CP5. 복제

Part 2. 분산 데이터

▼ 개요

- Part 1 에서는 단일 장비에서 데이터를 저장할 때 적용하는 데이터 시스템 측면을 다뤘다.
  Part 2 에서는 저장소와 데이터 검색에 여러 장비가 관여하면 무슨일이 발생할까? 주제 중심으로 다룬다.
- 여러 장비 간 분산된 DB를 필요하는 이유는 여러가지다.
  - 확장성
    데이터 볼륨, 읽기 부하, 쓰기 부하가 단일 장비에서 다룰 수 있는 양보다 커지면 부하를 여러 장비로 분배할 수있다.
  - 내결함성/고가용성
    장비 하나(또는 여러 장비나 네트워크, 전체 데이터센터)가 죽더라도 애플리케이션이 계속 동작해야 한다면 여러 장비를 사용해 중복성을 제공할 수 있다. 장비 하나가 실패하면 다른 하나가 이어 받는다. (fail over)
  - 지연시간
    전 세계 사용자가 있다면 사용자와 지리적으로 가까운 곳의 데이터센터에서 서비스를 제공하기 위해 전 세계 다양한 곳에 서버를 두고 싶을 것이다. 이를 통해 사용자는 네트워크 패킷이 지구를 반 바퀴 돌아서 올 때까지 기다릴 필요 없다.
- **고부하로 확장**
  - **공유 메모리 아키텍쳐**
    - 고부하 확장이 필요하다면 더 강한 장비를 구매하는게 가장 단순하다(수직 확장, 용량 확장) 많은 CPU, 메모리, 디스크를 하나의 운영체제로 함께 결합할 수 있다. 그래서 빠른 상호 연결로 모든 CPU 가 메모리나 디스크의 모든 부분에 접근 할 수 있다.
    - 공유 메모리 아키텍처에는 모든 구성 요소를 단일 장비처럼 다룰 수 있다.
    - 문제점은 비용이 선형적인 추세보다 훨씬 빠르게 증가한다.
      시스템 성능이 두 배를 내기 위해서는 비용이 두 배 이상이 소요된다.
      또한 병목 현상 때문에 두 배 크기의 장비가 반드시 두 배의 부하를 처리할 수 있는 것은 아니다.
    - 공유 메모리 아키텍처는 제한적인 내결함성을 제공한다. (장비를 중단 시키지 않고 스케일 업 할 수 있다) 하지만 완전히 하나의 지리적 위치로 제한된다.
  - **공유 디스크 아키텍쳐**
    - 공유 메모리 아키텍쳐와는 다른 접근 방식이다. 독립적인 CPU와 RAM 을 탑재한 여러 장비를 사용하지만 데이터 저장은 장비 간 공유하는 디스크 배열을 한다.

    - 여러 장비는 고속 네트워크로 연결된다. 일부 데이터 웨어하우스 작업부하이 이 아키텍쳐를 사용하지만 잠금 경합과 오버헤드가 공유 디스크 접근 방식의 확장성을 제한한다.
- **비공유 아키텍쳐 (수평 확장, 규모 확장, 스케일 아웃)**
    - DB 소프트웨어를 수행하는 각 장비나 가상 장비를 **노드**라고 부른다.
      각 노드는 CPU, RAM, 디스크를 독립적으로 사용한다.
      노드 간 코디네이션은 일반적인 네트워크를 사용해 소프트웨어 수준에서 수행한다.
    - 비공유 시스템은 특별한 하드웨어를 필요하지 않아 가격 대비 성능이 가장 좋은 시스템을 사용할 수 있다. 잠재적으로 지리적인 영역에 걸쳐 데이터를 분산해 사용자 지연 시간을 줄이고 전체 데이터센터의 손실을 줄일 수 있다.
    - Part 2 에서는 비공유 아키텍쳐에 중점을 둔다.
      비공유 아키텍쳐를 사용시 애플리케이션 개발자가 반드시 주의해야 하는 점이 있기 때문,
      데이터를 여러 노드에 분산하려면 분산 시스템에서 발생하는 제약 조건과 트레이드오프를 알고 있어야 한다. DB 스스로 이런 점을 숨길 수 없다
    - 대개 장점이 많지만, 애플리케이션 복잡도를 야기하고 때로는 데이터 모델의 표현을 제한한다. 경우에 따라 간단한 단일 스레드 프로그램이 100개 이상의 CPU 코어를 ㅅ용하는 클러스터 보다 효율적일 수 있다. 하지만 비공유 시스템은 매우 강력하다.
- **복제 대 파티셔닝**
    - 여러 노드에 데이터를 분산하는 방법은 일반적으로 두 개다.
        - 복제
            - 같은 데이터 복사본을 잠재적으로 다른 위치에 있는 여러 노드에 유지한다.
            - 복제는 중복성을 제공한다. 일부 노드가 사용 불가능한 상태라면 해당 데이터는 남은 다른 노드를 통해 여전히 제공될 수 있다. 복제는 성능 향상에도 도움된다.
        - 파티셔닝
            - 큰 DB를 파티션이라는 작은 서브셋으로 나누고 파티션은 각기 다른 노드에 할당한다. (**샤딩**)
    - 복제와 파티셔닝은 다른 매커니즘이지만, 서로 관련있다.
        - 파티셔닝과 복제를 같이 사용해 분산 시스템에서 필요한 어려운 트레이드오프를 설명할 수 있다.
        - 트랜잭션을 이해하면 데이터 시스템에 발생하는 많은 문제를 설명하는 데 도움을 준다.
        - 이후 장에서 복잡한 애플리케이션의 요구사항을 만족하기 위해 어떻게 다양한 (분산된) 데이터 저장소를 가져와 대규모 시스템을 통할할 수 있는지 설명한다.

5. 복제