

Hadoop

하둡이란 무엇인가?

- 하둡이란?

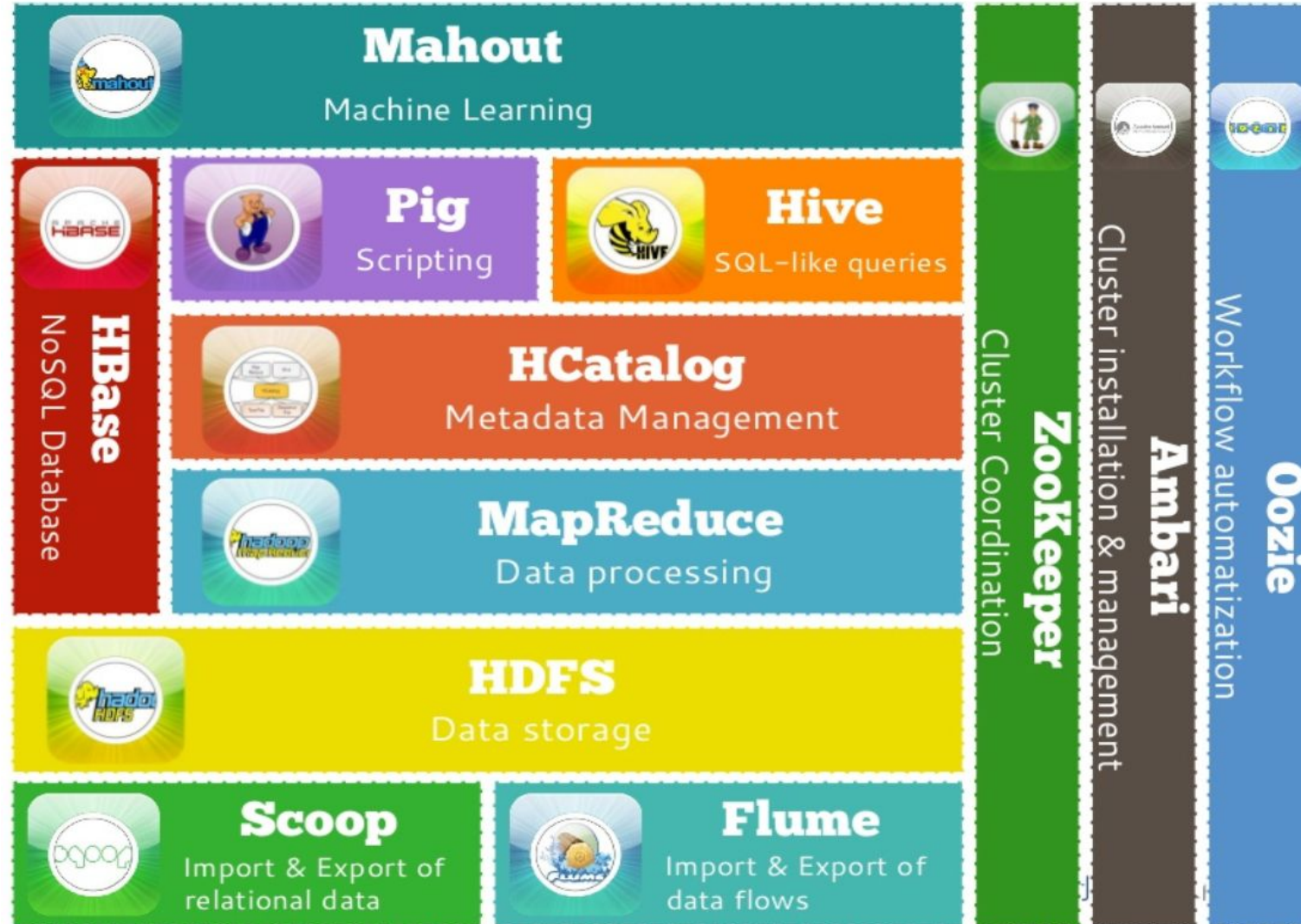
- 대규모 검색 색인을 구축하려고 자바로 개발된 오픈 소스 분산 컴퓨팅 플랫폼
- 하둡의 원래 개발 목적은 검색 색인에 있었지만, 사람들은 곧 하둡의 핵심 개념을 다른 일반적인 문제에도 폭넓게 적용
- 초창기부터 강조된 하둡의 핵심 기능은 장애 허용(fault tolerance)
 - 확장성(scalability)을 높이기 위해 장애를 당연히 발생할 수 있는 일로 간주
 - 기반 소프트웨어 시스템이 실패한 작업을 책임지고 재시도하게 설계
 - 이로 인해 다소 불안정하지만 저렴한 하드웨어로도 매우 안정적인 시스템을 구성할 수 있음

- 하둡의 핵심 기술

- 분산 파일 시스템
- 리소스 관리자와 스케줄러
- 분산 데이터 처리 프레임워크

하둡 에코시스템

- 하둡에코시스템



리소스 관리자와 스케줄러

- YARN(Yet Another Resource Negotiator)

- 하둡의 클러스터 관리 시스템
- 가장 효율적인 방법으로 계산 리소스를 할당하고 사용자 애플리케이션을 스케줄링 하는 시스템
- 스케줄링과 리소스 관리로 데이터 지역성을 극대화하고 계산량이 많은 애플리케이션이 리소스를 독점하지 않게 제어
- 교체 가능한 스케줄링 시스템을 지원
- 사용자당 리소스 제한이나 작업 대기열당 리소스 할당량 등 공용 리소스 시스템의 스케줄링에 필요한 기본적인 환경 설정을 스케줄러에 입력할 수 있음

- 구성

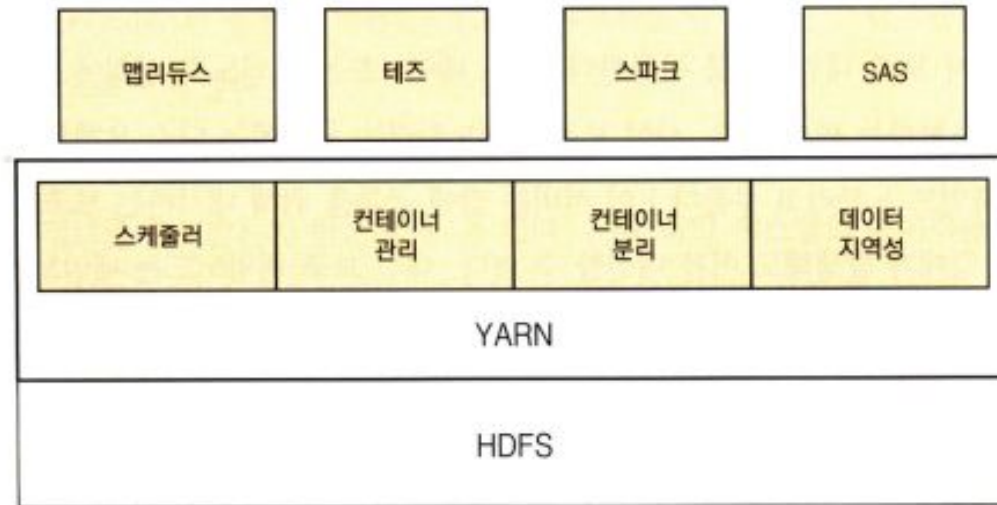
- Resource Manager라고 불리는 마스터 노드
 - 클러스터 전체의 계산 리소스를 관리하고, 클라이언트가 요구한 리소스를 NodeManager로부터 확보하도록 스케줄링 함
 - 요청된 이그제큐터 개수와 cpu 코어의 수, 메모리 양에 따라 이그제큐터를 하나 이상의 NodeManager로부터 확보하는 역할을 담당
- NodeManager라고 하는 여러 개의 워커 노드
 - 자신이 설치된 노드의 계산 리소스만을 관리

리소스 관리자와 스케줄러

- YARN

- 클러스터의 리소스를 컨테이너로 분할
- 컨테이너는 기본적으로 할당되는 CPU 코어 수와 메모리 용량으로 정의되며 추가 리소스를 포함할 수도 있음
- 실행중인 컨테이너들을 모니터링하면서 컨테이너가 리소스의 최대 할당량을 초과하지 않게 억제
- 클러스터의 리소스를 컨테이너로 관리함으로써 분산 시스템을 전체적으로 원활하게 운영하고, 클러스터의 리소스를 다수의 애플리케이션에 공평한 방식으로 공유
- 컨테이너를 비공개로 설정할 수도 있고, 사용자가 요청한 작업을 적절한 시점에 시작할 수 도 있음

▼ 그림 3-2 HDFS와 YARN, 그리고 다양한 처리 엔진(맵리듀스, 스파크, 테즈)을 포함하는 하둡 아키텍처



분산 데이터 처리 프레임워크

- 맵리듀스

- HDFS는 파일의 저장과 관리에 중점을 뒀다면 앞으로 살펴볼 맵리듀스는 HDFS에 저장된 파일에서 데이터를 추출해 가공, 분석, 병합해 의미 있는 결과를 도출하기 위한 일종의 프로그램이다.
- 자바 언어를 지원하는 맵리듀스 프레임워크는 분산 환경에서 병렬 처리를 쉽게 할 수 있게 돕는다.
- 맵리듀스에서 제공하는 몇 개의 클래스를 상속받아 규칙에 따라 코딩하면 HDFS에 저장된 데이터를 병렬 처리할 수 있음

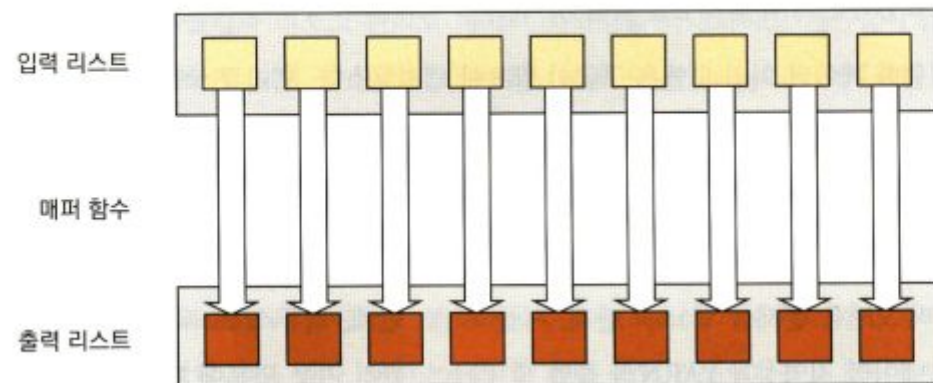
- 분산된 저장 환경에서 저장된 노드에서 실행하여 결과를 모으는 개념

분산 데이터 처리 프레임워크

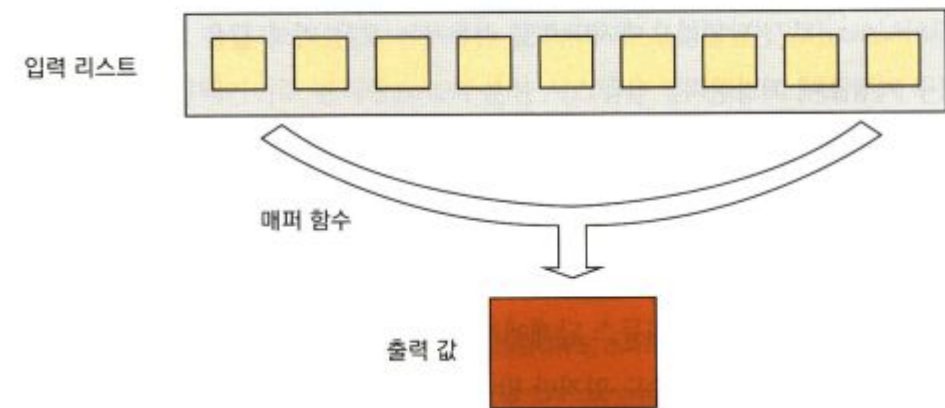
• 매키듀스

- 병렬 처리 모델은 문제를 map, shuffle, reduce 단계로 나눠 수행
- map
 - 입력 데이터가 클러스터에서 병렬로 처리되며 이 맵 단계를 수행하는 mapper 함수는 원시 데이터를 key와 value의 쌍으로 변환
- shuffle
 - 변환된 데이터는 키를 기준으로 정렬돼 bucket으로 셔플링된다.
- reduce
 - 모든 키의 값을 처리하며 결과를 HDFS나 다른 영구 저장소에 저장

♥ 그림 3-3 맵 단계: 입력 리스트는 독립적인 블록으로 나뉘어 HDFS에 저장돼 있다. 맵퍼 함수는 각 블록에 맞게 병렬로 실행된다. 출력 리스트는 키-값 쌍의 집합이다.



♥ 그림 3-4 리듀스 단계: 맵 단계의 출력 리스트가 리듀스 단계의 입력 리스트로 연결된다. 여러 리듀서가 사용될 경우, 입력 리스트는 키 값으로 묶여 특정 리듀서로 전달된다. 리듀서는 입력 리스트를 출력 값으로 병합(리듀스)한다.



아파치 스쿱(Sqoop)

- 아파치 스쿱

- 하둡과 정형(structured) 데이터베이스간의 효율적인 대용량 벌크 데이터 전송을 지원하는 도구
- 외부 시스템의 데이터를 HDFS로 가져와 하이브나 HBase 테이블에 삽입 할 수 있음
- 새로운 유형의 외부 데이터 소스를 연동할 수 도 있음
 - MySql, PostgreSQL, Oracle, SQL Server, DB2

아파치 플럼(Flume)

- 아파치 플럼

- 서버에서 생성되는 대용량 로그 데이터를 효율적으로 수집해 HDFS로 전송하는 안정적인 분산 서비스
- 단순하고 유연한 아키텍처는 소스(source)에서 저장소(sink)로 데이터를 보내는 역할을 하는 에이전트로 구성

- 플럼 사용

- 플럼 에이전트가 최소 두 개 필요(소스에 하나, 저장소에 하나)
- 여러 소스에서 데이터를 수집할 수 있으며 여러 플럼 에이전트를 파이프라인으로 구성할 수도 있음

- 아파치 우지(Oozie)와 팔콘(Falcon)

- 데이터 이동, 데이터 파이프라인 조정, 데이터 생명 주기 관리, 데이터 검색에 이르는 다양한 단계의 관리 기능을 제공
- 팔콘은 데이터의 최종 소비자가 하둡 클러스터에 저장된 데이터와 이 데이터를 처리하고 관리하는 작업을 빠르게 파악할 수 있는 데이터 관제(governance) 기능을 제공

아파치 하이브(Hive)

- 하이브

- SQL로 하둡 데이터를 분석하려는 페이스북 엔지니어들의 요구로 페이스북 내부에서 개발
- 오픈 소스로 공개돼 많은 개발자가 활발하게 참여하는 프로젝트가 되었음
- 현재는 실행 속도가 빨라져 대화형 쿼리와 실시간 쿼리까지 지원

- 기본적인 작동 원리

- 사용자가 SQL 쿼리를 작성하면 이것을 자동으로 맵리듀스 작업으로 변경해서 클러스터에서 실행

- 기본 구성과 특징

- 실행 부분 : 쿼리->맵리듀스 실행
- 메타데이터 정보 : Mysql과 같은 RDBMS에 저장

- Hive와 RDBMS의 차이점

- 작은 데이터일 경우 응답 속도가 느리다.
- 레코드 단위의 Insert, Update, Delete를 지원하지 않는다.
- 트랜잭션을 지원하지 않는다.
- 통계정보를 바로 확인할 수 없다.
- 입력값 오류도 바로 확인할 수 없다.

아파치 피그(Pig)

- 피그

- 하둡의 복잡한 추출, 변환, 적재(Extract, Transform, Load) 작업을 쉽게 구현하려는 요구로 야후에서 개발
- ETL 작업을 지원하기 위해 설계된 도메인 특화 언어
- 관계형 기본 함수를 제공하며, 최소 개수의 맵리듀스 또는 테즈 작업으로 구성된 논리 단계로 연산을 변환하고 실행하는 최적화 기능을 갖추고 있음
 - 스파크를 처리 엔진으로 사용하는 Pig-on-Spark라는 프로젝트가 존재

분산 파일 시스템

- 하둡의 핵심 파일 시스템으로 처음 개발된 HDFS(Hadoop Distributed File System)이 널리 사용됨
 - HDFS 내부에 블록 형태로 저장되며, 블록은 HDFS에 의해 투명하게 복제돼 여러 노드에 분산
 - HDFS의 리플리케이션(replication) 메커니즘은 데이터를 단순히 여러 노드에 저장하는 방식이 아닌, 데이터가 여러 랙(rack)에 분산 저장되도록 보장하는 다양한 전략을 사용함
 - 이러한 리플리케이션 전략은 단일 노드나 단일 랙의 장애가 데이터의 유실로 이어지는 사태를 방지
 - `hdfs-site.xml`에서 `dfs.replication`의 디폴트 값은 3이며, 이는 각 블록에 대한 복제계수이다.
- HDFS는 일반적인 파일 시스템을 가진 여러 노드를 묶어 하나의 분산 파일 시스템을 구축하도록 설계
 - 손쉽게 파일 시스템을 확장해 몇 페타바이트에 이르는 대용량 데이터까지 저장할 수 있음
- HDFS의 설계
 - 1. 데이터의 풀스캐닝을 지원하기 위해 파일 순차 읽기 속도가 빨라야 함
 - 2. 데이터가 계산이 수행되는 곳으로 옮겨지는 게 아니라, 데이터가 저장된 곳에서 계산이 수행될 수 있게 파일 시스템의 노드들이 각자 자신이 저장한 데이터의 위치 정보를 충분히 교환해야 함
 - 3. 노드의 장애를 소프트웨어 레이어에서 극복해야 함

hadoop cluster 3가지 mode

- hadoop cluster는 3가지 mode를 지원
 - Local (Standalone) Mode
 - 하나의 장비에 단일 java 프로세스로 실행되도록 구성
 - Pseudo-Distributed Mode
 - 하나의 장비에 분산 java 프로세스로 실행되도록 구성
 - Fully-Distributed Mode
 - 여러 장비에 분산 java 프로세스로 실행되도록 구성

하둡의 기본 클러스터
설정 파일은
core-site.xml

구분	하둡 설정 파일	
로컬모드	hadoop-local.xml	기본 로컬 파일시스템과 맵리듀스 잡을 실행하는 로컬(단일 JVM) 프레임워크에 적합한 하둡 설정을 가진다.
의사분산모드	hadoop-localhost.xml	로컬에서 작동하는 네임노드(namenode)와 YARN 리소스 매니저(resource manager)의 위치를 설정한다.
클러스터모드	hadoop-cluster.xml	클러스터의 네임노드와 YARN 리소스 매니저의 주소를 가진다. 실무에서는 예제에 사용된 'cluster'라는 이름 대신 실제 클러스터의 이름을 사용할 것을 권장한다.

hadoop cluster 3가지 mode

- cluster 3가지 환경 예시

•로컬모드

```
<?xml version="1.0"?>
<configuration>

  <property>
    <name>fs.defaultFS</name>
    <value>file:///</value>
  </property>

  <property>
    <name>mapreduce.framework.name</name>
    <value>local</value>
  </property>

</configuration>
```

•의사분산모드

```
<?xml version="1.0"?>
<configuration>

  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost/</value>
  </property>

  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

  <property>
    <name>yarn.resourcemanager.address</name>
    <value>localhost:8032</value>
  </property>

</configuration>
```

•클러스터모드

```
<?xml version="1.0"?>
<configuration>

  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://namenode/</value>
  </property>

  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

  <property>
    <name>yarn.resourcemanager.address</name>
    <value>resourcemanager:8032</value>
  </property>

</configuration>
```

NameNode, DataNode

- namenode

- HDFS에서 마스터 역할을 하며, 슬레이브 역할을 하는 datanode에게 I/O 작업을 할당한다.
- namenode가 손상되면 datanode에 저장된 블록으로부터 파일을 재구성하는 방법을 알 수 없기 때문에 파일시스템의 모든 파일을 찾을 수 없게 됨.
- datanode와 주기적으로 모니터링 하면서 datanode의 용량이 다 차면 다른 datanode로 블록을 이동 가능하도록 만듦
- 파일시스템 트리와 그 트리에 포함된 모든 파일과 디렉터리에 대한 메타데이터를 유지
- 이 정보는 네임스페이스 이미지와 에디트 로그라는 두 종류의 파일로 로컬 디스크에 저장

- secondaryNode

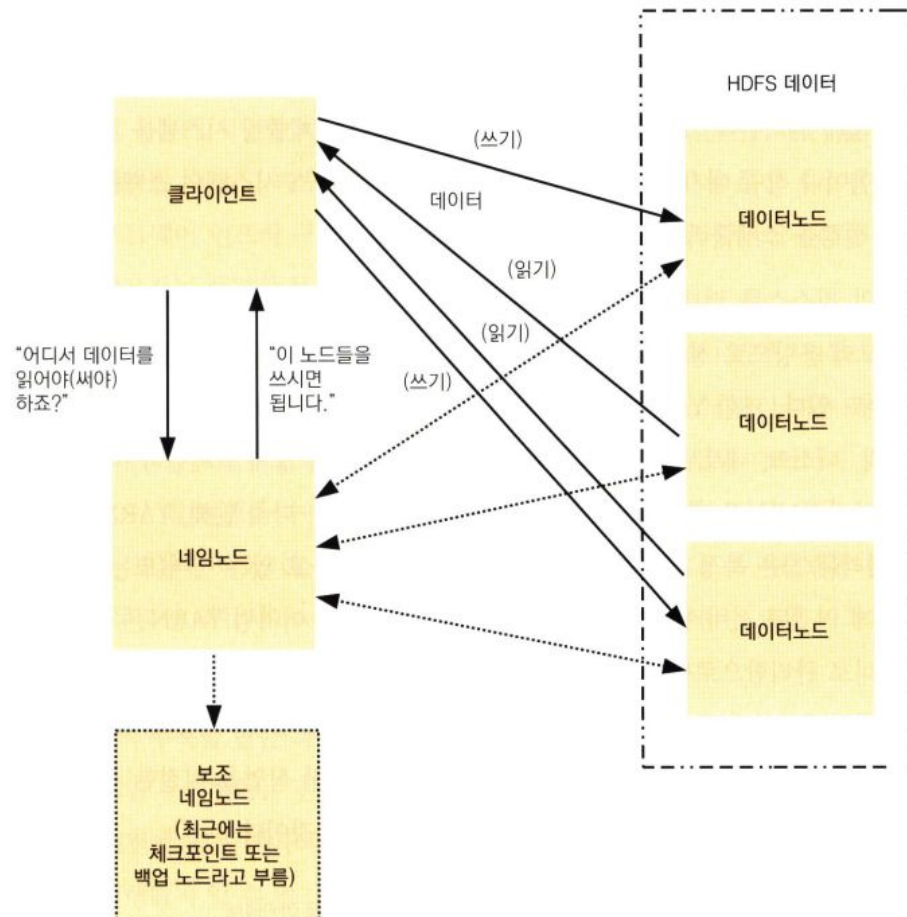
- namenode가 손상될 경우 대비하여 이중화를 구성

- datanode

- 클라이언트나 네임노드의 요청이 있을 때 블록을 저장하고 탐색하며, 저장하고 있는 블록의 목록을 주기적으로 네임로드에 저장

분산 파일 시스템

- 아래 그림은 HDFS의 아키텍처, 하둡 배포판 내 다양한 시스템의 역할을 보여준다.
- 메타데이터와 데이터의 흐름은 실선, 파일 시스템과 노드의 상태 정보 흐름은 점선으로 표시



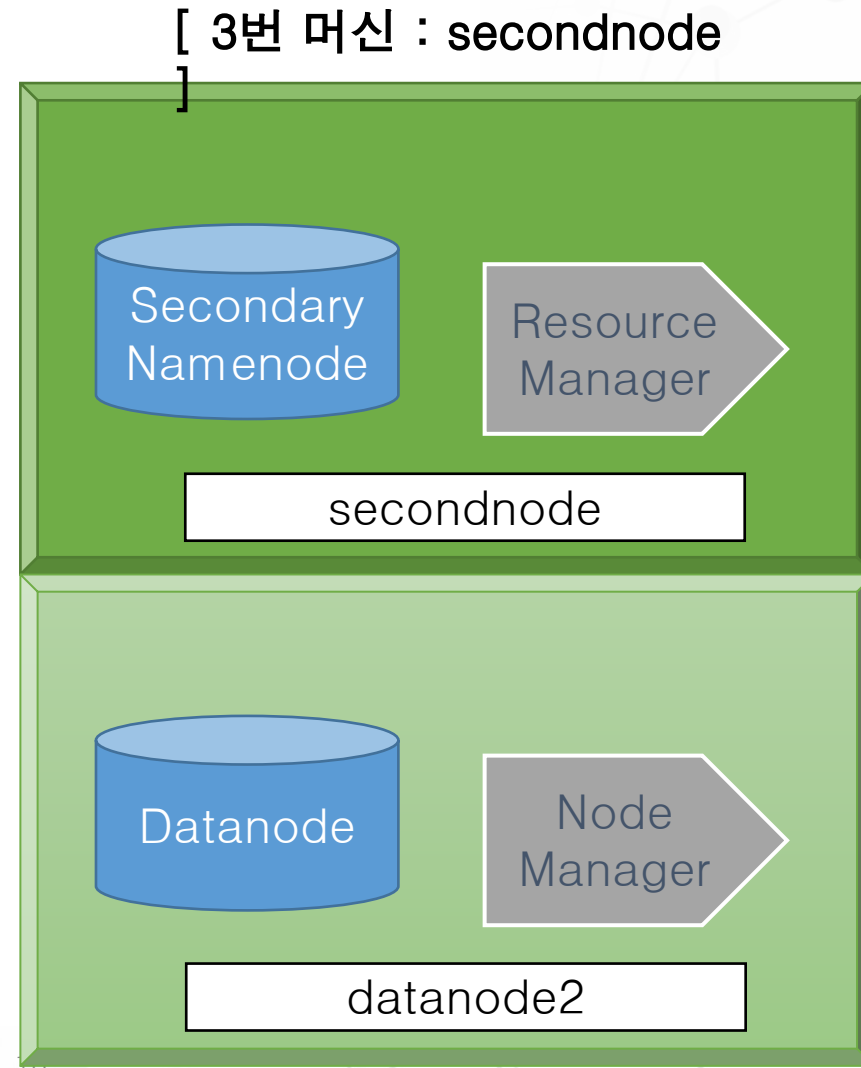
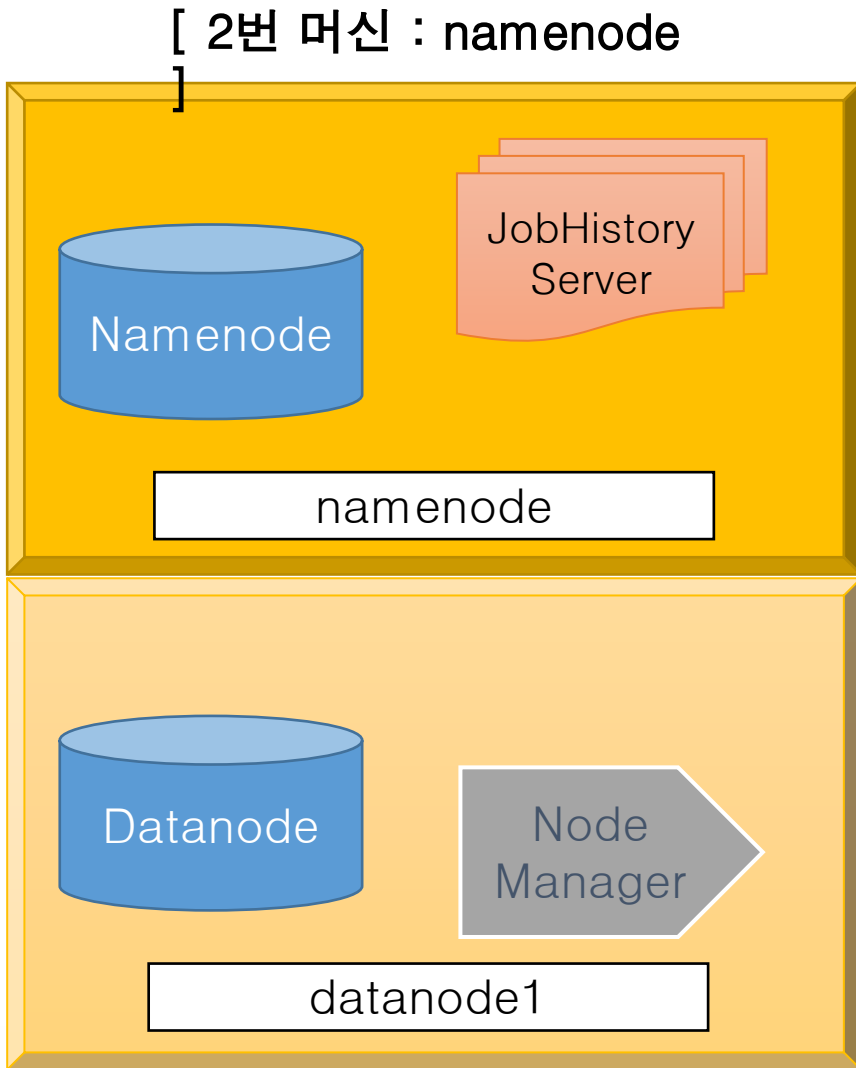
하둡 설정

- 하둡 설정 파일

파일명	설명
hadoop-env.sh	환경 관련 설정은 이곳에 들어간다. 현재 JDK가 시스템 경로 내에 있지 않다면 이 파일을 열어서 JAVA_HOME을 설정해야 한다. 또 이곳에서는 다양한 하둡 컴포넌트용 JVM 옵션을 지정할 수도 있다. 로그 디렉터리와 마스터 및 슬레이브 파일의 위치도 이곳에서 수정할 수 있다.
core-site.xml	시스템레벨의 하둡 설정 항목(예를 들어 HDFS URL, 하둡 임시 디렉터리, 락을 인식하는 하둡 클러스터용 스크립트 위치) 등을 포함한다. 이 파일에 들어 있는 설정은 core-default.xml에 들어 있는 설정보다 우선시된다. <ul style="list-style-type: none">- 하둡 공통 설정<ul style="list-style-type: none">- 네임노드 스킴 설정
hdfs-site.xml	<ul style="list-style-type: none">- 기본 파일 복제 횟수, 블록 크기, 권한 적용 여부 같은 HDFS 설정을 담고 있다.- 이 파일에 있는 설정은 hdfs-default.xml의 설정을 대신한다.- 네임노드, 세컨드노드, 데이터노드의 각종 포트 설정- 파일의 위치 지정
mapred-site.xml	기본 리듀스 작업 개수, 기본 최대/최소 메모리 크기, 투기적 실행같은 DFS 설정은 이곳에서 한다.
masters	하둡 마스터인 호스트 목록을 담고 있다. 이 파일명은 다소 오해의 소지가 있으며 secondary-masters라고 부르는 게 좀 더 적절하다. 하둡을 실행하면 하둡은 start 명령을 보내낸 로컬 호스트에서 네임노드와 잡트래커를 구동한다. 그런 다음 이 파일에 있는 모든 노드에게 ssh를 보내 SecondaryNameNode를 구동한다.
slaves	하둡 슬레이브인 호스트 목록이 들어 있다. 하둡을 실행하면 하둡에서는 이 파일 내 각 호스트로 ssh를 통해 데이터노드와 태스크트래커 데몬을 구동한다. <ul style="list-style-type: none">- 한 줄에 하나씩 호스트명을 지정

Hadoop

- aws 가상 머신



aws에서 하둡 설치하기

- visudo
 - 새롭게 생성된 hadoop 유저에 sudo 권한을 주기 위해서 사용하는 프로그램
 - ec2-user 계정에서 아래 명령어를 실행
 - sudo visudo

```
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
hadoop  ALL=(ALL)        ALL
```

- hadoop 계정을 위와 같이 추가

EC2 하둡 설정하기

- 개인키와 공개키를 생성

- `$ ssh-keygen -t rsa`

- 공개키를 키박스에 추가

- `$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys`

- 접근권한 변경

- `$ chmod 400 ~/.ssh/authorized_keys`

- 읽기전용으로 접근 권한을 변경

- 확인하기

- `$ ls -al .ssh`

- `$ ssh localhost`

- w

- exit

하둡 실행 순서 및 jps 확인

- mr-jobhistory-daemon.sh start historyserver
- mr-jobhistory-daemon.sh stop historyserver
- 서비스 시작 순서 : HDFS -> YARN -> MR-History Server
- 서비스 중단하기 순서 : YARN -> MR-History Server -> HDFS

```
[hadoop@secondnode ~]$ jps
```

```
1700 Jps  
1147 DataNode  
1560 NodeManager  
1235 SecondaryNameNode  
1466 ResourceManager
```

```
[hadoop@namenode ~]$ jps
```

```
2017 NameNode  
2557 Jps  
2427 NodeManager  
2139 DataNode  
3021 JobHistoryServer
```

하둡 설치 후 접속 사이트

- windows hosts 파일 추가
 - C:\Windows\System32\drivers\etc
 - 아래 ip는 ec2의 본인 machin의 public ip를 적어야 함
 - 54.180.112.202 namenode
 - 13.125.244.179 secondnode
- yarn 관리 페이지
 - <http://secondnode:8088>
- HDFS 기본 네임 노트 WebUI
 - <http://namenode:50070/dfshealth.html#tab-overview>
 - <http://namenode:9864/datanode.html>
 - <http://secondnode:9864/datanode.html>
- MR-Jobhistory server
 - <http://namenode:19888/jobhistory>

mapreduce 예제

- wordcount 예제

- client 터미널에서 실행(계정은 hadoop)

- input 폴더를 생성
 - `hdfs dfs -mkdir /input`
 - etc/hadoop 밑의 모든 xml 파일을 hdfs input 폴더에 복사
 - `hdfs dfs -put $HADOOP_HOME/etc/hadoop/*.xml /input`
 - wordcount 실행하여 hdfs output 폴더에 저장
 - `hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar grep /input /output '[a-z.]+'`
 - hdfs output 폴더의 결과 확인
 - `hdfs dfs -cat /output/*`

hdfs 명령어 정리

- `hdfs dfs -cat [경로]`
 - 경로의 파일을 읽어서 보여줌
 - 리눅스 `cat` 명령과 동리함
- `hdfs dfs -count [경로]`
 - 경로상의 폴더, 파일, 파일사이즈를 보여줌
- `hdfs dfs -cp [소스 경로] [복사 경로]`
 - hdfs 상에서 파일 복사
- `hdfs dfs -df /user/hadoop`
 - 디스크 공간 확인
- `hdfs dfs -du /user/hadoop`
 - 파일별 사이즈 확인
- `hdfs dfs -dus /user/hadoop`
 - 폴더의 사이즈 확인
- `hdfs dfs -get [소스 경로] [로컬 경로]`
 - hdfs 의 파일 로컬로 다운로드
- `hdfs dfs -ls [소스 경로]`
 - 파일 목록 확인

hdfs 명령어 정리

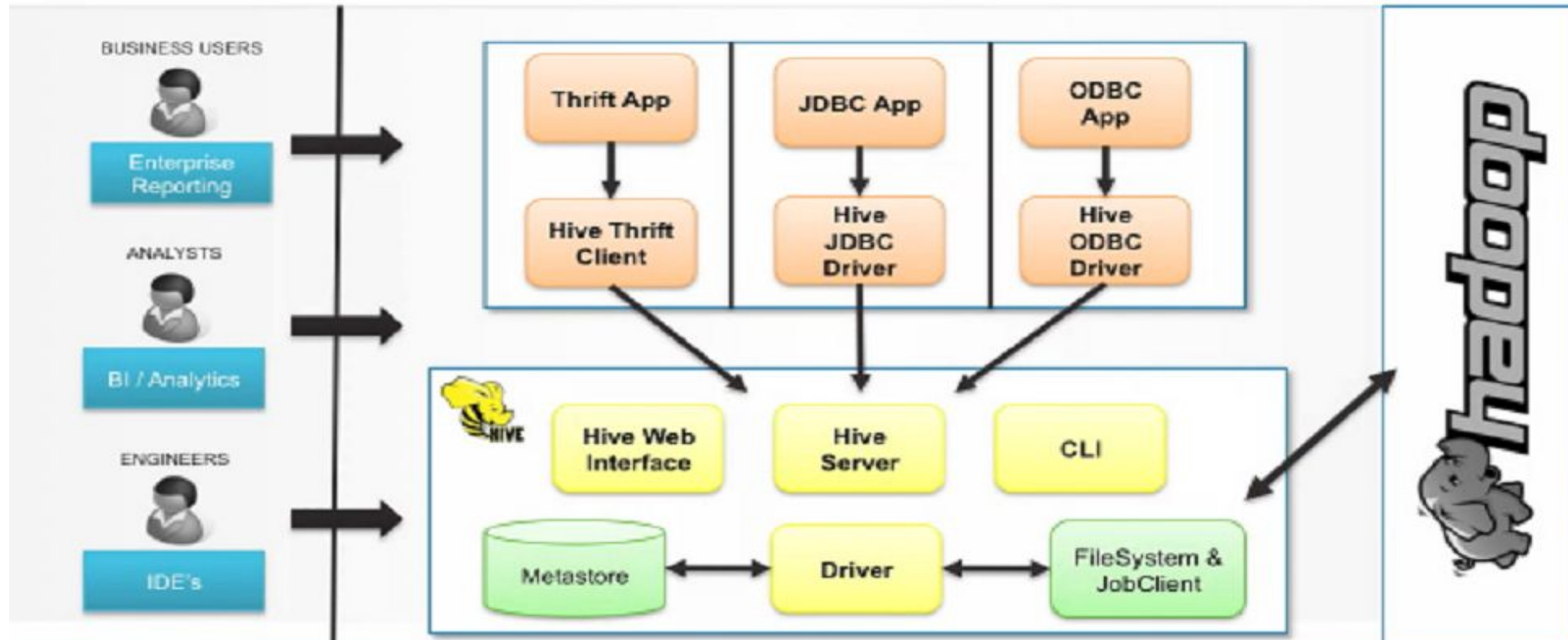
- `hdfs dfs -mkdir [생성 폴더 경로]`
 - 폴더 생성
- `hdfs dfs -mkdir -p [생성 폴더 경로]`
 - 폴더 생성, 부모 경로까지 한번에 생성해 준다.
- `hdfs dfs -put [로컬 경로] [소스 경로]`
 - 로컬의 파일 hdfs 상으로 복사
- `hdfs dfs -rm [소스 경로]`
 - 파일 삭제, 폴더는 삭제 안됨
- `hdfs dfs -rmr [소스 경로]`
 - 폴더 삭제
- `hdfs dfs -setrep [값] [소스 경로]`
 - hdfs 의 replication 값 수정
- `hdfs dfs -text [소스 경로]`
 - 파일의 정보를 확인하여 텍스트로 반환
 - gz, lzo 같은 형식을 확인후 반환해줌
- `hdfs dfs -getmerge hdfs://src local_destination`
 - **hdfs** 경로상의 파일을 하나로 합쳐서 로컬로 가져온다.
 - 리듀스 결과가 여러개일 경우 하나의 파일로 만들기 위해 사용할 수 있다.
 - 주의할 점은 로컬 경로로 가져온다는 것이다. **hdfs** 상에는 생성 불가이다.

Hive 개요

- 개요
 - SQL을 사용, 데이터 요약, 쿼리 및 분석을 수행, 하둡 기반의 데이터웨어하우스 시스템
 - 페이스북 주도로 개발
- 기본적인 작동 원리
 - 사용자가 SQL 쿼리를 작성하면 이것을 자동으로 맵리듀스 작업으로 변경해서 클러스터에서 실행
- 기본 구성과 특징
 - 실행 부분 : 쿼리->맵리듀스 실행
 - 메타데이터 정보 : Mysql과 같은 RDBMS에 저장
- Hive와 RDBMS의 차이점
 - 작은 데이터일 경우 응답 속도가 느림
 - 레코드 단위의 Insert, Update, Delete를 지원하지 않음
 - 트랜잭션을 지원하지 않음
 - 통계정보를 바로 확인할 수 없음
 - 입력값 오류도 바로 확인할 수 없음

Hive 개요

- DW에서의 Hive 역할



ec2에서 새로운 계정으로 로그인하기

- ssh 폴더 안에 id_rsa 파일을 확인

- [hadoop@namenode .ssh]\$ cat ~/.ssh/id_rsa
- -----BEGIN OPENSSH PRIVATE KEY-----
b3BIbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAAdzc2gtcn
.....
YiUIFtp/FXJoftAAAADWWhhZG9vcEBjbGllbnQBAgMEBQ==
-----END OPENSSH PRIVATE KEY-----

- 내용 복사하고 메모장에 붙여넣기

- 파일명을 pem 파일로 저장
 - 형식을 기본의 pem파일과 완전 동일하게 해야함

- puttygen으로 ppk로 변환

- putty에서 hadoop 계정으로 로그인