BIG DATA ANALYTICS

WEEK-13 | Big Data Overview

Yonsei University
Jungwon Seo

강의 목표

게시글 데이터가 **1000**억개 있을때, 조회수를 기준으로 상위 **N**개의 게시글을 추출하려고 한다!
한 컴퓨터에서 작업하면 **10**시간이 걸리는 작업을 **1**분 내로 처리할 수 있을까?

Big Data란?

- 얼마나 커야 빅데이터?
  - 100MB? 10GB? 1TB? 100TB?

2000년
HDD: 20GB
RAM: 64MB

2020년
SSD: 512GB
RAM: 16GB

하드웨어 동향

- 매년 저장소는 기하급수적으로 증가하는 반면 처리량은 선형적으로 증가
  - 하드 드라이브에 있는 데이터를 모두 처리하는데 걸리는 시간은 매년 증가
  - SSD의 개발로 더 복잡한 데이터 처리에 더 복잡한 알고리즘과 프레임워크를 사용할 수 있게되었지만, 아래와 같은 동향은 변하지 않음 (Wiktorski, 2019)

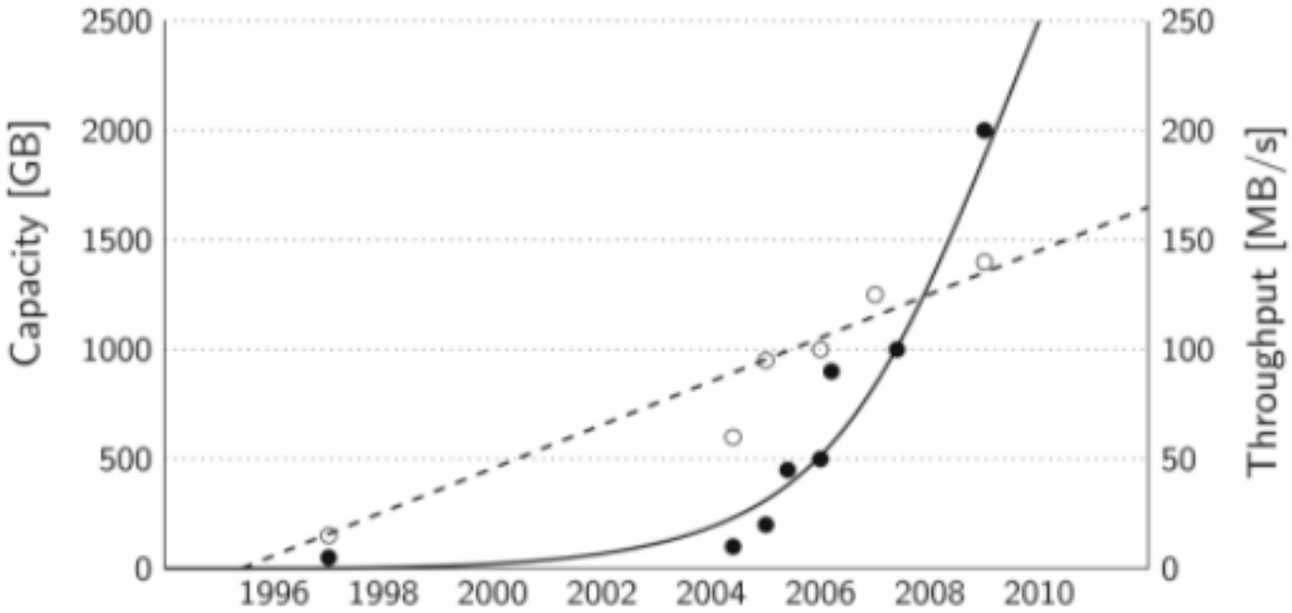


Fig. 2.2 Historical capacity versus throughput for HDDs. *Source* Leventhal (2009)

* Wiktorski, Tomasz. Data-intensive Systems: Principles and Fundamentals Using Hadoop and Spark. Springer, 2019.

“처리량의 증가 속도 < 저장량의 증가 속도” 일때
전체 데이터를 처리하는 시간을 일정하게 유지
할 수 있을까?

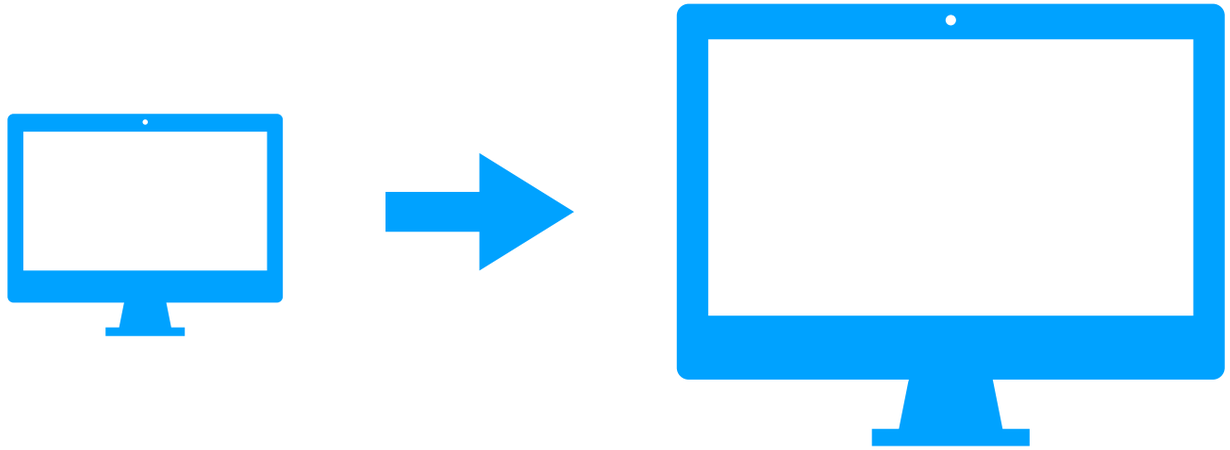
100GB, 10MB/s => 10,000s

200GB, 11MB/s => 18,181s

Vertical Scaling

더 좋은 하드웨어를 사용하자!

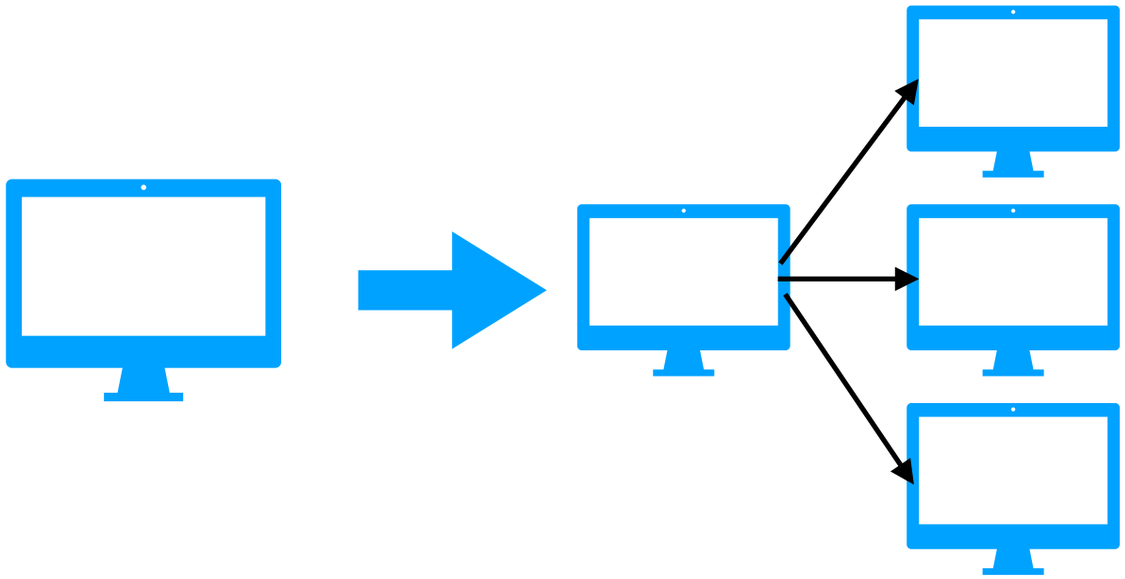
Vertical Scaling



더 좋은 하드웨어 = 더 많은 저장용량
처리량과 저장량의 갭은 더욱 커짐!

Horizontal Scaling

더 많은 하드웨어를 사용하자!

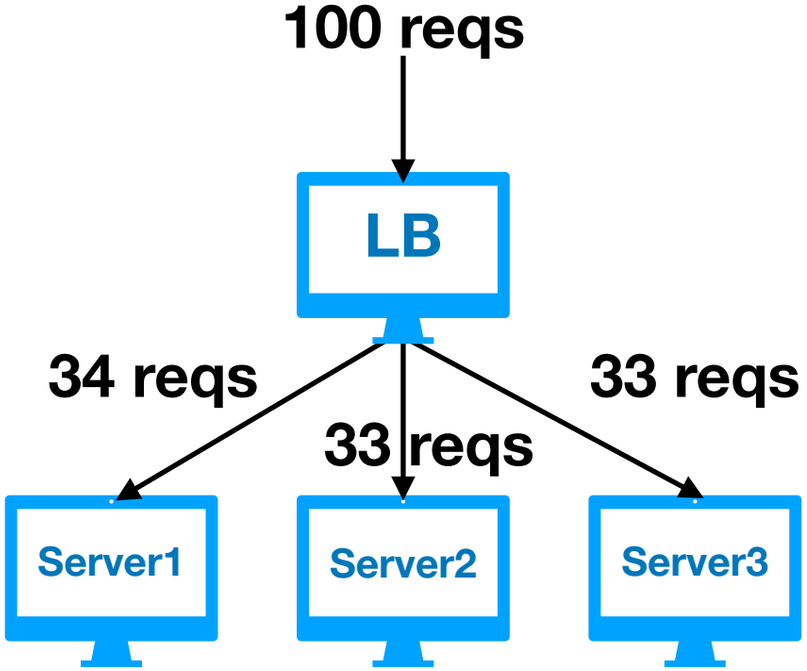


Horizontal Scaling

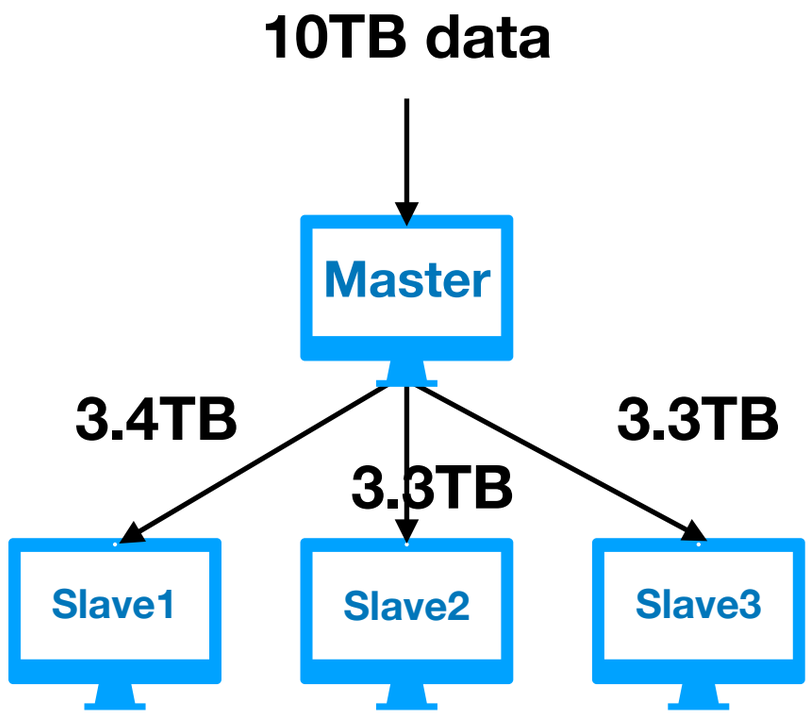
데이터 처리를 n개의 하드웨어에 분산시켜서 처리하자!
그런데 어떻게?

Horizontal Scaling의 어려움

웹 서버의 **Scaling**과 달리 작업들이 서로 독립적이지 않음



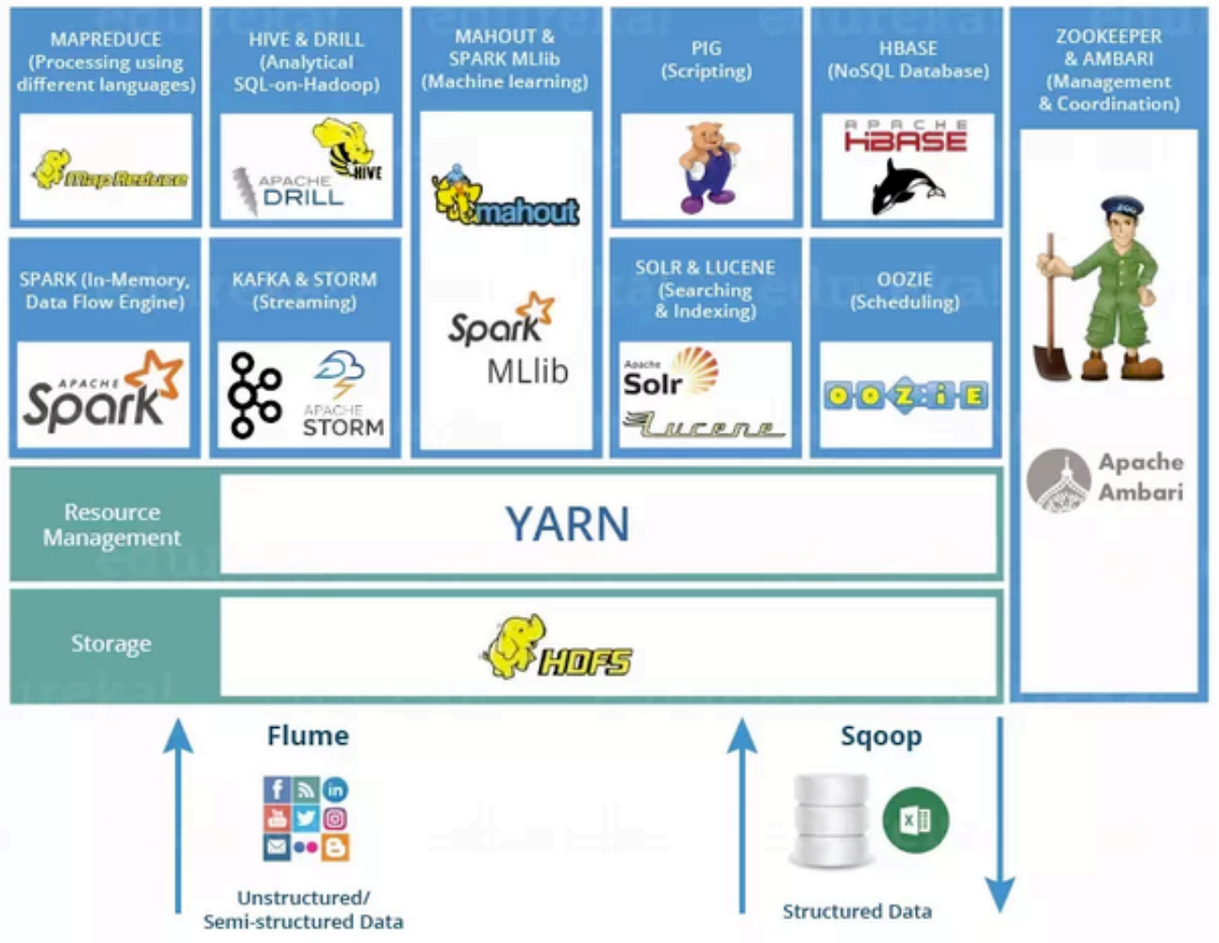
Web servers



Distribute Processing

Hadoop

- 여러 컴퓨터들을 대용량 데이터 처리(분산 처리)에 맞게 연결 시켜주는 프레임워크

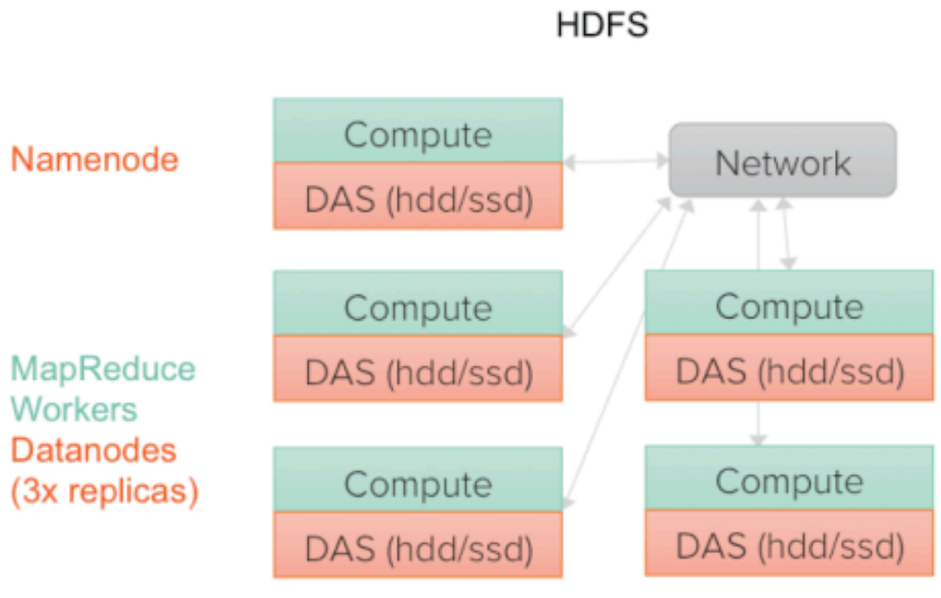


https://www.edureka.co/blog/hadoop-ecosystem

HDFS

- Hadoop Distributed File System
  - 대용량 파일을 여러 디스크에 분산에서 저장하고 분산 된 데이터를 빠르게 처리할 수 있는 파일 시스템
  - 저사양의 서버를 여러개 연결하여 고사양의 시스템을 구축 가능

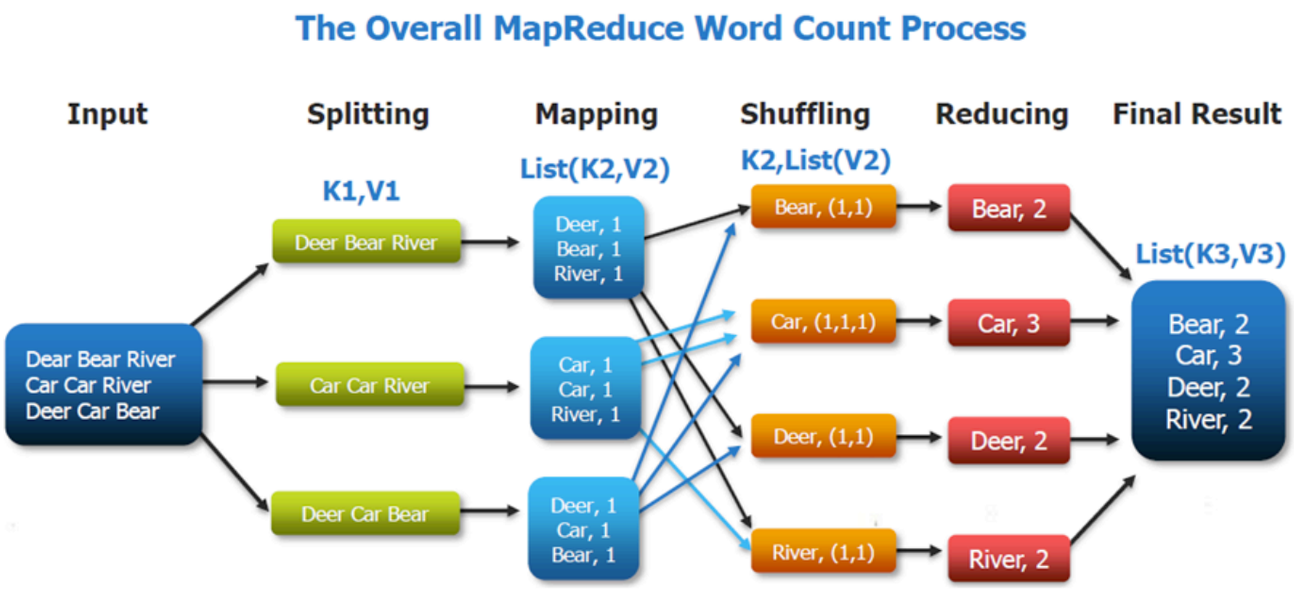
- HDFS의 대표적인 특징
  - Fault Tolerance: 한 시스템이 오류가 났을 때도 정상적으로 운영이 가능한가
  - Replication: 같은 데이터가 여러 서버에 복제되어 있는가



https://blog.purestorage.com/the-case-for-on-premise-hadoop-with-flashblade/

Map-Reduce

- 분산되어 있는 파일을 마치 하나의 파일 처럼 처리할 수 있는 프레임워크
  - HDFS에 저장된 파일은 NAS와 달리 하나의 파일이 여러 노드의 걸쳐 분산되어 있기 때문에, 일반적인 방법으로는 전체 파일에 효과적으로 접근하기가 어려움



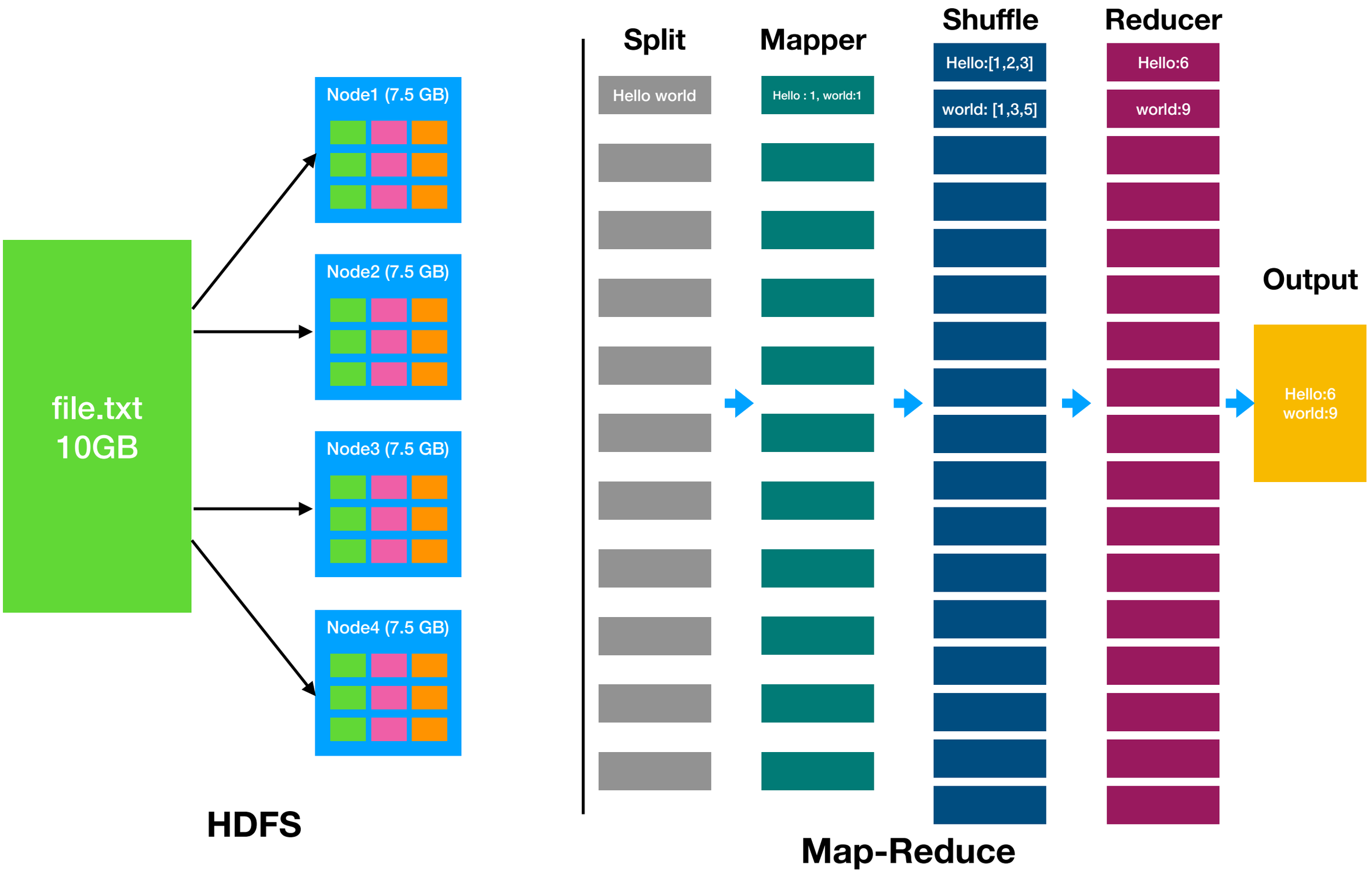
https://wikis.nyu.edu/download/attachments/74681720/WordCount%20MapReduce%20Paradigm.PNG?version=1&modificationDate=1462902481180&api=v2

Map-Reduce 한눈에 보기

- 10GB의 텍스트 파일이 HDFS에 저장
  - 4개의 Data Node가 존재
  - 복제본 3개를 기준으로 했을 때 각 Node당 7.5GB씩의 데이터를 지님
  - 각 노드의 기본 block size를 100MB로 설정=〉 노드당 75개의 block이 존재

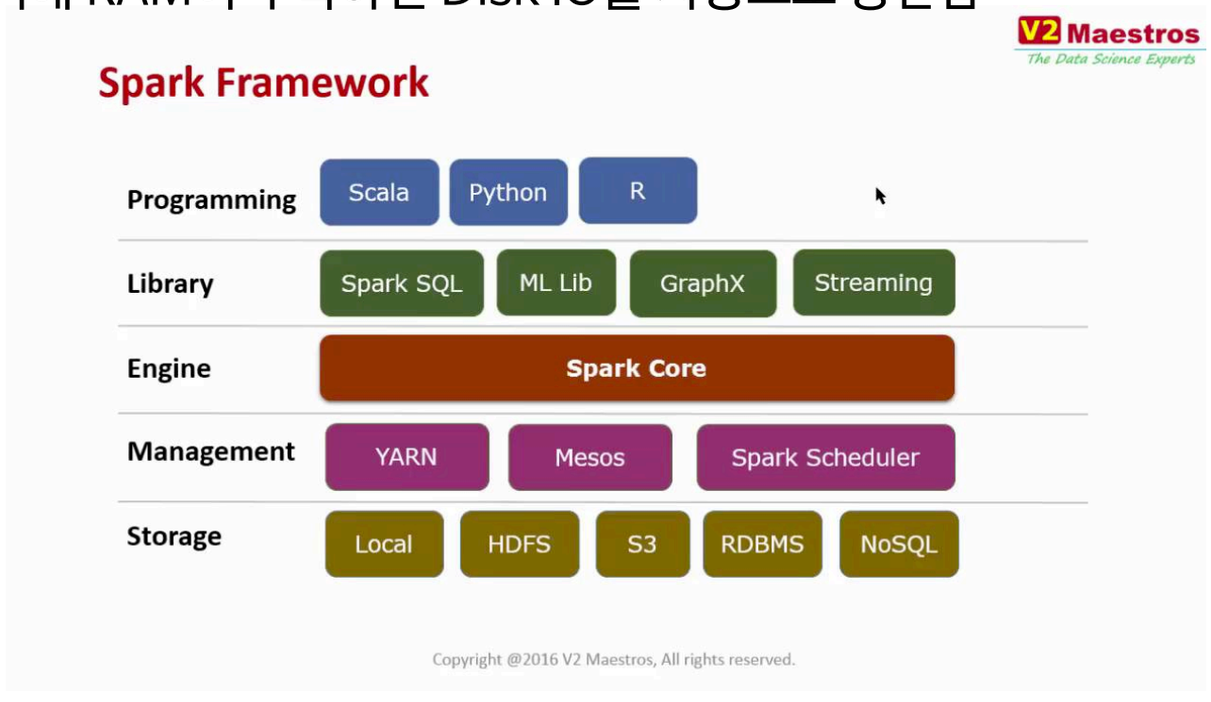
- Map-Reduce를 이용해 Word Count 시작
  - 각각의 data node가 75개로 split 함, mapper 75개 생성
  - 각각의 mapper는 각각의 block에 대한 처리
    - Mapper가 많을 수록 빠른게 아니라, Data node의 core수에 맞게 병렬처리가 이루어짐
  - block 별 unique한 단어수 Counting 완료
    - 예) “bigdata”, 3
  - shuffling을 통해 같은 key (이 경우에는 단어)로 데이터들이 묶임
    - 예) “bigdata”, [1,3,1,2]
  - reducer를 통해 묶이 데이터들에 대한 처리
    - 예) “bigdata”, 7

Map-reduce 한눈에 보기



Spark

- HDFS+MapReduce의 조합은 Disk IO를 항상 동반하기 때문에 속도가 느림
- Spark는 In-memory processing으로 일반적으로 10~100배 더 빠른 속도를 보임
- 만약 데이터에 비해 RAM이 부족하면 Disk IO를 자동으로 동반함



https://www.youtube.com/watch?v=ZTFGwQaXJm8

E.O.D