6장

키-값 저장소 설계

키-값 저장소

- 비 관계형 데이터 베이스
- 키에 매달린 값은 키를 통해서만 접근할 수 있다.
    - 키 : 일반 텍스트 or 해시 값 (짧을 수록 좋음)
    - 값 : 문자열 or 리스트 or 객체

문제 이해 및 설계 범위 확정

완벽한 설계는 불가능하다. 따라서 다음과 같은 부분들을 고려하면 좋다.

a. 읽기, 쓰기, 메모리 사용량 사이의 균형
b. 데이터의 일관성과 가용성 사이의 타협적 결정

이 챕터에서 고려할 키-값 저장소의 특성

- 키-값 쌍의 크기는 10KB 이하이다.
- 큰 데이터를 저장할 수 있어야 한다.
- 높은 가용성을 제공해야 한다. 따라서 시스템은 설사 장애가 있더라도 빨리 응답해야 한다.
- 높은 규모 확장성을 제공해야 한다. 따라서 트래픽 양에 따라 자동적으로 서버 증설/삭제가 이루어져야 한다.
- 데이터 일관성 수준은 조정이 가능해야 한다.
- 응답 지연시간이 짧아야 한다.

단일 서버 키-값 저장소

한 대의 서버만 사용하는 키-값 저장소를 설계하는 방법

1. 해시 테이블
2. 데이터 압축
3. 자주 쓰이는 데이터만 메모리에 두고 나머지는 디스크에 저장

분산 키-값 저장소

- 분산 해시 테이블이라고도 부른다.
- CAP 정리(Consistency, Availability, Partion Tolerance theorem)를 이해하고 있어야 한다.

CAP 정리

*데이터 일관성, 가용성, 파티션 감내 이 세 가지 요구사항을 동시에 만족하는 분산 시스템을 설계하는 것은 불가능*하다는 정리.

- 데이터 일관성 : 어떤 노드에 접속했느냐에 관계없이 언제나 같은 데이터를 보게 되어야 한다.
- 가용성 : 일부 노드에 장애가 발생하더라도 항상 응답을 받을 수 있어야 한다.
- 파티션 감내 : 네트워크에 파티션(두 노드 사이의 통신 장애가 발생하는 상황)이 생기더라도 시스템은 계속 동작하여야 한다.

그림에서 볼 수 있듯이 세 영역은 동시에 만족할 수 없다.

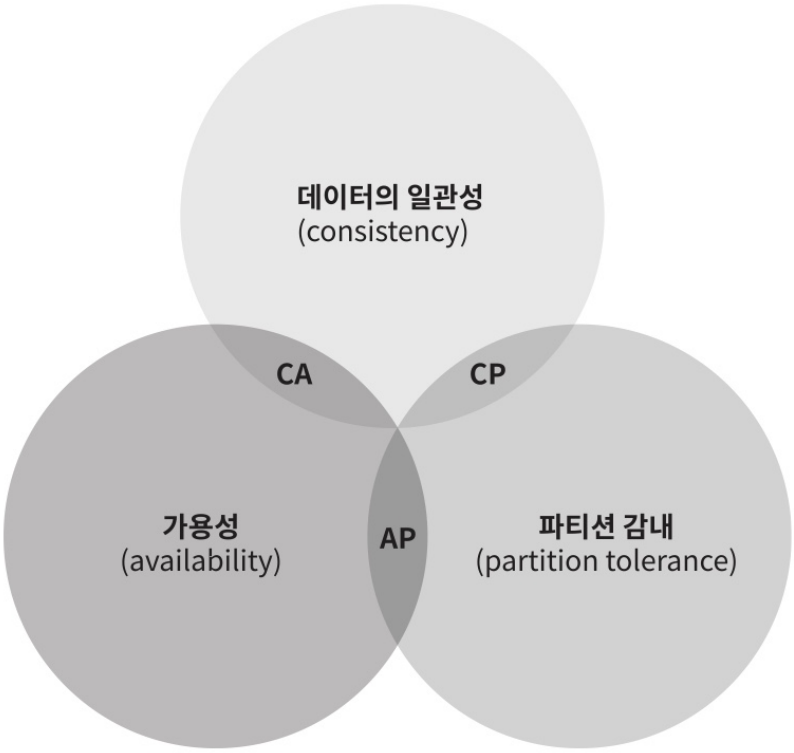

그림 6-1

이상적 상태

이상적 환경인 경우 네트워크가 파티션되는 경우는 발생하지 않는다.

- 3개의 저장소가 있는 경우 n1에 기록된 데이터는 자동적으로 n2와 n3에 복제된다.
- 데이터 일관성과 가요성도 만족된다.

실세계의 분산 시스템

- (가용성) n3에 장애가 발생한 경우
  - n3에 기록된 최신 데이터가 n1, n2로 전달되지 않음.
  - n1, n2에 기록된 데이터가 n3에 전달되지 않음.

- (일관성) n3에 장애가 발생한 경우
  - n1, n2에 대한 쓰기 연산을 중단 시킴.

어떤 특성을 유지할 것인지는 서비스의 성격에 따라 결정해야 한다. (ex. 은행-일관성)

시스템 컴포넌트

- 데이터 파티션
  - 대규모 애플리케이션의 경우 전체 데이터를 한 대 서버에 할당하는 것이 불가능하다. 가장 단순한 해결책으로 데이터를 작은 파티션들로 분할한 다음 여러 대 서버에 저장한다.

    고려할 사항

    a. 데이터를 여러 서버에 고르게 분산할 수 있는가

    b. 노드가 추가되거나 삭제될 때 데이터의 이동을 최소화할 수 있는가

    이 때 안정 해시 기술을 사용하면 좋다.

    장점

    a. 규모 확장 자동화 : 시스템 부하에 따라 서버가 자동으로 추가되거나 삭제되도록 만들 수 있다.

    b. 다양성 : 각 서버의 용량에 맞게 가상 노드의 수를 조정할 수 있다.

- 데이터 일관성
  - 여러 노드에 다중화 된 데이터는 적절히 동기화 되어야 한다. 이를 위해 정족수 합의 프로토콜을 사용하면 읽기/쓰기 연산 모두에 일관성을 보장할 수 있다.
    - 정족수 합의 프로토콜 : 사본 개수 N, 쓰기 연산에 대한 정족수 W, 읽기 연산에 대한 정족수 R을 기준으로 연산의 성공 여부를 판단한다.

  일관성 모델

  - 강한 일관성 : 모든 읽기 연산은 가장 최근에 갱신한 결과를 반환한다.
  - 약한 일관성 : 읽기 연산은 가장 최근에 갱신된 결과를 반활하지 못할 수도 있다.
  - 최종 일관성 : 약한 일관성의 한 형태로, 갱신 결과가 결국에는 모든 사본에 반영되는 모델이다.

비 일관성 해소법

- 버저닝과 벡터 시계를 통해 사본 간 틀어진 일관성을 해소한다.
  - 모든 데이터에 [서버, 버전]의 순서쌍을 매단다.
  - 어떤 데이터를 서버에 저장할 때 이미 존재하는 데이터라면 버전을 업데이트하고, 존재하지 않는 데이터라면 새 데이터를 저장한다.

  주의사항

  - 순서쌍 개수가 굉장히 빠르게 증가하므로 설정한 임계치 이상으로 길이가 늘어나면 오래된 순서쌍을 벡터 시계에서 제거해야 한다.

장애 처리

- 장애를 처리하는 방법
  - 장애 감지
  - 장애 해소

장애 감지

- 분산 시스템에서는 하나의 서버에 장애가 생겼다고 하여 바로 장애를 해결하지는 않는다.
  - 멀티캐스팅 : 가장 쉽지만 비효율적이다.
  - 가십 프로토콜 : 각 노드가 주기적으로 자신의 박동 카운터를 증가시키고, 무작위로 선정된 노드들에게 박동 카운터 목록을 보낸다. 이 때 지정된 시간 동안 갱신되지 않는 서버가 있다면 해당 멤버를 장애 상태로 간주한다.

일시적 장애 처리

- 단서 후 임시 위탁 : 가십 프로토콜로 감지한 장애 서버로 향하는 요청을 다른 서버가 잠시 맡아서 처리한다. 이후 장애 서버가 복구되면 변경사항을 일괄 반영하여 데이터의 일관성을 보존하는데 이를 위해 임시로 쓰기 연산을 한 서버에서 그에 관한 단서를 남긴다.

영구 장애 처리

- 반-엔트로피 프로토콜
  - 해시 트리라고도 불리는 머클 트리를 이용한다. 머클 트리는 각 노드에 그 자식 노드들에 보관된 값의 해시, 또는 자식 노드들의 레이블로부터 계산된 해시 값을 레이블로 붙여두는 트리다.
  - 머클 트리를 비교할 때에는 루트부터 비교한다. 루트 값이 같다면 두 서버는 같은 데이터를 갖고 있다고 판단한다. 값이 다른 경우는 왼쪽, 오른쪽 자식 노드를 차례로 비교한다.

데이터 센터 장애 처리

- 데이터를 여러 데이터 센터에 다중화한다.

쓰기 경로

1. 쓰기 요청이 커밋 로그 파일에 기록된다.
2. 데이터가 메모리 캐시에 기록된다.
3. 메모리 캐시가 가득차거나 임계치에 도달하면 데이터는 디스크에 있는 SSTable에 기록된다.

읽기 경로

1. 데이터가 메모리에 있는지 검사한다. 없으면 2로 간다.
2. 데이터가 메모리에 없으므로 블룸 필터를 검사한다.
3. 블룸 필터를 통해 어떤 SSTable에 키가 보관되어 있는지 알아낸다.
4. SSTable에서 데이터를 가져온다.
5. 해당 데이터를 클라이언트에 반환한다.