6장 키-값 저장소 설계

키-값 저장소

- 키-값 데이터베이스라고 불리는 비 관계형 데이터베이스
- 키는 고유 식별자
- 값은 문자열, 리스트, 객체 등

이번 장 목표

`put(key,value)` `get(key)` 연산을 지원하는 키-값 저장소 설계

키-값 저장소 설계 조건

- 키-값 쌍의 크기는 10KB 이하
- 큰 데이터를 저장할 수 있어야 함
- 높은 가용성 제공 → 시스템에 장애 있어도 빠르게 응답해야 한다
- 높은 규모 확장성 제공 → 트래픽양에 따라 자동으로 서버 증설, 삭제
- 데이터 일관성 수준은 조정 가능해야 함
- 짧은 응답 지연시간

단일 서버 키-값 저장소

모든 데이터를 해시 테이블로 저장하는 방법이지만 모든 데이터를 메모리 안에 두는 것은 불가능 한 일이다

개선책

- 데이터 압축
- 자주 쓰이는 데이터만 메모리에 두고 나머지는 디스크 저장

분산 키-값 저장소, 분산 해시 테이블

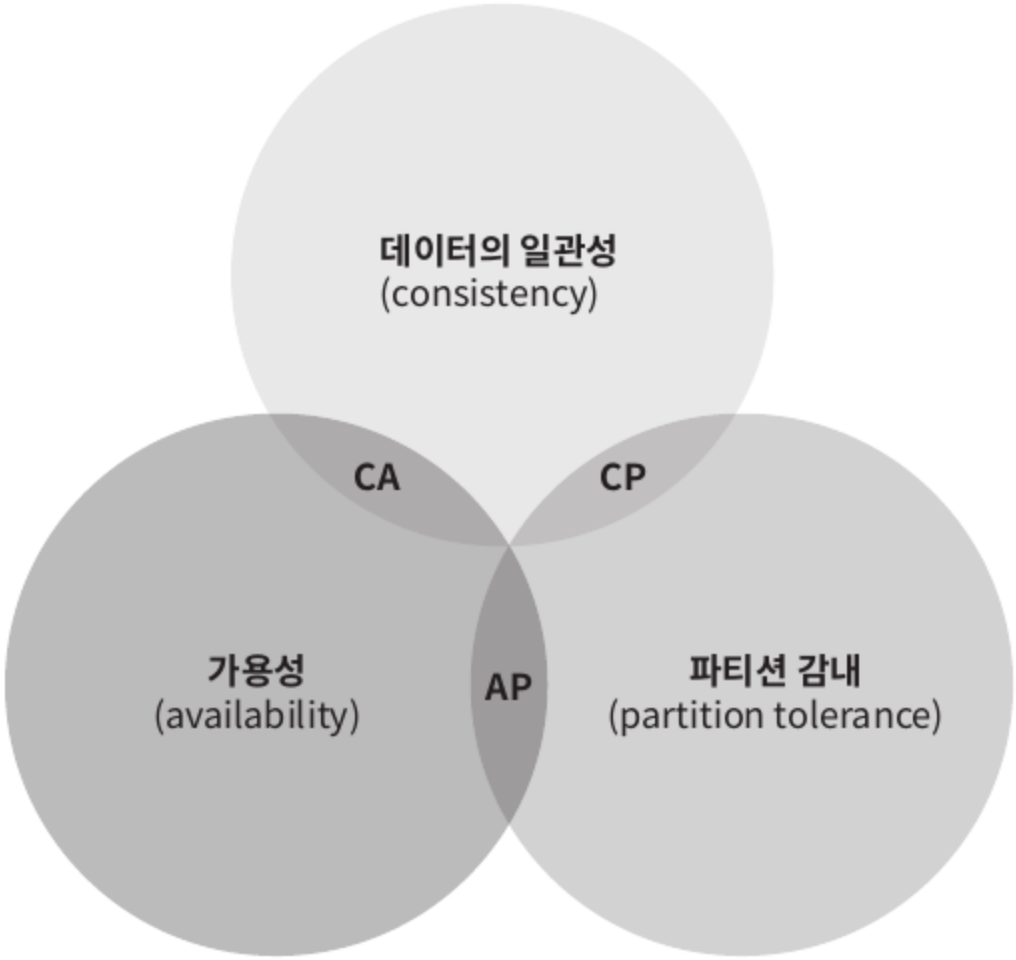
키-값 쌍을 여러 서버에 분산시키고 CAP 정리 사용

CAP 정리

Consistency, 데이터 일관성 : 분산 시스템에 접속하는 모든 클라이언트는 어떤 노드에 접속했느냐에 관계없이 언제나 같은 데이터를 보아야 함

Availability, 가용성 : 분산 시스템에 접속하는 클라이언트는 일부 노드에 장애가 발생하더라도 항상 응답을 받을 수 있어야 함

Partition tolerance, 파티션 감내 : 파티션은 두 노드 사이에 통신 장애가 발생하였음을 의미. 네트워크 파티션이 생기더라도 시스템은 계속 동작해야 함



CAP 세가지 요구사항을 동시에 만족하는 분산 시스템을 설계하는 것은 불가능하다(왜?) 그래서 위와 같이 3가지 중 두가지만 만족할 수 있도록 위와 같이 분류 가능

CP : A 희생

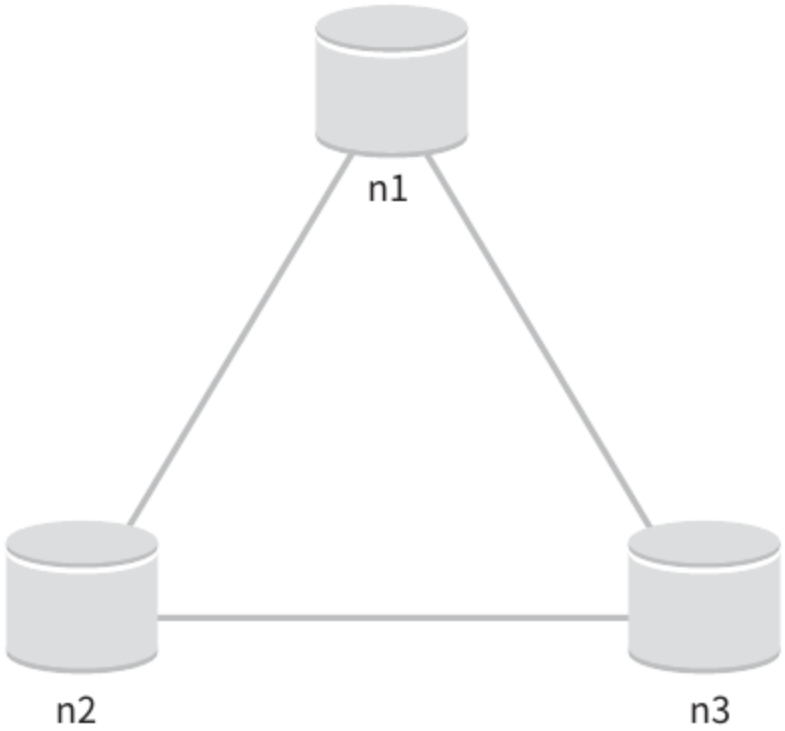
AP : C 희생

CA : P 희생, 하지만 네트워크 장애는 피할 수 없는 일이므로 분산 시스템은 반드시 피티션 문제를 감내할 수 있도록 설계해야 한다. 즉 실세계에서 CA는 존재하지 않는다.

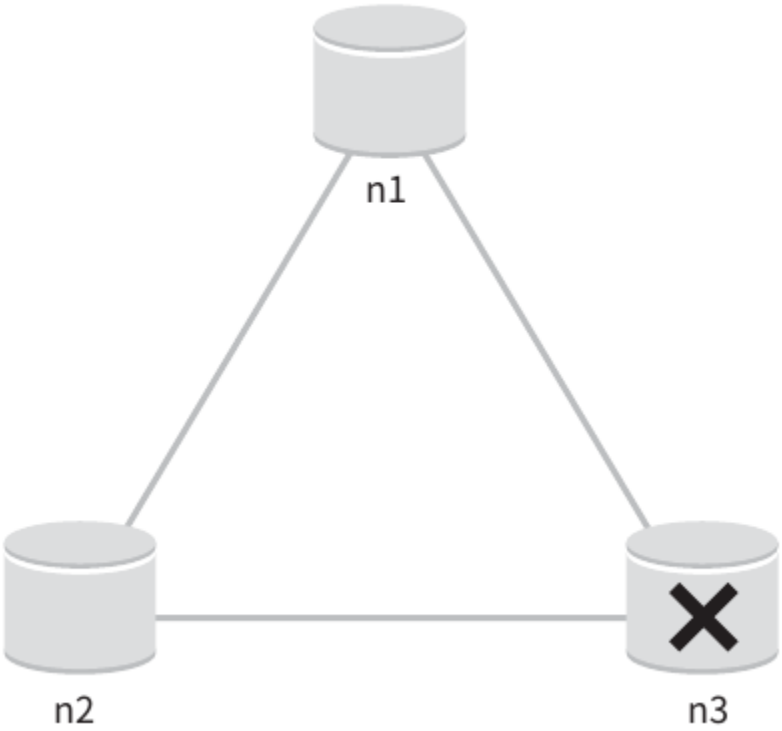
이상적 상태

분산 시스템에서 데이터는 보통 여러 노드에 복제되어 보관된다. 세 개의 복제 노든 n1, n2, n3에 데이터를 복제하여 보관하는 상황이 있다면 절대로 일어나지 않을 것

n1에 기록된 데이터는 자동적으로 n2, n3에 복제된다. 이는 데이터 일관성과 가용성을 만족한다



실세계의 분산 시스템



분산 시스템은 파티션 문제를 피할 수 없다. 만약 n3에 장애가 발생하여 n1, n2에 통신할 수 없는 상황이라면 n1,n2 / n3는 서로 데이터가 전달되지 않는다.

일관성을 선택했다면(CP시스템) 세 서버 사이에 생길 수 있는 데이터 불일치 문제를 피하기 위해 n1과 n2에 대해 *쓰기 연산을 중단시켜야 하는데, 그렇게 하면 가용성이 깨진다.*

반대로 **가용성을 선택한다면(AP시스템)** 낡은 데이터를 반환할 위험이 있다고 해도 계속 읽기 연산을 허용해야 한다.그리고 n1과 n2에서는 계속 *쓰기 연산을 허용*할 것이고, *파티션 문제가 해결된 후에 새 데이터를 n3에 전송*할 것이다.

분산 키-값 저장소를 만들 때에는 그 요구사항에 맞도록 CAP정리를 적용해야 한다.

시스템 컴포넌트

키-값 저장소 구현에 사용될 핵심 컴포넌트들 및 기술들에 대해 알아보자.

- 데이터 파이션
- 데이터 다중화
- 일관성

- 일관성 불일치 해소
- 장애 처리
- 시스템 아키텍처 다이어그램
- 쓰기 경로
- 읽기 경로

데이터 파이션

데이터를 작은 파티션들로 분할한 후 여러 대의 서버에 저장하는 것

파티션으로 나눌 때 중요하게 판단해야 하는 요소 2가지

- 데이터를 여러 서버에 고르게 분산할 수 있는가
- 노드가 추가되거나 삭제될 때 데이터의 이동을 최소화할 수 있는가

위 문제는 안정 해시로 적절하게 해결 가능하다. 안정해시를 사용하여 데이터 파티션을 하면 좋은 점은 다음과 같다.

규모 확장 자동화 : 시스템 부하에 따라 서버가 자동으로 추가되거나 삭제될 수 있다.

다양성 : 각 서버의 용량에 맞게 가상 노드의 수를 조정할 수 있다. (고성능 서버는 더 많은 가상 노드를 갖도록 설정할 수 있다)

데이터 다중화

높은 가용성과 안정성을 확보하기 위해 데이터를 N개 서버에 비동기적으로 다중화 해야 한다.

여기서 N은 튜닝 가능한 값

N개 서버를 선정하는 방법은 어떤 키를 해시 링 위에 배치한 후, 그 지점으로부터 시계 방향으로 링을 순회하면서 만나는 첫 N개의 서버에 데이터 사본을 보관하는 것

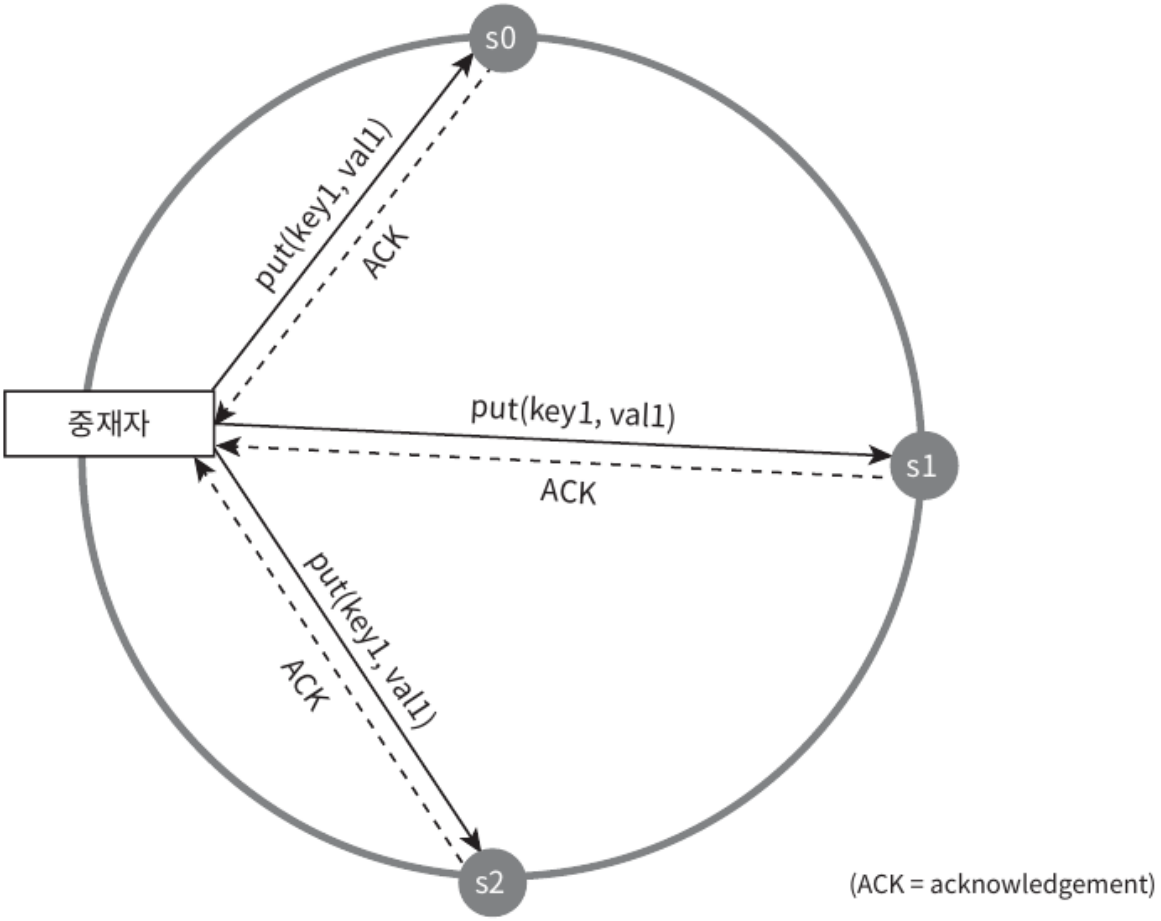
그러나 가상 노드를 사용하면 위와 같이 선택한 N개의 노드가 실제 대응될 물리 서버의 갯수 N보다 작아질 수 있다.

데이터 일관성

여러 노드에 다중화된 데이터는 적절히 동기화가 되어야 한다.

정족수 합의(Quorum Consensus) 프로토콜을 사용하면 읽기/쓰기 연산 모두에 일관성을 보장할 수 있다.

- N=사본 개수
- W=쓰기 연산에 대한 정족수. 쓰기 연산이 성공한 것으로 간주되려면 적어도 W개의 서버로부터 쓰기 연산이 성공했다는 응답을 받아야 한다.
- R=읽기 연산에 대한 정족수. 읽기 연산이 성공한 것으로 간주되려면 적어도 R개의 서버로부터 응답을 받아야 한다.



중재자는 클라이언트와 서버 사이에서 proxty 역할을 한다.

W=1의 의미는, 쓰기 연산이 성공했다고 판단하기 위해 중재자는 최소 한 대의 서버로부터 쓰기 성공 응답을 받아야 한다는 것이다.

즉, s1으로부터 성공 응답을 받았다면 s0, s2로부터의 응답을 기다릴 필요는 없다.

W, R, N의 값을 정하는 것은 응답 지연과 데이터 일관성 사이의 타협점을 찾는 전형적인 과정이다.

W=1 또는 R=1의 구성인 경우 한 대의 서버로부터의 응답만을 받으면 되기 때문에 응답속도는 빠를 것이다.

만약 이보다 큰 경우 데이터 일관성의 수준은 높지만 응답 속도는 느려질 것이다.

W+N>N의 경우에는 강한 일관성이 보장된다.일관성을 보증할 최신 데이터를 가진 노드가 최소 하나는 겹칠 것이기 때문이다.

- R=1, W=N : 빠른 읽기 연산에 최적화된 시스템
- W=1, R=N : 빠른 쓰기 연산에 최적화된 시스템
- W+R>N : 강한 일관성 보장
- W+R<=N : 강한 일관성의 보장이 없다.요구되는 일관성 수준에 따라 조정하면 된다.

일관성 모델

일관성 모델은 키-값 저장소를 설계할 때에 고려해야 할 또 하나의 중요한 요소이다.일관성 모델은 데이터 일관성의 수준을 결정하는데, 종류가 다양하다.

강한 일관성

- 모든 읽기 연산은 가장 최근에 갱신된 결과를 반환한다.다시 말해서 클라이언트는 절대로 낡은 데이터를 보지 못한다.

약한 일관성

- 읽기 연산은 가장 최근에 갱신된 결과를 반환하지 못할 수 있다.

최종 일관성

- 약한 일관성의 한 형태로, 갱신 결과가 결국에는 모든 사본에 반영(==동기화) 되는 모델이다.

- 일관성 불일치 해소
- 장애 처리

- 시스템 아키텍처 다이어그램
- 쓰기 경로
- 읽기 경로