■ 빅데이터 구현

- 빅데이터 아키텍처는 역할별로 수집 > 적재 > 처리 > 탐색 > 분석 > 응용 이라는 6개의 레이어로 나눌 수 있음

- 각 단계별 주요 기술은 아래 그림에서 확인 가능

- 구축 순서도 보통 수집 -> 적재 -> 처리 및 탐색 -> 분석 및 응용 순으로 진행

- '처리 및 탐색'과 '분석 및 응용' 단계는 필요시 반복 진행하면서 데이터의 품질과 분석 수준을 향상

- 빅데이터 아키텍처 요소 기술들은 적게는 10개에서 많게는 20여 개 정도 되고, 발생하는 데이터의 6V(Volume, Variety, Velocity, Veracity, Visualization, Value) 요건에 따라 최적화된 아키텍처를 구성해야 함

1) 데이터 수집

- 빅데이터의 수집 기술은 조직의 내외부에 있는 다양한 시스템으로부터 로우데이터(raw data)를 효과적으로 수집하는 기술
- 빅데이터 수집에는 기존의 수집 시스템보다 더 크고 다양한 형식의 데이터를 빠르게 처리해야 하는 기능이 필요한데, 그래서 선형확장이 가능하고 분산 처리가 가능한 형태로 구성
- 빅데이터 수집기는 로우(raw) 시스템의 다양한 인터페이스 유형(DB, 파일, API, 메시지 등)과 연결되어 정형, 반정형, 비정형 데이터를 대용량으로 수집
- 특히 외부 데이터(SNS, 블로그, 포털 등)를 수집할 때는 크롤링 등 비정형 처리를 위한 기술이 선택적으로 적용됨
- 수집 처리에는 대용량 파일 수집과 실시간 스트림 수집으로 나뉨
- 실시간 수집의 경우 CEP(Complex Event Processing: 시간에 따라 변화하는 이벤트를 빠르게 분석할 수 있는 기술), ESP(Event Stream Processing: 다양하고 복잡한 이벤트를 분석할 수 있는 기술) 기술이 적용되어 수집 중인 데이터로 부터 이벤트를 감지해 빠른 후속 처리를 수행
- 수집된 데이터는 필요 시 정제, 변환, 필터링 등의 작업을 추가로 진행해 데이터 품질을 향상시킨 후 빅데이터 저장소에 적재

6V	수집기술	중요성
Volume(크기)	대용량 데이터(테라바이트 이상) 수집 대규모 메시지(1000TPS 이상) 수집 *Transaction Per Second(초당 트랜잭션의 수)	상
Variety(다양성)	정형/반정형/비정형 데이터 수집 예) Log, RSS, XML, 파일, DB, HTML, 음성, 사진, 동영상	상
Velocity(속도)	실시간 스트림 데이터 수집	상
Veracity(진실성)	N/A	하
Visualization(시각화)	N/A	하
Value(가치)	N/A	하

- 빅데이터 수집 관련 소프트웨어는 플럼(Flume), 스크라이브(Scribe), 척와(Chukwa), 스쿱(Sqoop)
- 또한 실시간 스트림 데이터 처리를 위해 스톰(Storm)과 에스퍼(Esper)도 사용

2) 데이터 적재

- 빅데이터 적재 기술은 수집한 데이터를 분산 스토리지에 영구 혹은 임시로 적재하는 기술로 빅데이터의 분산 저장소는 총 4가지 유형

① HDFS(Hadoop Distributed File System): 하둡의 코어 기술로, 대용량 파일 전체를 영구적으로 저장할 수있다.

② NoSQL(Hbase, MongoDB, Casandra 등) : SQL을 사용하지 않는 Schema-less 데이터베이스로, 대규모 메시징 데이터 전체를 영구 저장

③ 인메모리 캐시(Redis, Memcached, Infinispan, Spark 등) : 메모리 기반의 데이터 저장소로, 대규모 메시징 데이터의 일부만 임시 저장

④ Message Oriented Middleware(Kafka, RabbitMQ, ActiveMQ 등): 분산 시스템 간 메시지를 주고 받는 기능을 지원하는 소프트웨어나 하드웨어 인프라로, 대규모 메시징 데이터 전체를 버퍼링 처리

- 대용량의 파일의 적재는 HDFS 저장소를 사용하면 됨
- 실시간 대량으로 발생하는 작은 메시지 데이터들을 HDFS에 저장할 경우 파일이 많아져서 저장소 효율이 크게 떨어짐
- 데이터의 성격에 따라 NoSQL, 인메모리 캐시, MoM 등을 선택

6V	수집 기술	중요성
Volume(크기)	대용량 데이터(테라바이트 이상) 적재 대규모 메시지(1000TPS 이상) 적재 *Transaction Per Second(초당 트랜잭션의 수)	상
Variety(다양성)	정형/반정형/비정형 데이터 수집	중
Velocity (속도)	실시간 스트림 데이터 적재	상
Veracity (진실성)	데이터의 품질과 신뢰성을 확보해 적재	상
Visualization (시각화)	N/A	하
Value(가치)	N/A	하

- 빅데이터 적재 기술은 6V 관점에서 데이터의 크기, 속도, 진실성을 효과적으로 처리
- '다양성'의 경우 로우 데이터(raw data)를 다양한 형식으로 변환해 적재할 수 있음
- 하지만 데이터의 일관성과 성능 측면에선 오히려 트레이드 오프(trade-off: 한쪽을 추구하면 다른 쪽을 포기해야 하는 상태) 가 발생할 수 있어서 주의
- 시각화 및 가치는 '탐색/분석 단계'에서 주로 활용

3) 빅데이터의 처리 / 탐색

- 대용량 저장소에 적재된 데이터를 분석에 활용하기 위해 정형화 및 정규화 하는 기술
- 데이터를 통해 가치를 발굴하기 위해서는 데이터를 이해하는 것이 선행돼야 하며, 이 과정에서 적재된 빅데이터를 지속적으로 관찰하고 탐색하는 탐색적 분석을 수행
- 탐색적 분석에서는 SQL on Hadoop(HDFS에 저장된 데이터를 SQL 혹은 SQL과 유사한 형태로 처리를 요청하고 분산 처리하는 시스템- 예: 하이브, Spark)이 주로 사용
- 대화형 애드혹(Ad-Hoc) 쿼리(동적 쿼리, 임시 쿼리)로 데이터를 선택, 변환, 통합, 축소 등의 작업을 수행
- 내외부의 정형/비정형 데이터를 결합해 기존에 기술적으로 한계로 만들지 못했던 새로운 데이터셋을 생성하는 중요한 작업이 진행됨
- 정기적으로 발생하는 처리/탐색의 과정들은 워크플로(workflow)로 프로세스화해서 작동화
- 워크플로 작업이 끝나면 데이터셋들은 DW(Data Warehouse)로 옮김
- DW로 옮겨진 데이터셋은 측정 가능한 구조로 만들어져 있어 빅데이터 분석을 편리하게 함

6V	수집기술	중요성
Volume (크기)	대용량 데이터(테라바이트 이상)에 대한 후처리 및 탐색	상
Variety(다양성)	N/A	하
Velocity(속도)	N/A	하
Veracity(진실성)	데이터의 품질과 신뢰성을 확보하기 위한 후처리 및 탐색	상
Visualization(시각화)	후처리된 데이터셋을 시각화해서 탐색	상
Value(가치)	N/A	중

- 빅데이터 처리 및 탐색 기술은 대규모로 적재된 데이터를 대상
- 크기에 대한 처리 기술이 중요
- 데이터 후처리 작업과 정규화 과정을 통해 데이터의 진실성을 확보하고 후처리된 데이터셋을 시각화툴로 더욱 쉽고 간편하게 탐색
- 처리/탐색 기술: 휴(Hue), 하이브(Hive), 피그(Pig), 스파크(Spark) SQL
- 후처리 작업 자동화하는 워크플로우 기술: 우지 (Oozie)

4) 빅데이터의 분석

- 대규모 데이터로부터 새로운 패턴을 찾고, 그 패턴을 해석해서 통찰력을 확보하기 위한 기술
- 빅데이터 분석은 활용 영역에 따라 통계, 데이터 마이닝, 텍스트 마이닝, 소셜 미디어 분석 등 다양하게 분류
- 빅데이터 분석/응용은 과거의 데이터로 부터 문제의 원인을 찾아 현재를 개선할 뿐 아니라,

인간의 힘으로 찾기 힘들었던 패턴들을 빅데이터 분석 기술로 찾아 알고리즘화 해서 미래를 예측하는 분석 모델을 만드는데 도움을 줌

- 빅데이터라는 용어가 사용되기 전에도 데이터 분석 기술과 도구가 많이 존재했지만, 모바일과 SNS의 발달 그리고 4차 산업혁명 시기에 접어들면서 데이터 양이 방대해지면서 기존의 기술로 이 데이터들을 처리하는데 무리가 있었음

- 빅데이터 분석은 선형적으로 확장이 가능하고 대규모 분산환경을 낮은 비용으로 구축할 수 있어서 기존의 분석기술의 한계점을 극복하였음

- 또한 머신러닝 기술을 활용해 군집(clustering), 분류(classification), 회귀(regression), 추천(recommendation) 등의 고급 분석 영역까지 확장할 수 있어 좀 더 똑똑한 소프트웨어를 만들 수 있음

- '대규모 배치 분석'이 인메모리 기반의 준실시간 분석으로도 가능해져 파일 기반의 배치 분석보다 수십 배 빠른 분석이 가능해짐으로써 활용범위가 더욱 커지고 있음

6V	수집기술	중요성
Volume (크기)	대용량 데이터(테라바이트 이상) 분석	상
Variety(다양성)	정형/반정형/비정형 등의 다양한 데이터 분석	상
Velocity(속도)	인메모리 기반으로 실시간 데이터 분석	상
Veracity(진실성)	신뢰도 높은 분석 결과를 비즈니스에 적용	상
Visualization(시각화)	분석 결과 및 창출된 가치를 시각화	상
Value(가치)	분석된 결과를 비즈니스에 적용해 가치 창출	상

- 빅데이터 분석/응용은 6V의 모든 항목이 적용

- 특히 '가치(Value)'는 빅데이터의 구축 사이클(수집, 적재, 처리/탐색, 분석/응용)에서 빅데이터의 최종 목표가 됨

- 빅데이터 기술은 5V(Volume, Variety, Velocity, Veracity, Visualization)로 비즈니스에 대한 통찰력을 갖게 되고, 이를 기반으로 조직의 혁신적인 1V(Value)를 창출하는 도구

- 분석/응용 기술: 임팔라(Impala), 제플린(Zeppelin), 머하웃(Mahout), 스파크ML(Spark ML)

■ 빅데이터와 보안

- 빅데이터 시스템도 일반적인 시스템에 적용하는 모든 보안 요소(시스템 보안, 네트워크 보안, 코드 보안, 데이터 보안 등)를 고려
- 보안은 범위와 기술이 워낙 다양하며 시스템의 주요 기능과 상충되는 경우도 많고 산업 분야와 업무 영역 또는 조직 문화에 따라 보안에 대한 경중이 다름
- 보안 담당자를 지정해 프로젝트 시작부터 끝까지 꼼꼼하게 챙기지 않으면 향후 심각한 컴플라이언스 이슈로 이어질 수 있음
- 여러 보안 요소들 중 데이터 보안, 접근 제어 보안은 빅데이터 시스템을 구축할 때 나타나는 주요 보안 요소

1) 데이터 보안

- 데이터 보안은 개인과 기업의 정보 보호를 위한 정책과 기술
- 정부의 개인정보 보호법은 빅데이터의 발전과 트레이드오프를 이룸
- 빅데이터 보안의 원칙은 우선 “개인 식별이 가능한 어떠한 정보도 수집하지 않는다”
- 개인을 식별 정보란 개인을 식별 할 수 있는 고유한 정보로서 이름, 직업, 성별, 주민등록번호, 전화번호, 주소, 여권번호, 위치 정보등 아주 다양
- 개인 식별 정보를 아예 수집하지 못할 경우 빅데이터 분석 자체가 의미가 없어지므로 이러한 개인 식별 정보는 다음 그림과 같이 수집과 동시에 비식별화 처리

이름	연령대	성별	거주지	직업	전화번호
홍**	20대	남	서울 강남구 신사동 OO	대학생	010-XXXX-1234
김**	30대	여	경기 과천시 중앙동 OO	공무원	010-XXXX-9876
이**	40대	남	부산 중구 광복동 OO	자영업	010-XXXX-6678
…	…	…	…	…	…

- 개인정보 비식별화 기술은 가명처리, 총계처리(개인 정보에 대하여 통계값(전체 혹은 부분)을 적용하여 특정 개인을 판단할 수 없도록 함), 데이터 값 삭제, 범주화, 마스킹 등
- 비식별화 뒤에 생기는 이슈 중 첫번째는 재식별화
- 비식별화 후 데이터를 사용하기 위해서는 재식별화를 해야하는데 재식별화는 주변의 다른 데이터와 결합했을 때 비식별화 한 데이터의 식별력이 높아지는 것을 말함
- 두번째는 빅데이터를 활용해 특정 고객을 대상으로 차별화된 전략 수립 및 적용 여부
- 기업은 빅데이터로 수집된 데이터를 정보화해서 분석한 뒤 직간접적인 수익을 창출
- 행동이력에 따른 실시간 상품 추천, 개인 정보 활용에 동의한 고객을 대상으로 한 1:1

마케팅에서 빅데이터 분석 결과를 토대로 개인을 고유하게 식별하는 키가 필요한데 이러한 고유키(핸드폰 번호, 주민번호 등)가 비식별화 되어 활용이 불가능함

- 이런 이슈를 해결하기 위해 대체키를 사용

- 일상에서 특정 인터넷 서비스를 등록하거나 상품을 신청할 때 회원가입과 동시에 내부적으로 고유한 키가 생성되며, 이 키가 개인의 식별키(이메일, 계좌번호, 전화번호, 주민등록번호, 여권번호, 운전면허번호 등)를 대체하는 대체키가 된다.

2) 접근제어 보안

- 현재 빅데이터 시스템의 접근제어(인증, 권한)를 완벽하게 처리하는 데는 많은 어려움

- 복잡하고 다양한 빅데이터 오픈소스 소프트웨어를 대상으로 일관된 접근 보안 정책을 적용하는 것은 배보다 배꼽이 더 큰 문제일 수 있고, 빠르게 발전하는 오픈소스 소프트웨어의 소스를 직접 수정하는 것도 매우 부담

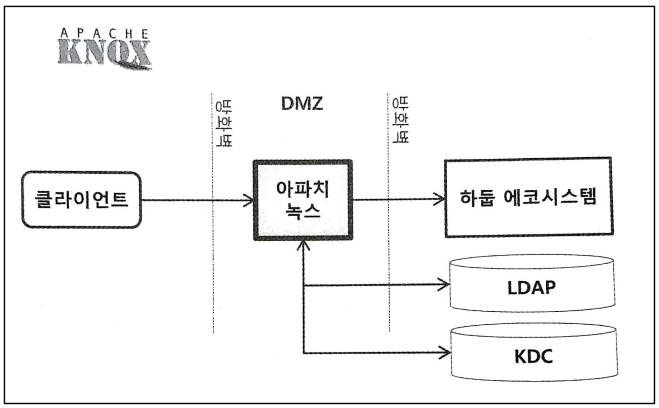
- 빅데이터 시스템의 물리적(네트워크) 위치는 대부분 기업 또는 조직의 매우 깊은 곳(네트워크 방화벽 안쪽) 이어서 외부 공격이나 불특정 다수가 접근하는 시스템이 아니며, 앞서 설명한 비식별화된 데이터로 저장되어 있어서 보통느 저수준의 접근제어(저수준 시스템 호출로 입/출력을 수행하는 것은 비효율 적으로 사용을 자제) 정책을 적용

- 일부 금융권 및 민감한 개인 정보를 다루는 빅데이터 시스템의 경우 고수준의 접근제어 정책과 보안 준수를 요구하고 있고, 이를 해결하기 위해 아파치 녹스(Apache Knox), 아파치 센트리(Apache Sentry), 아파치 레인저(Apache Ranger), 커베로스(Kerberos) 등을 활용

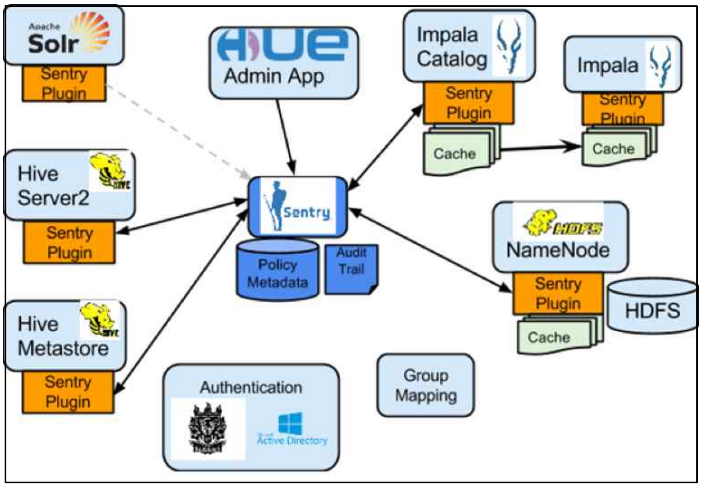
- '아파치 녹스'의 경우 네트워크 상의 DMZ(비무장지대:DMZ에는 외부 공격자의 침입으로부터 안전하게 보호되어야 하는 DB서버, 인증서 서버, 로그인 서버 등이 위치)에 위치

- 외부 클라이언트가 하둡 에코시스템에 직접 접근하는 것을 막고 항상 '아파치 녹스'를 거쳐 통신하게 하는 중간 게이트 역할로 사용

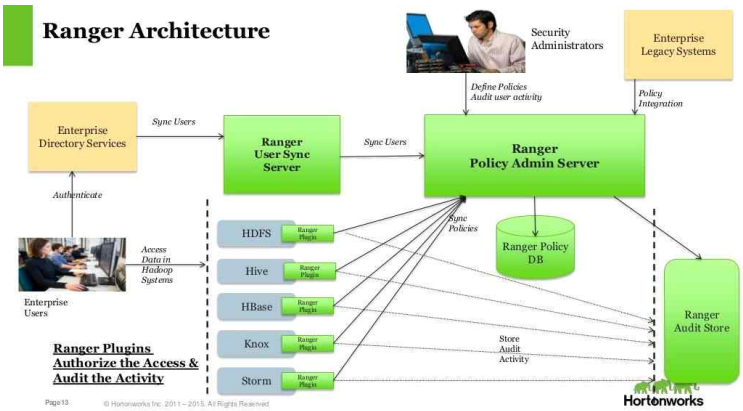
- 이때 들어온 요청에 대한 접근 인증을 LDAP(LightWeight Directory Access Protocol)과 KDC(Key Distribution Center)로 제공받음

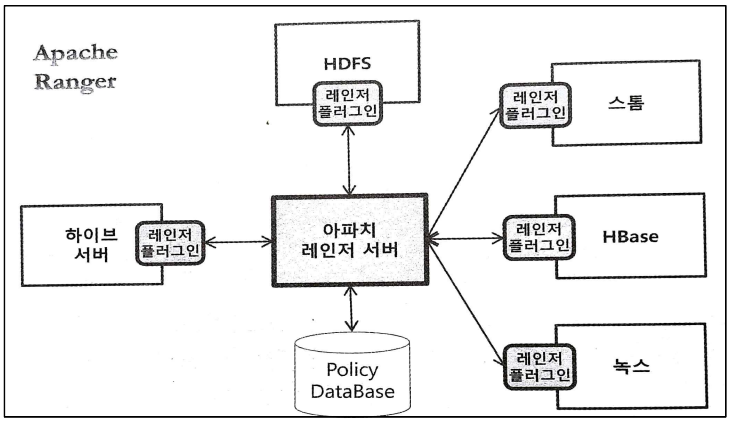


- ‘아파치 센트리’는 하둡 파일 시스템에 상세한 접근 제어(하둡 파일/디렉터리, 하이브 테이블 등)가 필요할 때 사용
- 하둡 데이터에 접근하려는 클라이언트는 센트리 에이전트를 반드시 설치해야 하고, 센트리 에이전트가 중앙에 있는 센트리 서버와 통신하면서 접근 권한을 획득
- 접근 이력을 관리하는 기능이 있어 향후 감사로그를 편리하게 조회

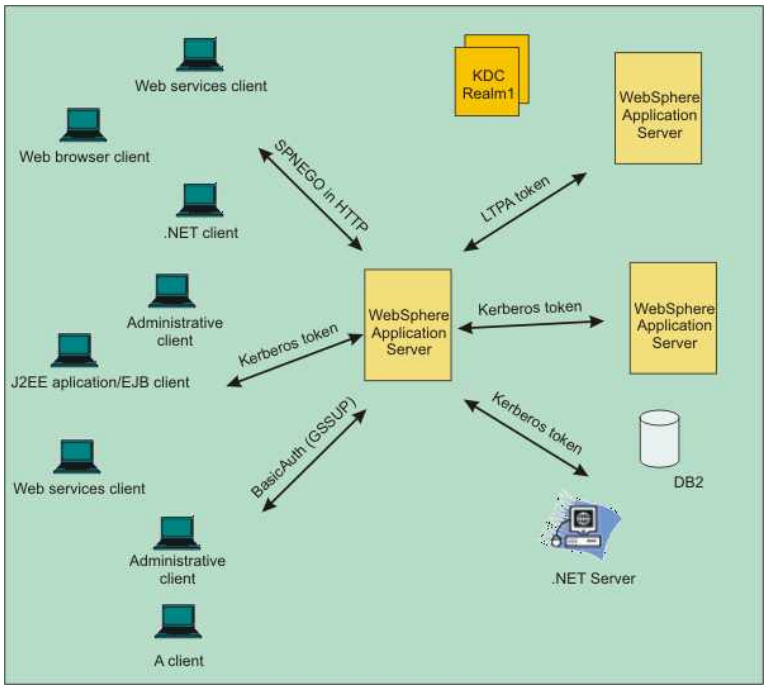


- ‘아파치 레인저’는 앞서 설명한 센트리와 유사한 역할을 하며 아키텍처에도 큰 차이가 없음
- ‘아파치 레인저’는 호트웍스에서, 센트리는 클라우데라에서 지원
- ‘아파치 레인저’가 지원하는 에코시스템이 많아 범용성이 좀 더 높음
- ‘아파치 레인저’를 사용할 겨우 프러그인을 통해 레인저 서버와 통신하게 되고 센트리와 마찬가지로 접근 이력을 관리하는 감사 로그 기능을 제공

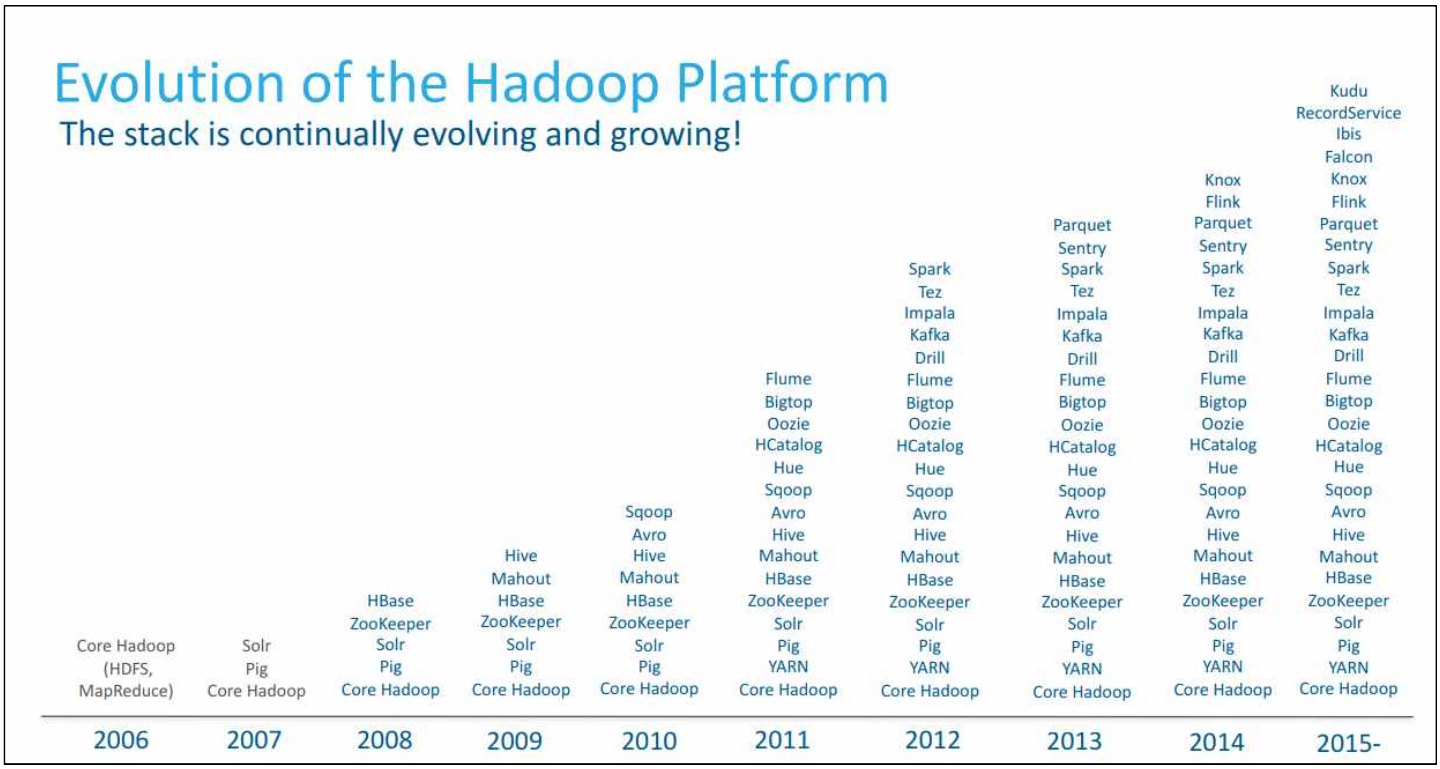




- ‘커베로스’는 KDC(Key Distribution Center) 시스템으로 불리며, 빅데이터 외에도 이미 다양한 곳에서 활용되고 있는 범용화된 인증 시스템
- ‘커베로스’는 크게 AS(Authentication Service)라는 인증 서버와 TGS(Ticket Granting Service)라는 티켓 발행 서버로 구성
- 하둡 파일 시스템에 접근하려는 클라이언트 에코시스템은 AS 인증 서버를 통해 최초 인증을 수행하고 TGS 티켓 발행 서버로부터 하둡 파일 시스템에 접근을 허용하는 티켓을받음
- 발행이후부터는 유효한 티켓만 있으면 하둡 파일 시스템에 인증없이 접근 가능



■ 하둡 에코시스템

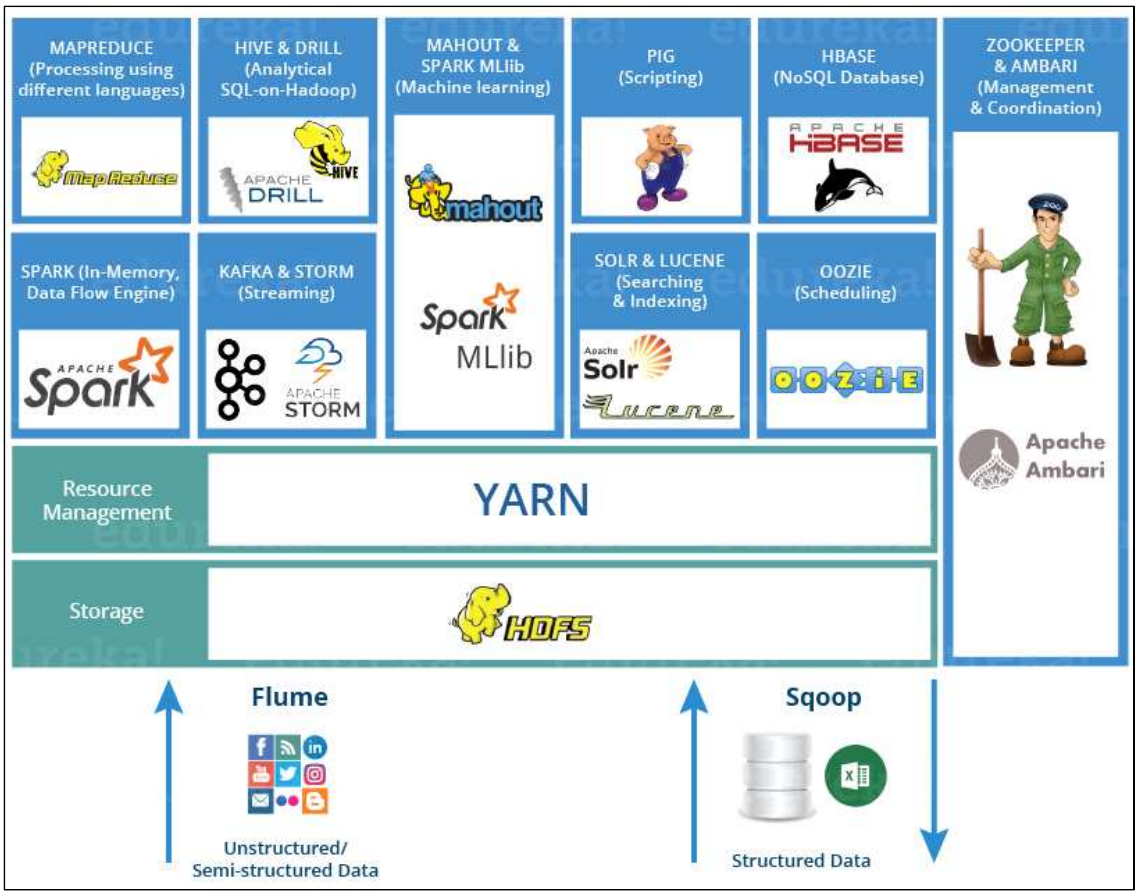


하둡의 기능을 보완하는 서브 오픈소스 소프트웨어들

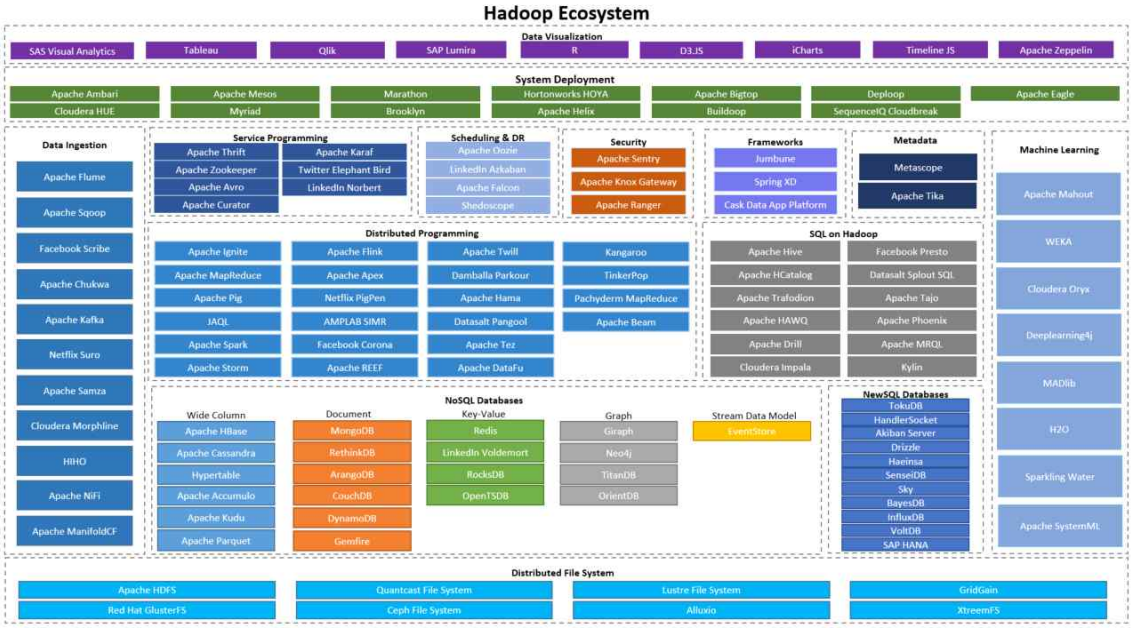
구분	주요 기술	기술 별 주요 기능
빅데이터 수집	▪ 플럼(Flume) ▪ 스쿱(Sqoop)	▪ 비정형 데이터 수집 ▪ 관계형 DB로부터 데이터 가져오기
빅데이터 저장, 활용	▪ Hbase	▪ 컬럼 기반 NoSQL 데이터베이스
빅데이터 처리	▪ 하이브(Hive) ▪ 피그(Pig) ▪ 마후트(Mahout)	▪ 유사 SQL 기반 빅데이터 처리 ▪ 스크리브 언어 기반 빅데이터 처리 ▪ 기계학습 알고리즘 기반 빅데이터 처리
빅데이터 관리	▪ 우지(Oozie) ▪ H카탈로그(HCatalog) ▪ 주키퍼(Zookeeper)	▪ 빅데이터 처리 과정(Process) 관리 ▪ 빅데이터 메타 정보 관리 ▪ 빅데이터 서버 시스템 관리

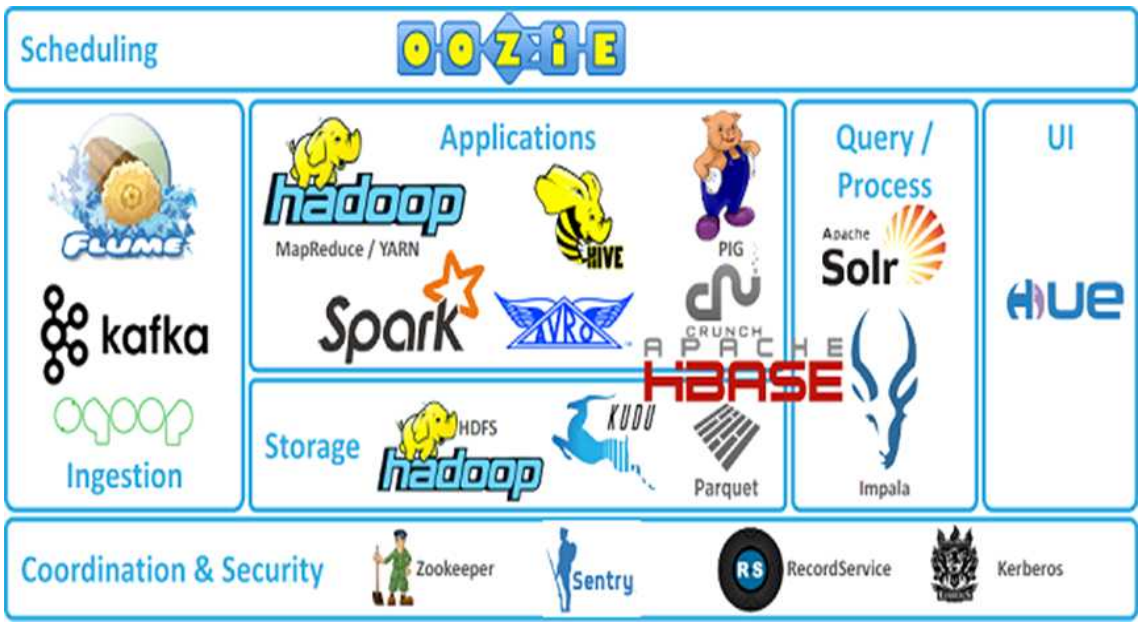
하둡 에코시스템

〈출처 : '빅데이터' 플랫폼의 미래', Technology Inside LG CNS R&D Journal, 이주열, 2013〉


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


In-Memory, Data Flow Engine
Analytical SQL-on-Hadoop
NoSQL Database
Machine Learning
Search
Scripting
Integration & Streaming
Management & Coordination
Pig
Ambari
Zookeeper
Resource Management
yarn
MESOS
Storage
hadoop HDFS
cloudera Manager





■ 하둡의 기본생태계(eco system)

구분	주요 기술	기술 별 주요 기능
빅데이터 수집	· Flume · Sqoop	· 비정형 데이터 수집 · 관계형 DB 로부터 데이터 가져오기
빅데이터 저장, 활용	· HBase HDFS	· 컬럼 기반 NoSQL 데이터베이스
빅데이터 처리	· Hive · Pig · Mahout	· 유사 SQL 기반 빅데이터 처리 · 스크립트 언어 기반 빅데이터 처리 · 기계학습 알고리즘 기반 빅데이터 처리
빅데이터 관리	· Oozie · HCatalog · Zookeeper	· 빅데이터 처리 과정(Process) 관리 · 빅데이터의 메타 정보 관리 · 빅데이터 서버 시스템 관리

1) 관리 : Zookeeper

2) 스케쥴링 : Oozie

3) 수집 : Flume(비정형), Kafka(실시간 데이터), Sqoop(RDBMS에서 정형데이터 수집)

4) 저장 : HDFS(하둡 기본 적재), HBase(NoSQL 데이터베이스), Redis

5) 처리 : Hive, Spark, Mahout

6) 가장 기본이 되는 생태계(현재는 많이 변하고 다양해졌음)

- 하둡이 비즈니스에 효율적으로 적용할 수 있게 제공한 다양한 서브 프로젝트가 상용화된 것이 하둡 에코시스템
- 하둡은 저장 기능인, HDFS와 처리 기능인 MapReduce / YARN 제공
- 하둡의 기능을 보완 및 효율적으로 적용하기 위한 다양한 서브 프로젝트

1) Zookeeper : 관리

- 분산 환경에서 서버 간의 상호 조정이 필요한 다양한 서비스를 제공
- 하나의 서버에만 서비스가 집중되지 않게 서비스를 알맞게 분산해 동시에 처리
- 하나의 서버에서 처리한 결과를 다른 서버와도 동기화해서 데이터 안정성 보장
- 운영(active)서버에 문제가 발생해서 서비스를 제공할 수 없을 경우 다른 대기 중인 서버를 운영 서버로 바꿔서 서비스 중지를 막음
- 분산 환경을 구성하는 서버의 환경설정을 통합적으로 관리
- 주키퍼 앙상블 (주키퍼를 여러 서버에 설치하는데 홀수로 구성)

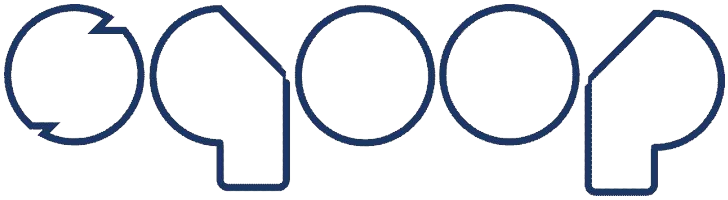
• 여러대의 Zookeeper 서버를 클러스터링(clustering:군집화) 한 것
• 고가용성을 지원
• 한대의 Leader와 여러대의 Follower로 구성
• Leader는 쓰기역할 수행

- 하둡2에서 역할

• 네임노드 HA 상태 정보를 저장하는 장소

- 공식 사이트: zookeeper.apache.org

2) Apache Sqoop : 수집



- RDBMS로부터 수집
- HDFS, RDBMS, DW, NoSQL 등 다양한 저장소에 대용량 데이터를 신속하게 전송
- Hive, Pig, Hbase 등로 바로 옮겨 확인 가능
- HDFS에 있는 데이터도 외부 RDBMS로 옮길 수 있음
- 오라클, MS-SQL, DB2등과 같은 상용 RDBMS와 MySQL, PostgreSQL과 같은 오픈소스 RDBMS 지원
- 공식 사이트: http://sqoop.apache.org/

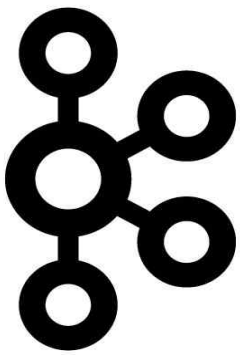
3) Apache Flume : 수집

- 대용량의 로그 데이터를 분산,안정성,가용성을 바탕으로 효율적으로 수집,집계,이동이 가능한 로그수집 솔루션
- 다양한 소스로 부터 데이터를 수집하여 다양한 방식으로 데이터를 전송이 가능하지만, 아키텍처가 단순하고 유연하며 확장 가능한 데이터 모델을 제공하여, 실시간 분석 애플리케이션을 쉽게 개발
- 언어는 Java
- 클러스터에 있는 모든 장치로부터 로그 파일들을 수집 후, HDFS와 같은 중앙 장소에 저장하는 로깅 시스템 구축에 제격
- 확장성, 안정성, 유연성, 실시간성 만족
- 공식 사이트: flume.apache.org

4) Facebook Scribe : 수집

- Facebook이 개발하여 오픈소스화한 로그수집서버. 대량의 서버로부터
- 실시간으로 스트리밍 로그 수집을 위한 솔루션
- 최종 데이터는 HDFS외에 다양한 저장소를 활용할 수 있음
- Facebook의 자체 Scaling 작업을 위해 설계되어 현재 매일 수백 억건의 메시지를 처리하고 있다. 클라이언트 서버의 타입에 상관없이 다양한 방식으로 로그를 읽어 들일수 있다.
- 언어: C++ 기반
- Apache Thrift는 필수
- Thrift 기반 Scribe API를 활용하여 확장 가능
- HDFS에 저장하려면 JNI(Java Native Interface) 사용
- 공식 사이트: github.com/facebookarchive/scribe

5) Apache Kafka : 수집

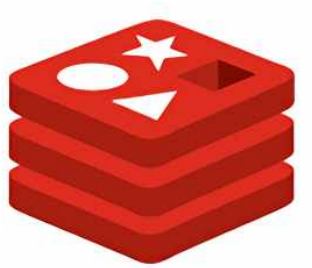


- MOM(Message Oriented Middleware) 소프트웨어
- 대규모로 발생하는 메세지성 데이터를 비동기 방식으로 중재
- 디스크기반이라 File System에 저장하는 방식으로 안전하고 언제든지 복구 가능(큐 방식)
- 단일 카프카가 1초당 몇 천개의 클라이언트로부터 유입되는 수백 메가바이트를 처리할 수 있어서 빠름
- Flume+Kafka로 연결하면 대용량의 실시간 데이터를 안전하게 수집 및 이동할 수 있음
- 공식 사이트: kafka.apache.org

6) HBase : 저장

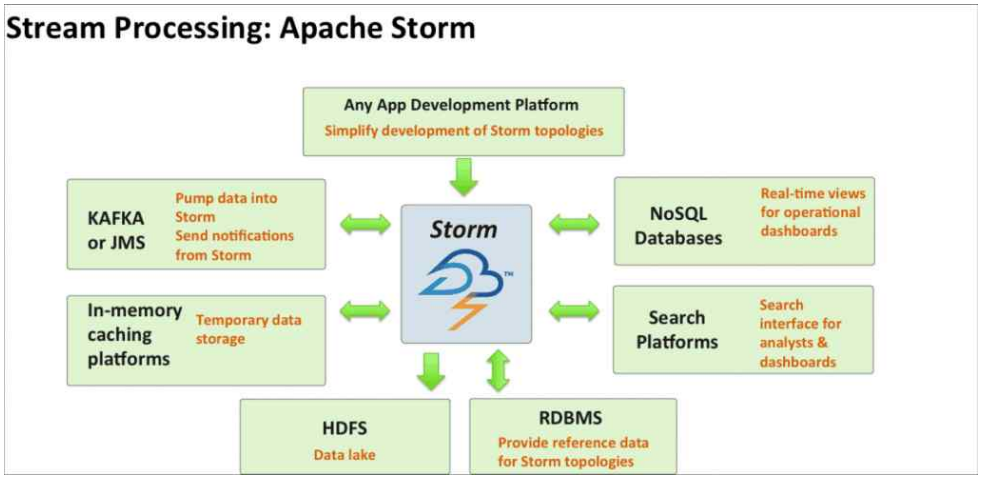
- 하둡기반의 칼럼 지향 NoSQL 데이터베이스 (데이터의 저장을 칼럼단위로 처리하는 데이터베이스 : 즉 테이블)
- 데이터를 키/값 형식으로 단순하게 구조화
- 쓰기 성능에 최적화
- 대용량 처리가 필요한 대규모 NoSQL 아키텍쳐 구성이 필요할 때 자주 사용
- HDFS 위에서 작동을 한다
- 공식 사이트: hbase.apache.org

7) Redis : 저장



- 메모리 기반의 키/값 Store(IMDG(InMemory Data Grid): 여러 시스템의 메모리에 데이터를 분산시켜 저장)
- 메모리 가반이라 빠름
- 인메모리 데이터를 영구적으로 저장하는 스냅샷 기능 제공(정합성 보장)
- 작지만 읽기가 쓰기보다 많은 공통 정보를 담을 NoSQL로 적합
- 데이터의 샤딩(Sharding)과 복제(Replication) 지원, HA 기능 지원
- 공식 사이트: redis.io

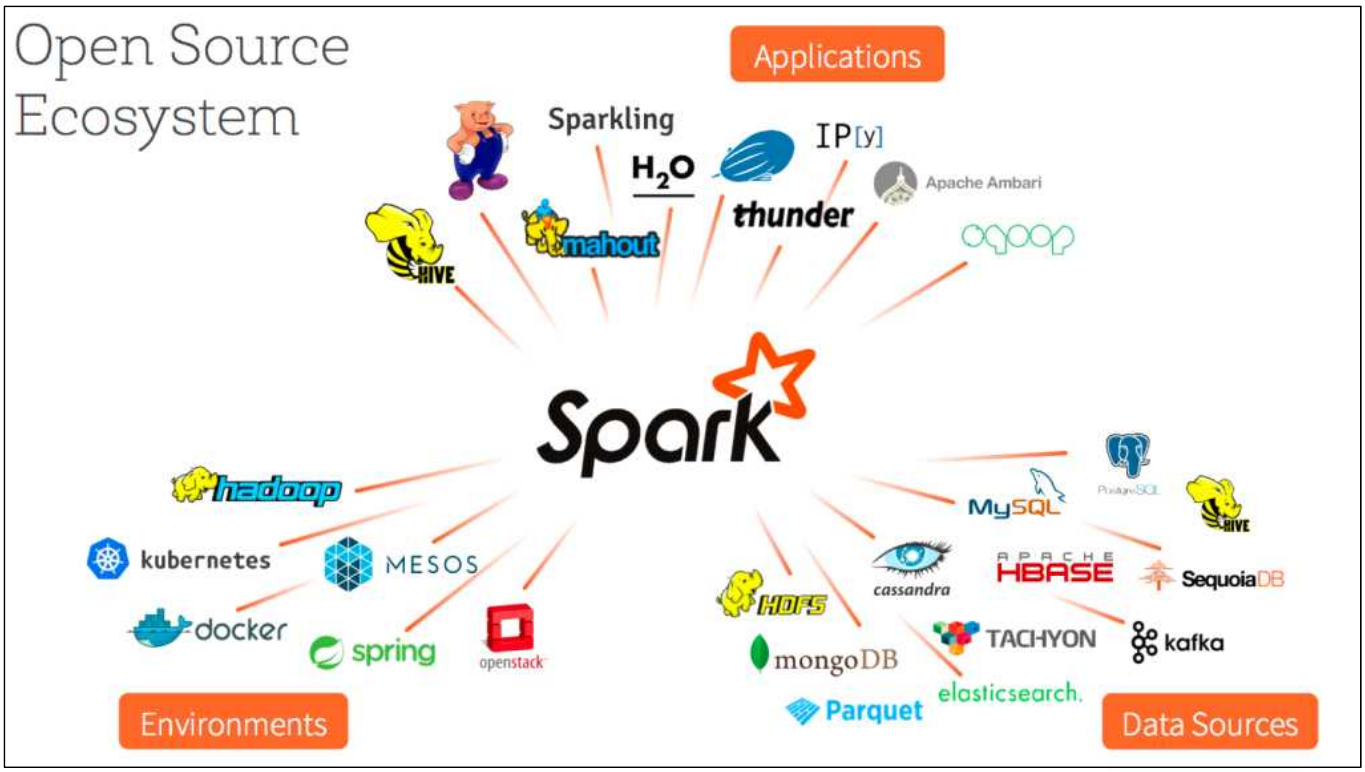
8) Apache Storm : 처리



- 실시간 데이터를 병렬 프로세스로 처리
- 실시간으로 데이터를 프로세싱(분리, 정제, 조합, 카운팅 등)
- 데이터를 적재하기도 전에 발행과 동시에 이벤트를 감지해서 처리
- 공식 사이트: storm.apache.org

9) Spark : 처리(새로운 에코시스템이 있음)

- 인메모리 기반의 범용 데이터 처리 플랫폼
- 배치 처리, 머신러닝, SQL 질의 처리, 스트리밍 데이터 처리, 그래프 라이브러리 처리 등
- 공식 사이트 : spark.apache.org



10) Hive : 처리

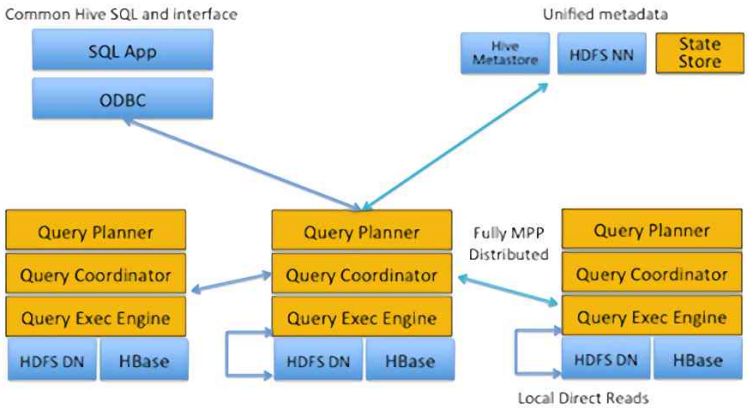
- 하둡 기반의 데이터웨어하우징용 솔루션
- SQL과 매우 유사한 HiveQL이라는 쿼리를 제공
- HiveQL은 내부적으로 MapReduce 잡으로 변환되어 실행
- 공식 사이트: hive.apache.org

11) Apache Chukwa : 분석

- 분산되어있는 노드들의 로그 데이터 수집하고 수집된 데이터를 저장하며 분석
- 수집하는 로그: 모니터링 로그, 응용프로그램 로그, 하둡 로그
- 테라바이트 단위 이상의 로그 데이터를 모니터링
- 실시간으로 로그를 관리/분석

12) Impala : 분석

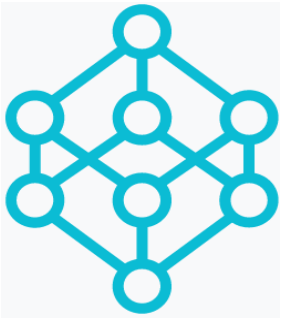
- HDFS에 저장돼 있는 데이터를 SQL을 이용해 실시간으로 분석할 수 있는 시스템
- 맵리듀스의 단점은 사용의 불편함 / 처리가 느림
- 맵리듀스의 보완이 'Hive'였지만 속도는 느림
- 임팔라는 속도가 매우 빠름



13) Pig : 분석

- 대용량 데이터 집합을 분석하기 위한 플랫폼
- 맵리듀스 프로그래밍
- 피그 라틴(Pig Latin)이라는 자체 언어를 제공

14) Apache Mahout : 머신러닝(분석)



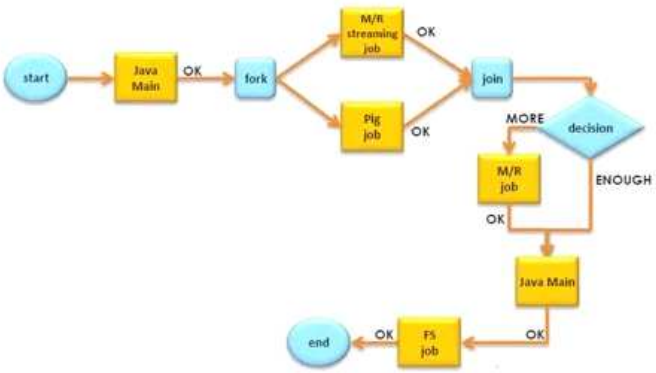
- 하둡 기반으로 데이터 마이닝 알고리즘을 구현한 오픈 소스
- 현재 분류 (classification), 클러스터링 (clustering), 추천 및 협업 필터링 (Recommenders/collaborative filtering), 패턴 마이닝 (Pattern Mining), 회귀 분석 (Regression), 차원 리덕션 (Dimension reduction), 진화 알고리즘 (Evolutionary Algorithms) 등 중요한 알고리즘을 지원
- 맵리듀스를 주로 이용
- 공식 사이트: mahout.apache.org

15) Zeppelin : 시각화 분석

- 빅데이터의 분석도구
- 다양한 언어로 분석코드를 짜고, 바로 Graph로 시각화하여 볼 수 있음
- 협업하여 사용 가능
- 공식 사이트: zeppelin.apache.org

16) Oozie : 스케쥴링

- 하나의 작업을 여러 단계로 나누면 일이 간단해짐
- 자동화하거나 스케쥴링을 도와주는 솔루션
- 공식 사이트: oozie.apache.org



17) 그 외에도 타조, 하마, 카산드라 등등 다양한 플랫폼이 존재했고, 나타나는 중