NoSQL 데이터베이스 최신 동향

권준호 (부산대학교)

목차

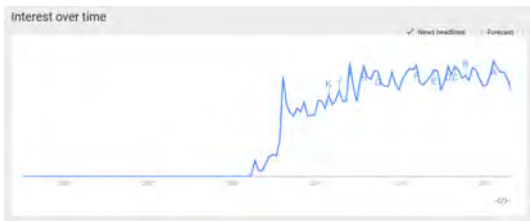
1. 서 론

전통적인 관계형 데이터베이스(RDBMS)는 다양한 응용에서 구조화된(structured) 데이터를 저장하기 위해 주로 사용되는 기술이다. RDBMS는 관계형 모델에 기반하여 SQL을 통한 임의 질의(ad-hoc) 질의 기능을 제공하기 때문에 많은 분야에서 채용이 되었다. 또한 일관성이 요구되는 환경에서 다수의 클라이언트들에 의해 접근 가능한 데이터 저장소로서 유일한 대안으로 여겨지고 있다[1].

빅 데이터와 사물 인터넷(Internet of Things) 기술의 분야의 발전으로 인해 다양한 형태의 데이터를 효율적으로 저장하고 처리하기 위한 새로운 데이터베이스 구조의 등장이 요구되었다. 전통적인 관계형 데이터베이스 시스템은(RDBMS)는 (그림 1)의 왼쪽과 같이 5단계의 추상화 레벨에 따르는 구조[2]를 가지고 있다. 구글의 맵리듀스[3,4]에 기반한 오픈소스인 하둡[5]의 데이터베이스 접근 구조는 (그림 1)의 오른쪽과 같이 3단계 추상화 레벨 구조를 따른다고 볼 수 있다.

새로운 구조를 가지는 NoSQL 데이터베이스는 빅 데이터, 클라우드, 분산/병렬 처리 기술의 발전에 따라 지난 몇 년간 지속적으로 발전하여 왔다. 원래 NoSQL이라는 단어는 1998년도에 SQL을 사용하지 않는 경량 데이터베이스 의미로 처음 등장하였고, 2009년도 오픈소스 분산데이터베이스의 논의 이벤트를 위한 이름으로 NoSQL이 지금과 비슷한 의미로 사용되었다[6]. 그 이후 (그림 2)와 같이 NoSQL 이라는 단어는 폭발적이고

(그림 1) RDBMS와 새로운 DB구조의 추상화 레벨 비교

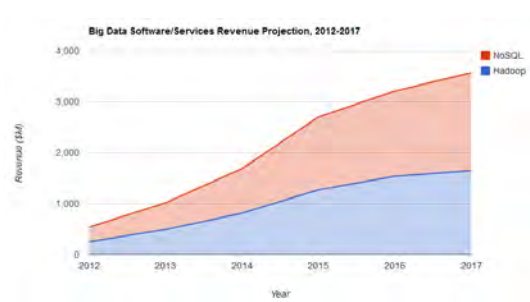


(그림 2) NoSQL 검색 빈도–Google Trends

지속적인 관심을 받고 있음을 알 수 있다.

Wikibon은 2013년도 전 세계 빅 데이터 시장의 규모를 (그림 3)과 같이 전망한 자료이다. 빅 데이터 관련 기술의 대표적인 기술로서 MapRedcue 프레임워크에 기반한 Hadoop과 NoSQL 데이터베이스 전체 시장 규모를 비교하고 있다. 비교 초창기에는 NoSQL 데이터베이스 관련 기술이 Hadoop 기술의 시장 규모가 비슷하지만, 시간이 지날수록 NoSQL 시장 규모가 빠르게 증가하여 Hadoop 기술의 시장 규모와의 격차가 증가함을 볼 수 있다.

(그림 4)는 NoSQL 데이터베이스를 활용할 수 있는 다양한 응용 분야와 NoSQL 데이터베이스의 시장 규모를 보여준다. NoSQL 데이터베이스가 다양한 응용 분야에 사용될 수 있음을 알 수 있다. 또한 2020년 NoSQL의 시장 규모는 3.4$로 예상이 되는데, 이는 2015년부터 2020년까지 6년 동안 연 평균 21퍼센트의 폭발적인 성장률을 의미한다.



(그림 3) NoSQL 데이터베이스 시장 규모 전망

(그림 4) NoSQL 데이터베이스 응용 분야

본 논문에서는 NoSQL 데이터베이스의 특성을 파악하고, 분류에 따른 최근의 인기 시스템들을 살펴 본다. 또한 성능 비교 결과 및 가이드라인을 소개하여 다양한 NoSQL 데이터베이스의 최근 발전 현황을 점검한다.

2. NoSQL 데이터베이스의 특성

NoSQL는 일반적으로 Not Only SQL의 약어로 알려져 있다. 하지만 일반적으로 통용되는 정의가 없으며 그 정의를 내리는 권위자도 존재하지 않는다. 따라서 NoSQL (데이터베이스)을 파악하고 이해하기 위해서는 아래와 같은 여러 시스템들의 공통적인 특성을 파악하는 것이 더 유용하다. NoSQL 데이터베이스들에서 발견할 수 있는 공통적인 특징은 다음과 같다[7].

- 관계형 모델을 사용하지 않음 (Not Using the relational model)
- 클러스터에서 잘 동작함 (Running well on clusters)
- 대부분 오픈 소스임 (Mostly open-source)
- 21세기 웹 환경을 위해 구축됨 (Built for the 21st century web estates)
- 스키마가 없음 (Schema-less)

NoSQL을 전통적인 관계형 데이터베이스가 해결할 수 없는 웹 기반 애플리케이션의 성능과 확장성 요구사항을 해결하기 위해 개발된 시스템들이기 때문에, 분산 처리 구조를 지니고 있다. 따라서 분산 시스템에 적용되는 CAP 이론에 대하여 이해하는 것도 필요하다.

CAP 이론은 Eric Brew가 주장한 것으로, 분산 시스템에서 보장해야 하는 측면을 다음과 같이 정의하고 있다[8].

- 일관성(consistency): 모든 클라이언트들은 항상 데이터들에 대한 같은 뷰를 가져야 한다. 즉 모든 노드들은 동시에 같은 데이터를 보아야 한다.
- 가용성(availability): 각각의 클라이언트들은 항상 읽고 쓸 수 있다. 모든 요청은 실패하든 성공하든 응답을 받는 것을 보장한다.
- 분할 용인(partition tolerance): 네트워크 실패로 인하여 임의의 분할이 발생해도 시스템은 계속적으로 동작해야 한다. 임의의 메시지들의 손실 또는 시스템의 부분 실패에도 불구하고, 시스템은 지속적으로 동작한다.

Brew의 CAP 이론은 분산 시스템은 consistency(C), availability(A) and partition toleration(P) 세 가지 특성들 중 2개만을 동시에 만족할 수 있다는 것을 의미한다[8]. 클러스터에서 잘 동작한다는 NoSQL의 공통 특징을 이해한다면, CAP 이론에서 분할 허용은 기본적으로 만족해야 하는 특성으로 생각할 수 있고, 따라서 NoSQL 시스템들은 일관성과 가용성을 절충해서 설계된 시스템이라고 판단할 수 있다[9,10].

전통적인 RDBMS는 일관성과 가용성 측면을 선택(C+A)하였고, 이를 트랜잭션(transaction) 개념을 사용하여 보장한다. 트랜잭션의 지녀야 하는 4가지 주요한 속성을 흔히 ACID 속성이라 한다. ACID는 원자성(Atomicity), 일관성(Consistency), 독립성(Isolation), 지속성(Durability)을 의미한다.

NoSQL은 CAP 이론의 분할 용인(P)를 기본적인 측면으로 취하기 때문에, A나 C를 선택하여 A+P나 C+P 전략에 기반을 둔다. NoSQL 시스템은 ACID 속성 대신에 BASE 속성을 따른다. BASE는 일관성보다 확장성을 우선시 하는 데이터베이스 디자인 철학으로, 기본적으로 확장적(Basically Available), 소프트 상태(Soft state), 결국 일관적(Eventually consistent)[11]을 의미한다. 기본적으로 확장적은 NoSQL 데이터베이스 시스템이 확장성을 보장한다는 것으로, 애플리케이션은 항상 동작을 해야 한다는 것을 의미한다. 소프트 상태는 애플리케이션들이 항상 일관적인 상태에 있을 필요가 없음을 의미한다. 점진적인 일관성은 리더(reader)들이 시간이 지남에 따라 쓰기 결과를 본다는 뜻이다. 안정 상태에서 시스템은 결국 마지막 쓰기 결과를 반환한다. 이는 클라이언트들은 변경이 진행중인 상황에서는 데이터의 불일치 상태를 겪을 수 있다

ACID와 BASE 속성을 정리하면 <표 1>과 같다[12].

많은 NoSQL 데이터베이스들은 개발자들이 그들의 필요에 맞게 시스템들 조정할 수 있도록 옵션들을 제공한다. 필수적인 세가지 변수는 R, W, N으로서 의미는 다음과 같다.

- R: 읽기 요청이 성공으로 여겨지기 전에 반드시 응답을 해야 하는 노드들의 수
- W: 쓰기 요청이 성공으로 여겨지기 전에 반드시 응답을 해야 하는 노드들의 수
- N: 데이터가 복제되어 저장되는 노드들의 수

이 세가지 변수들을 이용하여 시스템 설계자들은 분산 데이터를 위한 복제 전략으로 정족수(quorum) 시스템을 제안하였다. 데이터 저장소는

〈표 1〉 ACID와 BASE 속성 비교

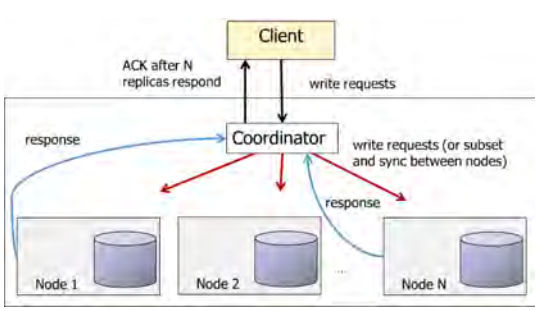
ACID		BASE	
원자성	전무 또는 전부 실행. 커밋과 롤백	낙관적 동작	데이터베이스의 일시적인 불일치를 허용
일관성	트랜잭션은 절대 불일치한 데이터를 보지 않음	기본적으로 가용적	복제를 통한 가용성
독립성	트랜잭션은 동시에 수행중인 트랜잭션을 의식하지 않음	소프트 상태	일관성을 제공하는 것인 사용자 프로그램의 역할임
지속성	승인된 트랜잭션은 모든 사건을 영구히 저장함	결국 일관성	약한 일치성. 결국에는 데이터베이스는 일관전인 상태가 됨

데이터 아이템을 복제자들의 집합(읽기 정족수라 부름)에게 보내어 저장을 한다. 읽기를 수행하기 위해서, 데이터 저장소는 다른 복제자들의 집합(읽기 정족수라 부름)으로부터 데이터 아이템을 가져 온다. 데이터 저장소는 복제자들이 반환한 값들의 집합을 비교하고, 버전들의 전 순서(total ordering)가 주어지면 가장 최근의 값을 반환한다.

(그림 5)는 Amazon Dynamo[13] 스타일의 정족수(quorum) 시스템의 제어 흐름을 보여 준다. N=3, W=2를 가정하고 있다. 클라이언트는 쓰기 요청을 중재자(coordinator) 노드에 보낸다. 중재자는 클라이언트의 요청을 모든 N 복제자(replicas)들에게 보낸다. 중재자가 복제자들로부터 W개의 ack을 받으면 쓰기를 반환한다.

정족수에 따른 구성과 그에 따른 일관성의 형태는 <표 2>와 같다.

페타(peta) 바이트나 엑사(exa) 바이트 이상의 빅 데이터를 다루게 된다면, 하나의 시스템의 용량을 초과하게 되므로, 필연적으로 데이터들을 분산하여 저장을 하게 된다. 분산 방법으로 복제(replication)와 샤딩(sharding)을 하나만 선택하거나 함께 적용할 수 있다. 복제는 같은 데이터를 여러 노드에 분산하여 복사하는 기법이고, 샤딩은 데이터를 중복되지 않게 여러 노드에 나누어서 저장하는 방법이다.



(그림 5) Dynamo Style 정족수에 기반한 쓰기[14]

많은 수의 NoSQL 데이터베이스는 샤딩(sharding)을 특징으로 내세우고 있다. 샤딩 기법을 사용할 때 중요한 문제중의 하나는 데이터 아이템을 어느 물리적인 노드에 배치하는 지를 결정하는 것이다. 가장 단순한 방법은 NodeID = hash(key) % TotalNodes 식을 사용하여 노드의 위치를 찾는 것이다. 새로운 노드의 추가나 기존 노드의 삭제가 야기하는 전체 데이터의 재정리(reshuffling)가 발생하는데, 하지만 나머지 연산은 이런 클러스터의 재조정을 효율적으로 지원할 수 없고 결과적으로 복제와 장애조치(failover)를 다루기 매우 어렵다.

이를 해결하기 위한 방법으로 NoSQL 데이터

〈표 2〉 정족수 구성과 일관성의 형태

R/W 구성	설명
W=N and R=1	읽기 최적화된 강한 일관성
W=1 and R=N	쓰기 최적화된 강한 일관성
W+R〈=N	결국 일관성. 읽기는 최근의 쓰기를 놓칠 수 있음.
W+R〉N	강한 일관성. 읽기는 적어도 하나의 가장 최근 읽기를 볼 수 있음

베이스들 중 일부는 consistent hashing을 사용한다. Consistent hashing[15]의 기본 아이디어는 데이터 아이템과 노드들이 같은 해시 함수를 사용한다는 것이다. 노드들에게 간격(interval)을 매핑하는데, 각 간격은 많은 수의 데이터 아이템 해시들을 저장한다. 노드가 클러스터에서 제거되면, 그 노드의 간격은 이웃한 간격을 지닌 노드가 가져간다. 다른 노드들에게는 아무런 변화가 발생하지 않는다.

NoSQL 데이터베이스는 고 확장성을 위하여 여러 개념과 기술에 기반을 하고 있기 다양한 시스템들이 시장에 출현하였다. 다음 장에서는 NoSQL 데이터베이스 시스템을 분류별로 정리를 한다.

3. NoSQL 데이터베이스 시스템

다양한 형태의 정형/비정형 데이터를 저장하기 위한 분산 구조 NoSQL 데이터베이스 기술이 등장하였다. NoSQL 데이터베이스 시스템은 스키마가 없으며(schema-free), 쉬운 복제를 지원하고, 간단한 API를 사용하며, 점진적인 일관성(eventually consistent)과 BASE 속성 등의 특성을 가진다.

NoSQL 데이터베이스를 데이터의 저장 방식에 따라 분류를 하면 (그림 6)과 같이 키-값 저장소, 넓은 컬럼 저장소(또는 컬럼 패밀리), 문서 저장소, 그래프 데이터베이스로 나눌 수 있다[16].1)

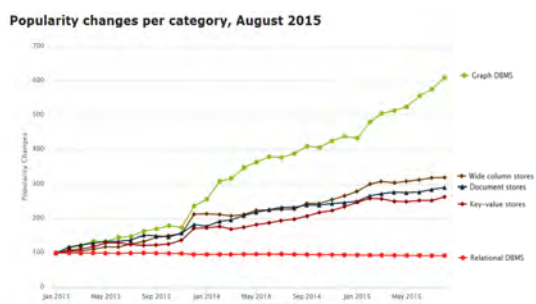
(그림 7)은 위 분류에 따른 분류별 인기도의 경향을 보여준다. 인기도는 매달 분류별 3개의 데이터베이스 시스템을 선택하여 그 랭킹의 평균을 계산하여 결정한 것이다. 모든 분류에 대하여 2013년 1월의 초기값은 100으로 설정하였다. 관

1) 각 NoSQL 데이터베이스는 제품명으로 쉽게 검색이 가능하므로, 이 논문에서는 참고 문헌으로 제시하지 않는다.

(그림 6) 분류별 NoSQL 데이터베이스

계형 데이터베이스는(RDBMS)는 지속적으로 100 근처의 값을 가지는데, 이는 인기도 증감이 크게 나타나지 않는 것을 의미한다. 이에 비하여 NoSQL 데이터베이스의 인기도는 지속적으로 상승하고 있음을 알 수 있다. NoSQL 데이터베이스 내에서는 그래프 데이터베이스, 컬럼 저장소, 키-값 저장소, 문서 저장소의 순으로 인기도의 상승을 확인할 수 있다.

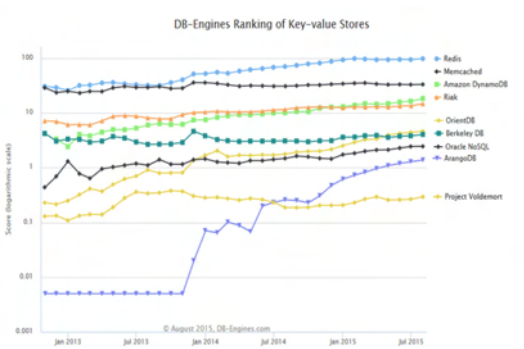
Sugam et.al[18]의 논문에서 설명한 방법을 따라서, 각 분류 별로 간단한 설명을 제시하고 그 분류에서 2015년 8월 현재 인기 있는 시스템을 인기도에 기반하여 그래프로 설명한다.



(그림 7) DB 분류별 인기도 (2015.8월)[17]

3.1 키-값 저장소(key-value stores):

가장 기본적인 저장 형식으로, 대부분의 NoSQL이 지원하는 저장 개념인 키/값(Key/Value) 쌍을 지원한다. 이 저장 모델은 유일한 키(key)에 하나의

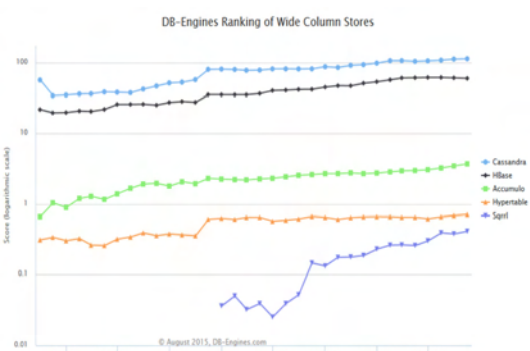


(그림 8) 인기있는 키-값 저장소 시스템[17]

값(value)을 저장하고 있는 형태이다. Memchahed, Redis, 아마존의 Dynamo와 그의 오픈소스 버전인 볼드모트(Voldemort), 오라클의 버클리 DB등의 솔루션이 있다.

3.2 넓은 컬럼 저장소(Wide Column stores)

넓은 컬럼 저장소(Column stores): 키/밸류 저장소의 확장 개념으로, 밸류에 컬럼 패밀리 형태를 사용한다. 넓은 컬럼 또는 컬럼 패밀리는 열들을 구조화 한 것으로, 자주 함께 접근되는 컬럼들을 하나로 값으로 저장을 하는 것이다. Cassandra, Hbase, Hypertable, Accumulo 등의 오픈소스가 존재한다.



(그림 9) 인기 있는 넓은 컬럼 저장소 시스템[17]

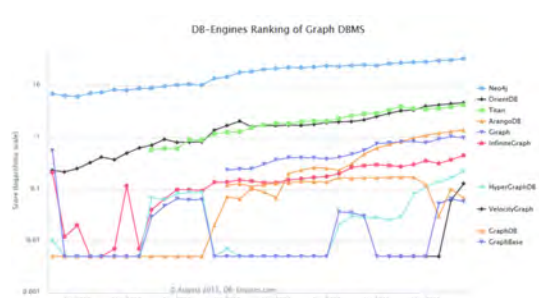
3.3 문서 저장소 (Document Stores):

키/밸류 저장소의 확장된 형태로, 하나의 키에 하나의 값을 저장하나, 값의 형태로 문서(document) 타입을 사용한다. 문서 형태로 XML과 JSON 형태의 문서를 사용하여 복잡한 계층 구조를 표현한다. MongoDB, CouchDB, Couchbase, Amazon DyanmoDB, OrientDB 등의 솔루션이 있다.

(그림 10) 인기있는 문서 저장소 시스템[17]

3.4 그래프 데이터베이스 (Graph Database)

그래프 형태의 데이터를 저장하기 위한 전용의 저장소이다. 테이블간의 조인 연산이 아니라 그래프에서의 경로 탐색 형태의 연산이 많이 사용되는 소셜 네트워크 분석 같은 그래프 응용 프로그램에 주로 사용한다. Neo4j, InfiniteGraph,



(그림 11) 인기있는 그래프 데이터베이스[17]

Titan, ArangoDB, Trinity등의 솔루션이 있다.

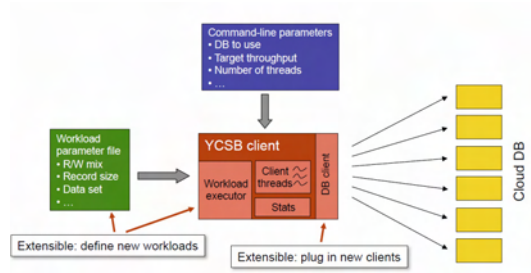
4. NoSQL 데이터베이스 성능 비교 동향

NoSQL 웹 사이트[16]에 따르면 2015년 8월 현재 150개 이상의 NoSQL 데이터베이스 시스템이 존재한다. 다양한 종류의 NoSQL 데이터베이스 시스템들의 존재는 오히려 연구자들이 어떤 NoSQL 시스템이 더 적합한지 결정하는 것을 어렵게 만든다. 이에 따라 NoSQL 데이터베이스의 성능 비교를 통한 수행한 연구들이 다 수 존재한다. 이장에서는 우선 공정한 성능 비교를 위한 벤치마크를 소개라고, 벤치마크를 이용한 비교 결과들을 제시한다. NoSQL 데이터베이스를 위한 정성적인 가이드라인도 언급한다.

4.1 야후 클라우드 서빙 벤치마크 (YCSB, Yahoo! Cloud Serving Benchmark)

YCSB는 공통적인 워크로드를 이용하여 다른 시스템을 비교하기 위하여 성능(performance)과 확장성(scalability)에 초점을 둔 벤치마크 도구이다. YCSB 프레임워크의 구조는 (그림 12)와 같다. 프레임워크는 워크로드를 생성하는 클라이언트와 성능 공간(읽기-집중 워크로드, 쓰기-집중 워크로드, 스캔 워크로드 등)의 흥미로운 부분을 담당하는 표준 워크로드 패키지로 구성되어 있다. YCSB의 큰 특징중의 하나는 새로운 유형의 워크로드를 정의하여 생성하기 쉬우며, 새로운 NoSQL 데이터베이스를 사용하기도 쉽다는 것이다. 워크로드는 중 인기 있는 테스트 패키지는 “50/50 읽기/업데이트”와 “95/5 읽기/업데이트” 케이스가 있다.

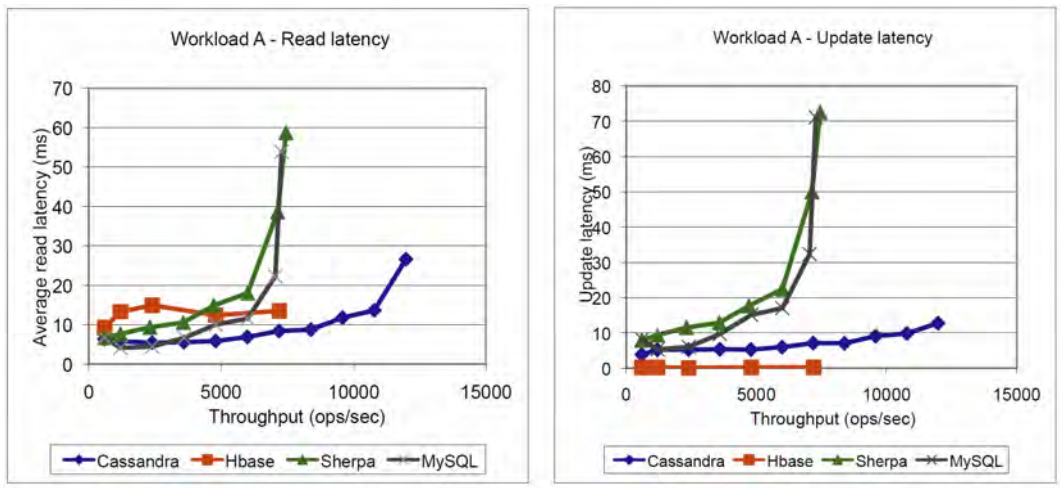
50/50 읽기/업데이트 케이스는 업데이트 중심 테스트 케이스이며, (그림 13)은 YCSB를 사용한



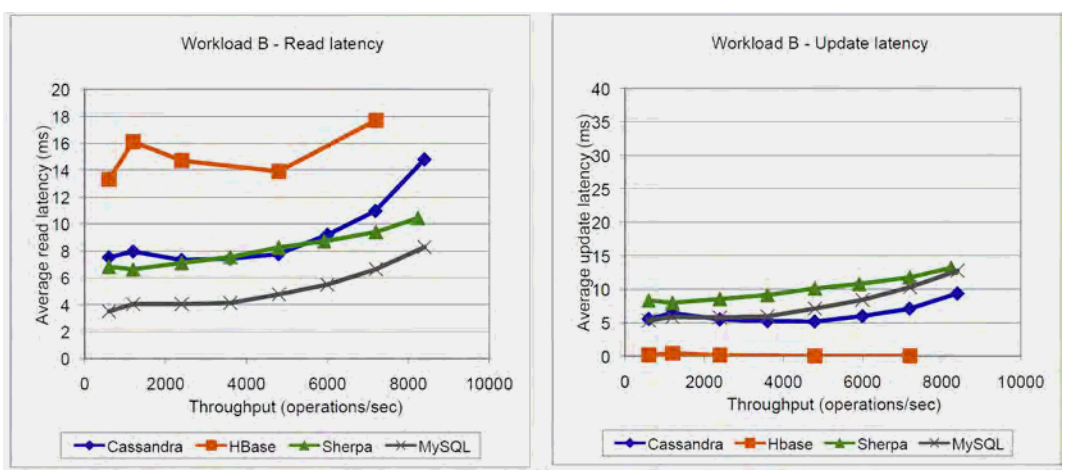
(그림 12) YCSB 클라이언트 아키텍처[19]

테스트 결과를 보여준다. 카산드라(Cassandra)는 쓰기에 최적화 되어 있으며, 읽기/쓰기 모두 높은 생산성과 낮은 지연율 가지고 있음을 알 수 있다. Sherpa와 MySQL은 상대적으로 비슷한 성능을 지니고 있는 것으로 나타났다. MySQL은 NoSQL 데이터베이스가 아님에도 비교를 하고 있는데, 읽기 및 쓰기 지연 시간이 4,000개 정도의 작업까지는 다른 NoSQL 데이터베이스와 비슷한 수준을 유지하지만 초당 작업이 5,000개를 넘어서는 순간 빠르게 늘어나서 성능이 저하됨을 볼 수 있다. Hbase는 읽기 시에는 레코드를 재구성해야 하기 때문에 높은 지연율을 보이나, 쓰기 시에는 제일 낮은 지연율을 보이고 있다.

95/5 읽기/업데이트 케이스는 읽기 중심의 테스트 케이스로서, (그림 14)는 실험 결과를 보여준다. 이 테스트 케이스는 정렬 칼럼 패밀리 저장소가 연속적인 범위 읽기에 있어서 가장 성능이 좋다고 주장한 이 장의 몇 가지 이론을 확인시켜준다. HBase는 초당 작업 개수와 상관없이 읽기에 대해 일관된 성능을 제공하고 있고, 업데이트는 경우 거의 지연 시간이 없다. MySQL은 읽기 전용 케이스에서 가장 좋은 성능을 전달한다. 이는 데이터가 캐시로부터 반환되기 때문이다. HBase를 Memcached나 멤베이스 같은 분산 캐시와 결합하면 MySQL의 읽기 성능에 견줄 만한 성능이 나오고, 작업량이 증가함에 따라 확장성



(그림 13) 갱심 중심 워크로드 비교

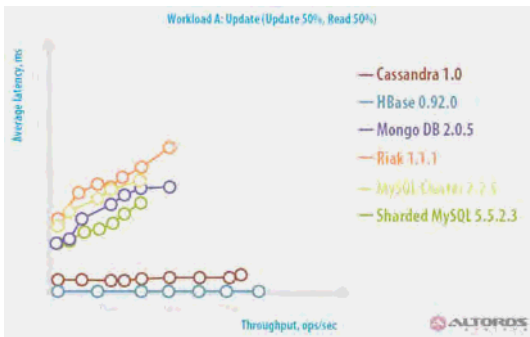


(그림 14) 읽기 중심 워크로드 비교

도 개선할 수 있다. 카산드라는 읽기의 경우 상당한 성능을 보여주고, 테스트에서 HBase를 능가했지만 쓰기의 경우 지연율이 발생한다는 것을 확인할 수 있다. Sherpa도 낮은 읽기 지연율과 높은 생산성을 보여 주고 잇다.

4.2 NoSQL 데이터베이스 성능 비교

Altoros사의 Sergey는 아마존 웹 서비스의 가상 머신 위에서 YCSB를 이용하여 NoSQL 데이터베이스에 대한 성능을 비교하였다[20]. YCSB 논문에서는 고성능의 하드웨어가 사용되었다는 점과 카산드라, HBase, PNUTS (Sherpa로 이름 변경), MySQL만을 비교하였다는 한계가 있다. Sergey는 평균적인 하드웨어서 유명한 NoSQL 데이터베이스의 성능을 확인하였다. 읽기 중심 테스트에서 (그림 15)와 같이 Cassandra와 HBase의 성능이 다른 NoSQL 데이터베이스 보다 우수



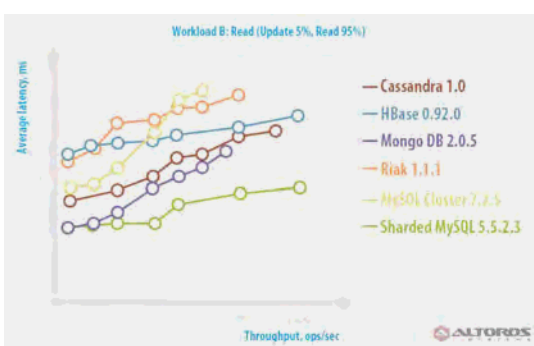
(그림 15) Update 중심 워크로드 비교

함을 알 수 있다.

일기 중심 테스트의 결과는 (그림 16)과 같다. Sharded MySQL이 제일 좋은 성능을 보였고, MongoDB가 Sharded MySQL에 근접하는 성능을 보였다. Cassandra도 Hbase나 Riak보다 더 빠른 성능을 보임을 알 수 있다.

Sergey 논문은 이 외에도 더 많은 결과들을 포함하고 있다. 이 벤치마크의 결론은 완벽한 NoSQL 데이터베이스는 없으며, 장단점이 존재하므로 개발자의 익숙함과 수행해야 할 작업의 유형에 따라 NoSQL 데이터베이스를 결정할 것을 권하고 있다.

EndPoint는 아마존 웹 서비스에서 EC2 인스턴스를 이용하여 상위 4개의 NoSQL 데이터베이스에 대하여 성능 비교를 수행하였다[21]. 선택된

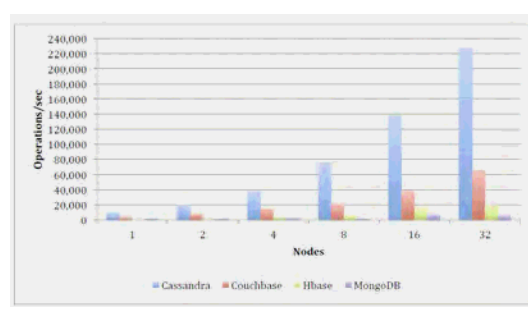


(그림 16) 읽기 중심 워크로드 비교

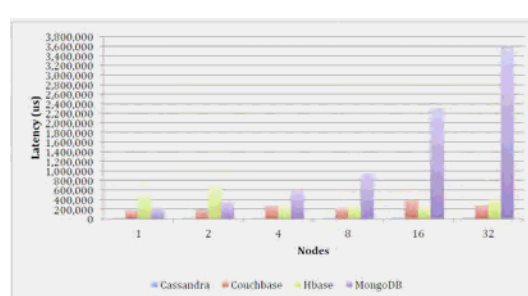
NoSQL 데이터베이스는 Cassandra, Couchbase, Hbase와 MongoDB이다. (그림 17)과 (그림 18)는 실험 결과의 샘플을 보여준다.

EndPoint의 실험에서는 Cassandra가 다른 NoSQL 데이터베이스들 보다 항상 좋은 성능을 보였다. CouchBase와 Hbase는 비슷한 성능을 보이는데 케이스에 따라 성능 순서가 바뀐다. MongoDB는 항상 제일 나쁜 성능을 보인다.

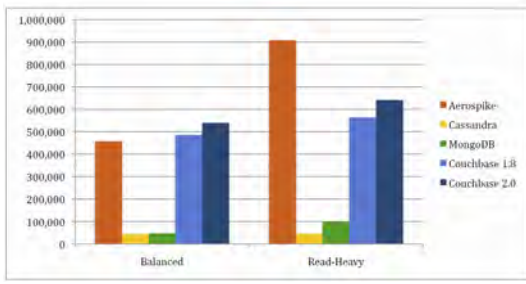
Thumbtack[22]은 Cassandra, Couchbase, MongoDB, Aerospike (SSD에 최적화된 NoSQL 데이터베이스)를 사용한 성능 평가 비교를 수행하였다. 앞의 세 성능 비교에서는 구체적인 사용 예(use case)에 집중하지 않고 범용적인 면에 집중을 하고 있고, 하드 디스크를 사용한 비교를 수행하였다는 것이 문제 상황이라 판단하였다. 그리하여 키-값 저장에 최적화된 형태의 데이터를 가정하고, 다



(그림 17) 읽기 중심 워크로드에서 생산성(throughput)



(그림 18) 균형있는 읽기/쓰기 혼합 워크로드에서 지연(latency)



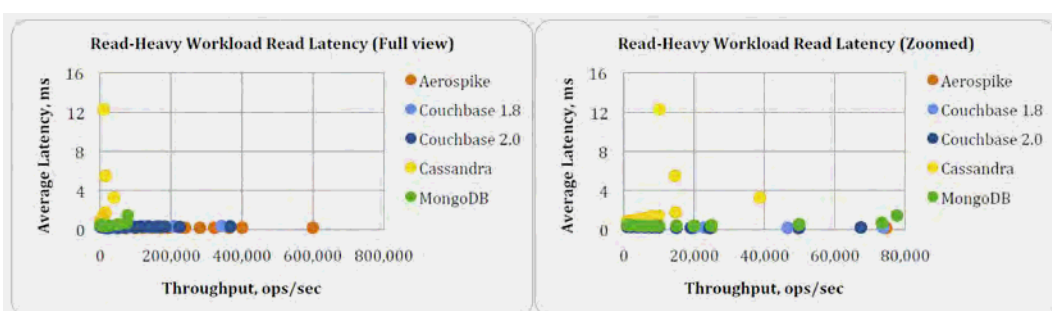
(그림 19) 최대 생산성– 인 메모리 데이터 셋

수의 클라이언트 환경 실험 시에 나타난 문제점을 개선한 YCSB 도구를 사용하여 성능 평가를 실시하였다.

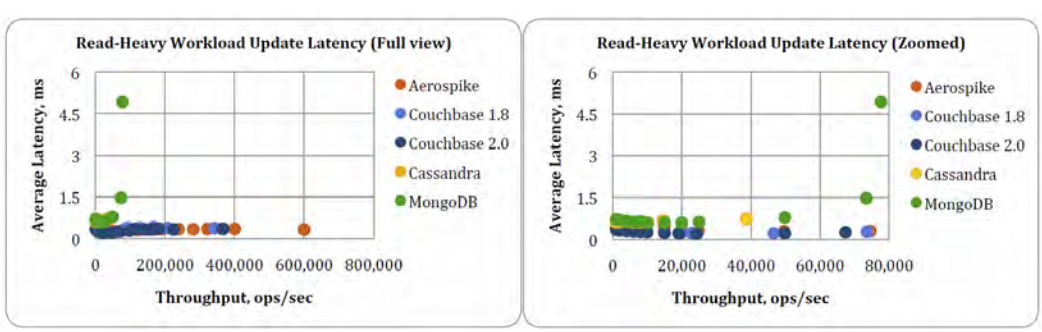
(그림 19)는 모든 데이터들이 메모리에 상주 가능할 때의 최대 성능을 측정한 결과이다. 균형 잡힌 경우 Couchbase가, 읽기 중심인 경우 Aerospike가 가장 높은 생산성을 보여 주었다. Cassandra와 MongoDB는 매우 낮은 생산성 결과를 보인다.

(그림 20)과 (그림 21)은 워크 로드에 따른 읽기 지연 결과를 보여준다. Aerospike와 Couchbases는 높은 생산성을 보일 때도 낮은 지연을 보인다. MongoDB의 경우는 쓰기 경우에는 지연이 나빠지지만 읽는 경우에는 일관성 있는 결과를 보여준다. Cassandra는 쓰기에 최적화되어 있기 때문에, 경우는 읽기 경우에는 지연이 나빠지지만 쓰는 경우에는 일관성 있는 결과를 보여준다.

모든 데이터들이 메모리에 상주하는 경우 Aerospike와 Couchbase가 좋은 성능을 보이고, Cassandra와 Mongdb는 성능이 떨어짐을 알 수 있다. 하지만 이 실험은 오직 키-값 상황에 적합한 데이터를 이용하여 수행을 하였기에 키-값 저장소의 성능이 더 좋은 결과가 나왔다. Cassandra와 MongoDB는 키-값 저장소 역할 시에는 성능



(그림 20) 읽기 중심 워크로드의 읽기 지연성 – 인 메모리 데이터



(그림 21) 읽기 중심 워크로드의 쓰기 지연성 – 인 메모리 데이터

이 느리지만, 훨씬 다양한 기능을 제공한다.

4.3 그래프 데이터베이스 성능 비교

그래프 데이터베이스는 연결된 데이터라는 저장 특성 때문에 다른 분류의 NoSQL 데이터베이스와의 직접적인 비교가 어렵다. 그래프 데이터베이스를 관계형 데이터베이스와 비교[23,24]하거나 그래프 데이터베이스들 간의 비교를 수행한 연구[25]가 존재한다.

Chad et.al.[23]의 연구 결과에 따르면, 그래프 구조에서 깊이가 4이상의 되는 탐색을 포함하는 질의의 경우 그래프 데이터베이스인 Neo4j가 좋은 성능을 보이고, 문자열이 포함되어 풀 텍스트 검색이 필요한 경우에도 Neo4j가 빠른 수행 시간을 보인다.

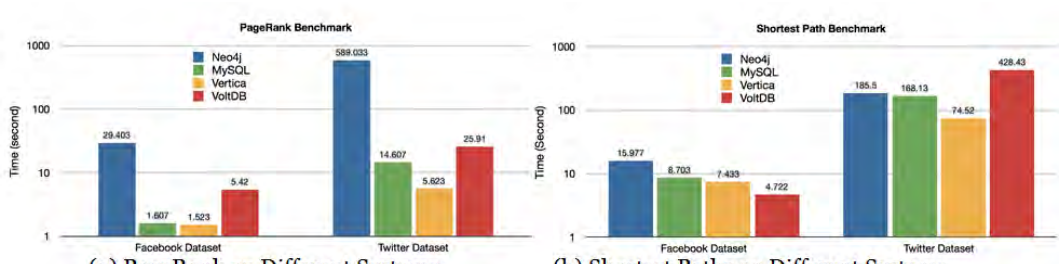
실제 그래프 데이터를 이용한 연구[24]에서는 위의 논문과 반대의 결과를 보인다. (그림 22)는 페이스북과 트위터의 데이터를 이용한 그래프 데이터베이스와 관계형 데이터베이스와의 비교 결과를 보여준다. PageRank의 경우 RDBMS인 MySQL의 성능이 월등히 좋음을 알 수 있다. PageRank의 경우 노드와 간선 테이블에 대한 풀 스캐닝과 조인을 필요로 하는데, 이는 관계형 데이터베이스가 매우 효율적으로 잘 수행하는 연산이기 때문이다.

최단 거리를 찾는 연산은 소스 노드로부터 시작하여 연속적으로 진출 간선을 탐색하는 과정을 수행한다. 이는 PageRank 연산과 달리 그래프 구조의 탐색 연산을 필요로 한다. 하지만, 여전히 관계형 데이터베이스인 MySQL이 그래프 데이터베이스인 Neo4j 보다 약간 더 좋은 성능을 보이고 있다.

4.4 NoSQL 데이터베이스 선택을 위한 가이드라인

앞에서 언급한 성능 비교 결과를 바탕으로 NoSQL 데이터베이스를 선택할 수도 있지만, 시스템 요구 사항에 맞추어서 적합한 NoSQL 데이터베이스를 선택할 수 있다. 다음은 NoSQL 데이터베이스 선택을 위한 일반적인 가이드라인[7]이다.

- 키-값 저장소는 세션 정보, 사용자 프로파일, 선호도, 쇼핑 카트 데이터를 저장하기에 일반적으로 유용하다. 데이터의 위한 질의가 필요하거나, 저장되는 데이터들이 관계를 가지고 있거나, 동시에 여러 키들을 다루어야 하는 경우 키-값 저장소를 사용하지 말아야 한다.
- 문서 저장소는 일반적으로 내용 관리 시스템, 블로깅 플랫폼, 웹 분석, 실시간 분석, e커머스 응용에 적합하다. 많은 연산들에 걸치는 복잡한 트랜잭션이나 다양한 집계 구조에 반하는



(그림 22) 그래프 데이터베이스 성능 비교[24]

질의를 필요로 하는 시스템의 경우 문서 저장소 사용을 피해야 한다.

- 컬럼 패밀리 데이터베이스는 내용 관리 시스템, 블로깅 플랫폼, 카운터의 유지, 사용의 만료, 로그 집계 같은 과도한 쓰기 볼륨에 유용하다. 초기 개발 단계이거나 질의 패턴이 변화는 시스템에서는 사용하지 않는 것이 좋다.
- 그래프 데이터베이스는 소셜 네트워크, 공간 데이터, 물건이나 돈을 위한 라우팅 정보, 추천 엔진과 같이 연결된 데이터가 필요한 문제들에 매우 적합하다.

5. 결 론

이 논문은 NoSQL 데이터베이스의 개념과 특징들에 대하여 기술하고, 저장 기법에 따라 네 가지 카테고리로 분류하였다. 해당 카테고리에 속하는 NoSQL 데이터베이스에서 최신의 인기 흐름을 파악하였으며, 성능 비교를 하는 연구들의 결과들을 소개하였다. 또한 개발자나 일반 사용자들의 편의를 위하여 NoSQL 데이터베이스 선택을 위한 가이드라인을 간단하게 제시하였다.

참 고 문 헌

[1] Christof Strauch, “NoSQL Databases,” http://www.christof-strauch.de/nosqldbs.pdf, 2011

[2] Härder, Theo. "DBMS Architecture - Still an Open Problem," In proceedings of Datenbanksystem in Business, Technologie und Web (BTW 2005), LNI P-65, pp 2-28, 2005.

[3] Jeffrey Dean, Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” In proceedings of the 6th Symposium on Operating Systems Design & Implementation (OSDI), pp. 137-150, Sanfrancisco, CA, USA, December 2004.

[4] Jeffrey Dean, Sanjay Ghemawat, “MapReduce: a flexible data processing tool,” Communication of ACM, Vol(53), No(1), pp. 72-77, 2010.

[5] Apache Hadoop, https://hadoop.apache.org/

[6] Pramod J. Sadalage, Martin Fowler, "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence," Addison-Wesley Professional, 2012

[7] Pramod J. Sadalage, “NoSQL Databases: An Overview,” http://www.thoughtworks.com/insights/blog/nosql-databases-overview, 2014. 10

[8] Eric Brewer, "Towards Robust Distributed Systems," Proc. 19th Ann. ACM Symp. Principles of Distributed Computing (PODC 00), ACM, 2000, pp. 7-10;

[9] Eric Brewer, "CAP Twelve Years Later: How the "Rules" Have Changed", Computer, vol.45, no. 2, pp. 23-29, Feb. 2012

[10] 이정행, “CAP Theorem, 오해와 진실”, http://eincs.com/2013/07/misleading-and-truth-of-cap-theorem/

[11] Werner Vogels, "Eventually Consistent," ACM Queue vol. 6, no.6, pp.14-19, 2008

[12] Deka Ganesh Chandram, "BASE analysis of NoSQL database," Future Generation Computer Systems, Vol(52), pp.13-21, 2015

[13] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: Amazon’s highly available key-value store,” In Proceedings of SOSP, pp. 205–220, 2007

[14] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, Ion Stoica , “Probabilistically Bounded Staleness for Practical Partial Quorums,” PVLDB 5(8), pp. 776787, 2012

[15] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, “Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web,” In Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing (STOC), pp. 654-663, 1997
[16] NoSQL, http://www.nosql-database.org/
[17] DB-Engines Ranking, http://db-engines.com/en/ranking
[18] Sugam Sharma, U S Tim, Shashi Gadia, Ritu Shandilya, and P Sateesh, “Classification and Comparison of Leading NoSQL Big Data Models,” International Journal of Big Data Intelligence (IJBDI), Vol. 2, No. 3, pp 201-221, 2015
[19] Brian F. Cooper,Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears, “Benchmarking cloud serving systems with YCSB”, Proceedings of the 1st ACM symposium on Cloud computing, pp. 143-154, 2010
[20] Sergey Bushik, "A Vendor-independent Comparison of NoSQL Databases: Cassandra, HBase, MongoDB, Riak", Altoros Systems, Inc., 2012
[21] EndPoint, “Benchmarking Top NoSQL Database,” http://www.datastax.com/wp-content/themes/datastax-2014-08/files/NoSQL_Benchmarks_EndPoint.pdf
[22] Denis Nelubin,Ben Engber, "Ultra High Performance NoSQL Benchmarking," Thumbtack Technology, 2013
[23] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, "A comparison of a graph database and a relational database: a data provenance perspective," In Proceedings of the 48th Annual Southeast Regional Conference, pp 1-6 2010.
[24] Y. Guo, M. Biczak, A. L. Varbanescu, A. Iosup, C. Martella, and T. L. Willke. How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis, In the IEEE International Parallel and Distributed Processing Symposium (IPDPS 2014). Phoenix, AZ, USA, 19-23 May 2014.
[25] Alekh Jindal,Benchmarking Graph Databases, http://istc-bigdata.org/index.php/benchmarking-graph-databases/

저 자 약 력

권 준 호

이메일 : jhkwon@pusan.ac.kr

- 2009년 서울대학교 전기컴퓨공학부 박사
- 2009년~2010년 차세대융합기술연구원 선임연구원
- 2010년~현재 부산대학교 빅데이터 협동과정 교수
- 관심분야 : 빅 데이터 처리 및 분석, 그래프 데이터베이스, XML 문서 필터링 및 인덱싱, IoT 데이터 저장 및 관리