🟨

The Log: What every software engineer should know about real-time data's unifying abstraction

1. Log란?

- 로그의 의미

  레코드가 시간별로 정렬된 일종의 테이블 혹은 파일

  → 시간 저장될 당시의 일과 시기를 기록하는 것 (물리적인 시계 X)

- 로그의 종류
  - application log : 응용 프로그램이 작동하면서 남기는 디버깅 메시지
  - data log : 데이터 사이언스에서 주로 수집되고 분석하는 로그

2. Transaction log

- 목적
  - 데이터를 일관된 상태로 관리하기 위해
  - 유지 관리하는 모든 데이터에 변경 사항을 적용하기 전에, 변경할 레코드에 대한 정보를 기록하기 위해

    → 데이터 변경 시 충돌이 발생할 경우 복원하는데 사용

3. 분산 데이터 시스템에서의 로그

- 목적

데이터 변경 순서를 정하거나 데이터 분산을 위해

- 역할

  여러 서비스들이 같은 process input을 받는다면, 결과도 같을 수 있도록 함

  → Deterministic algorithm

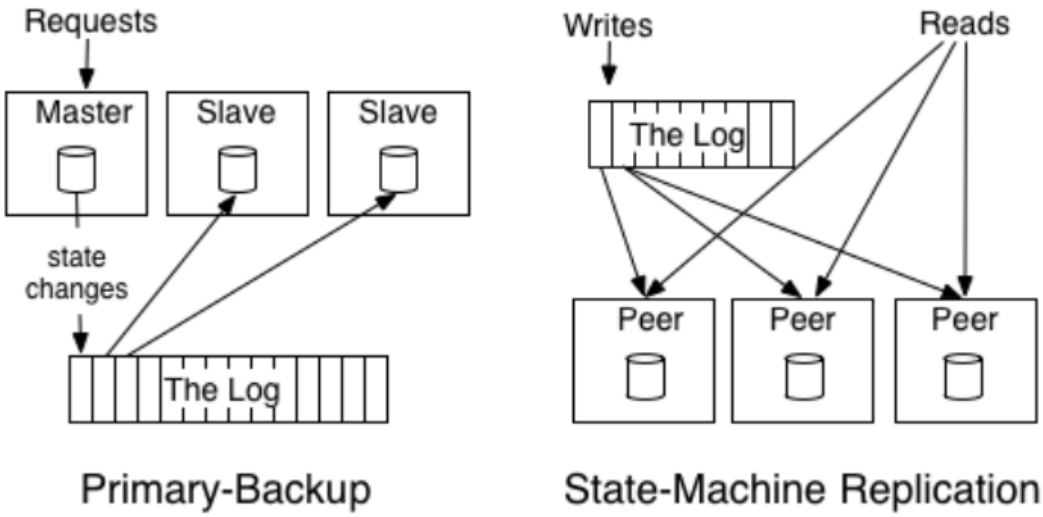
  → 타임 스탬프로서 인덱싱 역할을 함

- 접근 방식
  - `state machine model`

    incoming request와 각 request에 따른 replication process까지 로그를 갖는 모델

  - `primary-backup model`

    한 replica를 리더로 순서대로 process하고 이 change들을 log로 갖는 모델



3. 로그의 유용성

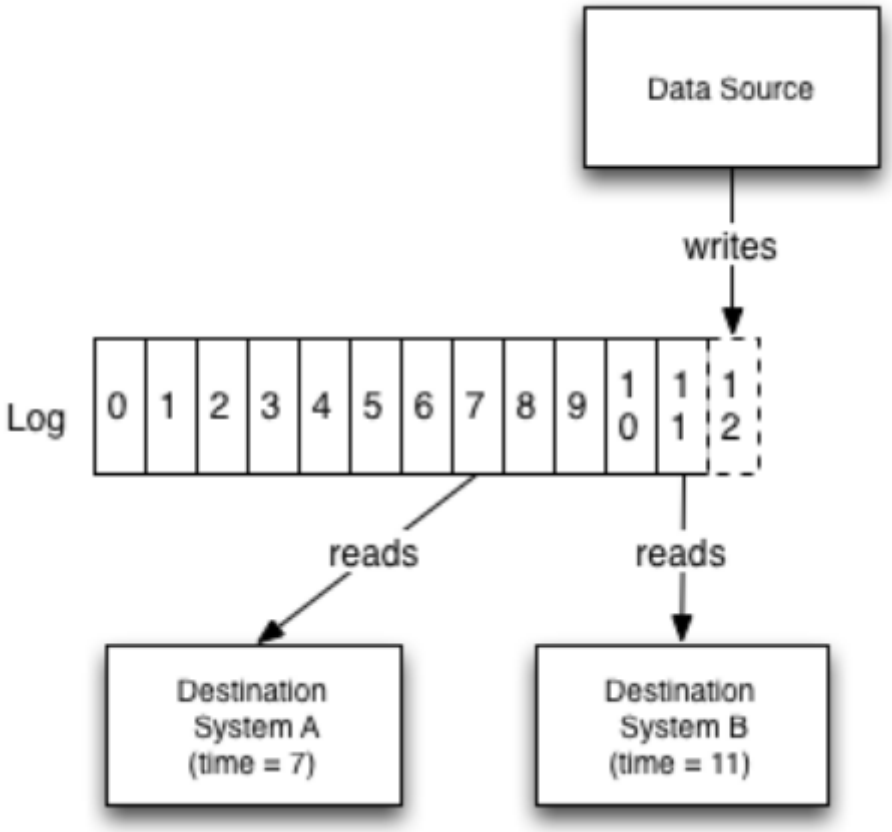
1. Data Integration
   - 모든 서비스와 시스템에서 조직의 모든 데이터가 사용될 수 있도록 하는 것

→ 저장된 데이터를 추출한 후, 요구 사항에 맞게 변형해 다른 곳에 사용할 수 있도록 전달

- 데이터 통합을 복잡하게 하는 요인
  - 이벤트 데이터 : 기존의 데이터베이스보다 수십 배로 더 크다
  - 특수 데이터 시스템 : 검색, 일괄처리 등

  → 이러한 상황에서 로그는 타임 스탬프 기준으로 인덱싱

  ⇒ 모든 데이터 소스들이 로그 파일로 정리되고, 그 다음에 데이터 시스템이 이 파일을 읽고 측정함으로써 데이터 양의 급격한 증가에도 손실 없이 소스 전달 가능

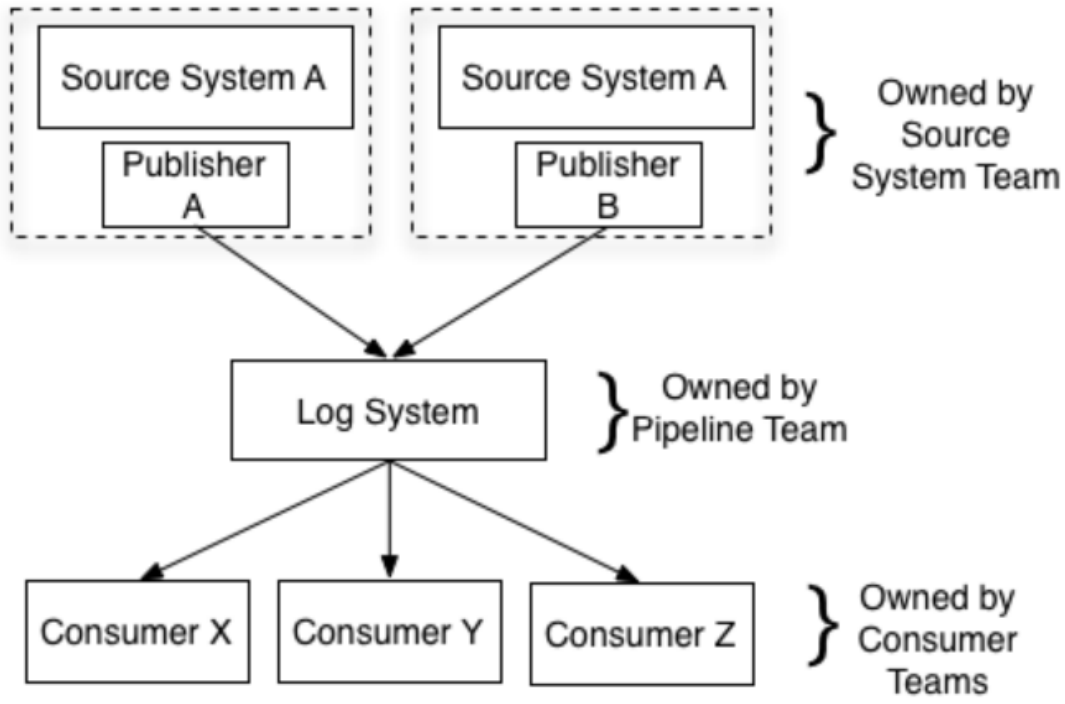


⇒ 각 시스템이 읽었을 당시의 시점이 남아있어 서로 다른 데이터 소스를 읽어도 추론 가능

- 같은 데이터라도 각 시스템과 파이프 라인에 따라 차이가 발생함

  → 정확한 데이터 활용 및 분석을 위해서는 모든 시스템에 동일하고 정확한 데이터 전달 필요

  → 전달하는 파이프라인 구축과 함께 단일 데이터 저장소가 필요했고, 로그가 그 역할을 함

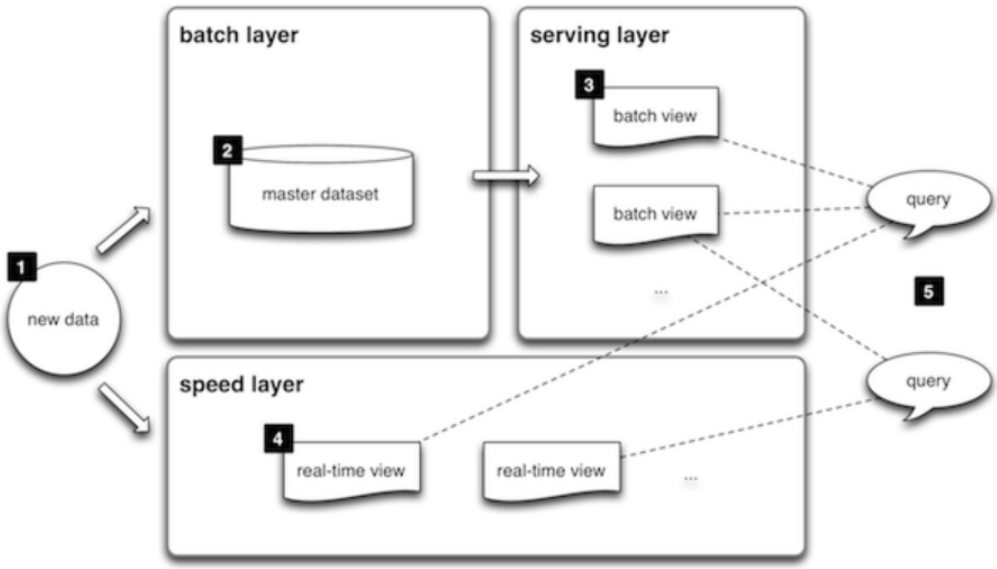


2. 실시간 데이터 처리
    - 스트림 처리
        - 지속적인 데이터 처리를 위해 필요한 인프라
        - RDS 데이터베이스를 기반으로 하는 데이터 배치 처리의 단점을 보완하는 처리 방식
        - 로그에서 읽고 로그 혹은 다른 시스템에 출력하는 모든 작업

          → 이렇게 통합된 로그를 활용하면 모든 데이터 변환 및 흐름을 파악할 수 있음
        - 로그의 역할
            - 각 데이터셋에 multi subscriber과 그 순서를 지정

              → 두 개 이상의 업데이트가 있을 때 정확한 output을 위한 역할
            - 스트림 처리 프로세스에 버퍼 역할
    - 람다 아키텍쳐
        - 로그지향적 데이터 모델링의 한 부류

- 대량의 데이터를 batch로 미리 만든 데이터와 실시간 데이터를 혼합해서 사용



- 결과 데이터를 만들어내는 코드가 수정되면 결과 데이터를 다시 산출 (재처리)

  → 코드 수정에 따른 영향도 분석

- 단점
  - 배치 시스템과 실시간 처리 두 곳에서 같은 결과를 산출

    → 배치 뷰 데이터와 실시간 뷰의 데이터가 중복되지 않도록 관리하는 것이 중요

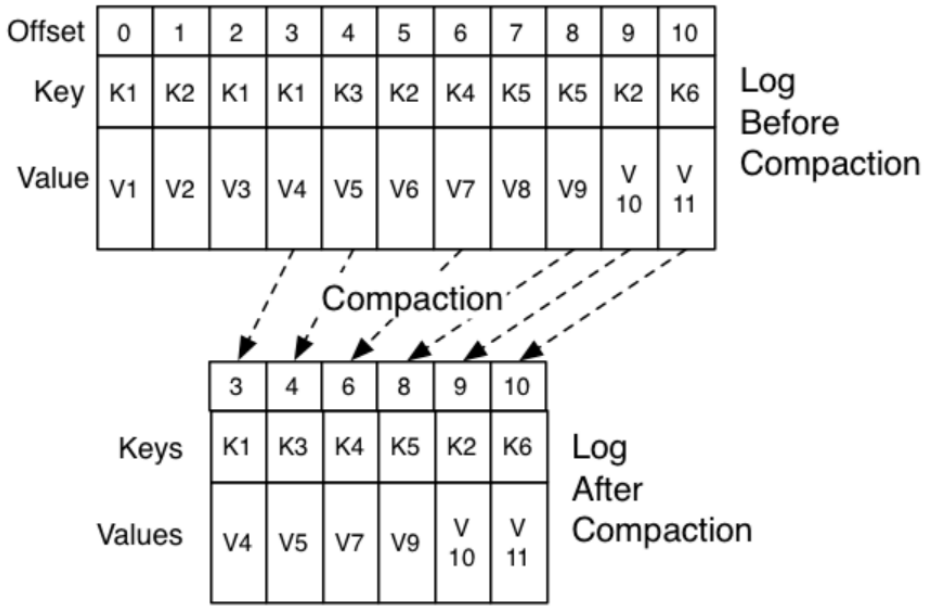
  ⇒ 대안

  스트림 처리 시스템에 서로 다른 두 시스템을 붙이는 것이 아니라, 스트림 처리 시스템에서 실시간 처리를 하면서 재처리하는 방식으로 병렬성을 늘리는 방안

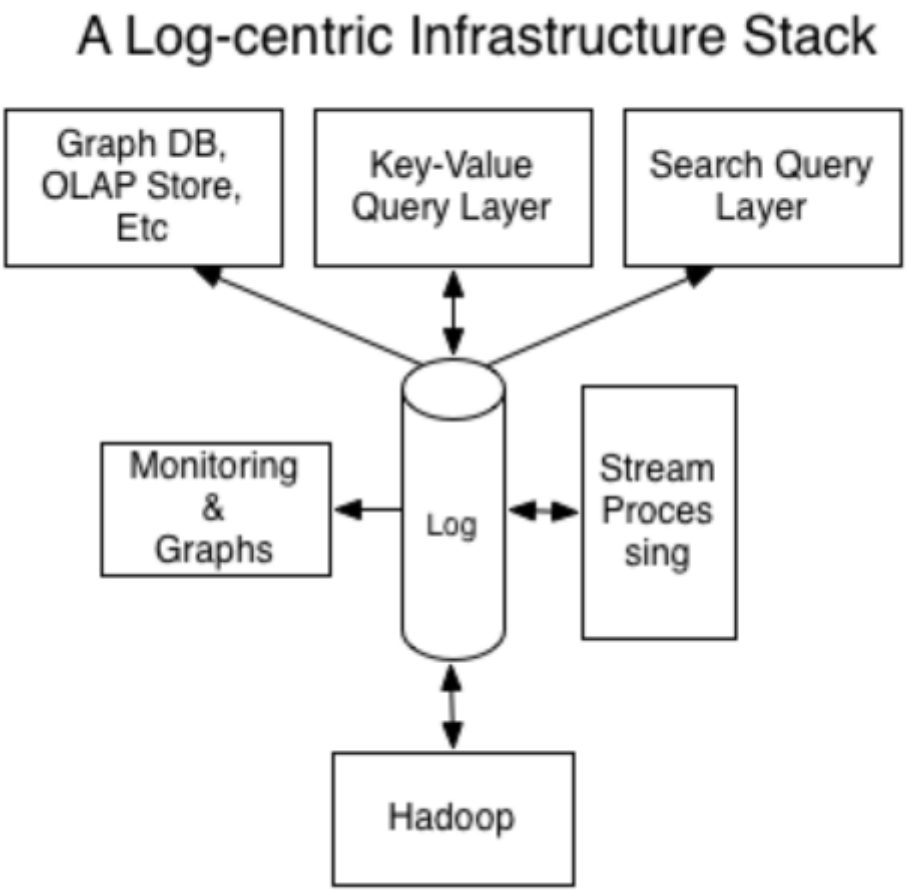
- 상태적 실시간 처리
  - 재처리 뿐만 아니라 상태 저장 또한 중요
  - 스트림 처리를 진행중인 프로세서에 상태 저장해야함
  - 로컬 테이블이나 인덱스에 입력스트림으로 저장된 후, 상태 복원용으로 로컬 인덱스의 변경 로그를 활용할 수 있음

- 로그 압축 (log compaction)

- 공간적 한계로 인해 수행
- 카프카의 경우
  - 데이터를 순수 이벤트 / 키별 업데이트 데이터인지 구분
  - 순수 이벤트 데이터 : 지정한 일자 동안 데이터 유지
  - 키별 업데이트 데이터 : 어떤 키에 의해 식별 가능한 엔티티의 상태 변화를 구체적으로 기록한 것 → 저장 공간이 많이 필요해 최신 업데이트한 레코드가 있다면 그 직전 레코드 삭제



- 분산 시스템 설계
  - 노드에 대한 동시 업데이트를 시퀀싱해 데이터 일관성 처리
  - 노드 간 데이터 복제 제공
  - 노드 간 데이터 재조정
  - 데이터가 손실된 복제본을 복구하거나 새 복제본을 부트스트랩하는 기능 제공


A Log-centric Infrastructure Stack
Graph DB, OLAP Store, Etc
Key-Value Query Layer
Search Query Layer
Monitoring & Graphs
Log
Stream Proces sing
Hadoop