빅데이터 시스템

구름

부동산융합대학원

한양대학교

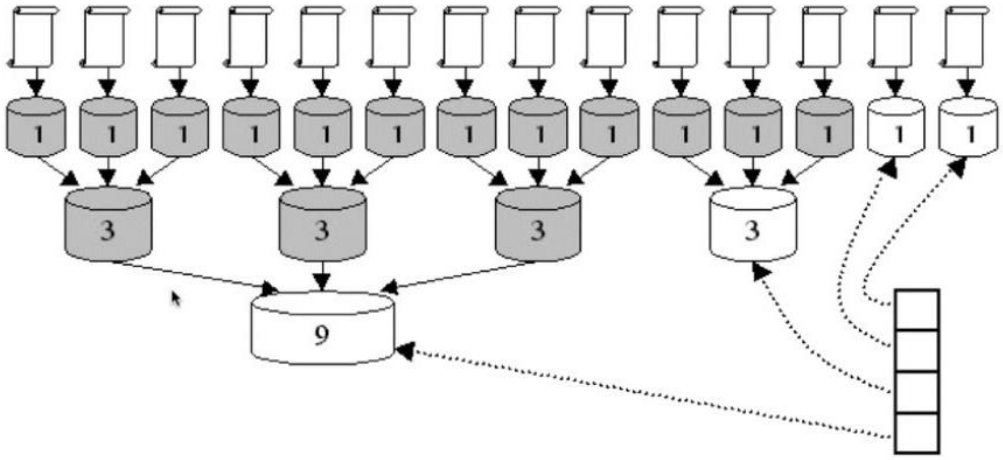
검색엔진 시장의 성장과 아파치 루씬 오픈소스 등장

99년 더그커팅의 루씬 프로젝트 개방을 통해 검색 엔진 확대
웹크롤링 기능의 너치 프로젝트도 함께 발표

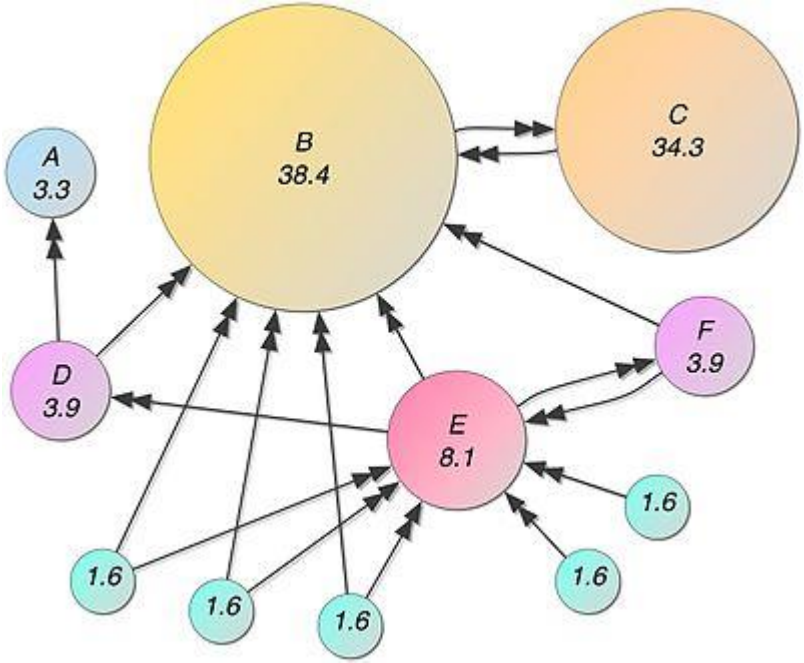
검색엔진의 핵심은 인덱싱!

늘어나는 웹페이지들의 인덱싱과 랭킹 계산을 위해 대용량 저장 매체가 필요

Lucene Indexing

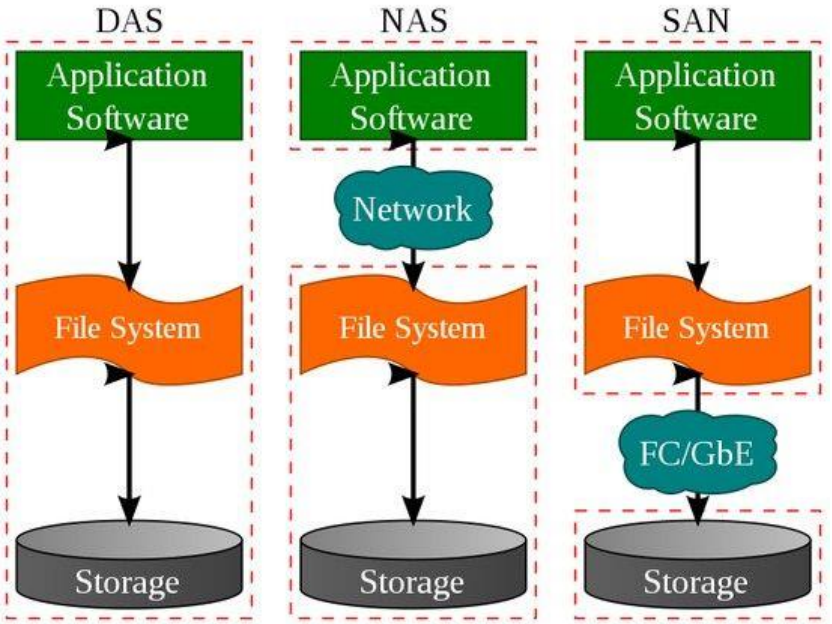


Google PageRank



대용량 저장을 위한 분산 저장 기술

확장 방식의 한계는 모두 존재



2003년 너치의 한계와 개선 목표 설정

1억 개 페이지를 RAID(DAS) 활용 스토리지로 처리가 한계

개선 목표

1. 데이터 형식이 자유롭다. (Schemeless)
2. 저장된 정보는 사라지지 않는다. (Durable)
3. 인간의 개입 없이 각종 장비의 오류를 처리한다.
4. 모든 장비의 사용을 균등하게 배분한다.

2003년 10월 Google File System을 공개

2003년 10월 구글이 자체 분산 저장 기술을 공개하는 Google File System 문서를 공개 (https://ai.google/research/pubs/pub51)

2004년 GFS를 참고하여 NDFS를 공개

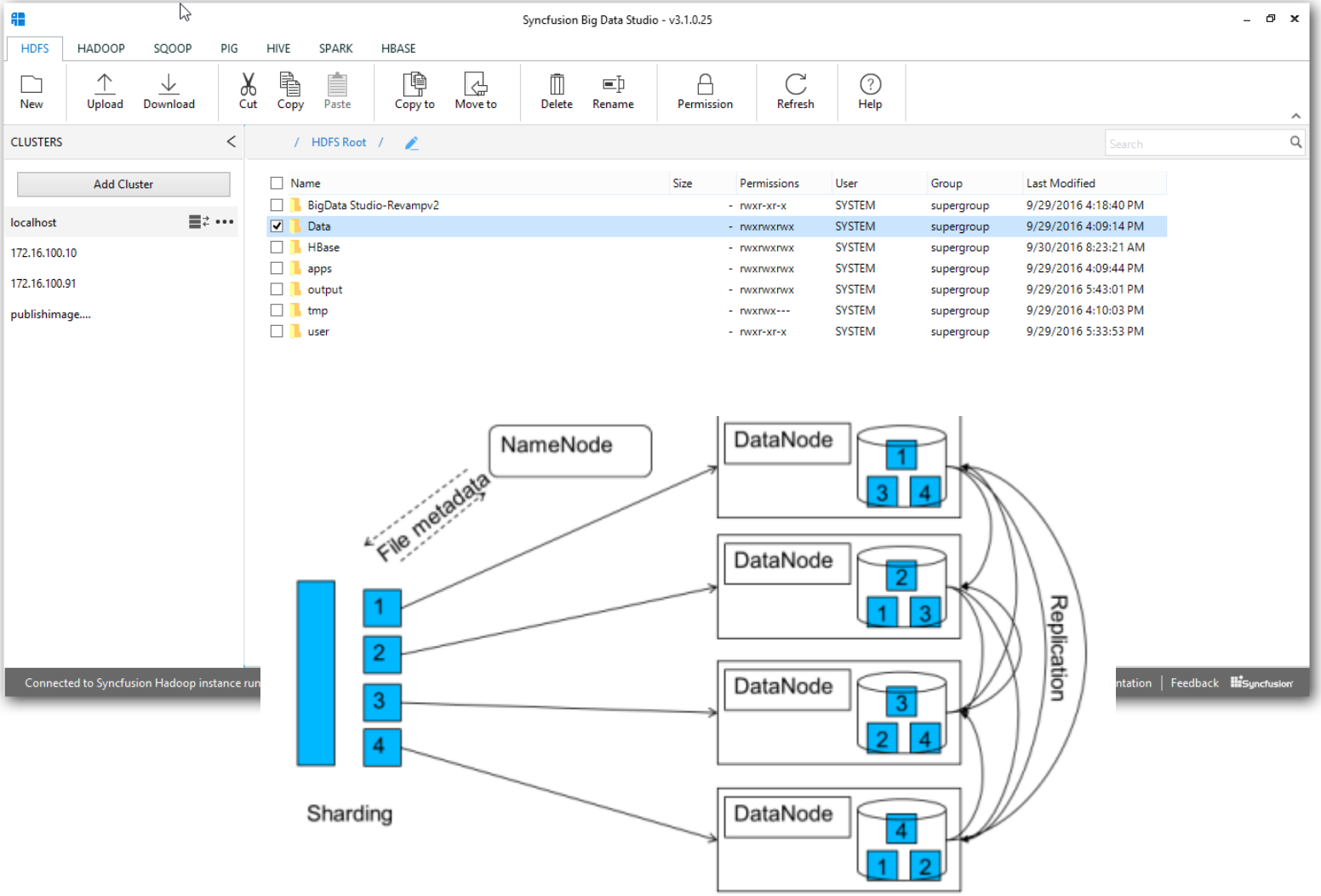
모든 데이터를 64MB 청크로 분할 저장

3개 이상의 복사본을 서로 다른 데이터 노드에 복제하여 데이터 손실 가능성을 제거

네임노드를 통해 추상화된 스토리지를 제공

정보 영속성, 공간 확장성, 오류 장비 처리 모두 해결

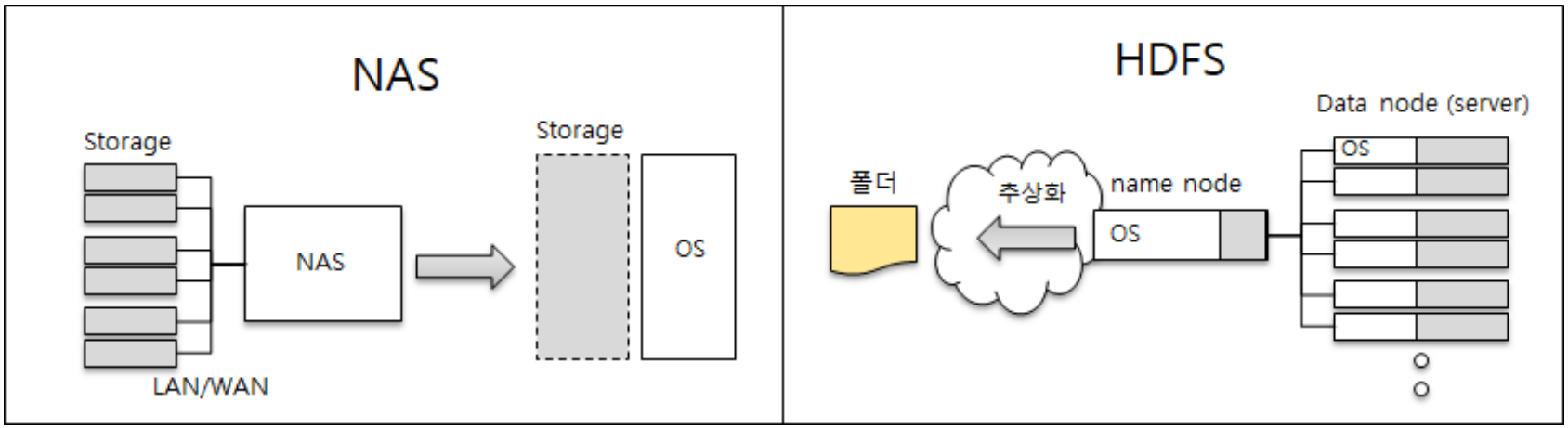
DFS 분산파일시스템은 사용자 입장에선 동일한 폴더



네트워크를 사용하는 NAS와 DFS의 차이점

LAN을 통한 데이터 전송에도 한계가 존재

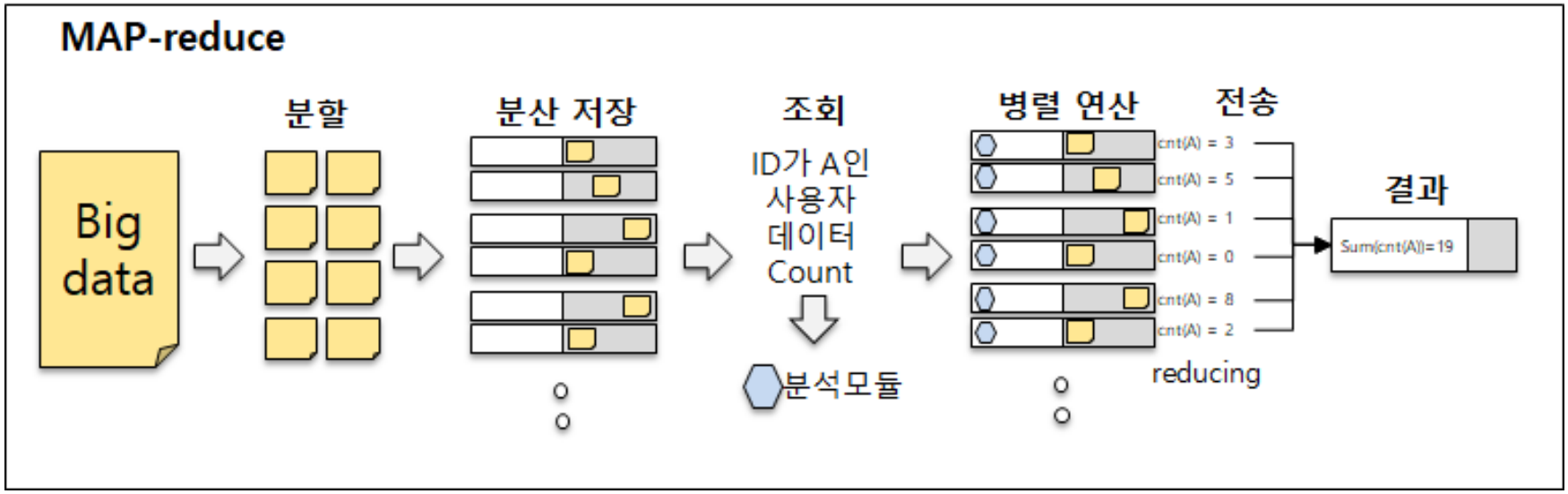
데이터 노드가 프로세서를 보유



Map-Reduce 의 등장

2004년 12월
구글에서 분산 저장 환경에서 효율적으로 데이터를 활용할 수 있는 맵리듀스 기술을 공개
(https://ai.google/research/pubs/pub62)

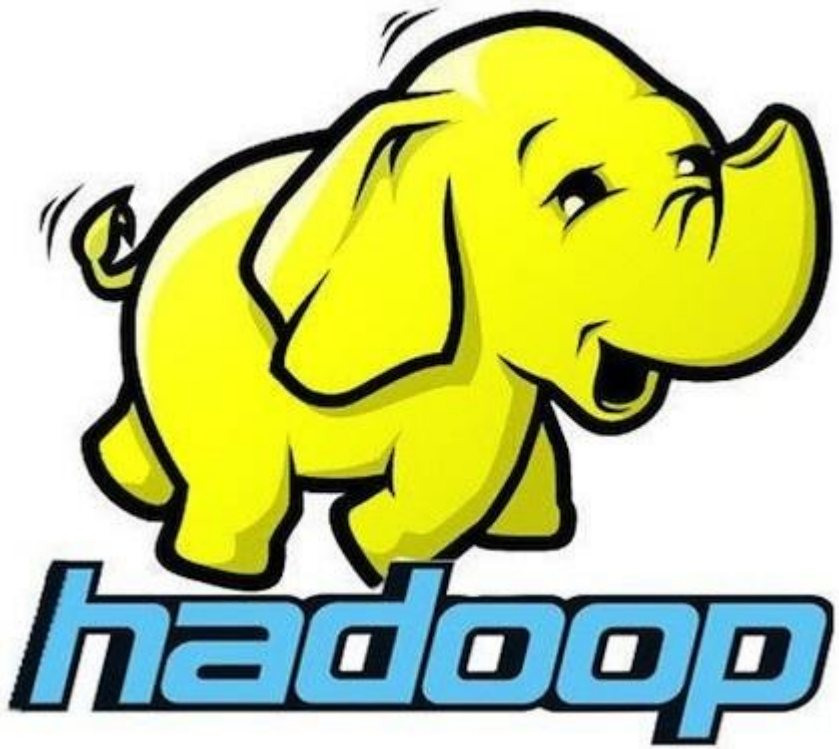
※ 맵리듀스는 개별 노드에 분석을 위한 연산과정을 모듈단위로 전달하고 연산과정 동안 개별 모듈이 마스터 서버에 지속적으로 핑을 알리도록 설계한 후 연산 모듈이 비정상적으로 종료되는 경우 해당 업무를 새로운 모듈에게 부여하는 방식으로 연산 오류를 해결

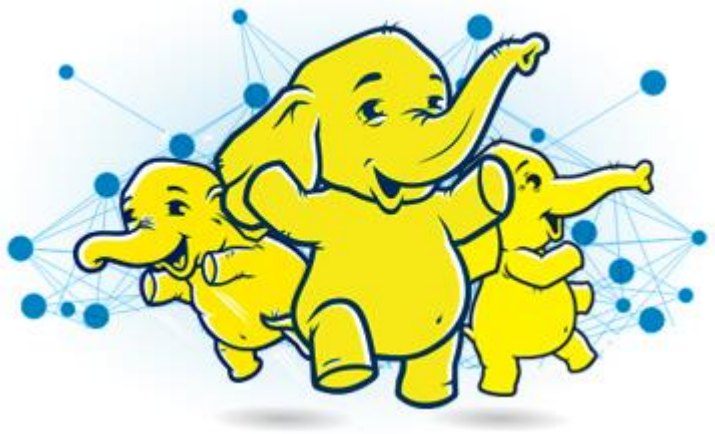


Hadoop의 등장!

2006년 2월
너치의 베이스코드와 NDFS, 맵리듀스를 루신프로젝트에서 분리하여 Hadoop이라 명함

Hadoop은 Hadoop Common, HDFS, Map-reduce로 구성





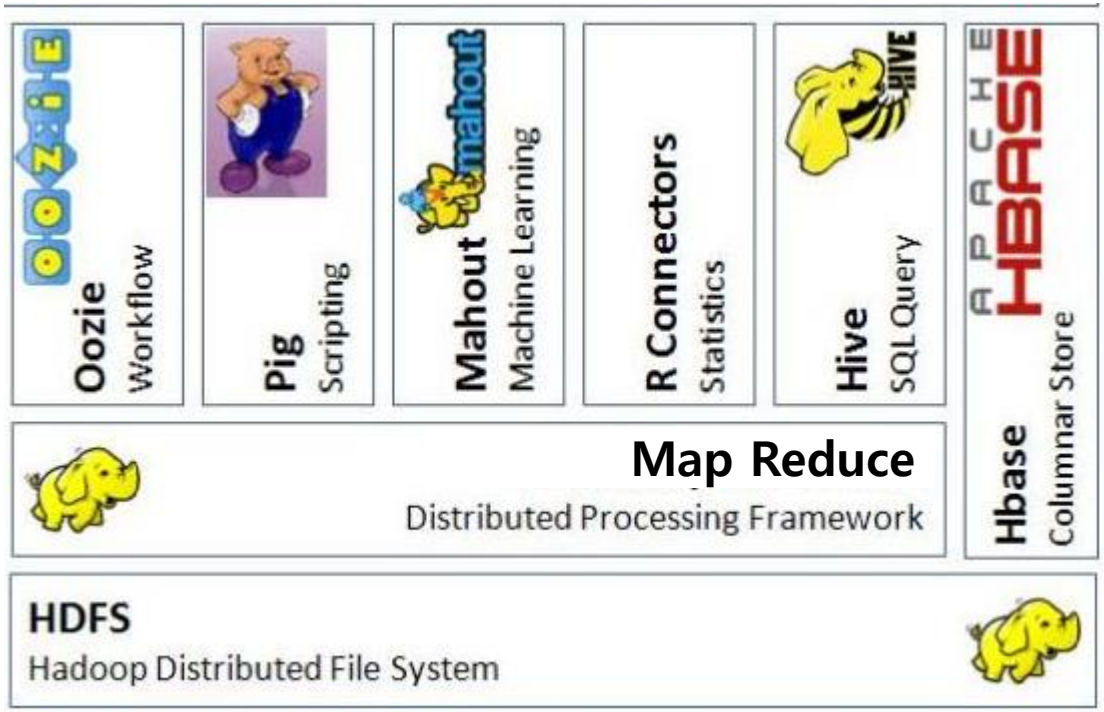
하둡 데이터 조회 에코시스템의 발전

2008년 아파치에서 HDFS를 활용한 데이터베이스로 **HBASE**를 발표

2008년 야후는 맵리듀스를 사용하여 HDFS데이터를 조회하는 **Pig**를 발표

(루씬과 하둡을 만든 더그커팅은 2006년 야후로 옮김)

페이스북은 SQL언어(HIVEQL)를 맵리듀스로 전환해 HDFS자료를 조회하는 **HIVE**를 발표

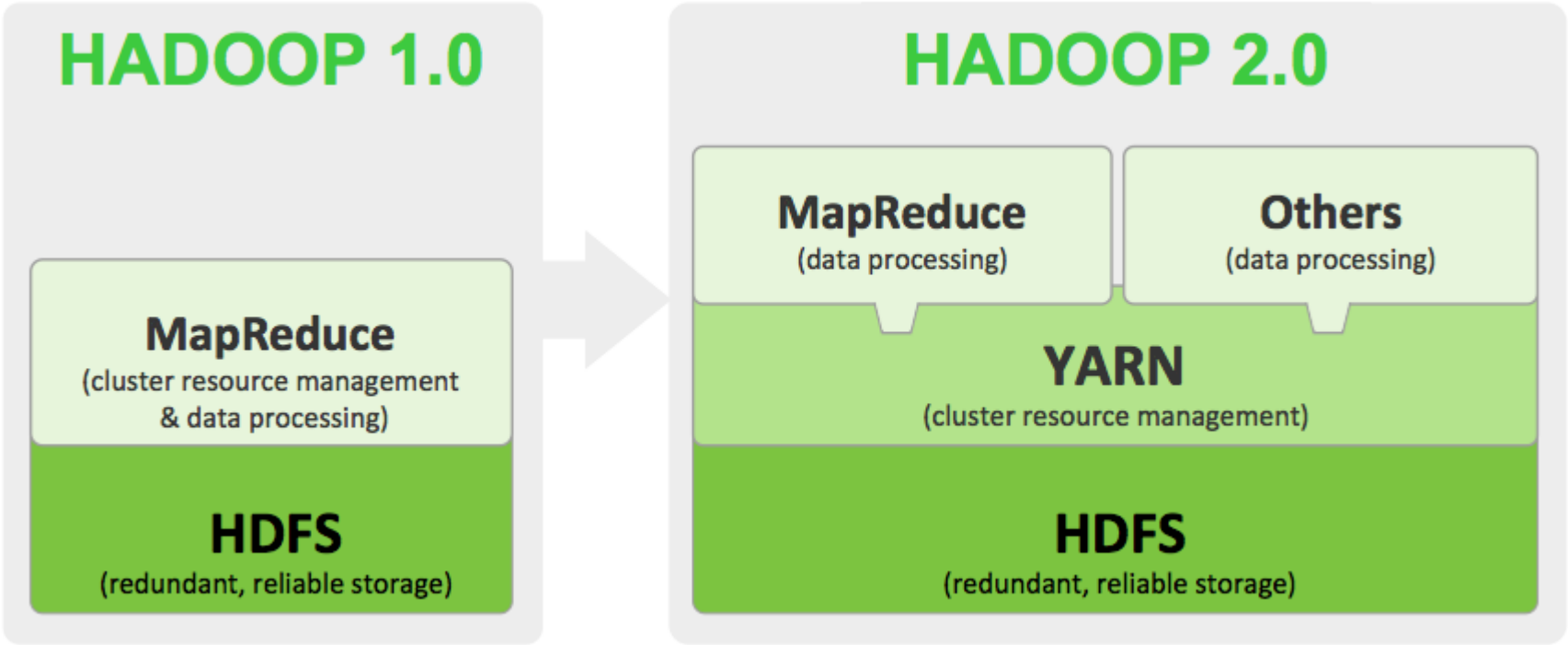


YARN의 등장

2012년 하둡은 맵리듀스V2를 YARN이라는 하위프로젝트로 독립시킴

과거 하둡은 데이터 분석과 자원 관리를 모두 고려해야 하는 어려움에서

자원 관리를 YARN이 모두 가져가면서 데이터 분석에 더 집중할 수 있게 됨



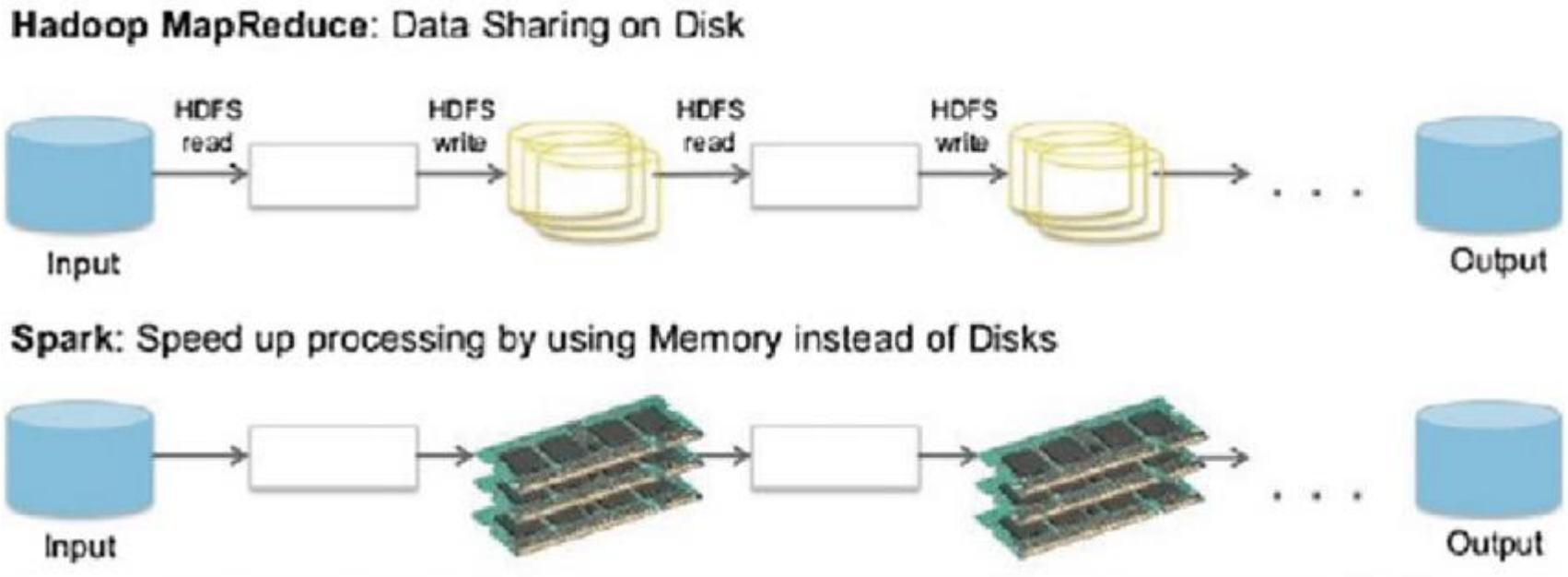
SPARK의 등장으로 분산 인메모리 컴퓨팅

2012년 UC버클리 박사과정에 재학중이던 마태 자하리아(Matei Zaharia)가 배포

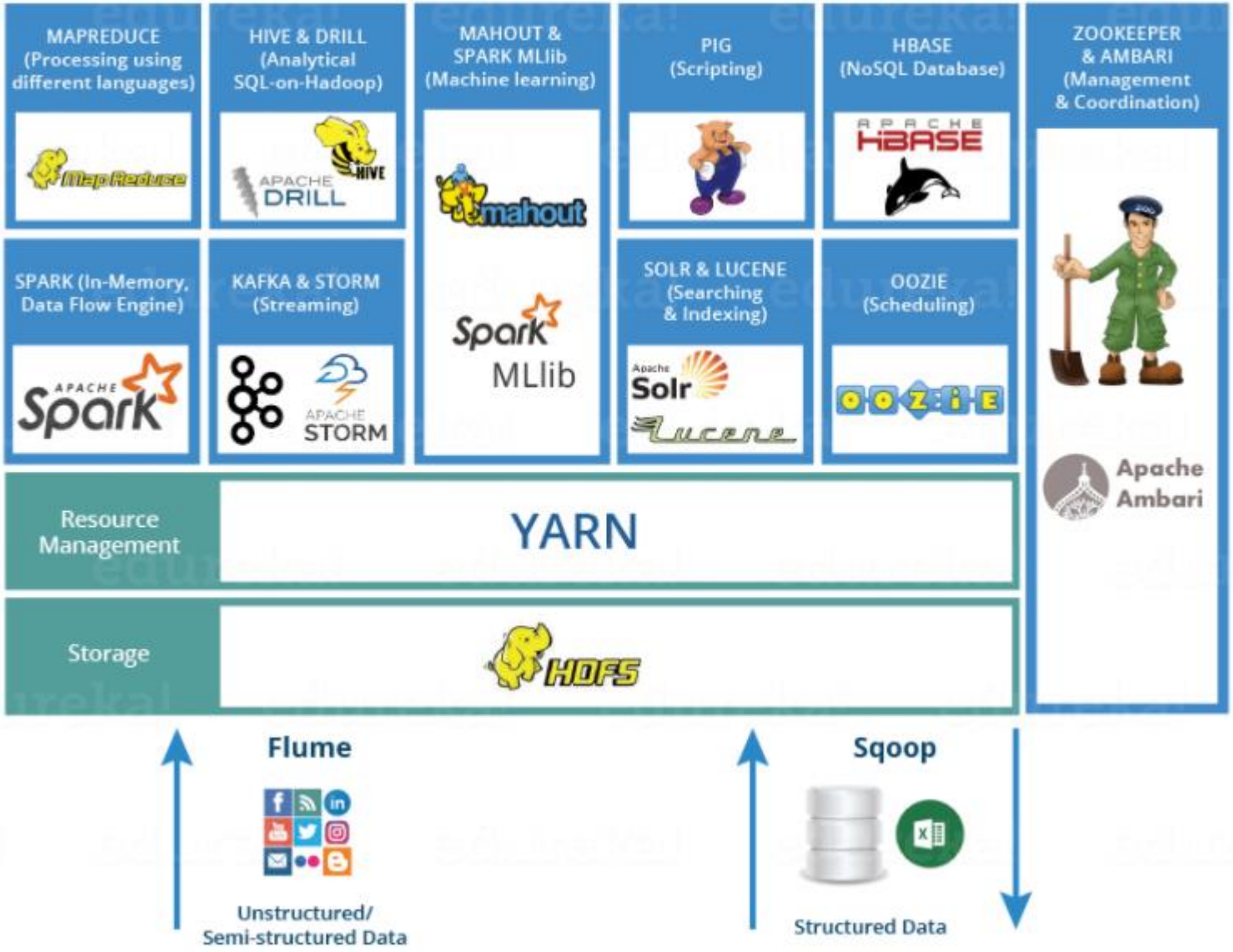
연산과정 동안 Disk I/O가 반복되는 문제를 해결하기 위해

코드 그룹을 받아서 한번에 인메모리 연산을 분산으로 수행하는 솔루션을 개발하고

아파치 그룹에 코어 소스를 기부함, 최근 발표된 spark3.0버전에서는 분산 GPU도 지원



에코시스템

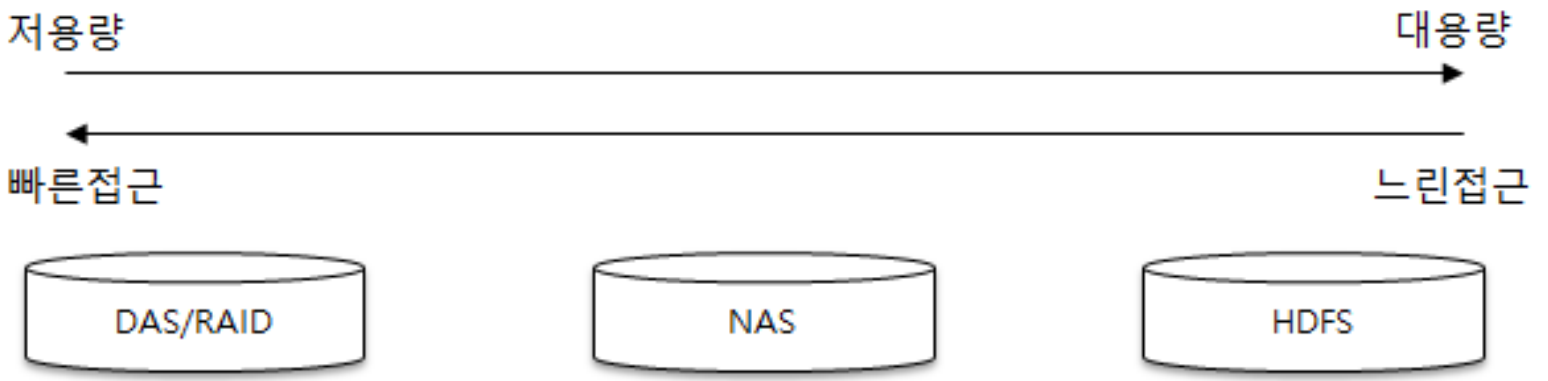

MAPREDUCE
(Processing using different languages)
HIVE & DRILL
(Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib
(Machine learning)
PIG
(Scripting)
HBASE
(NoSQL Database)
ZOOKEEPER & AMBARI
(Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM
(Streaming)
SOLR & LUCENE
(Searching & Indexing)
OOZIE
(Scheduling)
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/
Semi-structured Data
Sqoop
Structured Data
Apache Ambari

데이터를 중심으로 넓어진 선택의 폭

빅데이터는 이전 데이터 베이스를 압도하는 하나의 기술이 아니라

다양한 데이터에 맞춤복을 제공 가능한 수백 가지의 솔루션과 아키텍쳐의 조합

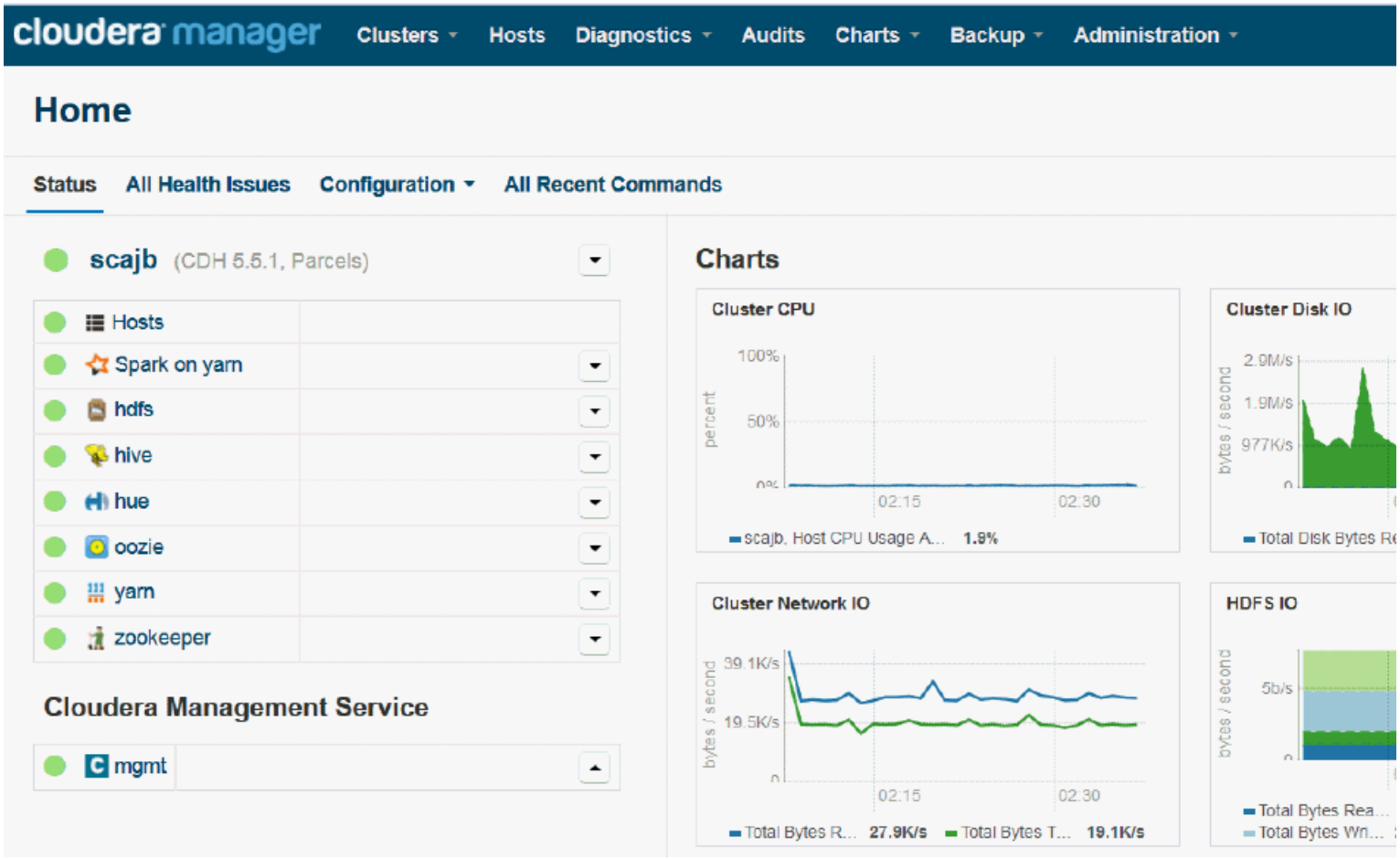
사용하는 데이터와 활용 목적에 따라 빅데이터는 다르게 적용해야 효율성이 극대화

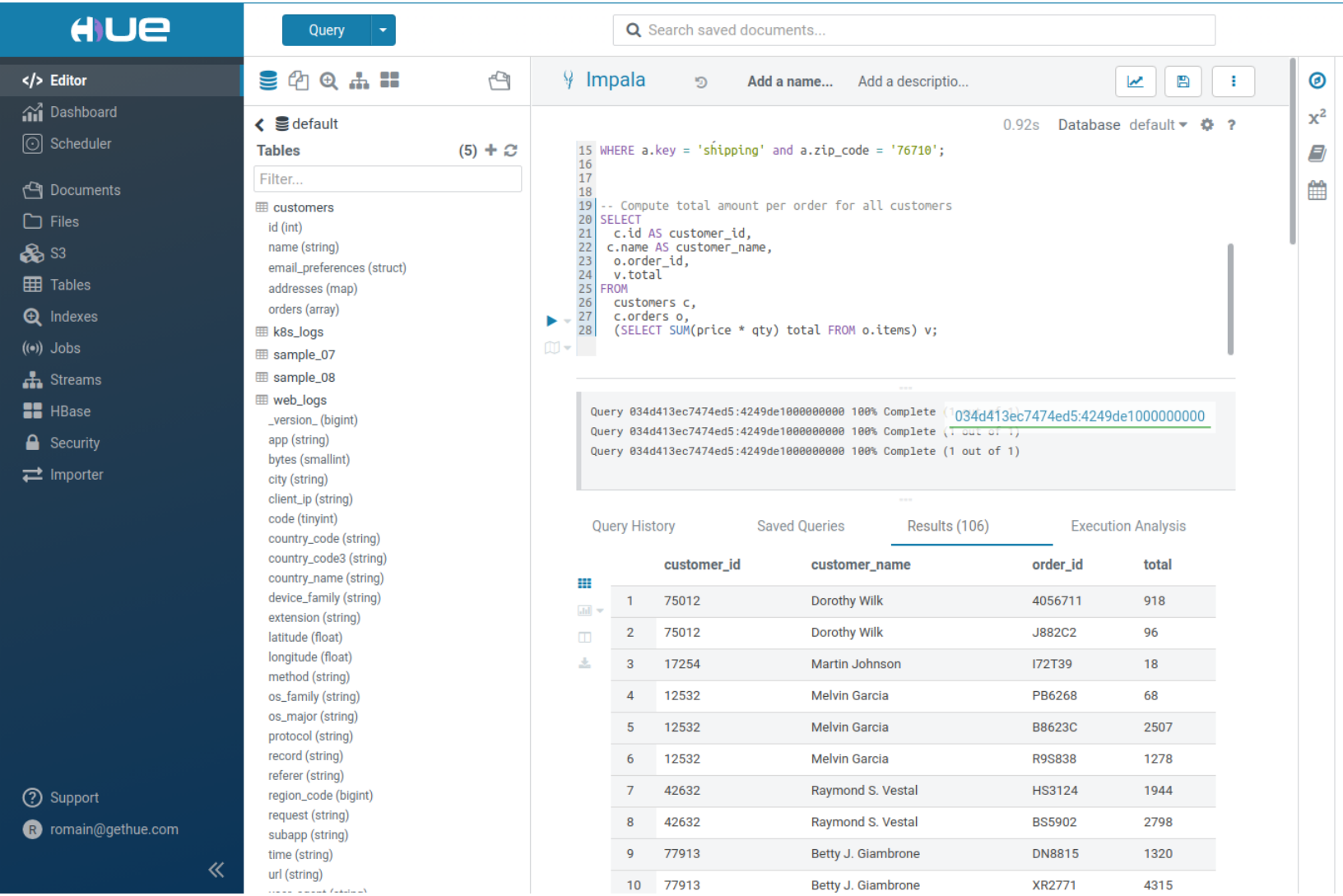


※데이터의 양이 임계치를 넘어서면 병렬 연산에 따라 HDFS의 처리속도가 월등히 빨라짐

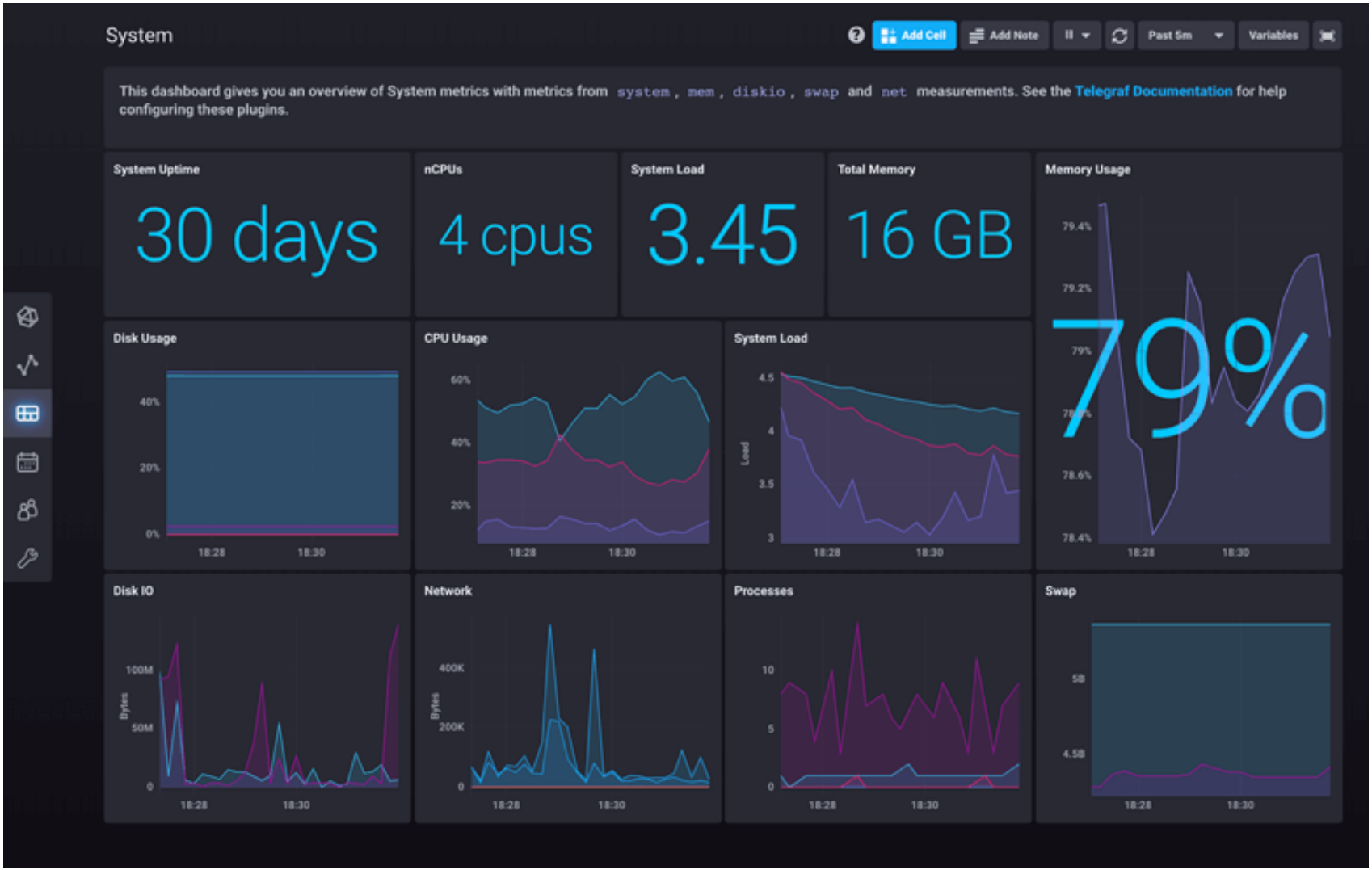
cloudera®

HORTONWORKS




HUE
Query
Search saved documents...
Jobs
Editor
Dashboard
Scheduler
Documents
Files
S3
Tables
Indexes
Jobs
Streams
HBase
Security
Importer
Support
romain@gethue.com
default
Tables (5)
Filter...
customers
id (int)
name (string)
email_preferences (struct)
addresses (map)
orders (array)
k8s_logs
sample_07
sample_08
web_logs
version (bigint)
app (string)
bytes (smallint)
city (string)
client_ip (string)
code (tinyint)
country_code (string)
country_code3 (string)
country_name (string)
device_family (string)
extension (string)
latitude (float)
longitude (float)
method (string)
os_family (string)
os_major (string)
protocol (string)
record (string)
referer (string)
region_code (bigint)
request (string)
subapp (string)
time (string)
url (string)
Impala
Add a name...
Add a descriptio...
0.92s
Database default
15 WHERE a.key = 'shipping' and a.zip_code = '76710';
19 -- Compute total amount per order for all customers
20 SELECT
21 c.id AS customer_id,
22 c.name AS customer_name,
23 o.order_id,
24 v.total
25 FROM
26 customers c,
27 c.orders o,
28 (SELECT SUM(price * qty) total FROM o.items) v;
Query 034d413ec7474ed5:4249de1000000000 100% Complete
034d413ec7474ed5:4249de1000000000
Query 034d413ec7474ed5:4249de1000000000 100% Complete (1 out of 1)
Query 034d413ec7474ed5:4249de1000000000 100% Complete (1 out of 1)
Query History
Saved Queries
Results (106)
Execution Analysis
customer_id
customer_name
order_id
total
1 75012 Dorothy Wilk 4056711 918
2 75012 Dorothy Wilk J882C2 96
3 17254 Martin Johnson I72T39 18
4 12532 Melvin Garcia PB6268 68
5 12532 Melvin Garcia B8623C 2507
6 12532 Melvin Garcia R9S838 1278
7 42632 Raymond S. Vestal HS3124 1944
8 42632 Raymond S. Vestal BS5902 2798
9 77913 Betty J. Giambrone DN8815 1320
10 77913 Betty J. Giambrone XR2771 4315
Tables
Statement 3/3
Filter...
default.customers
id int
name string
email_preferences struct
addresses map
orders array

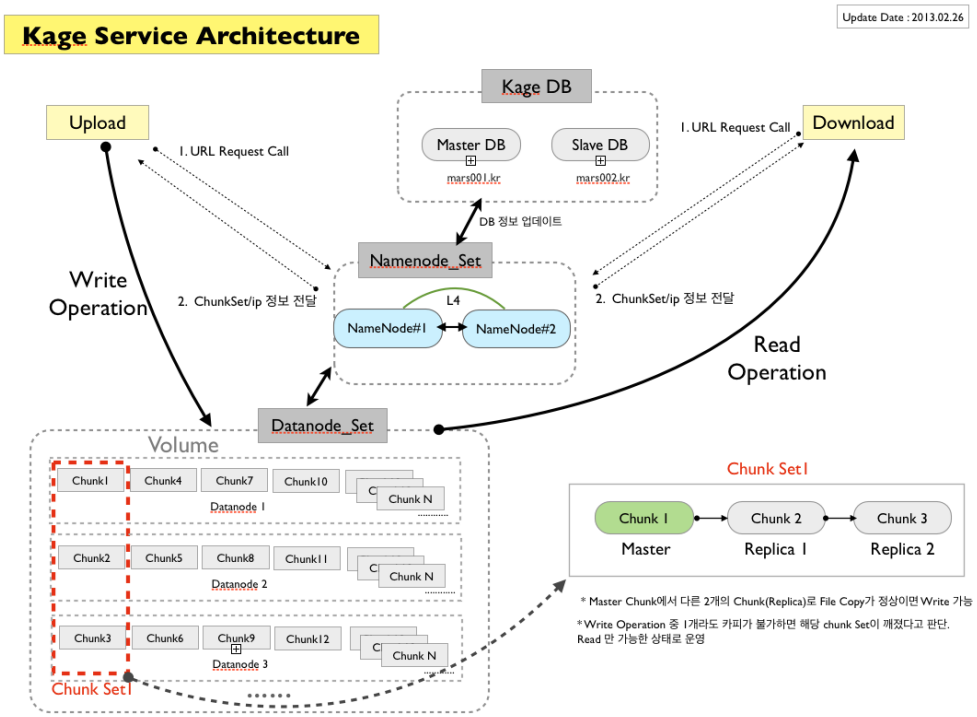
influxdb

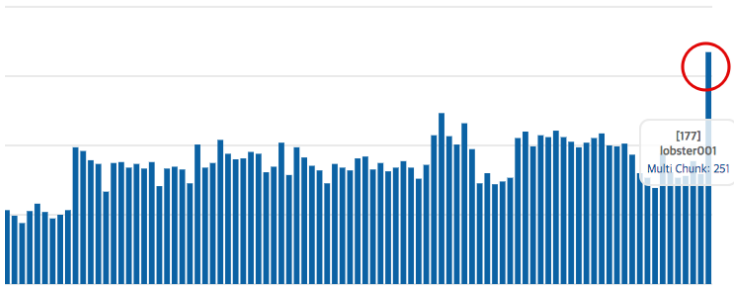

System
Add Cell
Add Note
Past 5m
Variables
This dashboard gives you an overview of System metrics with metrics from system, mem, diskio, swap and net measurements. See the Telegraf Documentation for help configuring these plugins.
System Uptime
30 days
nCPUs
4 cpus
System Load
3.45
Total Memory
16 GB
Memory Usage
79%
Disk Usage
CPU Usage
System Load
Disk IO
Network
Processes
Swap

https://tech.kakao.com/2017/01/12/kage/

카카오 빅데이터 시스템 KAGE

- 데이터를 분할하여 저장하고 추상화하는 방식은 하둡과 동일함
- 데이터의 저장 위치를 Kage-Key를 통해 관리하여 저장한 사용자 외에는 정보의 저장 위치를 알 수 없게 함
- 카카오 메신저 파일은 일정 시간이 지나면 삭제되므로 파일 저장시 Expire 기능을 지정하여 일정 시간이 되면 분산저장 된 파일이 자동 삭제되어 저장 공간을 재활용
- 특정 파일에 트래픽이 몰리는 경우 스케쥴링을 통해 청크를 데이터노드들에 추가 복제하여 트래픽이 골고루 분산될 수 있게 조정
- 그 외 이미지 변환과 동영상 트랜스코딩 기능 등 카카오서비스에 적합한 다양한 기능이 반영





http://csl.skku.edu/papers/kiise09.pdf

NAVER

네이버 빅데이터 시스템 OwFS

❍네이버는 대규모 분산 스토리지 시스템으로 자체 개발한 OwFS (Owner-based File System, http://csl.skku.edu/papers/kiise09.pdf) 사용

- 네이버 커뮤니티의 데이터 부분(메일 원문, 카페 및 블로그 이미지, 첨부파일 등)에 해당하는 정보만을 저장하는 보완적인 저장 장치로 개발
- 네이버의 이메일 서비스의 경우 95%의 메시지들이 55KB 미만으로 매우 많은 수의 작은 크기 파일들을 효율적으로 다룰 수 있게 구성
- 데이터 부분은 불변 파일들로 생성, 읽기, 삭제의 3종류 기능만 제공
- 네이버에서는 사례별 분산 스토리지 선택 기준을 아래와 같이 제시

표 1 주요 저장 시스템의 특징 비교

	NAS	Coda	Lustre	GFS	OwFS
파일 서비스 구현	Kernel-level	User-level	Kernel-level	User-level	User-level
POSIX API 지원	O	O	O	X	X
마스터 서버의 유무	X	X	O	O	O
마스터 서버 장애 대응	—	—	HA failover	HA failover	HA failover
데이터 서버 장애 대응	—	Replication	HA failover	Replication	Replication
복제 단위	—	Volume	—	Chunk	Owner
자동 장애 검출 및 복구	△	△	△	O	O
자동 데이터 이동	X	X	X	O	O
주요 용도	범용	유닉스 환경	고성능 컴퓨팅 환경	큰 파일 저장	다수의 불변파일 저장

http://www.bigvalue.co.kr/main

빅밸류 공간 빅데이터 BV-DFS

❍빅밸류는 자체 개발한 BVDFS(BigValue Distributed File System)을 사용

- 공간정보는 각 정보별 인덱스(지번주소, 도로명주소, 층, 호, 등기고유번호 등)의 정보를 보유함과 동시에 공간 좌표(위경도, 폴리곤 등)를 포함
- 빅밸류는 공간 좌표별 지정된 반경 거리내의 공간정보를 묶어서 분산 저장하는 방식으로 반경 단위 데이터 조회 속도를 높임
- 최신 공간정보와 과거 공간정보의 조회 빈도수 차이, 지역별 조회 빈도수 차이 등을 모니터링하여 조회가 활발한 정보는 HOT Storage로 이동하고 조회가 적은 정보는 Cold Storage로 이동시켜 스토리지 레벨을 제어한다.