일부 수정하여 사용한다[17].

[그림 3]은 제안하는 I/O tracer 구조도를 나타낸다. YCSB에서 워크로드에 대한 I/O 요청은 ArangoDB로 바로 입력이 되고 ArangoDB는 내부 정책에 따라 HDD나 SSD 같은 보조 기억 장치에 데이터를 읽거나 저장한다. 이때 보조 기억 장치에 대한 I/O 요청은 일정하게 처리되는 것이 아니라 운영체제의 커널 및 device driver에 따라 I/O 요청이 수행된다. BCC와 driver manager를 활용하여 VFS, block layer, device driver, device에 대해서 I/O 연산에 대한 처리 과정을 추적 및 기록한다. 기록 정보는 trace log로 저장되어 사용자가 읽기 쉬운 형태로 제공한다. 추후 가시화 작업을 수행하여 그래프 형태로 결과를 제공한다.

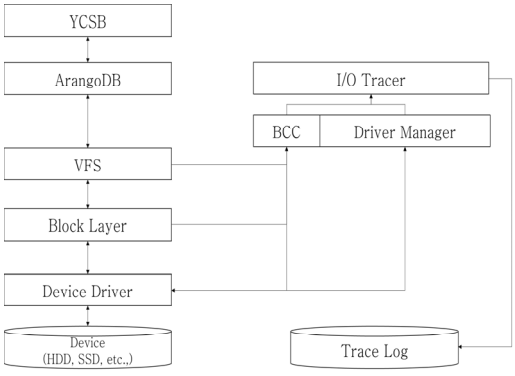


그림 3. I/O Tracer 구조도

[표 1]은 I/O tracer의 결과 파일을 나타낸다. time은 I/O가 추적된 시간을 의미하고 마이크로초 (us) 단위로 측정한다. mode는 I/O의 유형을 의미한다. R은 읽기 요청이고 W는 쓰기 요청이다. I/O size는 I/O의 크기를 나타내며 KB 단위이다. total latency는 I/O 요청에 대한 커널 계층에서의 총 지연시간을 의미한다. 커널 계층에서는 내부적으로 VFS, device driver, driver 계층으로 분리될 수 있으며 I/O tracer는 각 계층별로 처리 시간을 측정하여 어떠한 계층에서 병목 현상이 발생하는지 파악한다. 여기서 시간 단위는 나노초(ns) 단위로 측정한 값을 사용자에게 제공한다. 예를 들어, 첫 번째 행의 의미는 다음과 같다. 벤치마킹이 시작되고 0.478us 후에 읽기(R) 연산이 발생하였고, 해당 I/O를 처리하기 위해서는 총 93.4ns가 소요되었음을 의미한다. 이 때 총 시간 중 1.762ns는 VFS 계층에서, 1.785ns는 driver 계층에서, 89.909는 device 계층에서 소요되었을 의미한다. 이를 통해 device 계층에서 많은 시간이 소요됨을 확인 할 수 있다. 주로 device 계층에서의 시간이 가장 오래 걸리지만 2번째 행에서는 device driver에서 높은 지연 시간을 나타낸다. 계층별 I/O 추적 정보를 통해 병목 지점을 파악 할 수 있고, 튜닝 가능한 영역을 구분해낼 수 있다.

표 1. I/O Tracer 결과 파일

Time	Mode	I/O Size	Total Latency	VFS Latency	Driver Latency	Device Latency
0.478	R	4096	93.4	1.762	1.785	89.909
0.721	R	4096	123.279	2.492	25.536	95.251
0.807	R	4096	74.322	1.276	4.625	68.421
0.862	W	4096	24.857	1.351	3.535	19.971
0.916	W	4096	19.331	1.269	3.464	14.598

4. 워크로드 유형

본 논문에서는 실제 응용과 유사한 환경에서의 벤치마킹을 수행하기 위해서 세 가지 워크로드 유형을 제시한다. 세 가지 워크로드 유형은 기존의 NoSQL 기반의 응용 사례를 기반으로 만들어졌다. 대용량 멀티미디어 서비스 기반의 콘텐츠 캐싱 응용과 SNS의 개인 메신저 서비스 기반의 SNS 메신저 응용, 마지막으로 개인화된 추천 서비스를 제공하는 개인화 추천 응용이 있다.

초고속 인터넷 발달과 스마트 기기의 보급화로 인해 넷플릭스, 유튜브, 티빙, 왓챠 등 OTT(Over-The-Top media) 서비스가 활발하게 이루어지고 있다. 이러한 응용을 콘텐츠 캐싱 응용으로 정의한다. 멀티미디어 콘텐츠의 특성상 데이터 크기가 방대하며 인기 있는 콘텐츠일수록 더욱 자주 요청이 되는 특성이 있다. 콘텐츠의 업로드가 많지만 상대적으로 콘텐츠를 시청하는 일이 더욱 많기 때문에 이러한 특성을 반영한 워크로드가 정의되어

야 한다[18].

페이스북이나 인스타그램과 같은 소셜 네트워크 서비스는 DM(Direct Message)이라고 하는 1:1 메신저 서비스가 활발하게 사용된다. SNS 메신저 응용의 경우 데이터의 크기는 작고, 메시지를 읽는 요청과 쓰는 요청이 비슷한 양상을 나타낸다. 주로 최근 메신저 기록을 읽는 경향이 있으며, [19]에서는 전체 연산의 60%가 읽기 연산이고 34%가 쓰기 연산이며 6%가 삭제 연산으로 분석하고 있다.

개인화 추천 응용은 최신 뉴스, 개인화 뉴스 추천, 개인화 검색 결과를 제공하는 응용을 의미한다. 모바일 서비스의 발달로 인해 사용자는 다양한 개인화 서비스를 받고 있다. 뉴스의 경우 관심사를 선택하면 해당 관심사와 관련된 뉴스가 주로 추천되는 것을 확인할 수 있다. 개인화 추천 응용 또한 최신의 콘텐츠를 주로 접근하며, 읽기 연산이 쓰기 연산보다 많은 양상을 나타낸다[20].

[표 2]는 세 가지 워크로드 유형의 YCSB의 파라미터 정보를 나타낸다. 레코드 크기는 데이터베이스 시스템의 레코드 한 개의 크기를 의미하고, 레코드는 기본적으로 10개의 필드(속성)를 가지고 있으며, YCSB에서 생성하는 임의의 문자열 값이 저장된다. 레코드 수는 데이터베이스 시스템에 미리 적재할 레코드의 수를 의미한다. 추후 성능 평가에서 벤치마킹 시스템의 저장 크기를 고려하여 해당 수를 지정하였다. 연산 수는 데이터베이스 시스템에 대한 읽기와 쓰기 연산의 총합을 의미한다. 정의된 연산 수에서 연산 비율 (R:W)만큼 연산이 수행된다. 예를 들어, 콘텐츠 캐싱 같은 경우 200K개의 연산 중 90%인 180K만큼 읽기 연산이 수행되고 나머지 20K만큼 쓰기 연산이 수행된다는 것을 의미한다. 요청 분포는 읽기/쓰기 연산이 어떤 레코드에 주로 요청될 것인지를 지정하는 분포 값이다. hotspot은 자주 요청되는 레코드들의 목록이 존재하고, 이러한 레코드에 대해서 주로 요청이 되는 것을 의미한다. latest는 최근에 입력된 레코드일수록 많은 요청이 발생됨을 지정하는 것이다. YCSB에서는 이외에도 uniform, zipfian, sequental과 같은 다양한 요청 분포를 지정할 수 있다.

표 2. 워크로드 파라미터

파라미터	콘텐츠 캐싱	SNS 메신저	개인화 추천
레코드 크기	10MB	1MB	1MB
레코드 수	55K	550K	550K
연산 수	200K	500K	1.43M
연산 비율(R:W)	90:10	60:40	86:14
요청 분포	hotspot	latest	latest

5. 벤치마킹 수행 과정

YCSB는 사전에 정의한 전체 데이터의 크기만큼 삽입하는 적재(load) 단계를 먼저 수행하고, 적재된 데이터에 대해 연산 요청을 수행하는 실행(run) 단계 두 단계로 구분되어 수행한다. 먼저 적재 단계에서는 사전에 정의한 레코드 수만큼 데이터베이스 시스템에 입력이 이루어지고, 각 레코드의 크기 또한 사전에 정의한 크기만큼의 레코드가 입력된다. 실행 단계에서는 마찬가지로 정의된 연산 수, 연산 비율, 요청 분포를 바탕으로 특정 레코드에 대한 연산(비율에 따라 결정)이 수행된다. 실행 단계는 연산 수만큼의 연산이 수행되고 나면, 최종적으로 벤치마킹 결과를 생성하고 실행 단계를 종료한다.

[그림 4]는 제안하는 기법의 벤치마킹 실행 과정을 나타낸다. 가장 먼저 벤치마킹을 수행하기 위한 워크로드(3.4절에 정의한 3개의 워크로드 중 하나)를 지정한다. 그런 후 Block I/O를 추적하기 위해서 ArangoDB 클러스터의 노드 별로 I/O tracer가 실행된다. YCSB의 적재 단계와 실행 단계를 거치고 나면 YCSB 결과가 생성된다. 생성된 결과는 CSV 파일로 변환되어 I/O tracer 결과와 통합된다. 가시화 모듈은 통합된 최종 결과를 그래프 형태로 가시화한다.

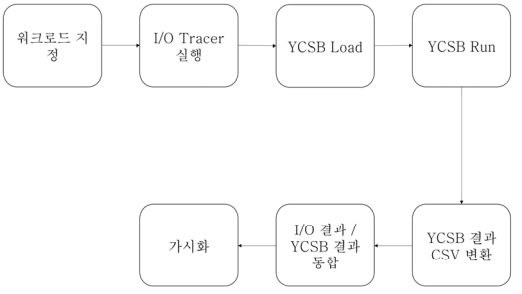


그림 4. 벤치마킹 실행 과정

IV. 성능 평가

제안하는 벤치마킹 시스템의 타당성과 실효성을 입증하기 위해서 ArangoDB 클러스터에 대한 벤치마킹을 수행한다. [표 3]은 본 논문에서 수행한 성능 평가 환경을 나타낸다. 동일한 성능을 가진 3대의 서버를 ArangoDB 클러스터로 구성하였으며, 리눅스 CentOS 환경에서의 성능 평가를 수행한다. 각 서버는 Intel Core i7-6700 CPU 3.4GHz 8 Core와 32GB의 메모리를 탑재하였다. 3.4절에서 정의한 워크로드 별로 벤치마킹을 수행한다.

표 3. 성능 평가 환경

이름	값
CPU	Intel Core i7-6700 CPU 3.4GHz 8 Core
Memory	32GB
OS	Cent OS 8.2.2004
Kernel Version	4.9.216 x86_64
YCSB Version	0.17.0
ArangoDB Version	3.6.3
# of servers	3

[그림 5]는 워크로드 별 벤치마킹 결과를 나타낸다. 콘텐츠 캐싱 응용은 각 레코드의 크기가 크기 때문에 상대적으로 높은 지연시간을 나타낸다. 콘텐츠 캐싱의 읽기 연산은 대략 225ms의 지연시간을 나태고, 쓰기 연산은 464ms의 지연시간을 나타낸다. SNS 메신저 응용의 경우 21ms의 읽기 지연시간과 48ms의 쓰기 지연시간을 나타낸다. 마지막으로 개인화 추천의 경우 20ms의 지연시간과 46ms의 지연시간을 나타냄을 확인할 수 있었다. 성능 평가 결과 콘텐츠 캐싱의 응용보다는 개인화 추천이나 SNS 메신저 응용에 더욱 적합한 데이터베이스 시스템임을 알 수 있었다.

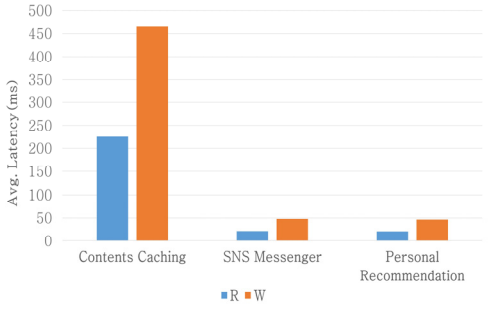


그림 5. 워크로드 별 벤치마킹 결과

[그림 6]은 YCSB의 상세한 벤치마킹 결과를 나타낸다. YCSB는 적재 단계와 실행 단계에서의 연산 별 요청 시간만을 측정하기 때문에 벤치마킹을 통해서 총 3가지의 결과만을 확인할 수 있다. 먼저 적재 단계에서는 쓰기 연산만 발생하기 때문에 쓰기 연산의 응답 시간만을 확인할 수 있다. 적재 단계 초기에는 ArangoDB가 warm-up 되어있지 않기 때문에 일부 높은 응답 시간을 나타내지만, 시간이 지날수록 안정된 응답 시간이 나오는 것을 확인할 수 있다. 실행 단계에서는 읽기 연산과 쓰기 연산이 동시에 수행되기 때문에 각 연산에 대한 결과 그래프를 얻을 수 있다. 읽기 연산은 전체적으로 고른 분포를 나타내지만 쓰기 연산 같은 경우에는 특정 구간에 높은 응답 시간을 나타낸다. 쓰기 연산이 특정 구간에서 급격하게 증가하는데 이러한 부분을 통해, 해당 응용(워크로드)에서 튜닝이 가능한 부분을 확인할 수 있다.

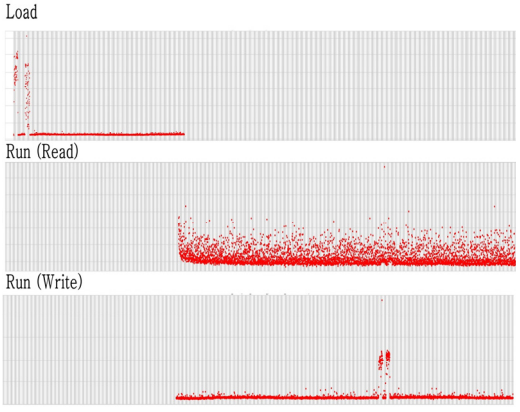


그림 6. YCSB 벤치마킹 결과

[그림 7]은 노드 별 I/O 연산 수를 나타낸다. 제안하는 시스템은 노드별로 I/O tracer가 동작 되기 때문에 노드 별로의 I/O 수를 측정 할 수 있다. 또한, I/O 연산이 읽기인지 쓰기인지도 확인 할 수 있기 때문에 읽기 연산, 쓰기 연산 별 횟수를 가시화 할 수 있다. 그림에서 푸른색은 쓰기 연산의 수이며 주황색은 읽기 연산의 수이다. YCSB의 적재 단계에서는 쓰기 연산만 발생하기 때문에 쓰기 연산이 3개 노드에서 많이 발생하는 반면, 실행 단계에서는 쓰기 연산의 수가 50%이상 급격히 줄어드는 것을 확인 할 수 있다. 노드 2를 중점으로 읽기 연산이 많이 발생하고, 노드 1에서는 특정 구간 이후에는 I/O 요청이 없어짐을 확인할 수 있다. 이를 통해 ArangoDB에서의 I/O 요청의 부하를 분산시켜야 함을 도출할 수 있다.

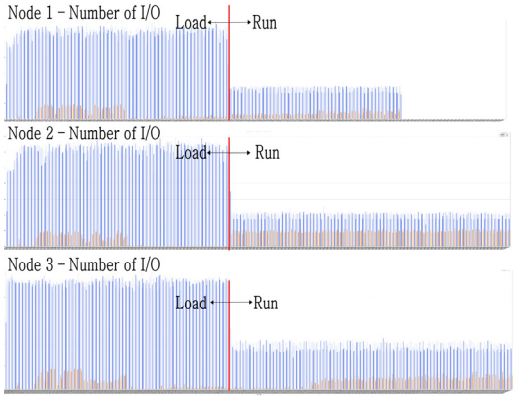


그림 7. 노드 별 I/O 연산 수

[그림 8]은 노드 1의 I/O tracer 추적 결과를 나타낸다. 추적 결과는 노드별로 측정 가능하며, 설명의 편의를 위해 본 논문에서는 노드 1개의 결과만을 가시화 하였다. 이전 결과와 마찬가지로 YCSB의 적재 단계에서 많은 쓰기 연산이 발생하여 커널 내부의 모든 I/O 처리 층에서 I/O의 응답 시간이 높게 나오는 것을 확인할 수 있다. 특히 GC(Garbage Collection)가 실행 단계보다 높게 발생하기 때문에 I/O 처리 시간이 증가하는 것으로 분석 할 수 있다. 이를 통해 제안하는 기법은 응용 단계뿐만 아니라 커널 영역의 I/O 응답 시간 모두 측정 및 가시화가 가능함을 입증하였다.

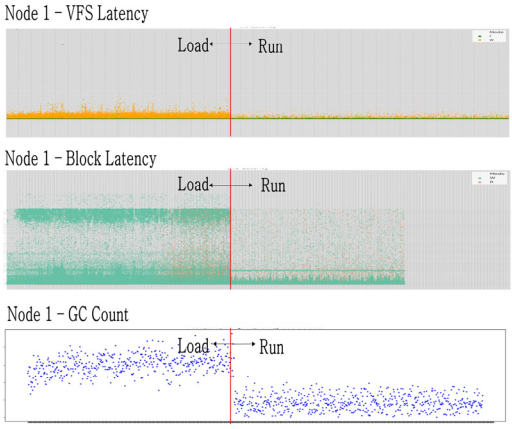


그림 8. 노드 1의 I/O Tracer 결과

V. 결 론

본 논문에서는 실제 환경과 유사한 ArangoDB 벤치마킹 시스템을 설계하고 구현하였다. 제안하는 시스템은 클러스터 환경에서 벤치마킹을 수행하기 위해서 기존 벤치마킹 도구인 YCSB를 일부 수정하였다. 제안하는 기법은 Block I/O를 추적하기 위한 I/O tracer를 제안하였다. 실험 결과 분석을 통해 I/O tracer 기반의 커널 계층 I/O 연산의 성능이 측정 가능함을 입증하였다. 추가로 본 논문에서는 실제 응용을 바탕으로 하는 실제 응용 워크로드들을 정의하였다. 정의한 워크로드 기반의 성능 평가를 통해 실제 환경과 유사한 환경에서의 벤치마킹이 가능함을 입증하였다. 성능평가 결과 세 가지 워크로드 유형 중 ArangoDB에서 활용 가능한 워크로드를 도출할 수 있었다. 또한, 응용 계층의 성능 결과를 통해 적재 단계와 실행 단계에서 특정 구간에 병목 현상이 발생하는 지점을 파악할 수 있었다. 커널 계층의 결과 분석을 통해서도 벤치마킹 수행 도중 생기는 I/O의 변화 패턴을 확인함으로써, 기존에는 응용 계층의 응답 시간만으로 분석하였던 정보들 이외에도 커널 계층의 평가 결과와의 연계를 통해 벤치마킹에 따른 성능 분석의 다양한 시각을 제공 할 수 있었다. 제안하는 기법의 실제 적용을 통해 ArangoDB로 이전 작업이 필요한 상황에서의 적용 가능성과 리스크 검토가 가능할 것으로 기대된다.

본 논문의 한계점은 다음과 같다. 먼저, YCSB에서의 가상 I/O 요청 연산은 멀티 스레드 형태로 생성되더라도, 단일 서버에서만 발생되기 때문에 실제 환경과 동일한 환경을 구현했다고는 볼 수 없다. 그리고, I/O 연산의 요청 형태가 단일 레코드, 특정 범위 레코드에 대한 읽기 쓰기 형태로 매우 단순하기 때문에 실제 환경에서의 질의 유형을 분석하고 구현해야 할 필요가 있다. 마지막으로, 응용 계층에서의 I/O 연산과 실제 Block I/O의 연관성을 보여주지 못하는 단점이 존재한다. 향후에는 실제 환경에서의 질의 유형 분석을 통해 실제 환경과 유사한 연산을 구현할 예정이다. 또한, YCSB 클러스터링을 통해 다중 클라이언트 환경을 고려한 벤치마킹 시스템을 설계할 예정이다.

참 고 문 헌

[1] https://www.arangodb.com/, 2021.05.01.

[2] C. F. ANDOR and B. PÂRV, "NoSQL Database Performance Benchmarking-A Case Study," Studia Informatica, Vol.63, No.1, pp.80-93, 2017.

[3] T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan, "Linkbench: A Database Benchmark based on The Facebook Social Graph," Proc. the 2013 ACM SIGMOD International Conference on Management of Data, pp.1185-1196, 2013.

[4] S. Ray, B. Simion, and A. D. Brown, "Jackpine: A Benchmark to Evaluate Spatial Database Performance," In 2011 IEEE 27th International Conference on Data Engineering, pp.1139-1150, 2011.

[5] G. Kang, D. Kong, L. Wang, and J. Zhan, "OStoreBench: Benchmarking Distributed Object Storage Systems Using Real-World Application Scenarios," In Benchmarking, Measuring, and Optimizing: Third BenchCouncil International Symposium Bench 2020, pp.90-105, 2020.

[6] L. Sfaxi, and M. M. B. Aissa, "Babel: A Generic Benchmarking Platform for Big Data Architectures," Big Data Research, Vol.24, 100186, 2021.

[7] S. Kashyap, S. Zamwar, T. Bhavsar, and S. Singh, "Benchmarking and Analysis of Nosql Technologiesm," Int J Emerg Technol Adv Eng, Vol.3, No.9, pp.422-426, 2013.

[8] D. Fernandes and J. Bernardino, "Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB," In DATA, pp.373-380, 2018.

[9] L. Meiling, "Benchmarking Multi-model Databases with ArangoDB and OrientDB," 2017.

[10] R. Gunawan, A. Rahmatulloh, and I. Darmawan, "Performance Evaluation of Query Response Time in The Document Stored NoSQL Database," In 2019 16th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering, pp.1-6, 2019.

[11] S. Chen, A. Ailamaki, M. Athanassoulis, P. B〉 Gibbons, R. Johnson, I. Pandis, and R. Stoica, "TPC-E vs. TPC-C: Characterizing the New TPC-E Benchmark via an I/O Comparison Study," ACM Sigmod Record, Vol.39, No.3, pp.5-10, 2011.

[12] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," Proc. the 1st ACM symposium on Cloud computing, pp.143-154, 2010.

[13] S. Patil, M. Polte, K. Ren, W. Tantisiriroj, L. Xiao, J. López, and B. Rinaldi, "YCSB++ Benchmarking and Performance Debugging Advanced Features in Scalable Table Stores," Proc. the 2nd ACM Symposium on Cloud Computing, pp.1-14, 2011.

[14] S. Friedrich, W. Wingerath, F. Gessert, N. Ritter, E. Pldereder, L. Grunske, and D. Ull, "NoSQL OLTP Benchmarking: A Survey," In GI-Jahrestagung, pp.693-704, 2014.

[15] O. Erling, A. Averbuch, J. Larriba-Pey, H.

Chafi, A. Gubichev, A. Prat, and P. Boncz, “The LDBC Social Network Benchmark: Interactive Workload,” Proc. the 2015 ACM SIGMOD International Conference on Management of Data, pp.619-630, 2015.

[16] A. D. Brunelle, and Alan D, “Block IO Layer Tracing: blktrace,” HP, Gelato-Cupertino, 2006.

[17] https://github.com/iovisor/bcc, 2021.05.01.

[18] https://medium.com/netflix-techblog/revisiting-1-million-writes-per-second-c191a84864cc, 2021.05.01.

[19] T. Harter, D. Borthakur, S. Dong, A. S. Aiyer, L. Tang, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, “Analysis of HDFS under HBase: A Facebook Messages Case Study,” Proc. USENIX conference on File and Storage Technologies, pp.199-212, 2014.

[20] Y. Chen, X. Qin, H. Bian, J. Chen, Z. Dong, X. Du, Y. Gao, D. Liu, J. Lu, and H. Zhang, “A Study of SQL-on-Hadoop Systems,” Proc. Workshops on Big Data Benchmarks, Performance Optimization, and Emerging, pp.154-166, 2014.

저 자 소 개

최 도 진(Do-Jin Choi) 정회원

- 2014년 2월 : 한국교통대학교 컴퓨터공학과(공학사)
- 2016년 2월 : 한국교통대학교 컴퓨터공학과(공학석사)
- 2020년 2월 : 충북대학교 정보통신공학과(공학박사)
- 2020년 3월 ~ 2020년 08월 : 충북대학교 정보통신공학과 박사후연구원 (Postdoc)

〈관심분야〉 : 연속 질의 처리, 그래프 스트림, 빅데이터

백 연 희(Yeon-Hee Baek) 준회원

- 2019년 2월 : 충북대학교 경영학부(복수전공은 빅데이터 연계전공)(학사)
- 2021년 2월 : 충북대학교 빅데이터 협동과정(석사)

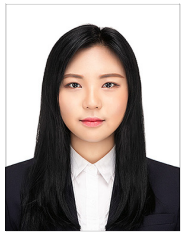
〈관심분야〉 : 소셜 네트워크, SIoV, 빅데이터 처리, 데이터베이스

이 소 민(So-Min Lee) 준회원

- 2019년 2월 : 충북대학교 정보통신공학부(공학사)
- 2021년 8월 : 충북대학교 정보통신공학과(석사)

〈관심분야〉 : 그래프 처리, 연속 질의 처리, 빅데이터, 기계학습

김 윤 아(Yun-A Kim) 준회원



- 2020년 2월 : 청주대학교 통계학과(이학사)
- 2020년 3월 ~ 현재 : 충북대학교 빅데이터협동과정(석사)

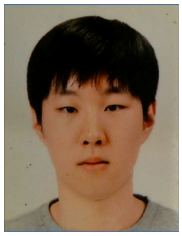
〈관심분야〉 : 소셜 네트워크, 기계학습, 빅데이터 처리, 데이터베이스

김 남 영(Nam-Young Kim) 준회원

- 2020년 2월 : 청주대학교 통계학과(이학사)
- 2020년 3월 ~ 현재 : 충북대학교 빅데이터 협동과정(석사)

〈관심분야〉 : 그래프 처리, 그래프 스트림, 빅데이터, 소셜 네트워크

최 재 용(Jae-Yong Choi) 준회원



- 2020년 2월 : 충북대학교 정보통신공학부(공학사)
- 2020년 3월 ~ 현재 : 충북대학교 정보통신공학과(석사)

〈관심분야〉 : 소셜 네트워크, 빅데이터 처리

이 현 병(Hyeon-Byeong Lee) 정회원

- 2016년 8월 : 한국교통대학교 컴퓨터공학과(공학사)
- 2018년 8월 : 한국교통대학교 컴퓨터공학과(공학석사)
- 2019년 3월 ~ 현재 : 충북대학교 정보통신공학과(박사과정)

〈관심분야〉 : 그래프 스트림, 빅데이터, 데이터베이스 시스템

임 종 태(Jong-Tea Lim) 정회원

- 2009년 2월 : 충북대학교 정보통신공학과(공학사)
- 2011년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2015년 8월 : 충북대학교 정보통신공학과(공학박사)
- 2015년 9월 ~ 2019년 8월 : 충북대학교 정보통신공학과 Postdoc.
- 2019년 10월 ~ 현재 : 충북대학교 전자정보대학 정보통신공학부 초빙 조교수

〈관심분야〉 : 소셜 미디어, 빅데이터, 시공간 데이터베이스, 위치기반 서비스 등

복 경 수(Kyoung-Soo Bok) 종신회원

- 1998년 2월 : 충북대학교 수학과(이학사)
- 2000년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2005년 8월 : 충북대학교 정보통신공학과(공학박사)
- 2005년 3월 ~ 2008년 2월 : 한국과학기술원 정보전자연구소 Postdoc
- 2008년 3월 ~ 2011년 2월 : 가인정보기술 연구소 차장
- 2011년 3월 ~ 2019년 8월 : 충북대학교 전자정보대학 정보통신공학부 초빙교수
- 2019년 9월 ~ 현재 : 원광대학교 SW 융합학과 조교수

〈관심분야〉 : 데이터베이스 시스템, 이동 객체 데이터베이스, 이동 P2P 네트워크, 소셜 네트워크 서비스, 빅데이터 처리 등

송 석 일 (Seok-Il Song) 종신회원

- 1998년 2월 : 충북대학교 정보통신공학과(공학사)
- 2000년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2003년 2월 : 충북대학교 정보통신공학과(공학박사)
- 2003년 7월 ~ 현재 : 한국교통대학교 컴퓨터공학과 교수

〈관심분야〉 : 데이터베이스, 센서 네트워크, 스토리지 시스템 등

유 재 수(Jae-Soo Yoo) 종신회원

- 1989년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 1991년 2월 : KAIST 전산학과(공학석사)
- 1995년 2월 : KAIST 전산학과(공학박사)
- 1995년 2월 ~ 1996년 8월 : 목포대학교 전산통계학과 전임강사
- 1996년 8월 ~ 현재 : 충북대학교 전자정보대학 정교수

〈관심분야〉 : 데이터베이스 시스템, 멀티미디어 데이터베이스, 센서 네트워크, 바이오 인포메틱스, 빅데이터 등