NoSQL

주차	15주차
스터디 일자	@2024/01/25

NoSQL

> *비관계형 데이터베이스 유형*

- 관계형 DB의 한계를 벗어나 빅데이터 처리를 위해 데이터의 읽기보다 쓰기에 중점을 둔,
- 수평적 확장이 가능하며 다수 서버들에 데이터 복제 및 분산저장이 가능한 DBMS (Not Only SQL)

💡 RDB는 SQL을 사용하여 데이터를 조작하는 데이터베이스 시스템이고,
NoSQL은 SQL을 사용하지 않는 데이터베이스 시스템이라,
RDB(SQL) | NoSQL 로 생각하면 된다.

특징

- RDB 형태의 관계형 데이터베이스가 아닌 **다른 형태의 데이터 저장 기술**을 의미
- RDBMS와는 달리 **테이블 간 관계를 정의 X**
- 데이터 테이블은 그냥 하나의 테이블이며 테이블 간의 관계를 정의하지 않아 **일반적으로 테이블 간 Join도 불가능**

> **등장 배경**
>
> 점점 빅데이터의 등장으로 인해 데이터와 트래픽이 기하급수적으로 증가함에 따라 RDBMS의 단점인 성능을 향상시키기 위해서는 장비가 좋아야 하는 **Scale-Up**의 특징이 비용을 기하급수적으로 증가시키기 때문에 데이터 일관성은 포기하되 비용을 고려하여 여러 대의 데이터에 분산하여 저장하는 **Scale-Out**을 목표로 등장

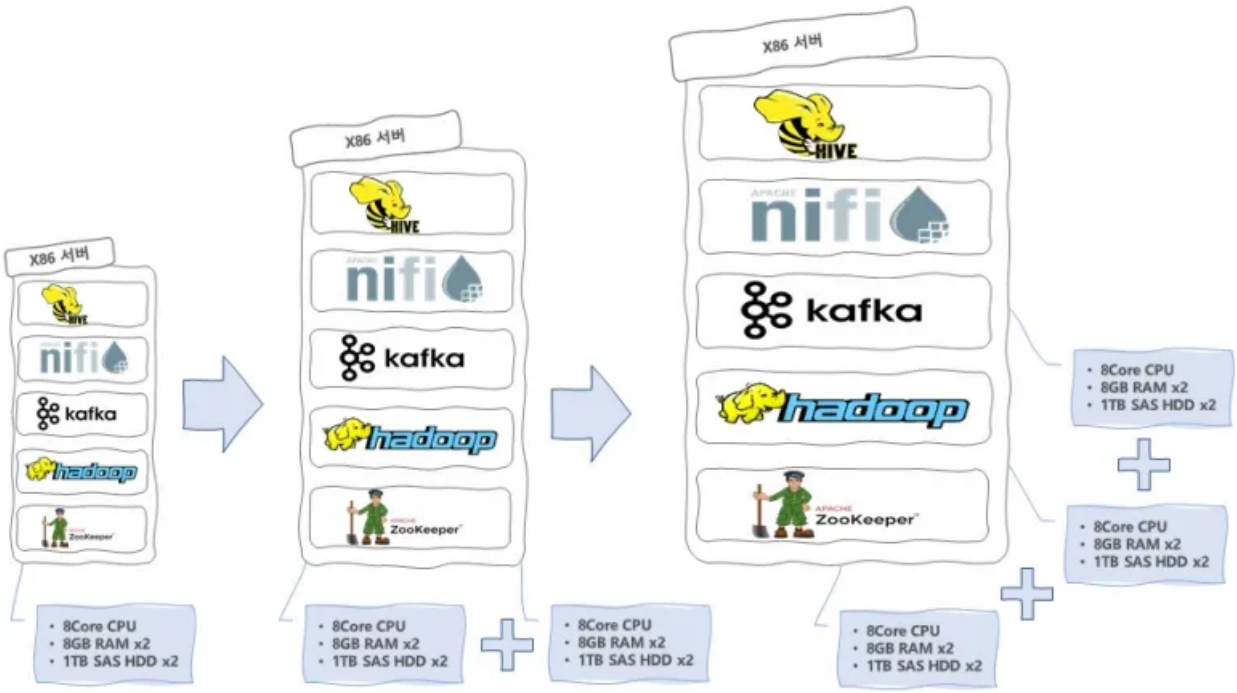
Scale-Up | Scale-Out 뭔데 그래?

Scale-Up과 **Scale-Out**은 인프라 확장을 위한 두 가지 방법

서버를 운영하다 보면 이용자가 증가하거나 사업을 확장할 때 많은 서버 용량과 성능이 필요하게 됨.

→ **Scale-Up**과 **Scale-Out**으로 인프라 확장 문제를 해결 가능!

Scale-Up (수직 확장)



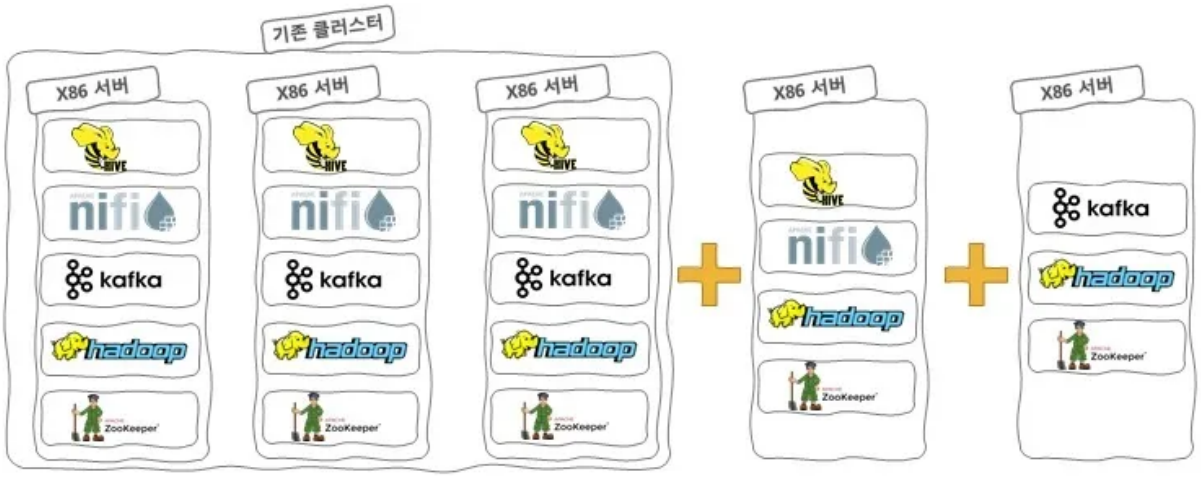
기존 서버의 **사양을 업그레이드해** 시스템을 확장하는 것

CPU나 RAM 등을 추가하거나 고성능의 부품, 서버로 교환하는 방법

⇒ 하나의 서버의 사양을 업그레이드 하기 때문에 수직 스케일로 불리기도 함.

☝ Scale-Up의 소프트웨어적인 예로는 AWS의 EC2 인스턴스 사양을 micro → small, small → medium 등으로 높이는 것으로 생각하면 된다 !

Scale-Out (수평 확장)



스케일 아웃은 **서버를 여러 대 추가하여** 시스템을 확장하는 것

서버가 여러 대로 나뉘기 때문에 각 서버에 걸리는 부하를 균등하게 해주는 **로드밸런싱**이 필수적으로 동반되어야 함.

⇒ 여러 대의 서버로 나눠 시스템을 확장하기 때문에 수평 스케일로 불리기도 함.

그럼 무조건 돈 적게 드는 Scale-Out을 하면 되잖아?

SCALE UP (스케일업)	VS.	SCALE OUT (스케일아웃)
	구성	
CPU 변경, RAM 추가 등으로 하드웨어 장비의 성능을 높임 수직 확장이며, 성능 확장에 한계가 있음	확장성	하나의 장비에서 처리하던 일을 여러 장비에 나눠서 처리함 수평 확장이며, 지속적 확장 가능
성능 증가에 따른 비용 증가폭이 큼	비용	비교적 저렴한 서버 사용으로 비용 부담이 적음
한 대의 서버에 부하가 집중되어 장애 영향도가 큼	장애	읽기/쓰기가 여러 대의 서버에 분산 처리 장애 시 전면 장애의 가능성이 적음

장단점

Scale-Up

- **장점**
  - 스케일 업 아키텍처에서는 추가적인 네트워크 연결 없이 용량을 증강 가능
  - 스케일 아웃보다 관리 비용이나 운영 이슈가 적고, 사양만 올리면 되는 것이기 때문에 비교적 쉬움.
- **단점**
  - 성능 향상에 한계가 있으며 성능 향상에 따른 비용부담이 큼.
  - 서버 한 대가 부담하는 양이 많아서 자연재해 등의 다양한 이유로 서버에 문제가 생기면 큰 타격

Scale-Out

- **장점**
  - 확장의 유연성 ⭐
    - 서버를 운영하면서는 향후 확장 가능성에 대비해 서버를 **현재 필요한 만큼보다 더 많은 용량이나 성능을 확보**해 놓는 경우가 많음.
    - **But,** 이런 경우 예상했던 정도와 요구되는 정도가 다르거나, 확장의 필요성이 없어졌을 경우 추가로 **확보해놓은 만큼의 손해가 발생**하게 됨.
    - Scale-Out 방식으로 시스템을 구축한 상황에서는 서버를 필요한 만큼만 도입해 놓고, 장기적인 용량 증가 추이를 예측할 필요 없이 그때 그때 필요한 만큼 서버를 추가해 용량과 성능을 확장(pay-as-you-grow)할 수 있게 된다.
- **단점**
  - 여러 노드를 연결해 병렬 컴퓨팅 환경을 구성하고 유지하려면 아키텍처에 대한 **높은 이해도 요구**
  - 서버의 수가 늘어날수록 관리가 힘들어지는 부분이 있음

- 여러 노드에 부하를 균등하게 분산시키기 위해 로드 밸런싱이 필요하고, 노드를 확장할 수록 **문제 발생의 잠재 원인** 또한 추가한 만큼 늘어나게 됨.

오케이 이제 알겠어. 근데 RDB로 Scale-Out을 하면 안돼?

(왜 굳이 NoSQL을 만들어서 씀?)

RDB의 수평 확장

> 예를 들어 위의 경우 서버를 2개를 사용하여 수평확장을 한다면
> A서버: 1, 2
> B서버: 3, 4
> 이렇게 데이터를 나누어 저장했다.

여기에 **테이블**과 **서버**를 추가한다면

커뮤니티 게시글

ID	제목	내용
1	~	~
2	~	~
3	~	~
4	~	~

커뮤니티 댓글

ID	내용
1	~
2	~
3	~
4	~

> 이런 커뮤니티 댓글 테이블과 C서버를 추가한다고 가정
>
> *게시글 테이블*
> A서버: 1, 2
> B서버: 3
> C서버: 4
>
> 댓글 테이블
> A서버: 1

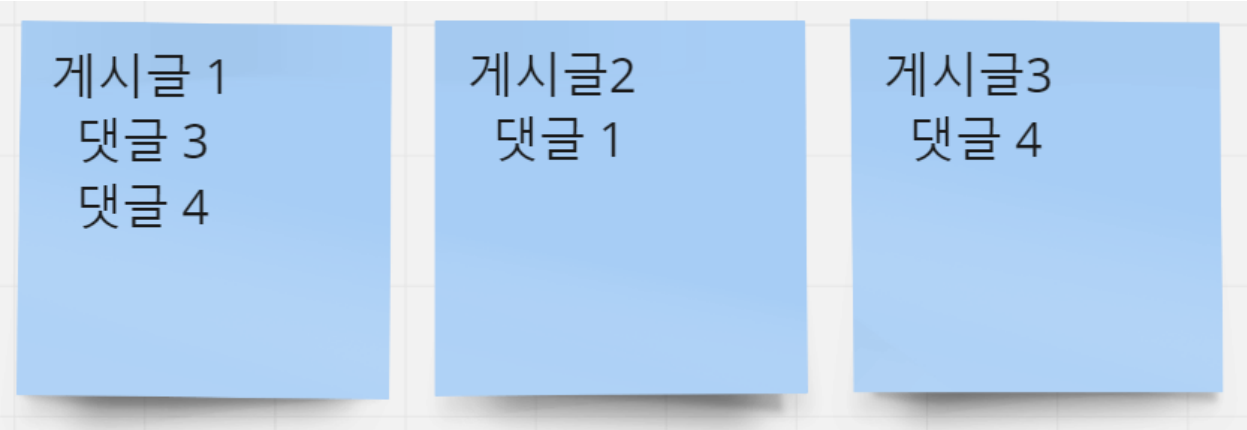
> B서버: 2, 3
> C서버: 4

이렇게 되면 이제 더 복잡해지는데, 데이터를 어떤 기준으로 나누어 저장할지부터 (샤딩)

만약 데이터를 나눴는데 A서버에 있는 게시글1에 댓글 3과 4가 속해 있다면 모든 서버를 뒤져야 원하는 결과를 가져올 수 있음.

이렇게 **관계가 생길수록 데이터가 늘어날수록 RDB에서는 수평확장에 대한 관리가 복잡해짐.**

NoSQL의 수평 확장



> 이렇게 게시글이 주체가 되어 구성돼있는 NoSQL 컬렉션이라면
> A서버: 1
> B서버: 2
> C서버: 3
> 이렇게 단순히 구간만 잘라서 데이터를 나누어 줄 수 있음.

NoSQL은 주체가 되는 객체의 데이터에 집중해서 데이터를 저장하기 때문에 해당 **주체가 누구인지만 알면** 데이터를 한 번에 다 가져올 수 있다는 장점이 있다.
이는 NoSQL이 RDB처럼 관계에 집중하는 것이 아니라
'주체가 되는 객체' 에 집중하는 구조이기 때문이다.

RDB vs NoSQL 고르기

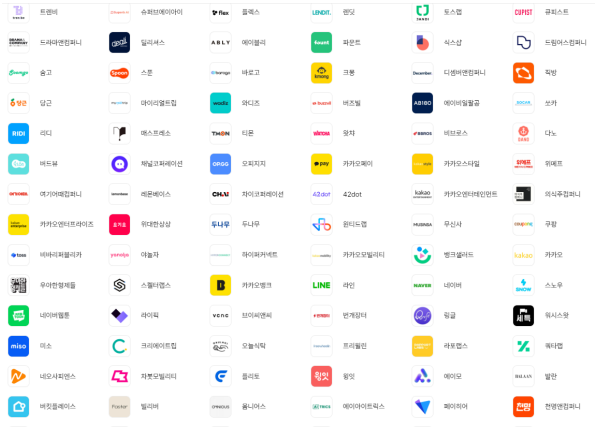
RDB는 주로 정형화된 데이터를 저장하고 복잡한 쿼리 및 데이터 조인을 수행할 때 사용됨.
⇒ 주문 관리 시스템과 같이 고객 정보, 주문 정보, 상품 정보 등을 관리하는 경우 등이 적합

NoSQL은 유연한 스키마 구조로 데이터를 저장하고, 대용량의 데이터 처리 및 분산 처리를 효율적으로 수행할 수 있음.
⇒ SNS나 로그 데이터와 같이 비정형 데이터를 다루는 경우 적합

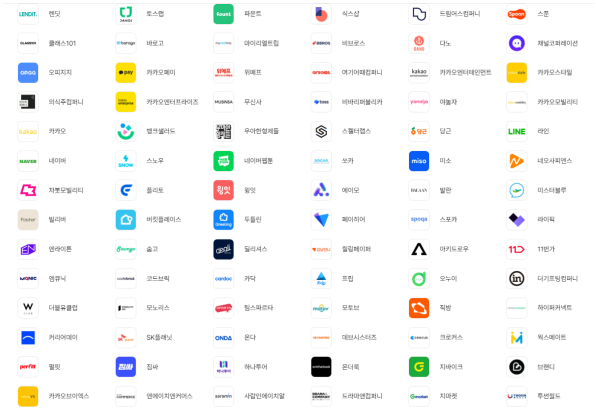
NoSQL은 특별한 이슈(데이터, 트래픽 증가 등)에 대응 하기 위해 나온 기술이기 때문에 처음에 특별한 이유가 없다면 RDB를 쓰다가 이슈가 생겼을 때 솔루션을 찾아도 늦지 않다.

(니꼴라스 쌤이 그랬어요)

NoSQL 사용하는 수많은 기업들 ~~*NoSQL 배워야 할지도? ..*~~



Redis



MongoDB

RDB와 NoSQL의 Scale-Out 이해 도움 블로그

https://broccoli45.tistory.com/46

참고한 곳

https://tecoble.techcourse.co.kr/post/2021-10-12-scale-up-scale-out/

https://velog.io/@cis07385/DB-NoSQL-그리고-RDBMS와의-차이점-장단점

https://www.youtube.com/watch?v=cnPRFqukzek