빅데이터

빅데이터 시대 도래


메인 프레임
컴퓨터
PC 시대
인터넷 ·
모바일 시대
스마트 시대
인공지능
사물 정보 통신
증강 현실
모바일 혁명
SNS
웹 2.0
1인 1PC
브로드밴드
빅데이터
시대 도래
1970
1980
1990
2000
2010
2020
2030

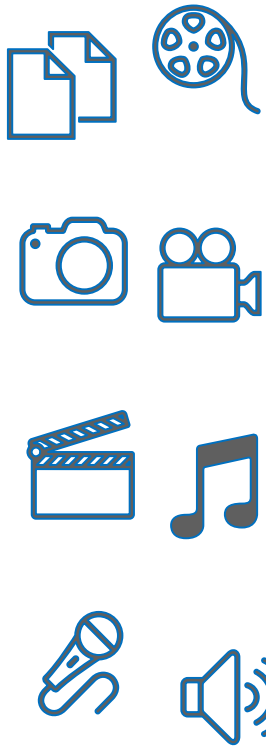
인간과 데이터

Real World

Perceive

Human

Express



Data

센서와 데이터

Weather

Sense

Thermometer

Report

3°C, 15°F

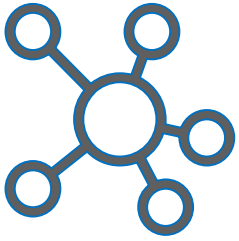
Locations

Sense

Geo Sensor

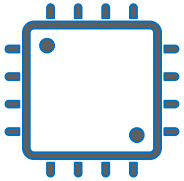
Report

132°E, 37°N



Networks

Sense



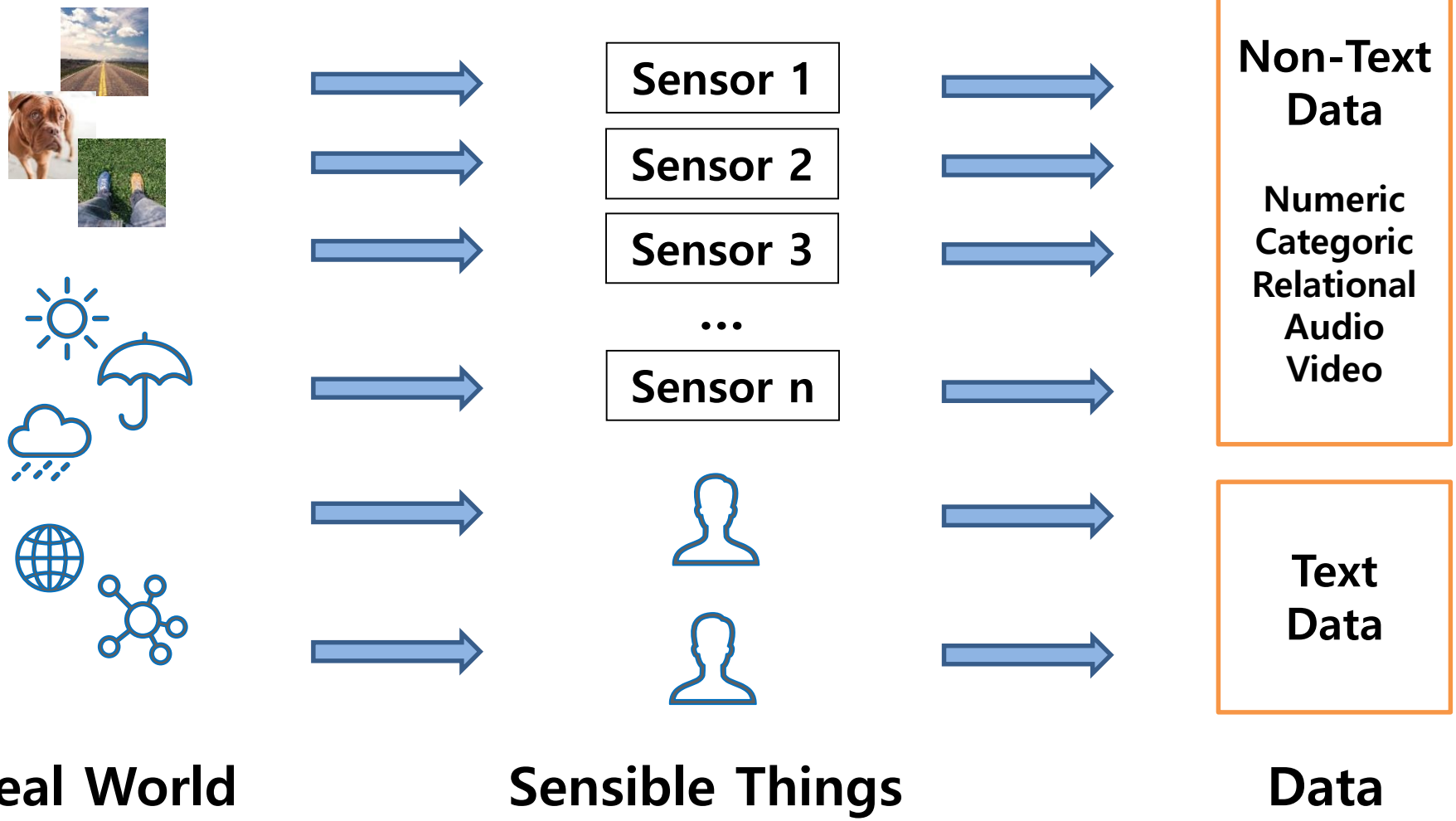
Network Sensor

Report

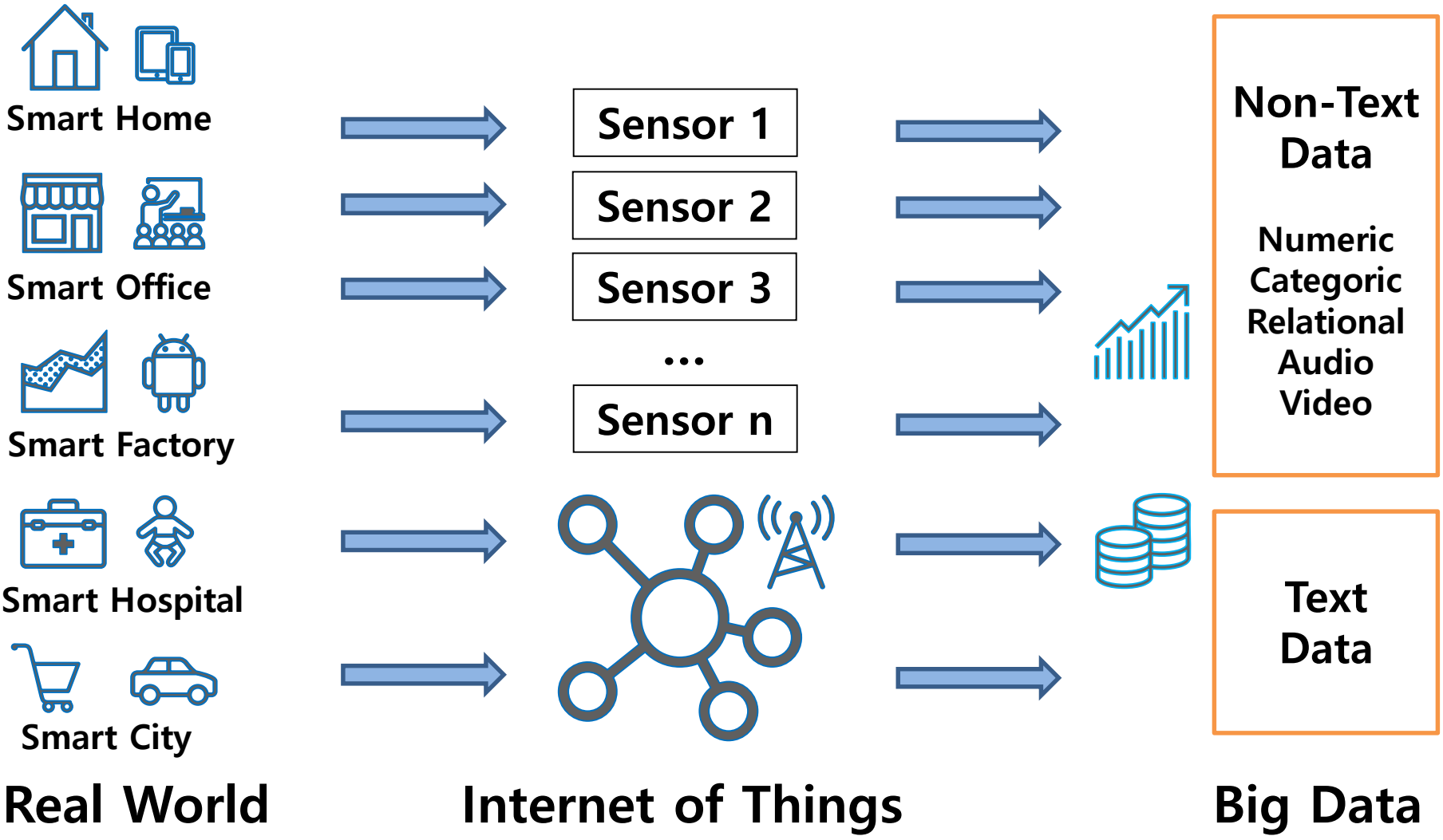
100011001

Data

센싱가능한 사물과 데이터



사물인터넷과 빅데이터


Smart Home
Smart Office
Smart Factory
Smart Hospital
Smart City
Sensor 1
Sensor 2
Sensor 3
...
Sensor n
Non-Text Data
Numeric
Categoric
Relational
Audio
Video
Text Data
Real World
Internet of Things
Big Data

빅데이터 특성

빅데이터 특성 (3V)

➢ **데이터 볼륨 증가 (Volume)**

✓ 지난 10년간 생성된 데이터보다 최근 2년간 생성된 데이터양이 훨씬 많음

✓ 보통 수백 테라바이트(TB: Tera Byte)에서 수 페타바이트(PB: Peta Byte) 이상의 대용량

KB (키로) < MB (메가) < GB (기가) < TB (테라) < PB (페타) < EB (엑사) < ZB (제타) < YB (요타)

빅데이터 특성 (3V)

➢ 데이터 포맷 다양성 증가 (Variety)

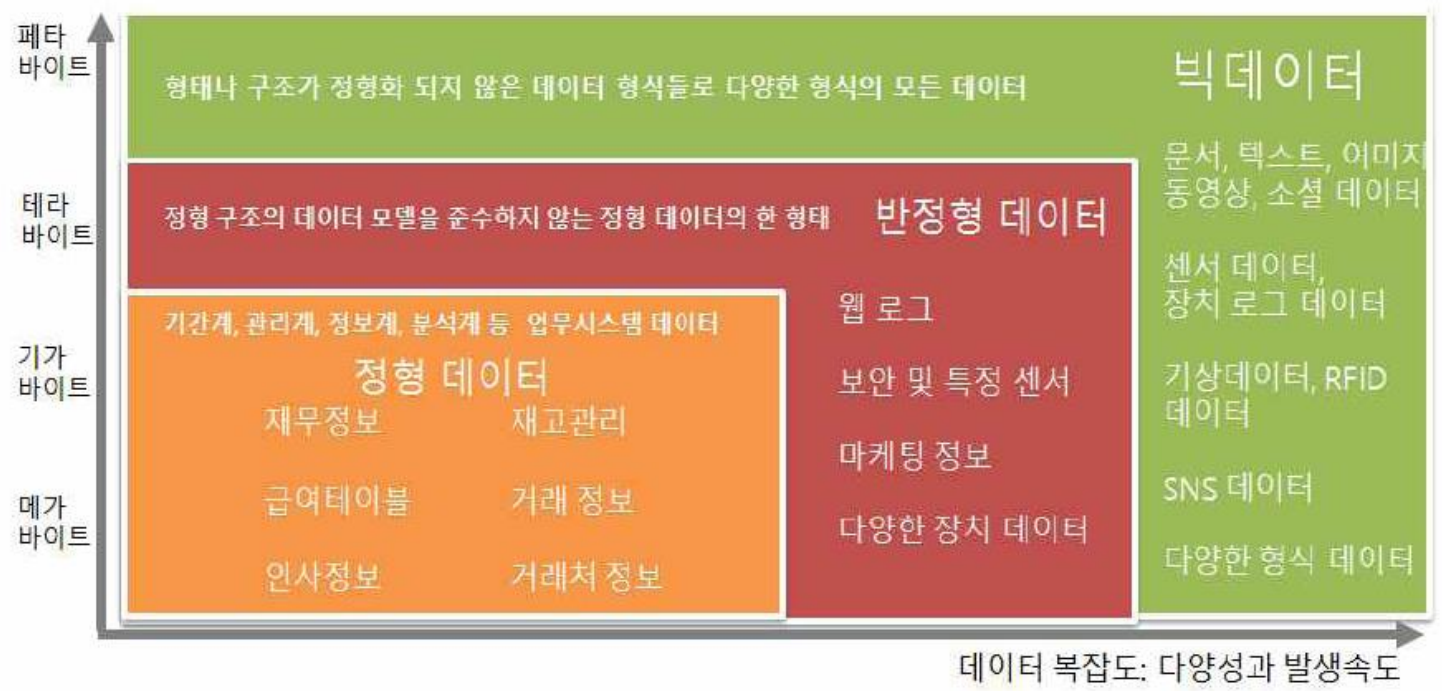
✓ 기존의 정형화된 데이터 뿐만 아니라 그림, 동영상, 음성, 로그, 센서 데이터 스트림 등 비정형화된 다양한 데이터들이 생성됨

종류	설명
정형	고정된 필드에 저장된 데이터 예 관계형 데이터베이스, 스프레드시트
반정형	고정된 필드에 저장되어 있지는 않지만, 메타데이터나 스키마 등을 포함하는 데이터 예 XML, HTML 텍스트
비정형	고정된 필드에 저장되어 있지 않은 데이터 예 텍스트 분석이 가능한 텍스트 문서, 이미지 · 동영상 · 음성 데이터

빅데이터 특성 (3V)

➢ 데이터 발생 속도 증가 (Velocity)

✓ 오늘날 데이터양의 85% 이상은 모바일, SNS, 센서, RFID 태그 등을 통해서 과거와 비교할 수 없는 속도로 데이터들이 생성됨



새로운 빅데이터 처리 프레임워크 필요

➢ 빅데이터 3V 특성에 맞는 새로운 빅데이터 처리 프레임워크가 필요함

- ✓ 많은 데이터를 처리해야 함
- ✓ 실시간 데이터를 처리해야 함
- ✓ 저비용으로 처리해야 함
- ✓ 결함 허용이 되는 시스템이 필요함

프레임워크(Framework)

- 소프트웨어 어플리케이션이나 솔루션의 개발을 수월하게 하기 위해 소프트웨어의 구체적 기능들에 해당하는 부분의 설계와 구현을 재사용 가능하도록 협업화된 형태로 제공하는 **소프트웨어 환경**

- 소프트웨어 패러다임
  - ✓ 하드웨어 중심 → 프로그램 언어 중심 → 객체지향 및 웹 중심
  - ✓ 현재는 컴포넌트, 프레임워크, 오픈소스 소프트웨어 중심

실시간 데이터 처리 기술 필요

➢ 실시간 데이터 처리 → 통합 분석할 수 있는 데이터 처리 기술이 필요

- ✓ 트위터나 페이스북 같은 SNS에서 발생하는 실시간 데이터의 처리
- ✓ 데이터를 실시간으로 분석 → 사용자의 패턴 파악 → 의사 결정
- ✓ RFID, IoT 센서 데이터 및 각종 소스에서 발생하는 데이터의 처리

이벤트 스트리밍 데이터 처리 필요

- 이벤트를 관리하면서 데이터 질의 및 분석을 통합 처리할 수 있어야 함
  - ✓ 은행에서 결제 및 지불 관련 이상 거래를 자동으로 탐지
  - ✓ 개인 정보 유출에 대비 → 비정상적인 방법으로 데이터 접근
  - ✓ 대량의 이벤트를 신속하게 해독하고 분석
  - ✓ 저장된 데이터에 대해 쿼리를 실행하는 것이 아니라 데이터 관리 및 분석 루틴을 저장하고 이러한 쿼리를 통해 대량의 이벤트 스트리밍 데이터를 처리하면서 실시간으로 데이터를 필터링, 정규화, 취합하고 패턴을 탐지

스트리밍 데이터

- 수천 개의 데이터 소스에서 연속적으로 생성되는 데이터를 말하며, 보통 데이터 레코드를 작은 크기(KB 단위)로 동시에 전송함

  - ✓ 모바일이나 웹 애플리케이션을 사용하는 고객이 생성하는 로그 파일, 전자 상거래 구매, 게임 내 플레이어 활동, 소셜 네트워크의 정보, 주식 거래소, 데이터 센터의 계측 등의 데이터

빅데이터 처리 시스템

- 대용량 데이터를 분산 병렬 처리하고 관리하는 시스템으로 사용자에게 데이터의 유형에 따라서 실시간(Real-time) 처리나 배치(Batch) 처리를 할 수 있도록 하는 프레임워크

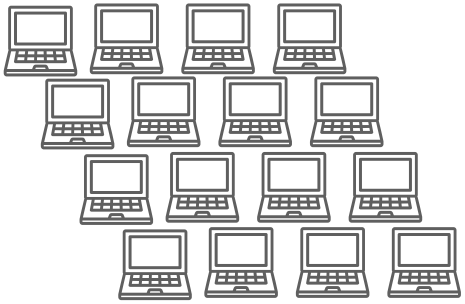
- 대규모 양의 데이터의 수집, 관리, 유통, 분석을 처리하는 일련의 분산 병렬처리 프레임워크

빅데이터 처리 시스템 설계 원칙 (1)

➢ 결함 허용 시스템(Fault Tolerance)

1대의 컴퓨터 → 1년에 한번씩 장애발생

365대의 컴퓨터 → 하루에 한번씩 장애발생



빅데이터 처리를 위한 클러스터(Cluster) → 매시간에 한번씩 장애발생

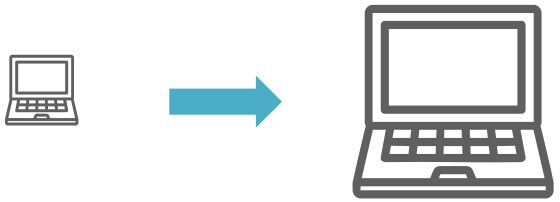
빅데이터 처리 시스템 설계 원칙 (1)

- **결함 허용 시스템**(Fault Tolerance)
  - ✓ 모든 시스템은 애플리케이션 문제 또는 하드웨어 리소스 문제로 인해 장애가 발생할 수 있음
  - ✓ 이러한 장애가 발생하여도 시스템이 대체 시스템이나 고장 대응체계를 통해서 시스템 운영을 계속하는 것
  - ✓ 빅데이터 처리를 위해서는 장애가 발생하여도 버티고 수행하는 능력인 "결함 허용 시스템" 을 갖추어야 함

빅데이터 처리 시스템 설계 원칙 (2)

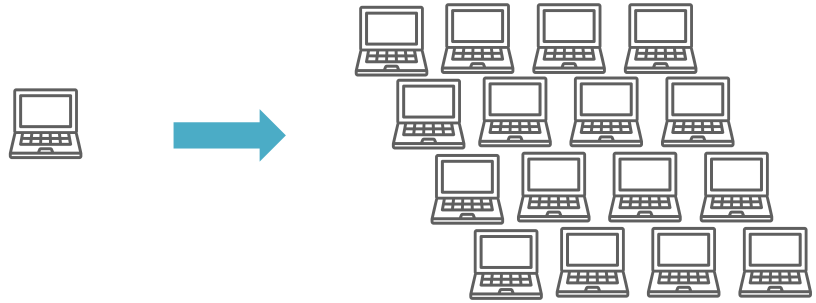
➢ 저 비용 시스템(Cost Effective System)

- 수직 확장 (Scale Up)



❖ 확장성 한계

- 수평 확장 (Scale Out)



❖ 확장성 증대

빅데이터 처리 시스템 설계 원칙 (2)

➢ **저 비용 시스템**(Cost Effective System)

✓ **빅데이터라고 해서 항상 큰 비용이 드는 것은 아님**

✓ **작업에 적합한 도구를 적절하게 선택하여 데이터 처리에 드는 비용을 절감할 수 있음**

✓ **처리할 데이터 구조, 답변시간, 처리량에 따라서 적절한 비용에 따라 툴을 선택**

빅데이터 처리 시스템 설계 원칙 (2)

➢ **저 비용 시스템**(Cost Effective System)

✓ **시스템 구축 시 장점**

- **초기 도입 비용이 저렴하여 가격 경쟁력이 높음**
- **저렴한 비용의 성능이 낮은 서버로 시스템 구성이 가능**
- **오픈 소스 도입으로 상용 소프트웨어 도입으로 인한 라이선스 비 절감**

빅데이터 처리 시스템 설계 원칙 (3)

➢ 기존 시스템과 연계성

- ✓ 소셜, 시스템 로그, 텍스트, 멀티미디어, 센서 로그 등 다양한 데이터를 수집하고 이를 적절한 시스템에 저장하여 처리하게 됨

- ✓ 빅데이터 처리 시스템은 이러한 다양한 데이터 종류에 대한 수집 및 처리를 할 수 있어야 하며, 이는 기존 시스템과 연계성이 필요함

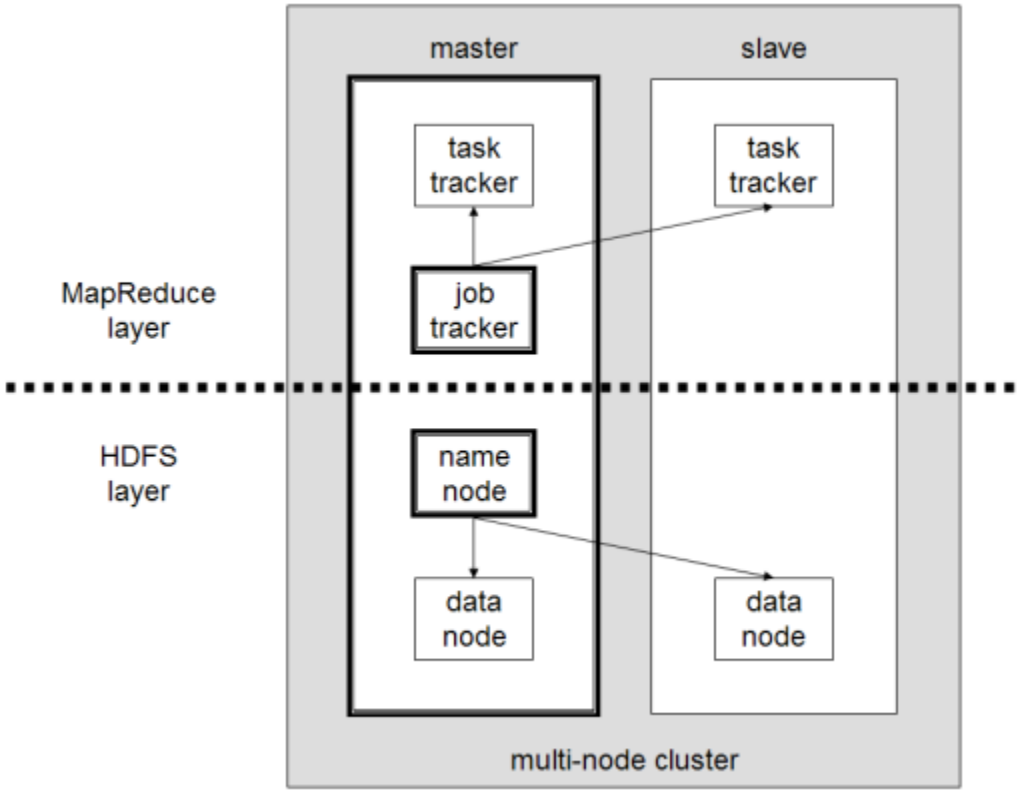
- ✓ 예를들면, 기존 DBMS와 하둡 시스템을 연계할 수 있어야 함

빅데이터 처리 시스템의 원조 : 하둡 (Hadoop)

- 2003년 – 구글 파일 시스템 (GFS)
- 2004년 – 구글 맵리듀스 (MapReduce)
- 2005년 – 아파치 넛지 (Nutch) 프로젝트 (더그 커팅, 마이크 카페렐라)
- 2006년 – 야후 하둡 프로젝트 탄생 (더그 커팅)
- 2008년 – 클라우데라 (Cloudera) 창립
- 2011년 – 호튼웍스 (Hotonworks) 창립

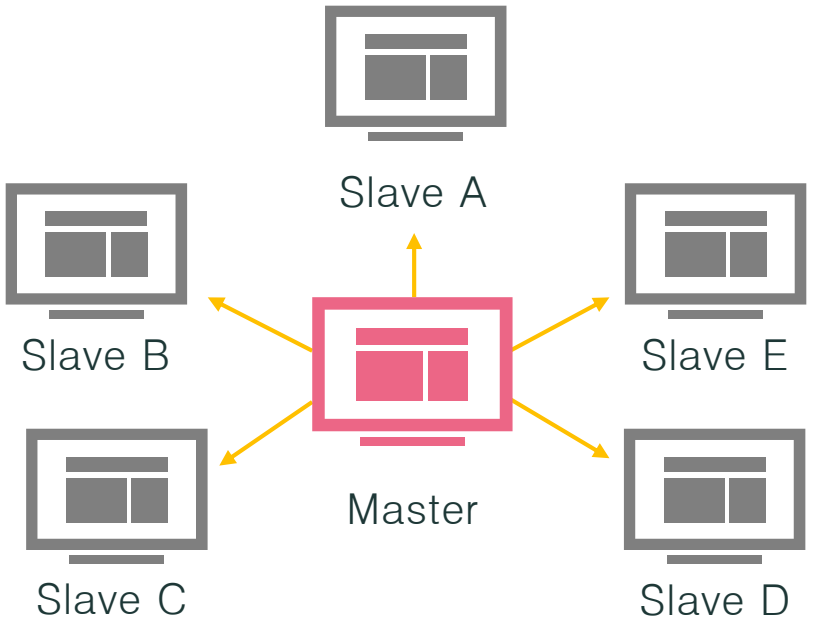
하둡 1.0

- 맵리듀스 (MapReduce) 분산 처리 계층
- 하둡 분산 파일 시스템 (HDFS) 계층



하둡(Hadoop) 맵리듀스 프레임워크

- 빅데이터 처리 시스템의 원조
- 분산 병렬 처리 방식으로 여러 개의 작업 노드에 작업을 분산하여 병렬 수행할 수 있는 프레임워크를 제공



맵리듀스 프레임워크

- 맵(Map)과 리듀스(Reduce) 함수를 합친 말

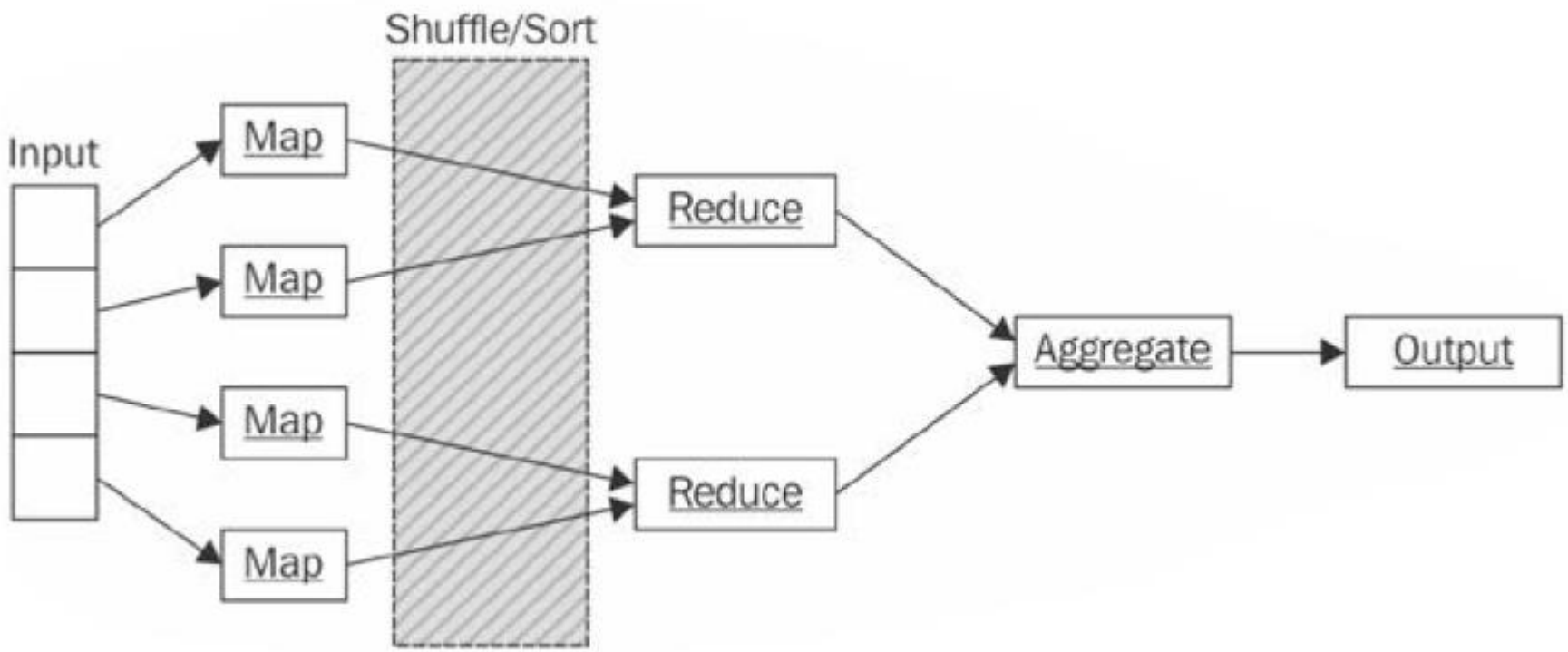
- 맵(Map) 단계
  - ✓ 분산된 데이터를 키(key)와 값(value)의 리스트로 모으는 단계

- 셔플(Shuffle and Sort) 단계
  - ✓ 맵 단계에서 나온 중간 결과를 해당 리듀스 함수에 전달하는 단계

- 리듀스(Reduce) 단계
  - ✓ 리스트에서 원하는 데이터를 찾아서 집계하는 단계

맵리듀스 프레임워크

➢ 데이터 흐름



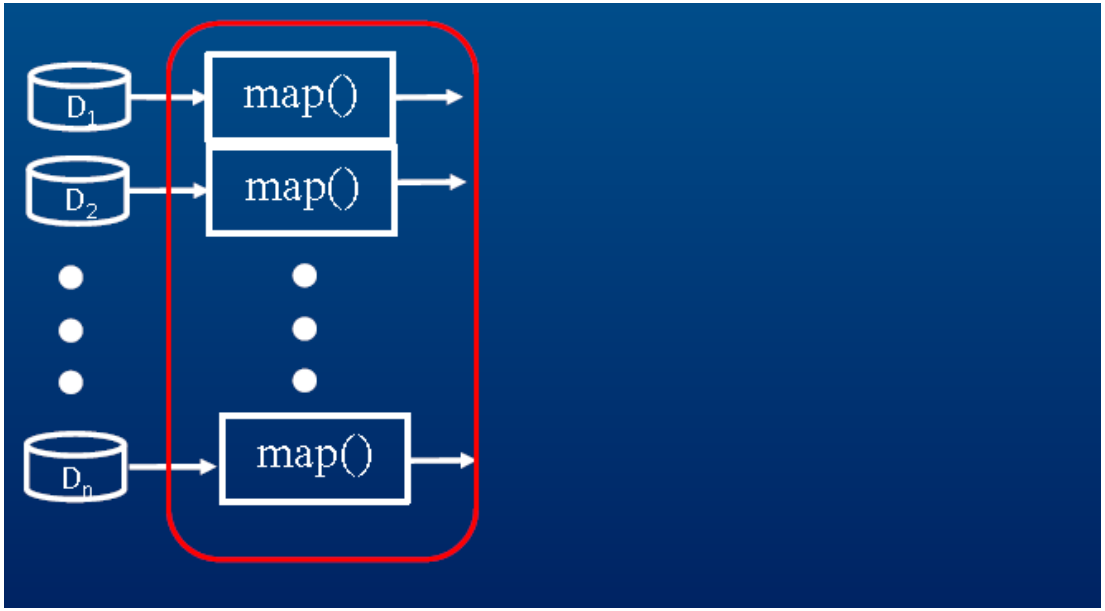
맵리듀스 분산 병렬 처리 방식

- 개발자 정의
  - ✓ 맵(Map) 함수
  - ✓ 리듀스(Reduce) 함수

- 분산병렬 처리는 시스템(하둡: Hadoop)이 자동으로 처리함

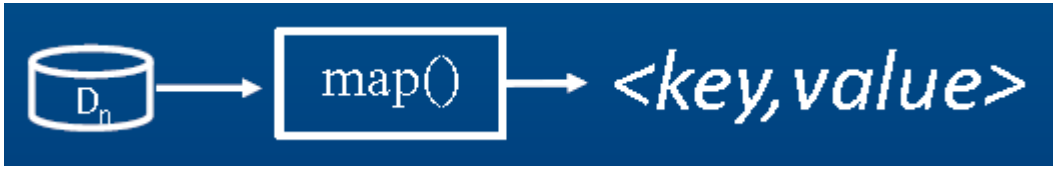
맵리듀스 분산 병렬 처리 방식

1) 시스템(하둡)이 map() 함수들을 데이터에 배포함



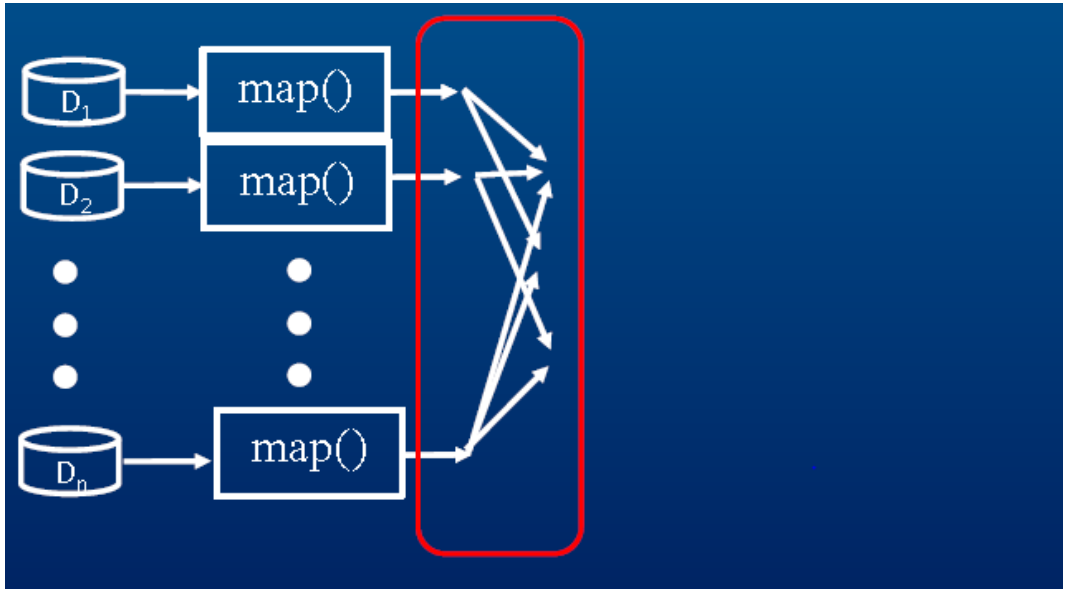
맵리듀스 분산 병렬 처리 방식

2) map() 함수는 데이터를 읽고 <key, value>를 출력함



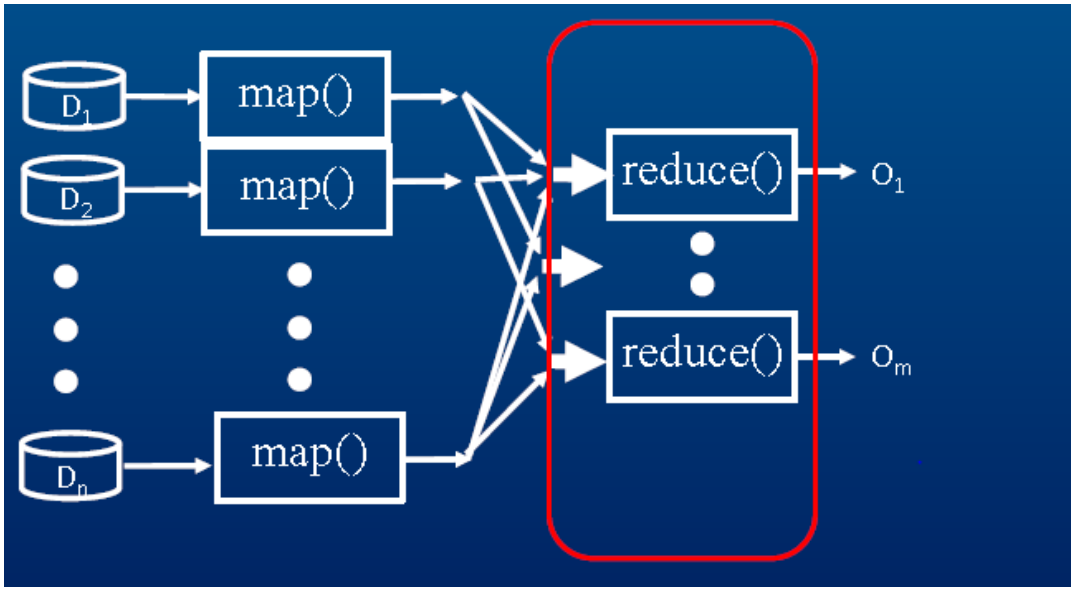
맵리듀스 분산 병렬 처리 방식

3) 시스템(하둡)이 <key, value> 데이터를 그룹핑함 → 셔플 및 정렬 (Shuffle and Sort)



맵리듀스 분산 병렬 처리 방식

4) 시스템(하둡)이 그룹핑된 데이터를 reduce() 함수들에 배포함

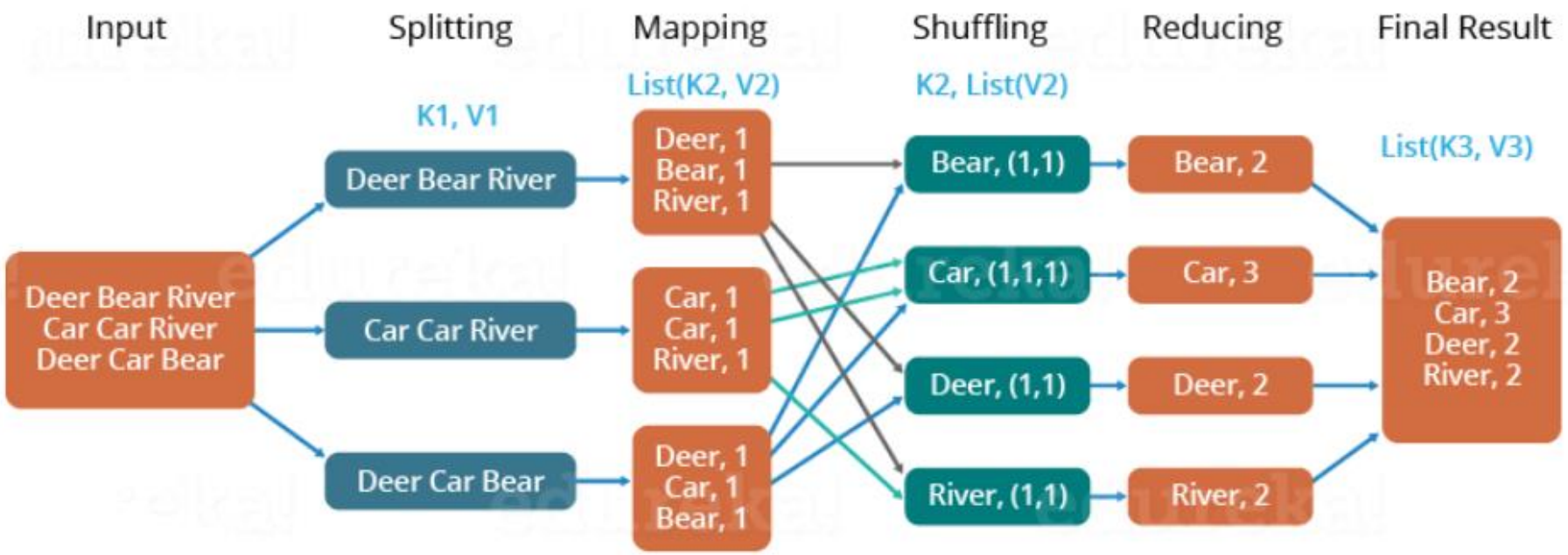


맵리듀스 분산 병렬 처리 방식

5) Reduce() 함수는 <key, value>를 읽고 결과를 출력함

맵리듀스 분산 병렬 처리 예제

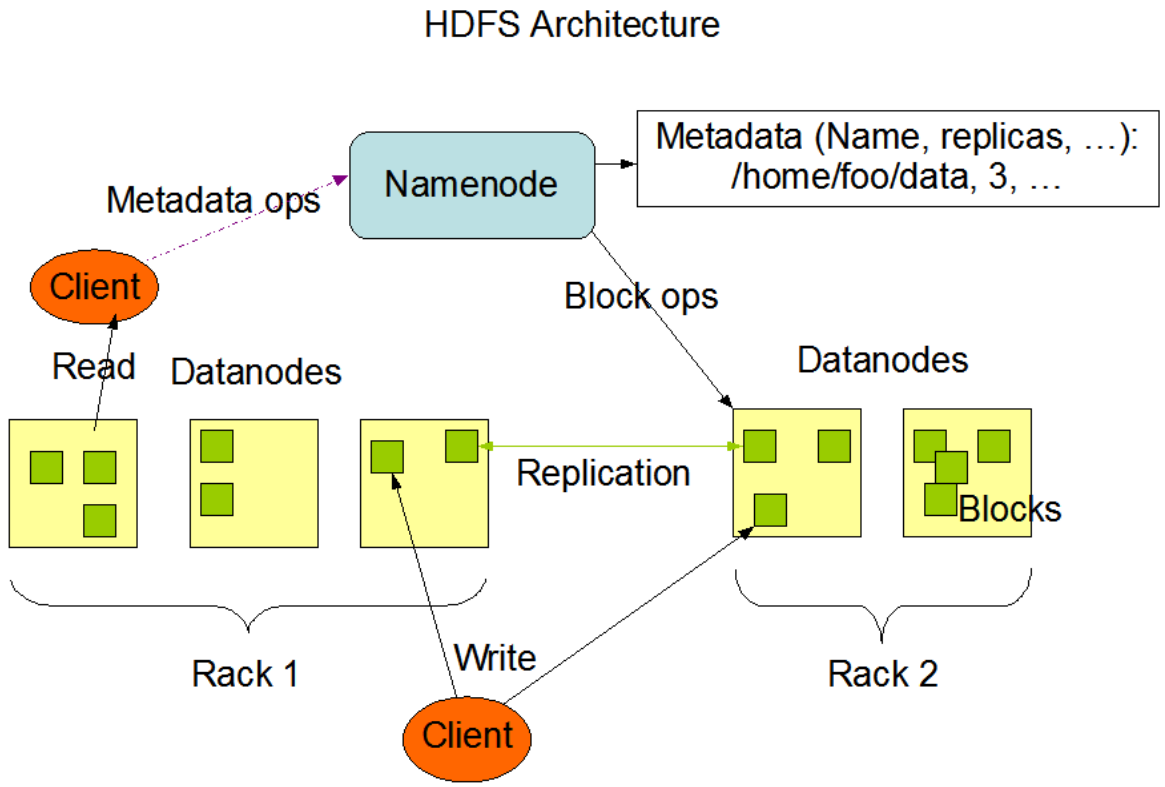
➢ Word Count 문제



출처: https://www.edureka.co/blog/mapreduce-tutorial/

하둡 분산 파일 시스템 (HDFS)

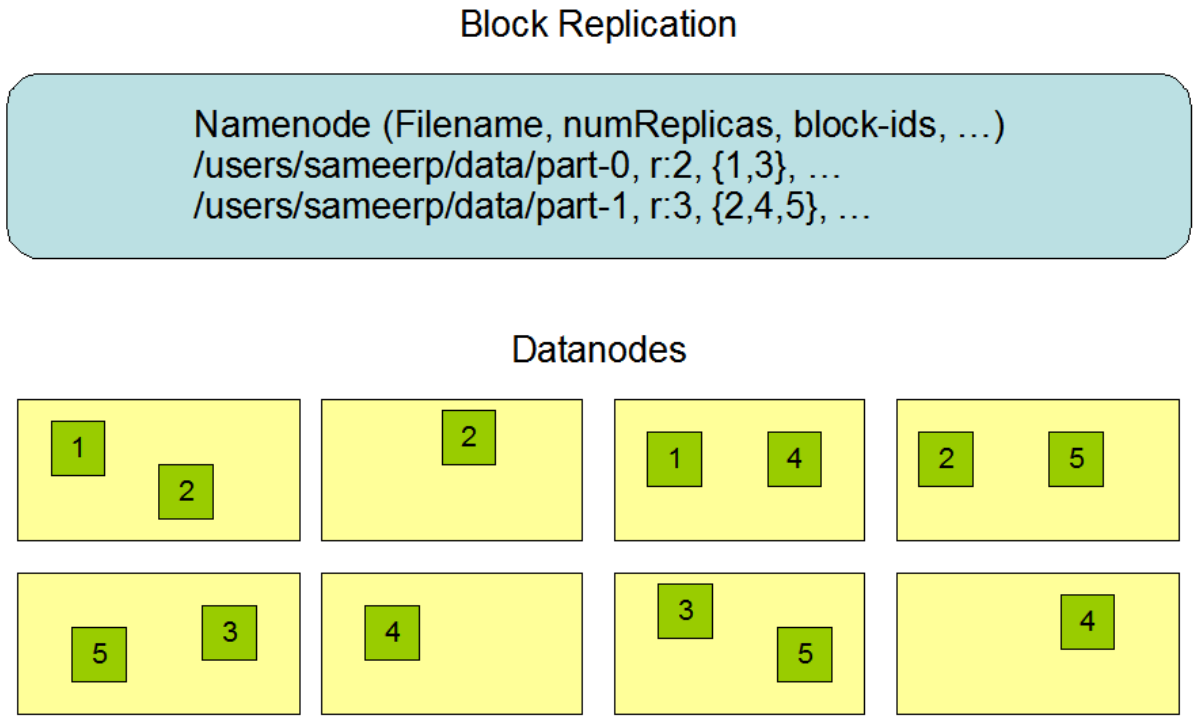
➢ 파일 저장 시 분산 저장 → 분산 파일 시스템



출처: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

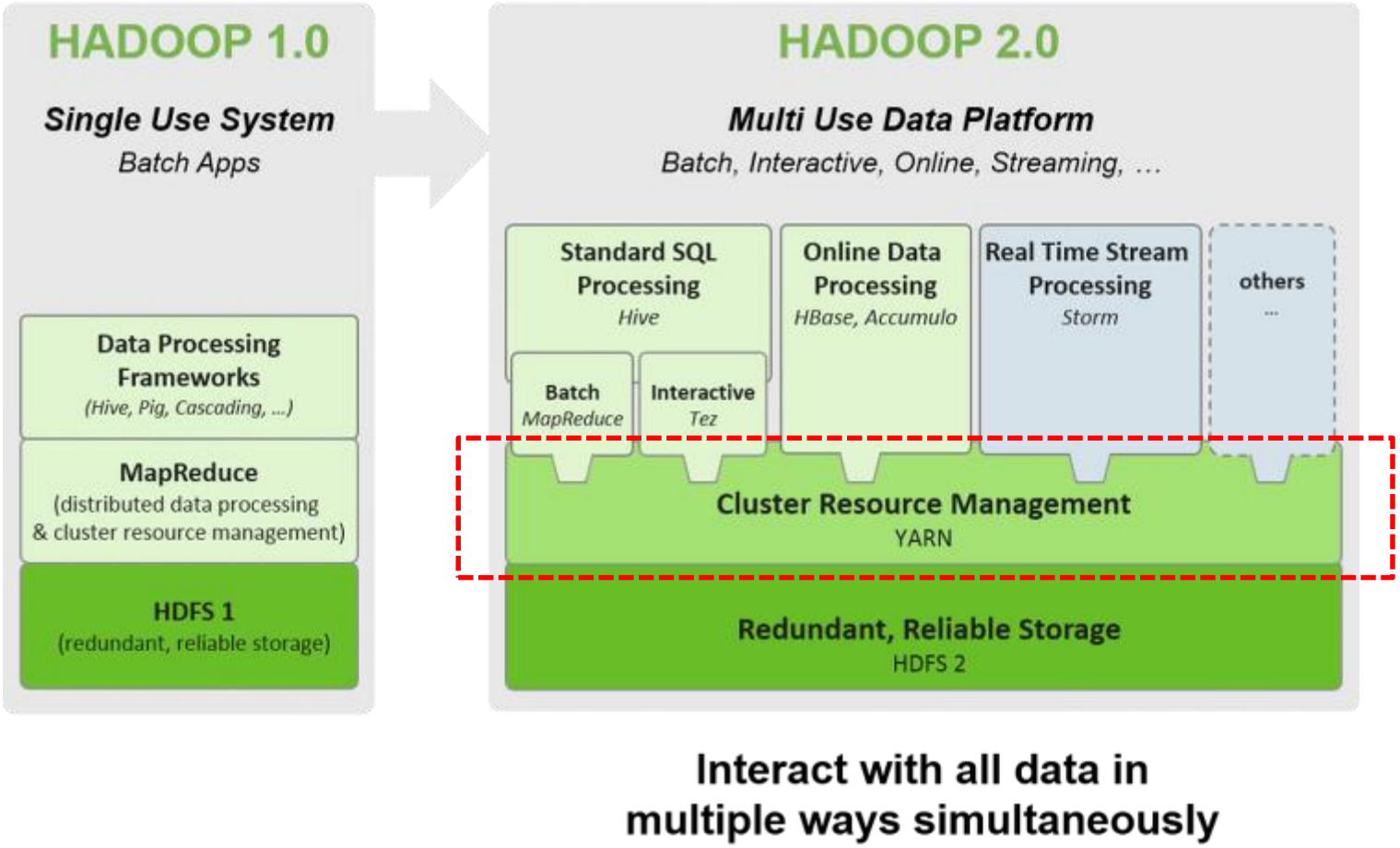
하둡 분산 파일 시스템

➢ 데이터 복제

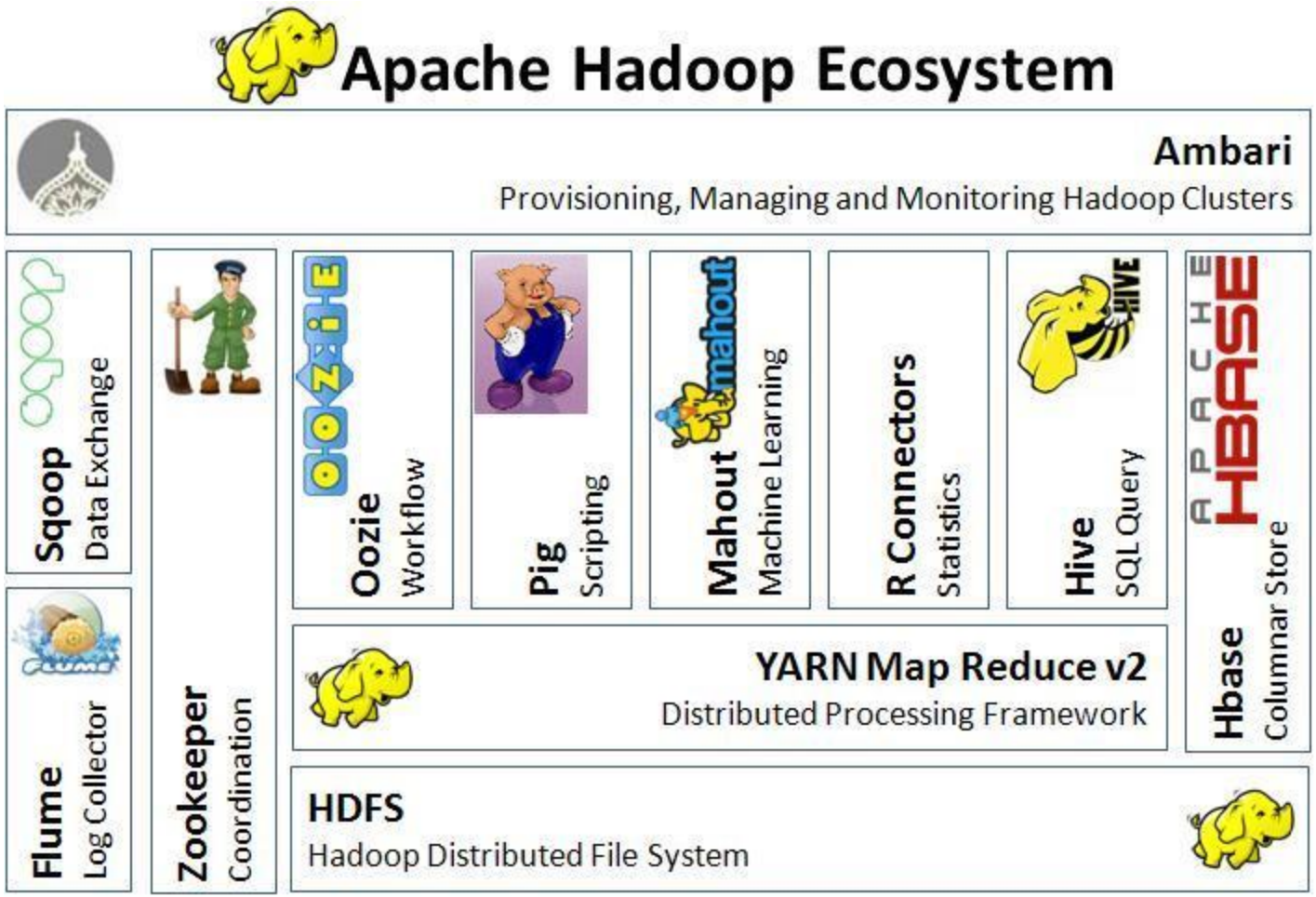


출처: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

하둡 2.0


HADOOP 1.0
Single Use System
Batch Apps
Data Processing Frameworks
(Hive, Pig, Cascading, ...)
MapReduce
(distributed data processing & cluster resource management)
HDFS 1
(redundant, reliable storage)
HADOOP 2.0
Multi Use Data Platform
Batch, Interactive, Online, Streaming, ...
Standard SQL Processing
Hive
Online Data Processing
HBase, Accumulo
Real Time Stream Processing
Storm
others
...
Batch
MapReduce
Interactive
Tez
Cluster Resource Management
YARN
Redundant, Reliable Storage
HDFS 2
Interact with all data in multiple ways simultaneously

하둡 에코 시스템


Apache Hadoop Ecosystem
Ambari
Provisioning, Managing and Monitoring Hadoop Clusters
Sqoop
Data Exchange
Flume
Log Collector
Zookeeper
Coordination
Oozie
Workflow
Pig
Scripting
Mahout
Machine Learning
R Connectors
Statistics
Hive
SQL Query
Hbase
Columnar Store
YARN Map Reduce v2
Distributed Processing Framework
HDFS
Hadoop Distributed File System