빅데이터 마스터과정
(DAM)

하석재
CEO, 2HCUBE
sjha72@gmail.com

이 수업에서 다루는 기술

- 하둡
  - HDFS/MapReduce
  - Flume/Sqoop/Hive
- 스파크
  - Spark RDD
  - Spark SQL
  - Spark Streaming(mini-batch, structured)
  - Spark MLlib(Machine learning)
- 카프카
  - Real-time streaming

빅데이터 프로세스(NCS)

- 빅데이터 프로세스(오픈소스), 수업대상(1-3)

1. 빅데이터 수집
   Flume, Sqoop, Fluentd, logstash, ...
2. 빅데이터 저장(HDFS)
3. 빅데이터 처리(쿼리)
   MapReduce, **Spark RDD/SQL**
4. 빅데이터 분석
   엑셀, 파이썬 데이터사이언스, R, Tensorflow, **Spark MLlib**, ...
   cf. R-Hadoop, Tensorflow-HDFS(Hadoop)
5. 빅데이터 시각화(모니터링)
   Prometheus/Grafana, Kibana, ...

빅데이터 트렌드

1. 하둡 HDFS+MapReduce -> 스파크 RDD -> 스파크 SQL(DataFrame/DataSet)
   SQL화

2. Batch(배치) -> Streaming(스트리밍/mini-batch) -> “Real-time” Streaming(실시간 스트리밍)
   Kafka

3. 하이브리드(Eco-system)
   Spark + Kafka => 실시간 스트리밍(구조화, 비구조화)

빅데이터 vs. RDBMS(VLDB)

- DBMS(OLTP vs OLAP vs 그 외)
  - On-line Tranaction Processing / On-line Analysis Processing
  - Data mining / DW(Data warehousing)
  - cf. datalake

- VLDB(OLTP) vs. BigData
  - 샤딩+복제
  - 저장용량/처리성능(read/write) 향상

- 차이
  - VLDB(DBMS 차원) - 복제(Replication) / 샤딩(sharding)
  - 하둡 - HDFS(파일시스템차원)

빅데이터기술(하둡,스파크,카프카)의 경쟁자

1. OLAP-Data Mining/Data warehousing(DW)

2. Splunk

3. Elastic Stack = Logstash + Elasticsearch + Kibana + ...

    cf. ES-Hadoop = Elasticsearch + HDFS

CDH/HDP

- 공식 배포판
  - Apache Hadoop, Apache Spark, Apache Kakfa
  - 에코시스템 Flume, Sqoop, Zookeeper, Parque, ...
  - 버전의존성(스파크-하둡 버전 2.7/3.2이상) 문제

- CDH(클라우데라 배포판)
  - 충돌문제 해결 + 간편
  - 최신기능 지원에 시간걸림
    - CDH6(Hadoop 3.0) vs. Hadoop 3.3
    - CDH6(Spark 2.2) vs. Spark 3.0.1
  - 무겁다

CDP

- https://kr.cloudera.com/products/cloudera-data-platform.html
- Cloudera Data Platform
  - CDH 와 HDP의 통합
- Public Cloud / Private Cloud 버전
- 최신버전 7.2.2
  - 유료화 / 60일 데모버전 배포
  - 무료버전인 CDP express 제공(6.3까지)

“BigData”의 의미

- 무엇이 “빅데이터”인가?
  - 현재는 상업적/**마케팅**적인 용어로 변화
  - 일단 큰 데이터를 대상으로 하면 다 빅데이터?
- **“크기”**와 **“처리방식”**을 둘 다 만족해야 함
  - 좁은 의미로는 HDFS와 같은 분산파일시스템과 MapReduce와 같은 병렬처리 프레임워크를 가지고 있어야 빅데이터 처리 시스템이라고 말할 수 있슴
    - 대표적으로 하둡(Hadoop)이 있슴

“Data”의 의미

- 대표적인 데이터를 다루는 기술 : DBMS(주로 RDBMS)
- 데이터를 저장(생성)하고 검색/수정/삭제
- CRUD(Create, Retrieve, Update, Delete)
    - 주로 테이블/레코드 형태로 관리(관계형)

RDBMS와 **NoSQL**과의 비교

- 보통 **일반적인 DBMS**는 **빠른 읽기**에 최적화(R >> CUD)
  - CUD가 일어나면 인덱스(Index) 수정*
- 데이터가 빈번히 생성/수정/삭제되는 시스템에 적합하지 않음
  - **NoSQL**의 등장 -> **빠른 쓰기**에 최적화
- 하지만 양쪽 다 데이터가 커지는 경우에는 특수한 처리 필요
  - 수 십 테라가 넘어가는 경우
- 빅데이터는 일반적인 **파일로 관리**(주로 **키/밸류** 방식)

"Big"의 의미 - 크기

- 일반 **DBMS**(주로 RDBMS)
  - "Large" 데이터를 다룸
  - 인덱스로 검색속도 해결
    - 인덱스를 달아도 느려진다면?

- **"Very Large" DBMS**(VLDB)
  - 사용되는 기술이 다름
  - 파티셔닝(DB안), 샤딩(DB밖), 복제(이중화)

“Big”의 의미 - 크기

- **"Big"** Data(데이터)
  - 일반 DBMS로 처리하려면 엄청난 비용
  - 처리가능하지만 여러 가지 제약조건

    * 외래키 제약/조인-정규화 문제

  - 일반적인 DBMS로 처리하지 못할 정도로 큰 양
    - 통상 수 십 테라 이상의 크기를 가지는 경우

“Big”의 의미 - 크기

- 정의 1.
  서버 한 대로 처리할 수 없는 규모의 데이터
  ex. 10TB 소팅 in 1 서버
- 정의 2.
  기존 소프트웨어로는 처리할 수 없는 규모의 데이터
  Scale Up vs. Scale Out
- 정의 3.
  3V(Volume, Velocity, Variety) -> 4V -> 5V
  cf. 20TB in 액센츄어(->30TB->40TB)

"Big"의 의미 - 처리방식

- 큰 데이터를 어떻게 RDBMS와 다른 방식으로 처리할 수 있는가?
  - **스케일러블(Scalable)한 방식으로 처리**
    - 시스템 수를 추가하면 계속 처리용량이 커지는 방식
  - **스케일 아웃(Scale Out)**
    - 하둡의 경우 (하둡v1) 4,000대,
      (하둡v2) 10,000대까지 연결가능
    - 처리가능한 데이터량 수 십-수 백 PB까지

  - 시스템 추가 처리는?
    - 간단한 설정으로 가능
    - (masters/slaves 파일에 주소 등록)

“Big”의 의미 - 처리방식

- 데이터를 나누어 저장하면서 생기는 문제
  - 여러 개의 시스템 중 어디에 원하는 내용이 있나
  - 일부 시스템/네트워크에 장애가 일어난다면?
- 해결책
  - **분산파일시스템**을 만들자
    - 여러 대의 시스템을 묶는 큰 파일시스템
    - **고가용성(HA:High Availability)**를 제공
      - 동일한 정보를 여러 군데에 중복해서 저장
      - 중복성/다중화(Redundancy)
  - **병렬처리방식**으로 처리성능을 높이자
    - 작업을 나눠 동시에 처리하는 방식
    - 분업화

아파치 하둡
(Apache Hadoop)

하둡의 역사

- **더그 커팅** : **하둡**의 아버지, **루씬** 제작자(검색엔진용 오픈소스 텍스트 인덱스 엔진)
- 구글 검색엔진과 같은 대형 검색엔진 제작에 관심
- 데이터를 대량으로 저장할 수 있는 빅파일시스템과 분산처리구조에 관심을 가지고 있었슴
- 구글의 두 가지 논문에 영감을 얻어 하둡을 제작
  - **The Google File System(2003)**
  - **MapReduce : Simplified Data Processing on Large Cluster(2004)**
- 2006년 부터 제작(Apache Top Level Project)
- 야후에 취직-> 클라우데라로 이직
- **GFS-> HDFS, MapReduce-> MapReduce**
- 하둡의 이름은 더그커팅의 아들이 가지고 놀던 노란색 코끼리 인형!
  - 하둡의 기술의 이름은 코끼리와 관련있는 이름을 지음

http://www.nytimes.com/2009/03/1

Hadoop/Mahout/Oozie/Horton

머하웃(Mahout) 코끼리를 모는 사람

하둡 아키텍처

- **하둡(Hadoop) = HDFS(저장) + MapReduce(처리)**

- 빅파일 시스템: **HDFS(Hadoop File System)**
  - 네임노드(마스터)/데이터노드(슬레이브)
  - 세컨더리 네임노드
    - SPOF(Single Point Of Failure)문제

- 분산처리 프레임워크 : **MapReduce**
  - 잡트래커(마스터)/태스크트래커(슬레이브)

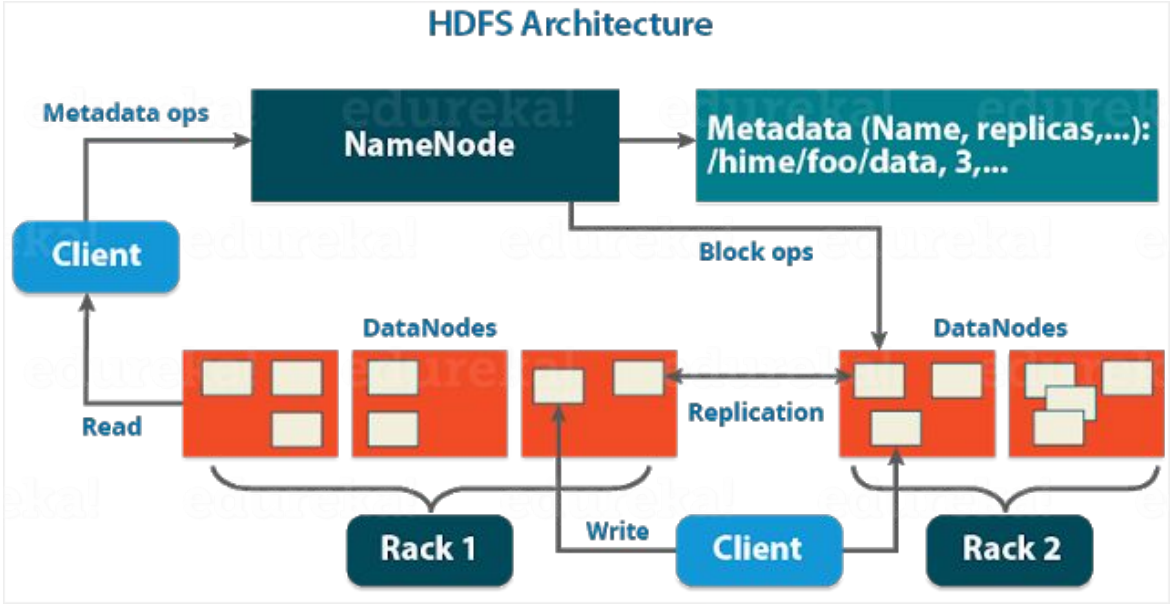
HDFS

- 하나의 HDFS 에 **하나의 네임스페이스** 제공
- 파일을 여러 개의 **블럭**으로 나누어 저장한다
  - 블럭사이즈: 64MB(실제로는 128MB 많이 사용)
- 큰 파일을 다루는 데 적합

- 운영체제의 파일 시스템을 그대로 사용한다
  - ext4fs(linux)
- **복제수(Replication Factor)**
  - 여러 군데에 같은 블럭을 복사**(HA:High Availability)**

- Write Once Read Many
- File Overwrite not Append

하둡 아키텍처(HDFS)

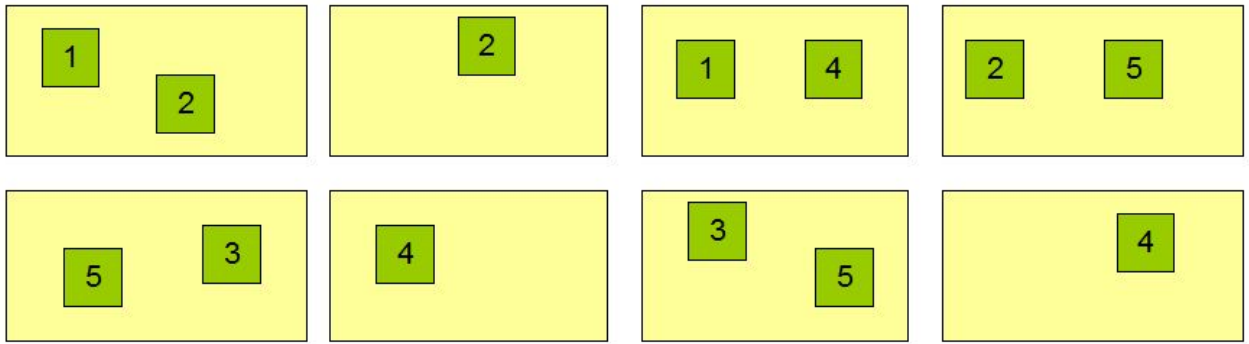

HDFS Architecture
Metadata ops
NameNode
Metadata (Name, replicas,...):
/hime/foo/data, 3,...
Client
Block ops
DataNodes
DataNodes
Read
Replication
Rack 1
Write
Client
Rack 2

하둡 아키텍처(HDFS)

Block Replication

Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …

Datanodes



https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

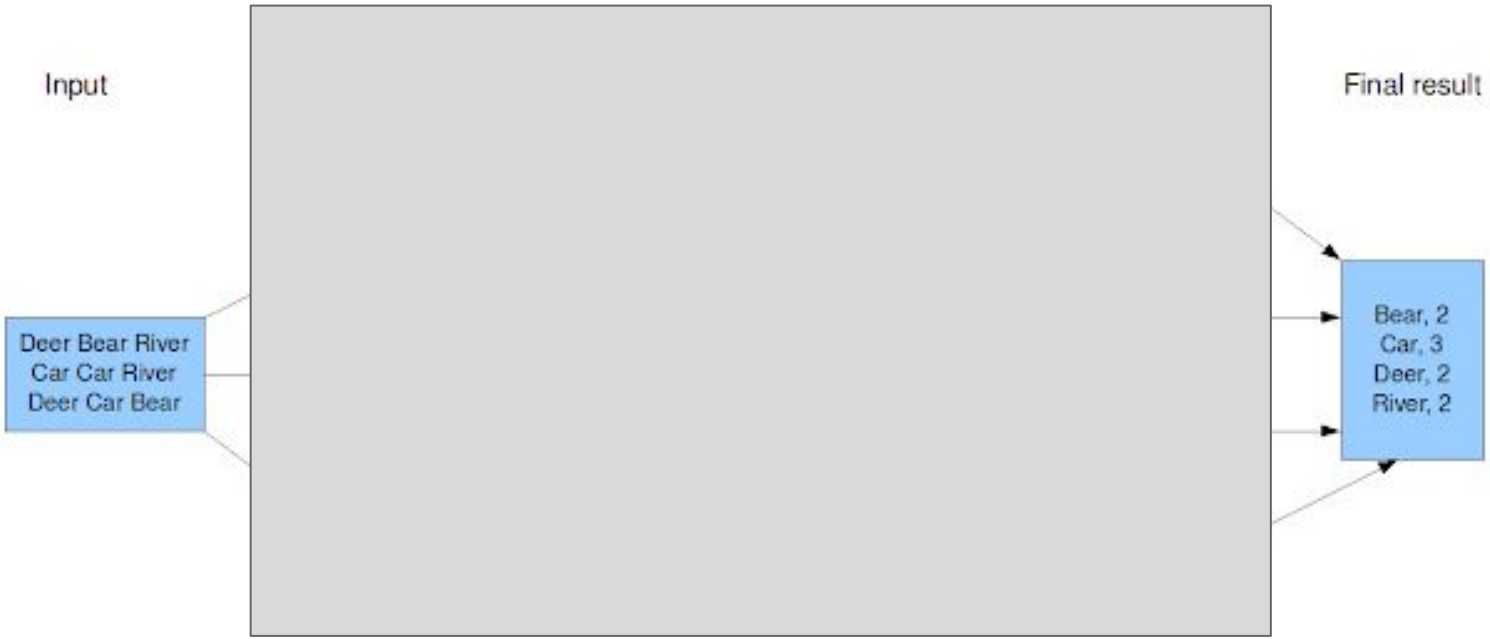
HDFS 파일시스템의 특징

1. 윈도우 파일시스템 드라이브마다 루트가 있슴(C:\, D:\, E:\, L:\)

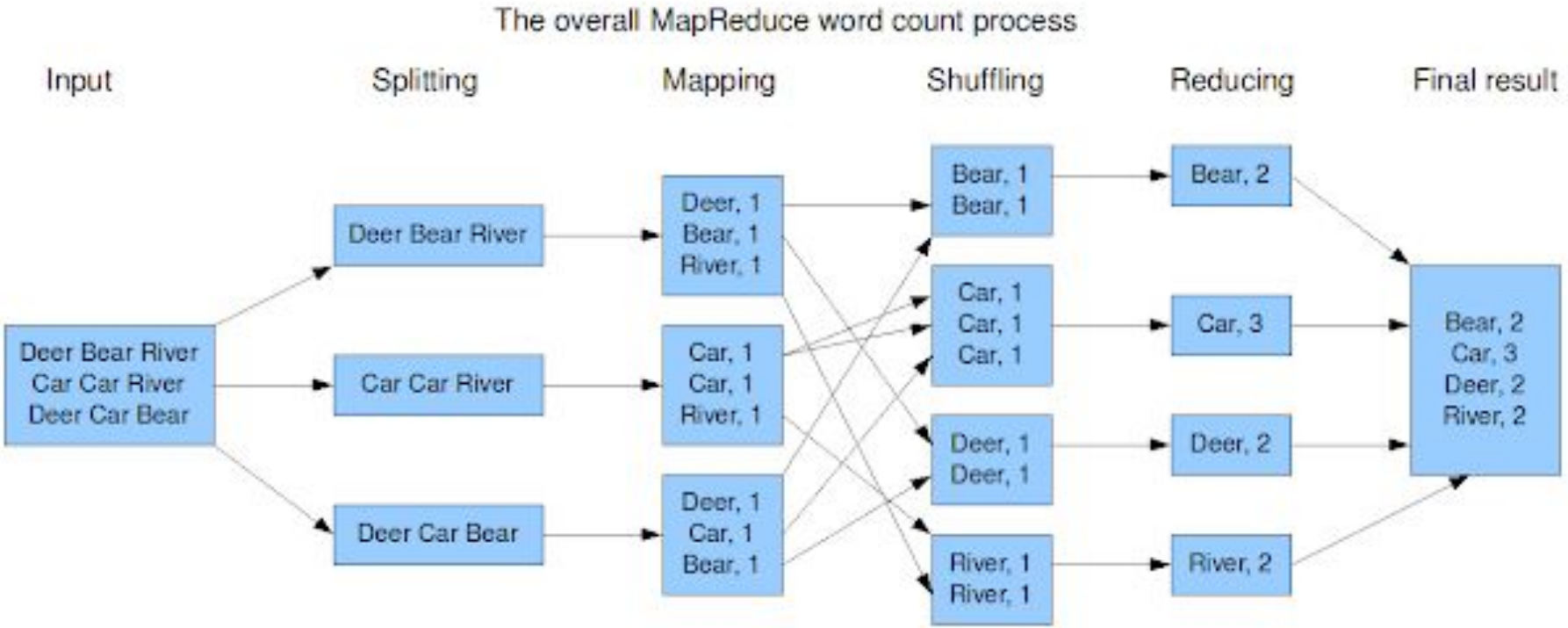
2. 리눅스 파일시스템 전체 시스템에 루트가 하나(/)
   /dev/hda1, /dev/hdb2, /dev/cdrom -> 마운트기반

3. 하둡 HDFS
   여러 대의 시스템을 묶어서 하나의 논리파일시스템을 구성
   전체 루트는 하나
   /a -> 1번 컴퓨터, /b -> 2번 컴퓨터
   파일을 조각을 내서 분산저장 -> 보기에는 하나로 표현
   네트워크와 시스템 장애에 대응 -> 데이터블럭으로 여러 개 복제
   읽기/쓰기 속도까지 향상

하둡 아키텍처(맵리듀스)


Input
Final result
Deer Bear River
Car Car River
Deer Car Bear
Bear, 2
Car, 3
Deer, 2
River, 2

하둡 아키텍처(맵리듀스)


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Hadoop 1의 약점

- **배치(Batch)에 최적화**
  - 스트리밍과 같은 작업은?
  - 맵/리듀스 형태의 작업에만 최적화
- **자원활용방법의 문제점**
  - 잡트래커가 맵/리듀스작업이 잘 되고 있는지 관리
  - 잡트래커가 시스템 사용효율도 같이 관리
    - 맵작업이 바쁠경우/리듀스 작업이 바쁠 경우
- 네임노드의 **SPOF의 문제점**
  - 네임노드와 세컨더리 네임노드가 동시에 장애를 일으키면?

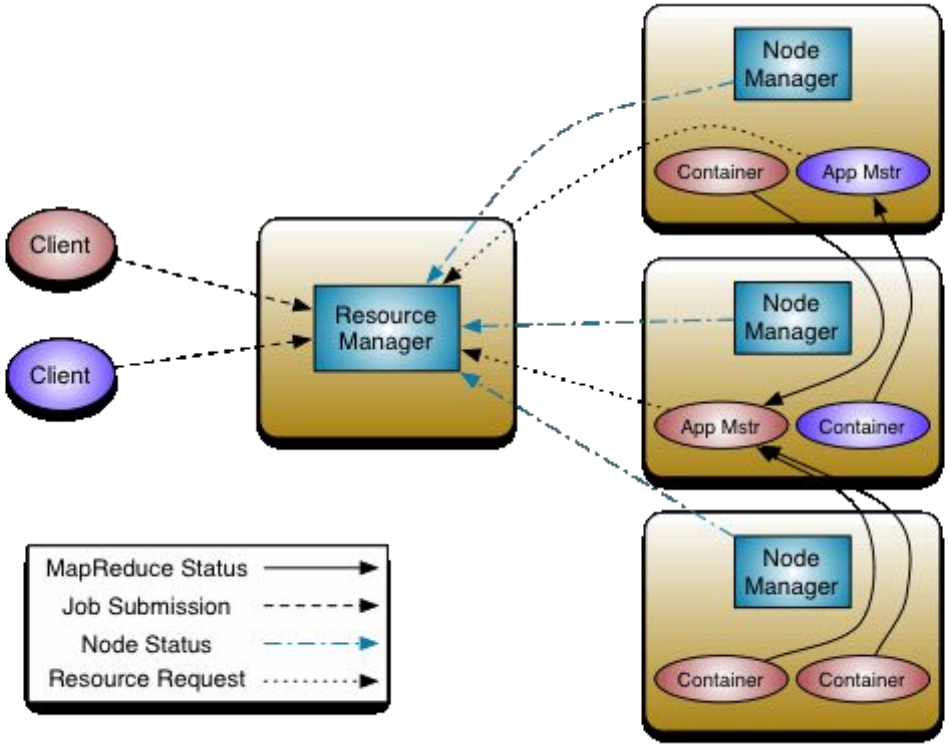
Hadoop 2의 해결책

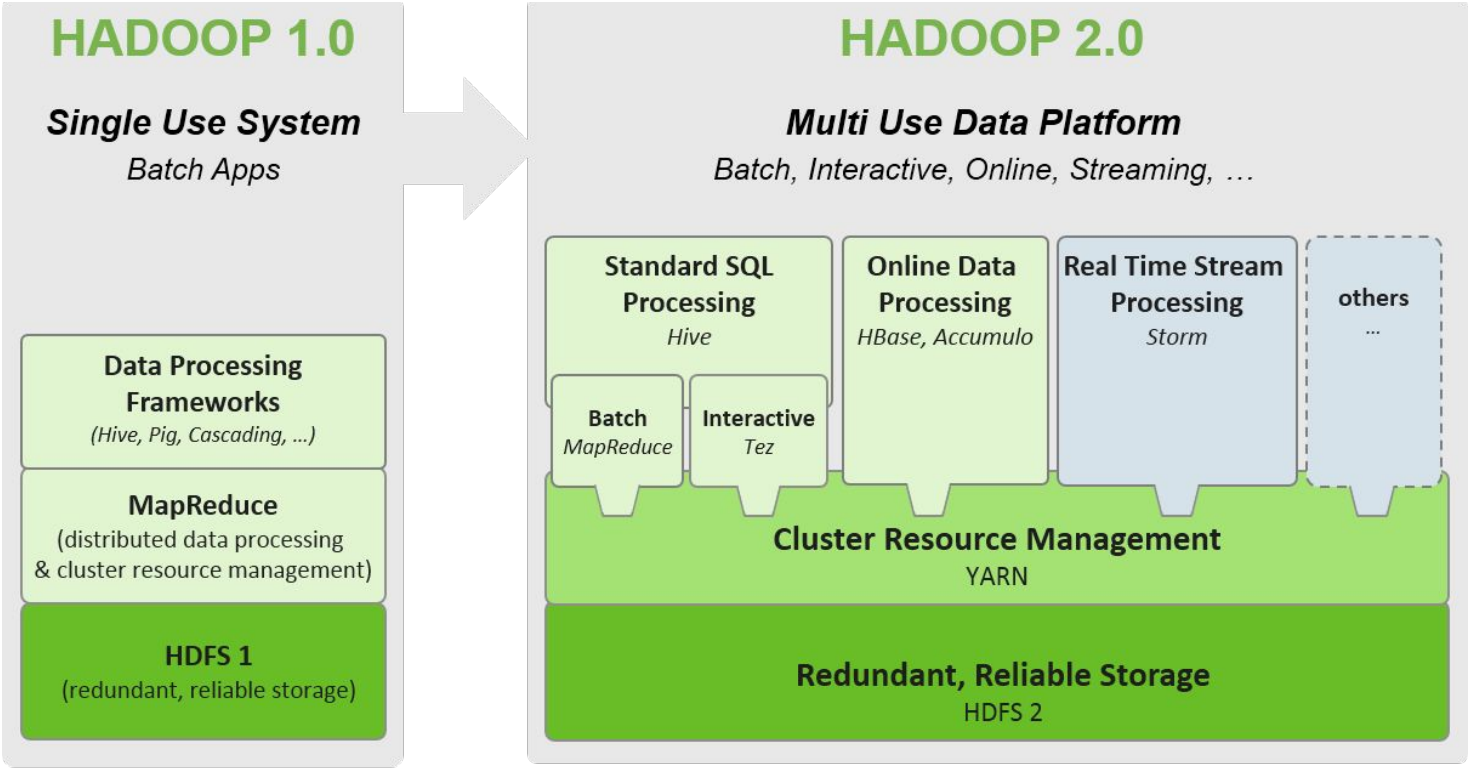
- 아키텍처의 변화
- **얀(YARN)**의 도입
  - YARN을 통해서 맵/리듀스작업 처리
  - **맵/리듀스작업은 YARN의 하나의 애플리케이션**이 됨
    - 다른 형태의 서버스 지원(스트리밍 포함)
    - 애플리케이션 마스터(AM:Application Master)
      - 애플리케이션별 작업관리
  - 잡트래커+태스크트래커->**노드매니저+리소스매니저**
  - **시스템 리소스활용과 잡(작업) 진행사항관리를 분리**

Hadoop 2의 해결책

- 아키텍처의 변화
- **주키퍼(Zookeeper)**의 도입
  - 네임노드 **고가용성(HA:High Availability)**가 가능하도록 구성
    - Master-Slave(수동)
    - Active-Standby(Hadoop 2)
      - 전환과정에서 문제발생 가능성
    - Active-Active(Hadoop 3)
      - 합의문제
  - election(선거)
  - 리소스매니져 이중화

하둡2 아키텍처(YARN)





http://asdtech.co/wp-content/uploads/2014/02/hadoop2.0.png

Interact with all data in multiple ways simultaneously

Hadoop 3의 변화점

- 블록복제(replication)
  - 데이타공간을 많이 필요한 문제
  - Erasure coding
    - 70% 저장공간으로 복제수 3의 효과(1.7-> 3효과)
    - cf. RAID 5,6 (체크섬, 패리티)
    - 계산량이 늘어남

- Zookeeper
  - Active-Active 모드
  - 처음부터 두 개 이상의 Active Namenode 허용
    - **합의**문제가 발생할 수 있슴(다수결)
      - 가급적으로 홀수로 구성(3,5,7,9,...)

스파크 프레임워크

스파크(Spark)란?

- **Apache Spark™**
  - **a fast and general engine for large-scale data processing** (이전)
  - a unified analytics engine for large-scale data processing (현재)

스파크(Spark)란?

- **Apache Spark™**
  - a fast and general engine for large-scale data processing (이전)
  - **a unified analytics engine for large-scale data processing** (현재)

스파크(Spark)란?

- **Apache Spark™**
  - a fast and general engine for large-scale data processing (이전)
  - **a unified analytics engine for large-scale data processing** (현재)

단순 빅데이터 처리 -> 본격 분석기술로의 발전

스파크(Spark)란?

- 스파크 = 빅데이터처리 + 데이터분석 + SQL

스파크(Spark)란?

- **스파크 = 빅데이터처리 + 데이터분석 + SQL**

- 빅데이터처리(하둡과 유사한 기술)
  - 배치처리 / 실시간처리(스트리밍) 지원

스파크(Spark)란?

- **스파크 = 빅데이터처리 + 데이터분석 + SQL**

- 빅데이터처리(하둡과 유사한 기술)
  - 배치처리 / 실시간처리(스트리밍) 지원
- 데이터 분석(R / 파이썬 데이터 사이언스)
  - 머신러닝(딥러닝은 별도의 라이브러리 필요)
  - https://www.nextobe.com/single-post/2017/08/01/상위-10가지-딥러닝-프레임워크

스파크(Spark)란?

- **스파크 = 빅데이터처리 + 데이터분석 + SQL**

- 빅데이터처리(하둡과 유사한 기술)
  - 배치처리 / 실시간처리(스트리밍) 지원
- 데이터 분석(R / 파이썬 데이터 사이언스)
  - 머신러닝(딥러닝은 별도의 라이브러리 필요)
  - https://www.nextobe.com/single-post/2017/08/01/상위-10가지-딥러닝-프레임워크

- 정형데이터처리(SQL)

스파크(Spark)란?

- **스파크 = 빅데이터처리 + 데이터분석 + SQL**

- 빅데이터처리(하둡과 유사한 기술)
  - 배치처리 / 실시간처리(스트리밍) 지원
- 데이터 분석(R / 파이썬 데이터 사이언스)
  - 머신러닝(딥러닝은 별도의 라이브러리 필요)
  - https://www.nextobe.com/single-post/2017/08/01/상위-10가지-딥러닝-프레임워크

- 정형데이터처리(SQL)
- 그래프관련 처리(GraphX)

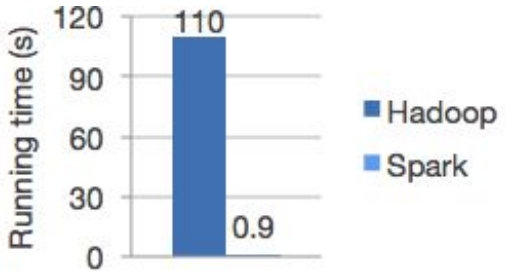
스파크(Spark)란?

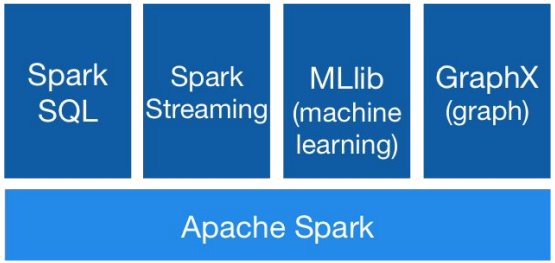
- 속도
  - 하둡보다 100배 이상 빠름(메모리 위주로 처리)
- 사용편의성
  - 자바, **스칼라**, 파이썬, R, SQL지원

```
df = spark.read.json("logs.json")

df.where("age > 21").select("name.first").show()
```

- SQL, 스트리밍, 분석(머신러닝) 지원
  - 자체 지원
- Scalable 서비스 지원
  - 하둡, 쿠버네티스, 클라우드 지원





스파크(Spark)와 하둡의 비교

- 하둡과의 비교
  - 하둡은 분산파일시스템(HDFS)와 분산처리시스템(MapReduce)의 조합

- **스파크는 별도의 파일시스템 없이 분산처리시스템으로 존재**
  - 하둡 관점에서는 HDFS와 같은 파일시스템 위에서 MapReduce 프레임워크를 대체
  - 스파크는 HDFS상에서 동작할 수도 있고 없어도 동작가능

- 하둡은 디스크기반으로 설계
  - 스파크는 **RDD라는 메모리기반 자료구조**를 사용

하둡과 스파크 비교

Word Count Example

```
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
  .map(word => (word, 1))
  .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

수십 라인의 Java 코드를 단 3줄로 만들어 버림

데이터분석의 괴로움 해결

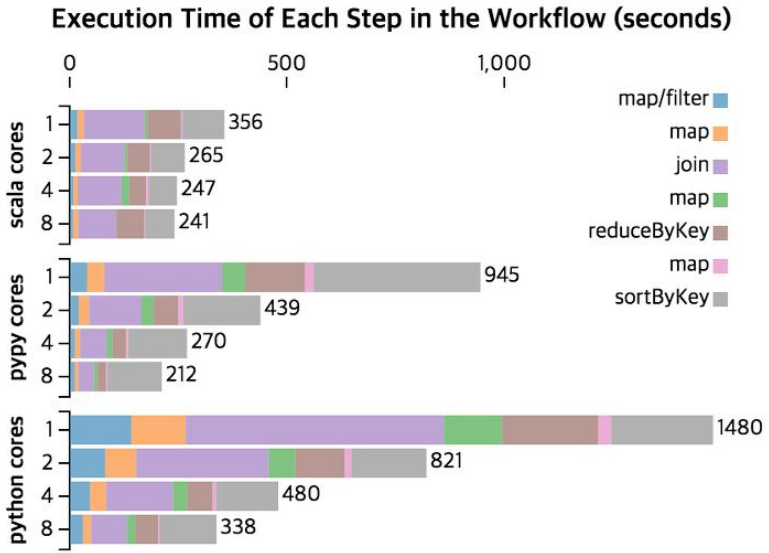
스파크(Spark)란?

- **스칼라(scala) 언어**로 만들어짐(JVM필요)

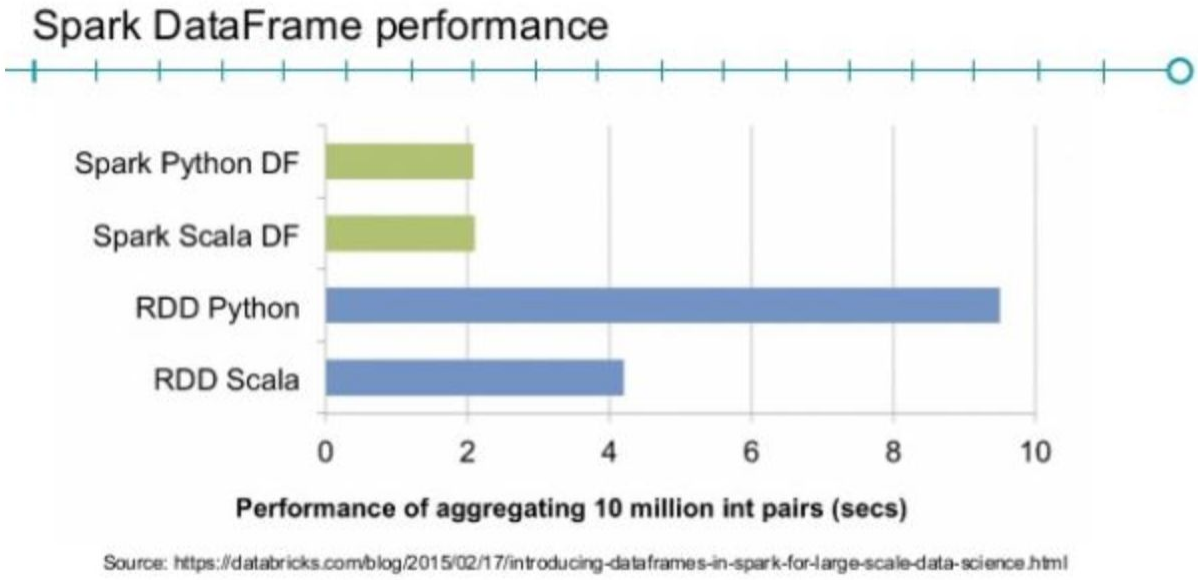
- 자바/파이썬/R 지원
  - **성능은 조금씩 차이가 남**
  - 스칼라보다 파이썬이 느림

- 쉘(Shell)
  - spark-shell(scala), pyspark(python)
  - 자바는 쉘이 없음
  - 한 줄 씩 실행모드

Scala vs. Python(Spark 1.x)



Scala vs. Python(Spark 2.x)



http://man
8%EC%84

Source: https://databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html

감사합니다