분산환경으로 갔을 때 challenge

- 단일 환경과는 다르게 어디서 어떻게 문제가 생기는 경우가 다양함
- 분산환경에서 터지는 문제들 여러가지를 고려해야함 (네트워크) - Network transfer rates
  - 랙 단위로 문제 생길 수 있음
  - Network transfer rates are slower than memory access - bottleneck 발생
- 노드가 여러개다 보니까 일부는 동작이 안될 수 도 있음 - Node Failure
  - 노드 오류 가능성 높아짐
- 로드밸런싱 정책, reprecation (partial failure) 에 대한 고려가 시스템안에서 해결이 되어야한다.

이러한 솔류션 - 하둡 에코시스템 (사용자가 쉽게 핸들할 수 있게끔 관리할 수 있게 해준다.) = 새로운 프로그램 패러다임이 필요함

데이터의 전처리 과정이 중요하다 **

- data가 비어있거나 서로 다른경우, 중복되는 경우 (당연함)
- Cleansing , preprocessing이 매우 종요하다.
- large data set은 사람이 일일이 확인 할 수 없다.
  - 알고리즘이 필요
- Feature engineering 이라고 함 (의미 없는 피처들을 제거하거나 그중에 의미 있는 피처만 selection , 피처들을 새로 조합해 새로운 피처를 만들거나)
- ex : PCA (principal computational analysis)

Iteration **

- 똑같은 data를 multiple passes
- disk io delay로 인한 전체적인 process가 느려짐

어떤 데이터를 풀 것인가(문제정의) **

- 문제가 있을 때 어떠한 데이터를 쓸 것인가 (데이터를 의미있게 만드는 것)
- 연구실과 실무에서의 차이 :
  - 연구 : 데이터를 이해하는 정도, 이론 테스트, 실험 (빠른 prototype)
  - 실무 : package models, improve accuracy, care about SLA(service-level agreement), uptime
- production system에 잘 fitting

아파치 스파크

- distributing programs across clusters of machines

아파치 스파크와 아파치 하둡의 유사점

- Maintain MapReduce's linear scalability and fault tolerance

스파크	하둡
directed acyclic graph(방향성이 있는 그래프 파이프라인) not sequential	MapReduce 사용 : sequential
In memory procecssing (한번 올려놓으면 다시 내리지 않고 사용 가능) 속도 빠름, Iteration 관점에서 굉장히 장점	Intermidterm 결과값을 매번 disk에 저장해야함 (disk io delay 야기) 메모리에 데이터 올렸다 내렸다 하는것 때문에 속도 느림 -> Iteration에서 bottle neck 야기
코딩 편함(Streamlined API 제공)	상대적으로 어려움

아파치 하둡

- 데이터를 scalable하게 저장(linear scalability ; 거의 근접함, 데이터가 늘어나면 컴퓨터가 비례해서 늘어나면 시간은 유사), economical - 구글에서 주장하는 대표적인 정책
    - comodity pc 여러대를 사용 (가격적으로 훨씬 저렴함 - economical, 슈퍼컴퓨터처럼 고성능의 효율을 낼 수 있게끔 분산환경 고려)
    - industry standard hardware
    - Break up work into small tasks

data bottleneck이 어디서 발생하는가

- 과거처럼 한군데서의 processing bottleneck을 처리한다고 해서 data를 잘 처리할 수 있는 것이 아니다.
- 기존의 processing bottleneck을 해결하는 것과는 굉장히 다른 양상

data 수집 소스 : database (log data, event data)

HDFS - 하둡 파일 시스템

HBase - 구글 빅테이블의 클론 형태

Impala

Floom

Hadoop MapReduce

- 하둡은 여러 형태로 사용됨 (밑 두가지로 나눠짐)
    - 하둡 에코시스템

    - 하둡 맵리듀스
        - 하둡 에코시스템 컴포넌트중에 하나 (스파크와 유사)
        - 스파크만큼 사용되지는 않음

Apache pig

- hdfs 기반으로 hbase가 있고 그위에 스파크
- 좀더 처리하기 위한 스크립트를 만들어놓은 것 - 아파치 피그
- pig와 hive의 차이점

Pig vs Hive - Differences

Pig	Hive
Procedural Data Flow Language	Declarative SQLish Language
For Programming	For creating reports
Mainly used by Researchers and Programmers	Mainly used by Data Analysts
Operates on the client side of a cluster.	Operates on the server side of a cluster.
Does not have a dedicated metadata database.	Makes use of exact variation of dedicated SQL DDL language by defining tables beforehand.
Pig is SQL like but varies to a great extent.	Directly leverages SQL and is easy to learn for database experts.
Pig supports Avro file format.	Hive does not support it.

Hue

- 브라우저형태의 ui로 표현되는 툴
- 원인분석이 안되면 user interface상에서는 raw data 확인이 불가능함,

우지

- Data가 들어왔을 때 맵리듀스를 통해서 어떤 형태로 변환이되고 어떤 인풋으로 들어가서 아웃풋으로 나오는 일련의 과정을 다이어그램으로 표현해줌

에코시스템

- 분산환경은 매우 다양한 프로세스가 복잡하게 이루어져있는데 유저가 편하게 이용할 수 있도록 컴포넌트를 제공함