가상 면접 사례로 배우는

대규모 시스템 설계 기초

이정민

6. 키-값 저장소 설계

문제 이해 및 설계 범위 확정

■ 설계 할 키-값 저장소의 특성

- **키-값 쌍의 크기는 10kb 이하**
- **큰 데이터를 저장 할 수 있어야 함**
- **높은 가용성을 제공해야 함 (장애가 있더라도 빨리 응답해야 함)**
- **높은 규모 확장성을 제공해야 함 (트래픽 양에 따라 자동적으로 서버 증설/삭제가 이루어 져야 함)**
- **데이터 일관성 수준은 조정이 가능해야 함**
- **응답 지연시간이 짧아야 함**

단일 키-값 저장소

- 가장 쉽고 간단한 방법
  - **키-값 쌍 전부를 메모리에 해시 테이블로 저장**
  - **빠른 속도를 보장 하지만 모든 데이터를 저장하기 불가능할 수 있음**
  - **개선 방법**
    - 데이터 압축, 자주 쓰이는 데이터는 메모리, 나머지는 디스크에 저장
  - **많은 데이터를 저장하려면 분산 키-값 저장소 필요**

분산 키-값 저장소 (1/7)

- CAP 정리
  - **아래를 동시에 만족하는 분산 시스템 설계 불가**
    - 어떤 두가지를 충족하려면 하나는 반드시 희생되어야함
  - **데이터 일관성(Consistency)**
    - 클라이언트는 노드에 상관없이 같은 데이터를 봐야함
  - **가용성(Availability)**
    - 일부 노드 장애 시에도 가 항상 응답을 받을 수 있어야 함
  - **파티션 감내(Partition tolerance)**
    - 두 도느 간 통신 장애 시에도 시스템은 계속 동작해야함
  - **CP 시스템 or AP 시스템, CA 시스템(미존재)**

- 사례
  - **이상적 상태**
    - 이상적 환경에서는 네트워크 파티션은 없음 (C, A도만족)
  - **실세계의 분산 시스템**
    - 분산 시스템은 네트워크 파티션을 필할 수 없음
    - 파티션 발생 시 C, A 중 하나를 선택해야함
    - CP : 데이터 불일치를 피하기 위해 쓰기 연산 중단
    - AP : 낡은 데이터라도 쓰기/읽기 연산을 허용

분산 키-값 저장소 (2/7)

■ 핵심 구성요소 및 기술

- 데이터 파티션
- 데이터 다중화(replication)
- 일관성(consistency)
- 일관성 불일치 해소(inconsistency resolution)
- 장애 처리
- 시스템 아키텍처 다이어그램
- 쓰기 경로(write path)
- 읽기 경로(read path)

분산 키-값 저장소 (3/7)

- 데이터 파티션
  - **대규모 애플리케이션의 경우 전체 데이터를 작은 파티션으로 분할하여 여러대의 서버에 저장**
  - **고려사항**
    - 데이터를 여러 서버에 고르게 분산할 수 있는가?
    - 노드가 추가/삭제 시 데이터 이동을 최소화할 수 있는가?
  - **안정 해시를 사용하여 데이터 파티션**
    - 규모 확장 자동화 (부하에 따라 서버가 자동 추가/제거)
    - 다양성 (각 서버별로 가상 노드의 수를 조정)

- 데이터 다중화
  - **높은 가용성과 안정성 확보를 위해 데이터를 N대의 서버에 비동기적으로 다중화**
    - 해시 리에서 시계 방향으로 순회하면서 만나는 N개의 서버에 데이터 사본을 보관
  - **고려사항**
    - 가상노드 사용시 N개의 노드레 동일 물리서버 중복 발생 가능 (동일 물리 서버 중복을 피해야함)
    - 동일 데이터 센터의 노드는 동시에 동일 문제 발생가능 (안정성을 위해 고속 네트워크로 연결된 다른 센터이용)

분산 키-값 저장소 (4/7)

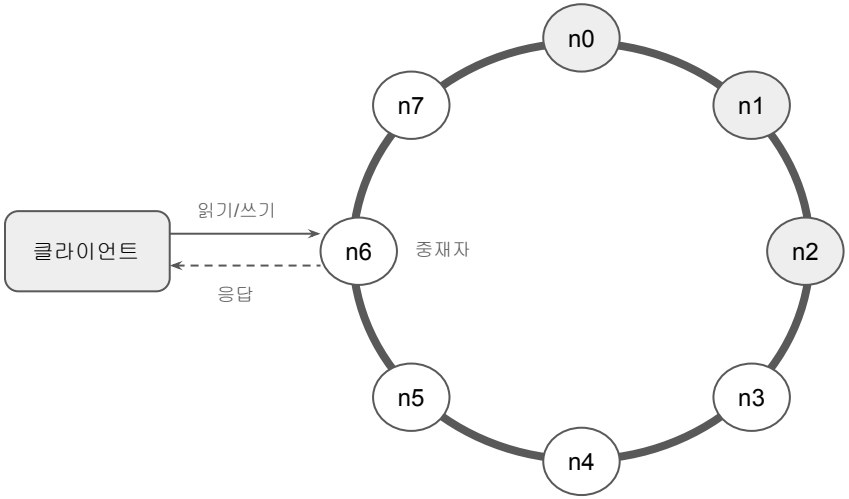
- 데이터 일관성
  - **여러 노드에 다중화된 데이터는 동기화 필요**
    - 정족수 합의 프로토콜로 읽기/쓰기 연산에 일관성 보장
    - 중재자가 클라이언트와 노드 간 프락시(proxy) 역할
  - **동작 원리**
    - N = 사본 개수
    - W = 쓰기 연산 정족수 (최소 W개 서버로부터 성공 응답)
    - R = 읽기 연산 정족수 (최소 R개 서버로부터 성공 응답)
    - R=1, W=N : 빠른 읽기 연산에 최적화
    - W=1, R=N : 빠른 쓰기 연산에 최적화
    - W+R >= N : 강한 일관성 보장됨 (보통 N=3, W=R=2)
    - W+R <= N : 강한 일관성 미보장
    - 요구되는 일관성 수준에 따라 W, R, N의 값을 조정
  - **일관성 모델 (데이터 일관성 수준 결정)**
    - 강한 일관성 : 가장 최근 갱신 결과반환 (낡은 데이터 x), 모든 사본에 쓰기 연산 반영까지 읽기/쓰기 금지 (고가용성 부적합)
    - 약한 일관성 : 가장 최근 갱신 결과반환 미보장
    - 최종 일관성 : 약한 일관성의 형태로, 결국에는 동기화
  - **비일관성 해소 기법 : 데이터 버저닝**
    - 사본간 일관성이 깨질 가능성을 해소하기 위한 기술
    - 버저닝은 데이터 변경시 마다 새로운 버전을 생성하는 기법
    - 벡터시계는 버전 간의 충돌을 해소하는 기법 (선후행 판단)

분산 키-값 저장소 (5/7)

- **장애감지**
  - 분산시스템에서는2대 이상 서버가 장애 보고시 장애로 간주
  - 모든 노드 간 멀티캐스팅 채널 구축 : 쉽지만 비효율적
  - 가쉽 프로토콜 : 주기적으로 최신 박동정보를 갱신/전송
- **일시적 장애처리**
  - 엄격한 정족수에서는 장애로 감지된 시스템은 읽기와 쓰기 연산을 금지해야 함
  - 느슨한 정족수 접근법 : 가용성을 위하여 조건을 완화
  - 임시 위탁 기법 : 장애 서버로 가는 요청을 일시적으로 다른 서버가 처리
- **영구 장애 처리**
  - 일시적 장애가 아닌 영구적인 장애 처리
  - 반-엔트로피 프로토콜 : 노드 간 일관성 유지를 위한 방법, 노드간 데이터의 차이점을 찾아서 동기화. 영구적인 장애 발생 시 다른 서버들에 보관된 데이터 복제본을 사용해 손실된 데이터를 복구
  - 머클 트리 : 변경이 필요한 데이터 부분만 식별하여 전체 데이터를 전송하지 않고 효율적으로 데이터를 동기화 할 수 있음
- **데이터센터 장애 처리**
  - 정전, 네트워크 장애, 자연재해 등 발생을 대비하여 여러 데이터 센터에 다중화 필요

분산 키-값 저장소 (6/7)

- 시스템 아키텍처 다이어그램
  - **주요 기능**
    - 클라이언트 : 키-값 저장소가 제공하는 API 사용
    - 중재자 : 클라이언트에게 키-값 저장소의 프록시 역할
    - 노드 : 안정 해시 링 위에 분포
    - 시스템은 완전 분산됨 (자동으로 노드자 추가/제거 가능)
    - 데이터는 여러 노드에 다중화됨
    - 모든 노드가 같은 책임을 지므로, SPOF는 존재하지 않음
  - **분산 설계에서 아래 기능 지원 해야 함**
    - 클라이언트API, 장애감지, 데이터 충돌 해소, 장애복구 메커니즘, 다중화, 저장소 엔진 등



분산 키-값 저장소 (7/7)

■ 쓰기 경로

- **쓰기 절차 (카산드라 사례)**
  - 쓰기 요청이 커밋 로그 파일에 기록함
  - 데이터가 메모리 캐시에 기록함
  - 메모리 캐시가 가득차거나 임계치 도달 시 디스크의 SSTable(Sortered-String Table)에 기록함

■ 읽기 경로

- **읽기 절차 (카산드라 사례)**
  - 캐시에 데이터가 있는지 확인 (있으면 데이터 반환)
  - 없는 경우 디스크에서 가져옴 (보통 블룸필터 기법 사용)
  - 블룸필터를 통해 어떤 SSTable에 키가 있는지 확인
  - SSTable에서 데이터를 가져옴
  - 해당 데이터를 클라이언트에 반환

요약

- 분산 키-값 저장소 기능과 구현 기술
  - 대규모 데이터 저장 : 안정 해시를 사용해 서버들에 부하분산
  - 읽기 연산에 대한 높은 가용성 보장 : 데이터를 여러 데이터센터에 다중화
  - 쓰기 연산에 대한 높은 가용성 보장 : 버저닝 및 벡터 시계를 사용한 충돌 해소
  - 데이터 파티션, 점진적 규모 확장성, 다양성 : 안정 해시
  - 조절 가능한 데이터 일관성 : 정족수 합의
  - 일시적 장애 처리 : 느슨한 정족수 프로토콜과 단서 후 임시 위탁(hinted handoff)
  - 영구적 장애 처리 : 머클 트리(merkle tree)
  - 데이터 센터 장애 대응 : 여러 데이터 센터에 걸친 데이터 다중화