⊙ 벌크형과 스트리밍 형의 data 수집.

빅데이터의 대부분은 확장성이 높은 분산 storage에 저장.

기본은 (객체 storage이다) — HDFS (Hadoop)
S3 (Amazon)

⊙ 객체 storage는 다수의 컴퓨터를 사용하여 파일을 여러디스크에 복사함.

ㄴ 중복화 , 부하분산.

⊙ 데이터수집.

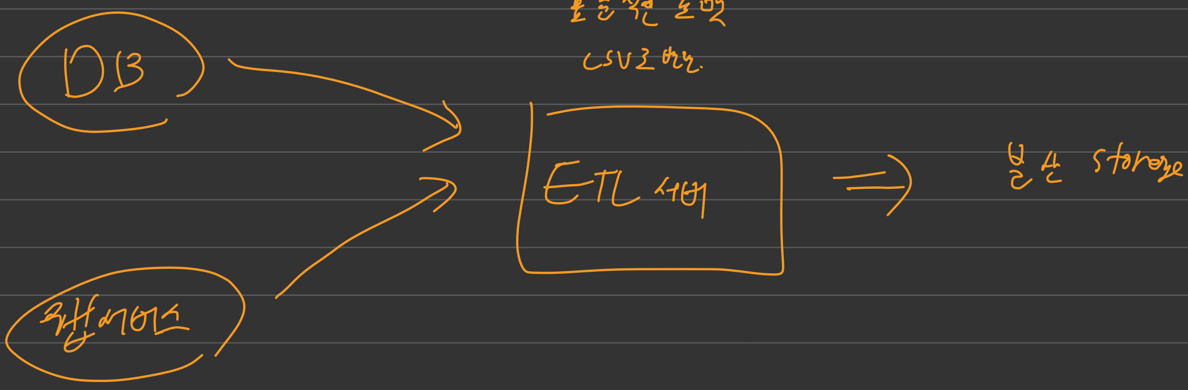
ㄴ time series 데이터, 작은 데이터는 축적하여 모아서 하나의 큰 파일로 만듦.

ㄴ 너무 큰 data는 적당히 나눈다.

객체 storage에서 효율적으로 처리할 수 있는 파일크기는
대략 1MB ~ 1GB

⊙ 벌크형 data 전송.

표준적인 포맷
csv로 변환.



① 스트리밍 쪽의 Data 전송

└ 네트워크 성능이 중요!!!

└ 수많은 Client에게 짧은 시간에 짧은 Data가 전송. (Message Delivery)

(높은 처리성능 ?

→ 2가지방법 ① NoSQL

② 메시지큐, 메시지브로커들의 플랫폼들이 존재.

1) 웹브라우저 에서의 Message Delivery

a) Fluentd, Logstash, 웹서버에서 전송.

b) Javascript 사용, Web event tracking 스크립트, event 수집처리에서 전송.

2) 모바일 앱으로부터 delivery.

└ 웹브라우저와 동일.

└ 서버를 직접 만든 라이브러리, MBaaS 나 플랫폼의 종속 서비스를 이용.

3) 디바이스로부터의 메시지 처리.

└ MQTT (TCP/IP) 를 사용해 데이터를 전송하는 Protocol

① 성능 × 신뢰성 메시지 배송의 trade off.

메시지 배송에 의해 보내진 데이터를 분산스토리지에 저장한 때로 들어가 좋다.

대량의 메시지를 안정적으로 받기 위해서는 빈번한 쓰기에도 견딜 수 있도록
성능이 매우 높고 필요에 따라 성능을 얼마든지 높일 수 있는 storage가
필요하다.

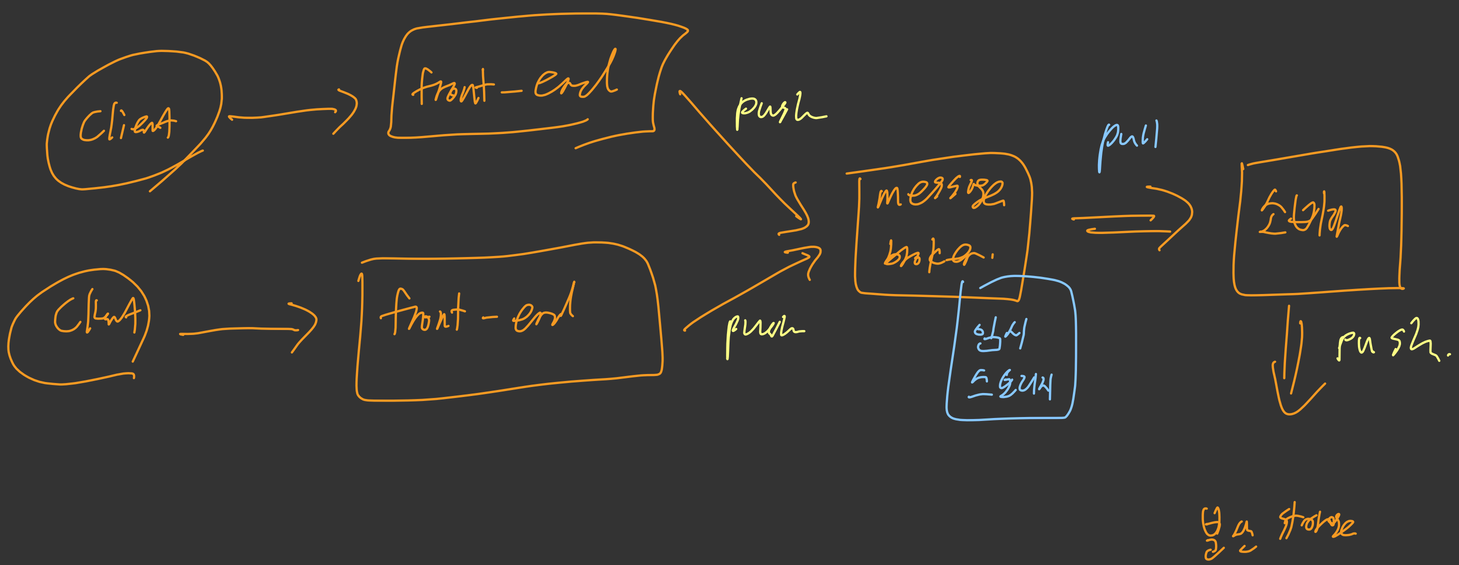
\# 빅데이터에서는 종종 Data를 일시적으로 축적하는 중산층 가 설치.
(Message Broker)

빅데이터를 위한 분산형 메시지 브로커는 오픈소스의 경우
Apache Kafka, Amazon Kinesis 등 유명하다.

② Push vs Pull

Push	Pull
송신측의 제어로 데이터를 보내는 방식	수신측 주도로 데이터를 가져온다.

② 풀 방식의 메시지 배송.



메시지 브로커는 높은 빈도로 데이터를 쓰는 것에 최적화되어 있고, 여러대의 노드에 부하분산 함으로써 성능을 끌어올릴 수 있는 뛰어난 확장성을 가지고 있다.

└> push 형의 메시지 배송은 모두 메시지 브로커에 집중시키고, 일정한 빈도로 꺼낸 데이터를 분산 스토리지에 기록하여 성능문제를 피한다.

pull 형은 메시지 파일사이즈 적정화에도 도움이 된다.

④ 메시지 라우팅 - 메시지 브로커에 써넣은 데이터는 복수의 다른 소비자에서 읽어들일 수 있다. 이를 통해 메시지가 복사되어 여러경로로 분기.

신뢰성 문제 ... 3가지 설계 방식.

- 모바일 회선과 같은 신뢰성이 낮은 네트워크 에서는 반드시 중복, 누락이 발생한다.

1) at most once → 메시지는 한번만 전송, 실패해서 사라질 가능성이 있음

2) exactly once → 메시지는 손실되거나 중복 없이 1번만 전달한다.

3) at least once → 메시지는 확실히 전달된다. 여러번 전달될 가능성이 있다.

중복 제거는 높은 비용의 오퍼레이션.

ㄴ tcp에서는 메시지에 시퀀스 번호를 붙이나, 분산 시스템에서는 시퀀스번호 그닥... 소용 X

모든 메시지에 일련번호 넣기는 힘들, 성능 향상 제한됨...

→ 이를 해결하기 위해!?

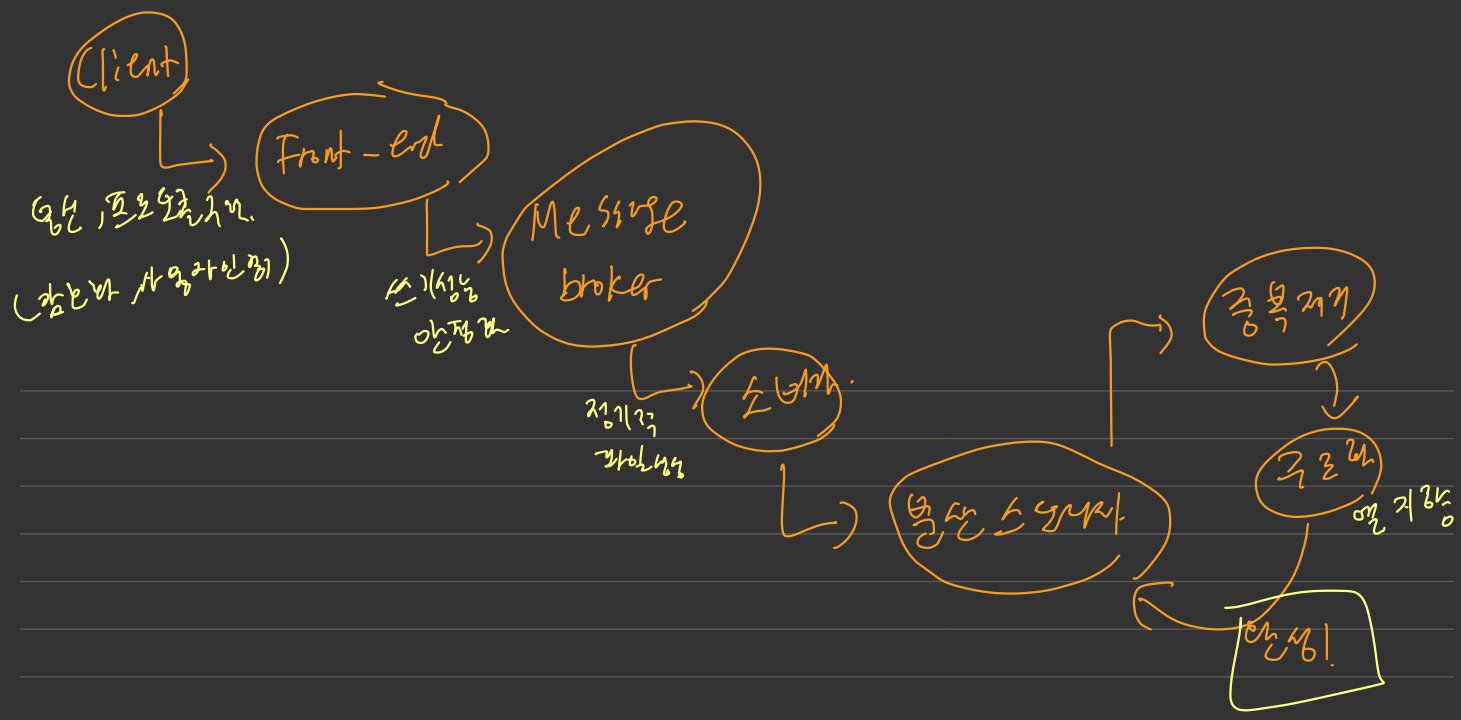
ㄴ 1) 오프셋을 이용한 중복 제거. (bulk형에 적합)
  - 전송 과정에 파일명을 보내며, 덮어쓴다.

2) 고유 ID에 의한 중복 제거
  - 최근에 받은 ID 남을가짐, 그 보다 늦게 온 메시지의 중복은 처리.

3) 종단간의 신뢰성 (End to End)

- 종단간의 보장 at least once로 하고, 정확한 곳에서 중복제거

① 데이터 수집의 파이프라인 (깨끗한 데이터 빨리 적합한 장소로)



@ 시계열 데이터의 최적화

ㄴ 스트리밍 형식 메시지 배송에는 "도착 시간 지연"이 문제다.

- 이벤트 시간 - 클라이언트 상에서 메시지가 생성된 시간
- 프로세스 시간 - 서버가 처리하는 시간

데이터 분석의 대상이 되는 시간

@ 프로세스 시간에 의한 분할과 문제점.

분산 스토리지에 데이터를 넣는 단계에서는 이벤트 시간이 아니라, "프로세스 시간"으로 사용하는 것이 보통이다.

ex) 2017년 1월 1일에 도착한 데이터 안에, 다수의 과거 데이터가 포함되어 있다. (이벤트 기준)

→ 이 경우 매우 많은 파일을 열어야 원하는 이벤트를 찾을 수 있다

◦ 다수의 파일을 모두 검색하는 쿼리는 풀 스캔 (full scan). 시스템의 부하를 크게 높이는 요인이 된다.

◦ 시계열 인덱스 — 이벤트 시간에 의한 집계의 효율화

ㄴ RDB 에서 index를 만드는 것과 마찬가지로, 이벤트 시간에 인덱스를 만든다.

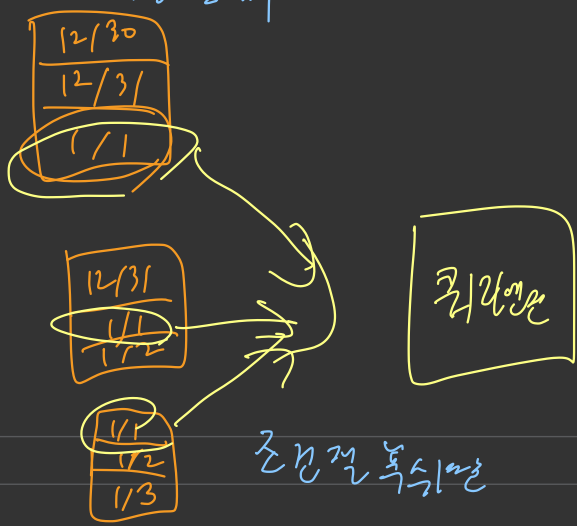
ㄴ 시계열 index에 대응하는 카산드라 분산 DB를 이용하면, 시간으로 인덱스된 table 생성 가능.

ㄴ 장기적으로는 좋지않음.

◦ 조건절 푸시다운 — 이벤트 시간에 의한 집계의 효율화

ㄴ 이벤트 발생 시간으로 정렬 하여 저장.

예) 열 지향 스토리지

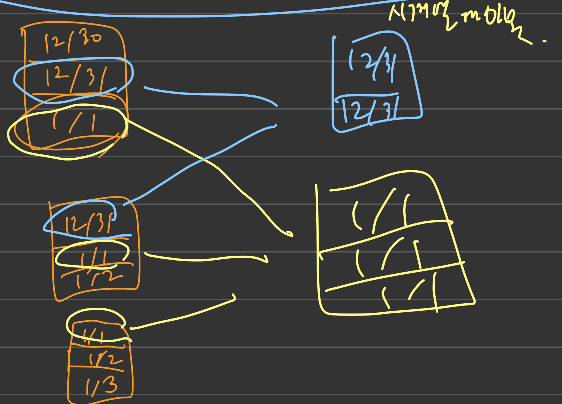


조건절 푸시다운

데이터가 충분히 연속적으로 배치되어 있어야 최적화 효과가 ↑

ㄴ→ 짧은 주기로 만들면 X (데이터가 적으면 많은 load가 발생)

◦ 이벤트 시간에 의한 분할 — 테이블 파티셔닝, 시계열 테이블



몇 년에 걸쳐서 data를 보내올 가능성...

◦ 데이터마트는 이벤트 시간으로 정렬하기

ㄴ 데이터마트 만드는 단계에서는 활용하도록 한다.

③ 비구조화 데이터의 분산 스토리지

ㄴ NoSQL 사용시, 데이터는 단순히 보관해 저장뿐 아니라, application으로 온라인 이용, 실시간 접근가능.

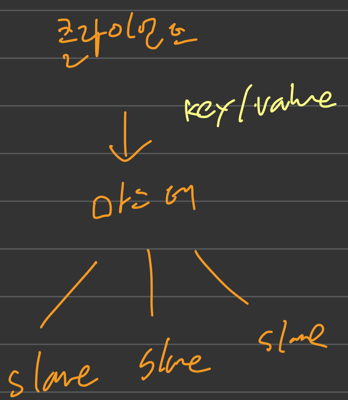
작성에 유리.(write)

◦ 분산 KVS – 디스크로의 쓰기 성능을 높이기.

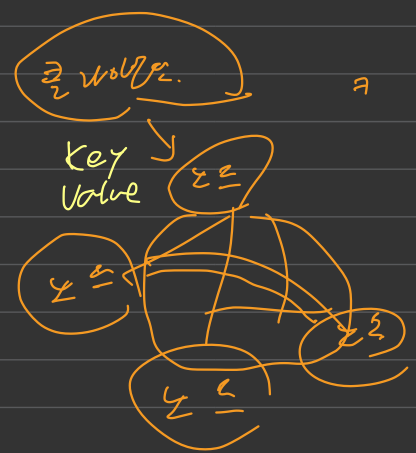
ㄴ distributed key value store

ㄴ 모든 데이터를 키-value 쌍으로 저장하도록 설계된 DB.

① Master/slave



② P2P형



- Amazon Dynamo DB
  - 고확장 안정적 암기쓰기 성능을 제공하도록 디자인된 분산형 NoSQL DB.
  - P2P 방.

- 테이블 컬럼 스토어 - 구조화 데이터를 분석 목적으로 저장,
  - 일반 KVS를 확장시켜 고객이의 성능의 키에 데이터를 저장할 수 있도록 한 것
    - Google Bigtable
    - Apache HBase
    - Apache Cassandra.
  - Row의 column 명과, value 를 같이 저장한다.

- Apache Cassandra.
  - 테이블 컬럼 스토어, (RDB)
    스키마는 정의해 둘 수도 있으나,
    insert into (추가할) 으로 동적.
    동일키 레코드는 덮어쓰기
    P2P 형.

- 도큐먼트 스토어 - 스키마리스 데이터 관리하기.
  - 주로 데이터베이 유형을 뜻함.

JSON 등의 복잡하게 처리한 스키마리스 데이터를 그대로의 형태로 저장하고 쿼리를 실행할 수 있도록 함.

검색엔진 - NoSQL라는 성격이 조금 있음.

- text, schemaless data 집계하는데 자주 사용

풀스캔에 의한 전문검색. (역색인) inverted index

Elastic Search.

- 오픈 소스 검색엔진.
- 임의의 JSON 데이터 저장 가능.

text data는 역색인이 중요.

Splunk