



20220502

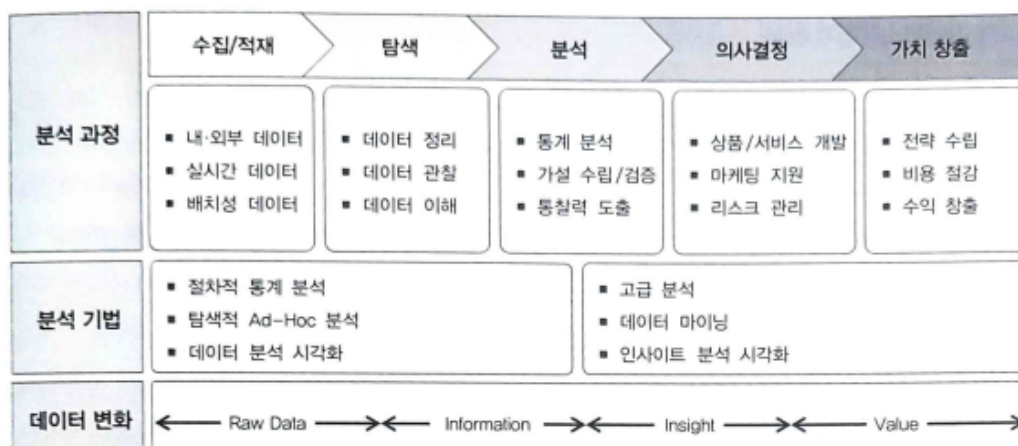
빅데이터 분석

빅데이터 탐색 단계가 데이터를 관찰하고 이해하며 전처리하는 과정이라면 빅데이터 분석은 탐색과 분석을 반복하며 의미 있는 데이터를 추출해 문제를 검증하고 주요 의사결정을 내리는 과정이다.

- 기술 분석: 분석 초기 데이터의 특징을 파악하기 위해 선택, 집계, 요약 등 양적 기술 분석을 수행
- 탐색 분석: 업무 도메인 지식을 기반으로 대규모 데이터셋의 상관관계나 연관성을 파악
- 추론 분석: 전통적인 통계분석 기법으로 문제에 대한 가설을 세우고 샘플링을 통해 가설을 검증
- 인과 분석: 문제 해결을 위한 원인과 결과 변수를 도출하고 변수의 영향도를 분석
- 예측 분석: 대규모 과거 데이터를 학습해 예측 모델을 만들고, 최근의 데이터로 미래를 예측



- 다양한 분석 기술을 통해 빅데이터의 가치는 “Raw 데이터 → 정보 → 통찰력 → 가치” 순으로 변하게 된다.




빅데이터 장점

- 저비용 고효율
 - 오픈소스의 강력한 분산 기술과 낮은 하드웨어/소프트웨어 비용으로 수평적 선형 확장이 가능하다.
- 내부(정형)데이터와 외부(비정형)데이터를 결합해 분석 시 다양한 관점을 제공할 수 있다.

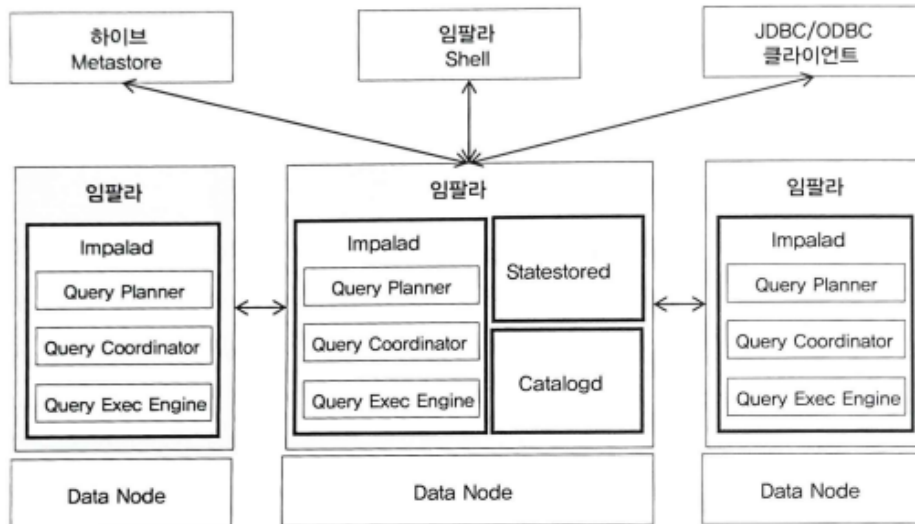
빅데이터 분석에 활용 기술

임팔라

- 실시간으로 분석하기 위한 오픈소스

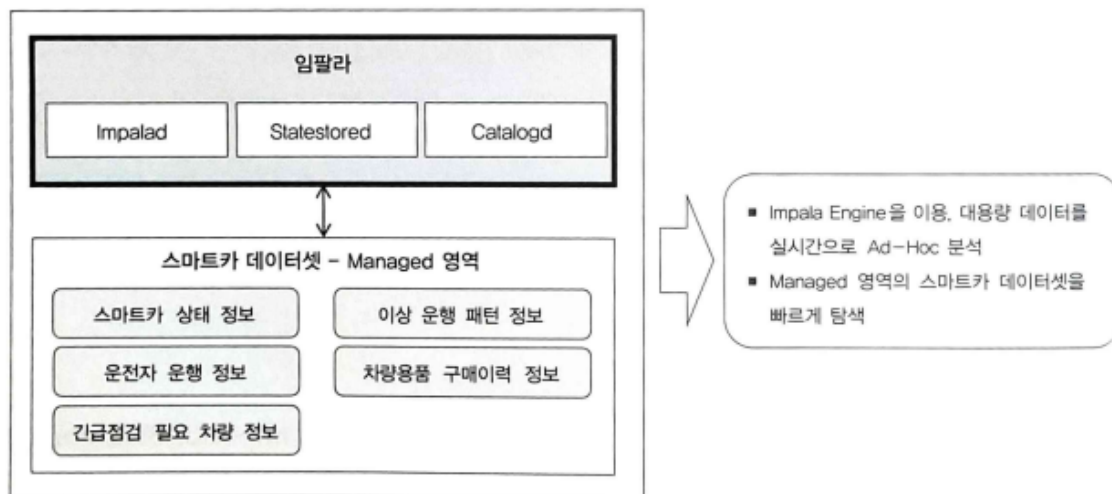
공식 홈페이지		http://impala.apache.org/
주요 구성 요소	Impalad	하둡의 데이터노드에 설치되어 임팔라의 실행 쿼리에 대한 계획, 스케줄링, 엔진을 관리하는 코어 영역
	Query Planner	임팔라 쿼리에 대한 실행 계획을 수립
	Query Coordinator	임팔라 잡리스트 및 스케줄링을 관리
	Query Exec Engine	임팔라 쿼리를 최적화해서 실행하고, 쿼리 결과를 제공
	Statestored	분산 환경에 설치돼 있는 Impalad의 설정 정보 및 서비스를 관리
	Catalogd	임팔라에서 실행된 작업 이력들을 관리하며, 필요 시 작업 이력을 제공
라이선스	Apache	
유사 프로젝트	Tez, Spark SQL, Drill, Tajo	

- 임팔라 아키텍처
 - 하둡의 분산 노드에서 대규모 실시간 분석을 위해 Impalad, Statesored, Catalogd라는 컴포넌트가 설치된다.
 - Impalad는 HDFS의 분산 노드 상에서 실행 계획과 질의 작업을 수행하는 데몬이고,
 - Statestored는 Impalad의 기본 메타 정보로부터 각 분산 노드에 설치돼 있는 Impalad를 관리하는 역할을 한다.
 - Catalogd는 Impalad와 Statestored와 통신하면서 임팔라 SQL의 실행과 변경 이력을 관리




• 임팔라의 활용방안

- 하이브 쿼리를 임팔라 쿼리로 바꾸고, 데이터셋을 실시간 탐색한다.



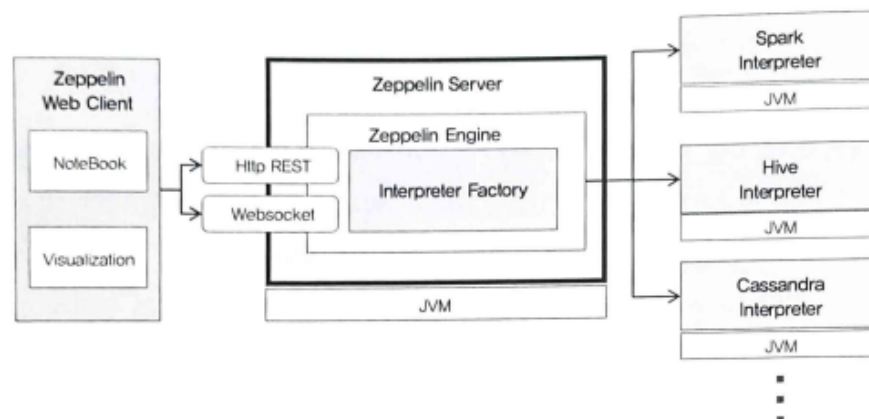
제플린

- 대용량 데이터를 효과적으로 탐색 및 분석하기 위해서는 대용량 데이터셋을 빠르게 파악하고 이해하기 위한 분석 및 시각화 툴이 필요하다.
- RHive, RHadoop, RHipe 같은 도구로 HDFS와 직접 연결해 병렬처리가 가능하도록 구성할 수 있지만 복잡도가 높아지고 안정적인 사용을 위해서는 추가 비용이 발생한다. 이러한 요건들을 해결하고자 스파크를 기반으로 하는 제플린(Zeppelin)이 탄생했다.

공식 홈페이지	 Apache Zeppelin	http://zeppelin.apache.org
주요 구성 요소	NoteBook	웹 상에서 제플린의 인터프리터 언어를 작성하고 명령을 실행 및 관리할 수 있는 UI
	Visualization	인터프리터의 실행 결과를 곧바로 웹 상에서 다양한 시각화 도구로 분석해 볼 수 있는 기능
	Zeppelin Server	NoteBook을 웹으로 제공하기 위한 웹 애플리케이션 서버로서 인터프리터 엔진 및 인터프리터 API 등을 지원
	Zeppelin Interpreter	데이터 분석을 위한 다양한 인터프리터를 제공하며, 스파크, 하이브, JDBC, 셀 등이 있으며 필요 시 인터프리터를 추가 확장
라이선스	Apache	
유사 프로젝트	Jupyter, Theia, RStudio	

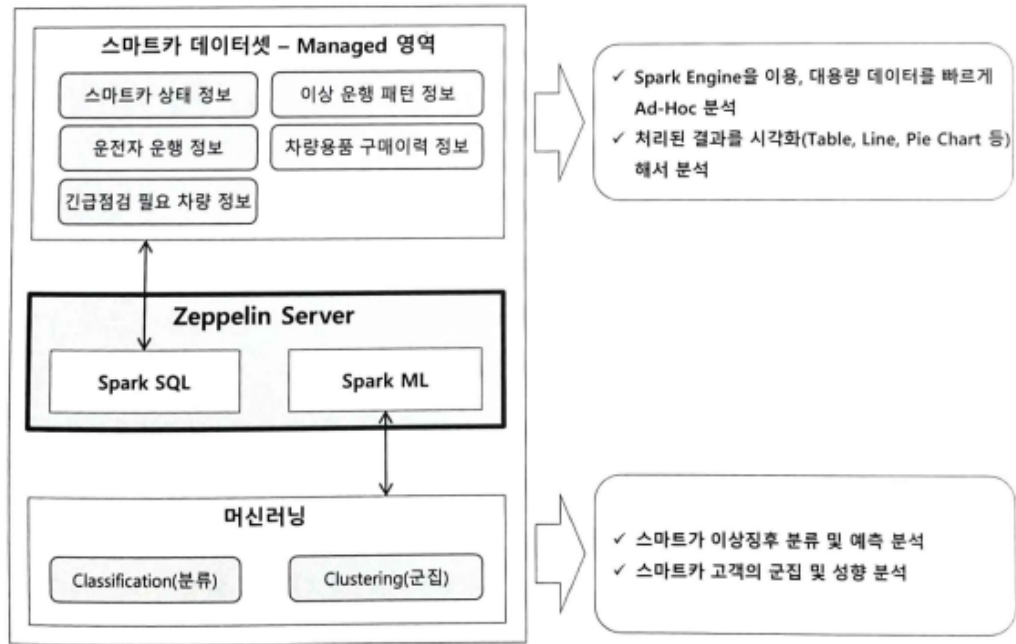
• 제플린 아키텍처

- 웹 UI의 NoteBook에서 스파크 또는 스파크 SQL을 작성해 하둡 클러스터에 작업을 요청, 처리 결과를 다시 웹 UI에 시각화해서 볼 수 있다.
- 이때 제플린의 클라이언트와 서버 사이에 REST또는 웹소켓 통신을 요청, 요청된 결과에 해당하는 인터프리터가 작동해서 타깃 시스템에 작업을 요청하게 된다.




• 제플린 활용 방안

- 마트 데이터를 대상으로 제플린에서 스파크 SQL을 이용해 다양한 에드혹 분석을 수행
- 스파크ML을 이용해 머신러닝의 분류와 군집으로 예측과 성향 분석 작업 진행



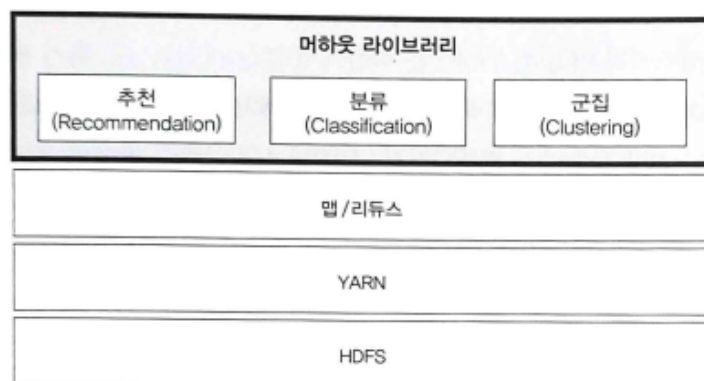
머하웃

- 머신러닝 기법을 이요해 데이터 마이닝을 수행하는 툴
- 대규모의 데이터셋을 분석할 수 있게 설계되어 있지 않고 분산 환경에서 실행하기 어렵다.
- 머하웃은 하둡에서 분산 머신러닝을 하기 위해 개발

공식 홈페이지		http://mahout.apache.org
주요 구성 요소	추천 (Recommendation)	사용자들이 관심을 가졌던 정보나 구매했던 물건의 정보를 분석해서 추천하는 기능으로, 유사한 사용자를 찾아서 추천하는 "사용자 기반 추천"과 항목 간 유사성을 계산해서 추천 항목을 생성하는 "아이템 기반 추천" 등이 존재
	분류 (Classification)	데이터셋의 다양한 패턴과 특징을 발견해 레이블을 지정하고 분류하는 기능으로, 주요 알고리즘으로 나이브 베이지안, 랜덤 포레스트, 로지스틱 회귀 등을 지원
	군집 (Clustering)	대규모 데이터셋에서 새로운 특성으로 데이터의 군집들을 발견하는 기능으로, 주요 알고리즘으로 K-Means, Fuzzy C-Means, Canopy 등을 지원
	감독학습 (Supervised Learning)	학습을 위한 데이터셋을 입력해서 분석 모델을 학습시키는 머신러닝 기법으로, 학습된 분석 모델을 이용해 예측하고 최적화하는 데 사용하고, 분류와 회귀 분석 기법이 이에 해당
	비감독학습 (Unsupervised Learning)	학습 데이터셋을 제공하지 않고 데이터의 특징적인 패턴을 발견하는 머신러닝 기법으로서 사람이 구분 및 그루핑하기 어려운 현상들을 자동으로 그루핑하는 데 사용하며, 군집 기법이 여기에 해당
라이선스	Apache	
유사 프로젝트	R, RapidMiner, Weka, TensorFlow, Spark ML, Scikit-learn	

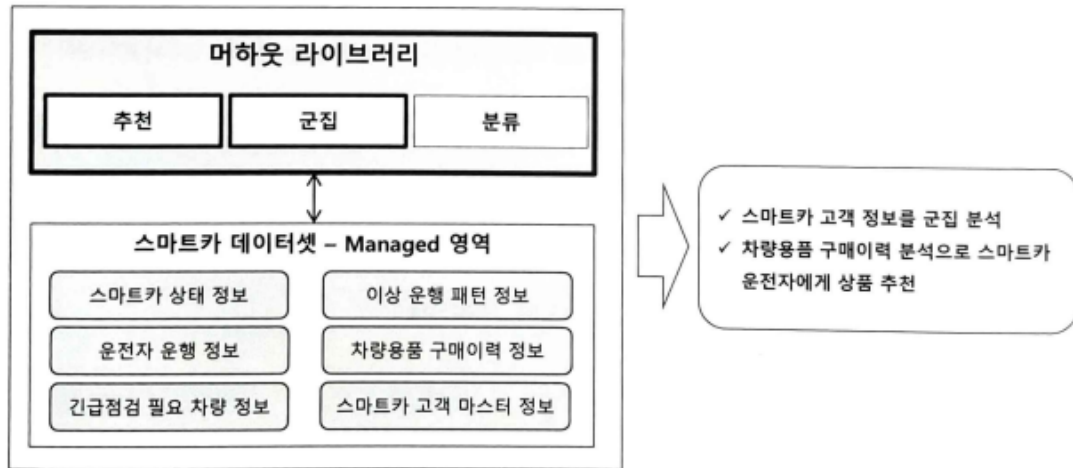
• 머하웃 아키텍처

- 하둡의 분산 환경 위에 맵리듀스를 기반으로 고급 분석을 지원하는 라이브러리 패키지다.




• 머하웃 활용 방안

- 정보를 대상으로 군집 분석을 진행해 고객군의 적정 개수를 파악하는 데 활용한다.



스쿱

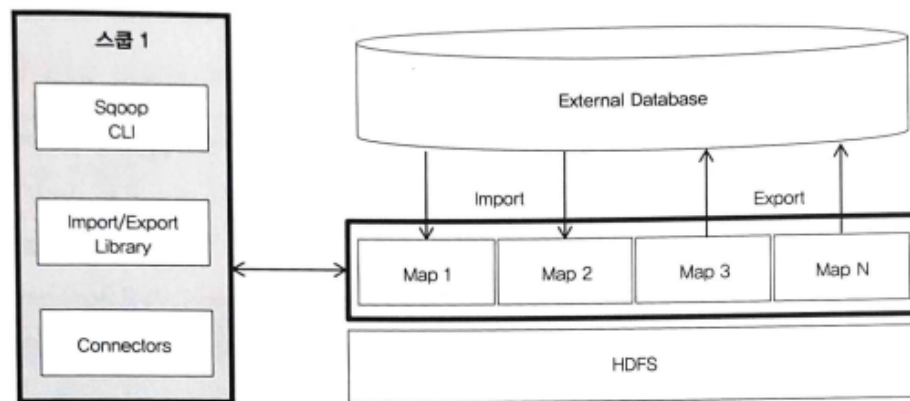
- RDBMS와 HDFS 사이에서 데이터를 편리하게 임포트하거나 익스포트해주는 소프트웨어

공식 홈페이지		http://sqoop.apache.org
주요 구성 요소	Sqoop Client	하둡의 분산 환경에서 HDFS와 RDBMS 간의 데이터 임포트 및 익스포트 기능을 수행하기 위한 라이브러리로 구성
	Sqoop Server	스쿱 2의 아키텍처에서 제공되며, 스쿱 1의 분산된 클라이언트 기능을 통합해 REST API로 제공
	Import/Export	임포트 기능은 RDBMS의 데이터를 HDFS로 가져올때 사용하며, 반대로 익스포트 기능은 HDFS의 데이터를 RDBMS로 내보낼 때 사용
	Connectors	임포트 및 익스포트에서 사용될 다양한 DBMS의 접속 어댑터와 라이브러리를 제공
	Metadata	스쿱 서버를 서비스하는 데 필요한 각종 메타 정보를 저장
라이선스	Apache	
유사 프로젝트	Hiho, Talend, Kettle	

• 스쿱 아키텍처

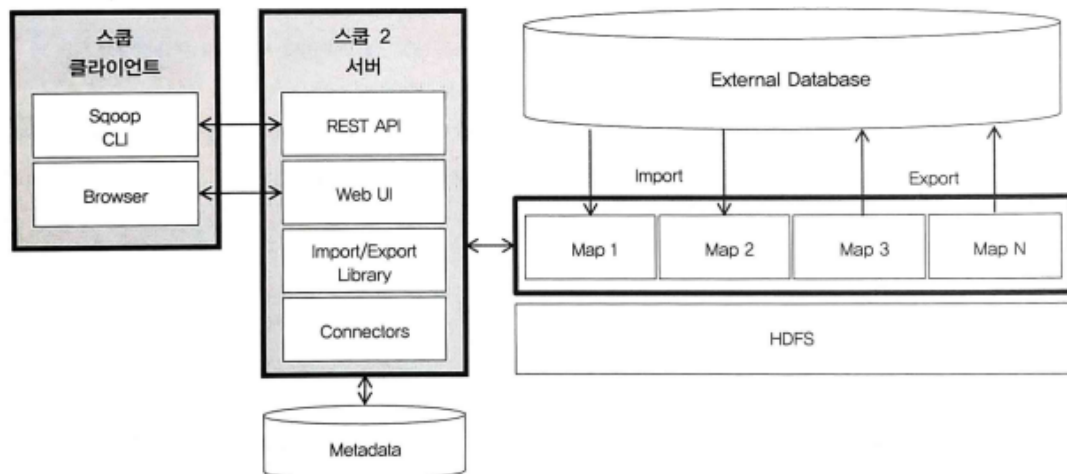
◦ 스쿱1

- 스쿱의 CLI로 임포트, 익스포트 명령을 하둡에 전달하면 맵 태스크가 병렬로 실행되어 외부 데이터베이스와 HDFS 사이에서 대량의 데이터를 임포트 및 익스포트 할 수 있는 아키텍처를 제공



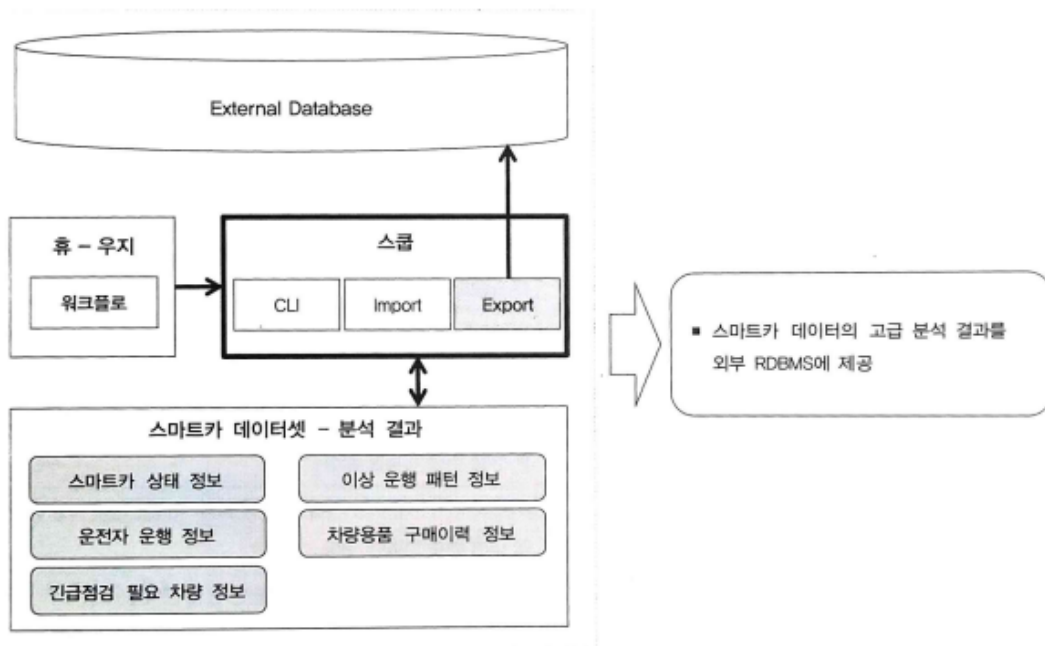
◦ 스쿱2

- 스쿱 1의 아키텍처를 확장해서 스쿱 서버를 추가한 것이다.



• 스쿱 활용 방안

- 하둡 생태계에선 수집(Import) 기술로 분류, 분석 결과를 외부에 제공(Export)

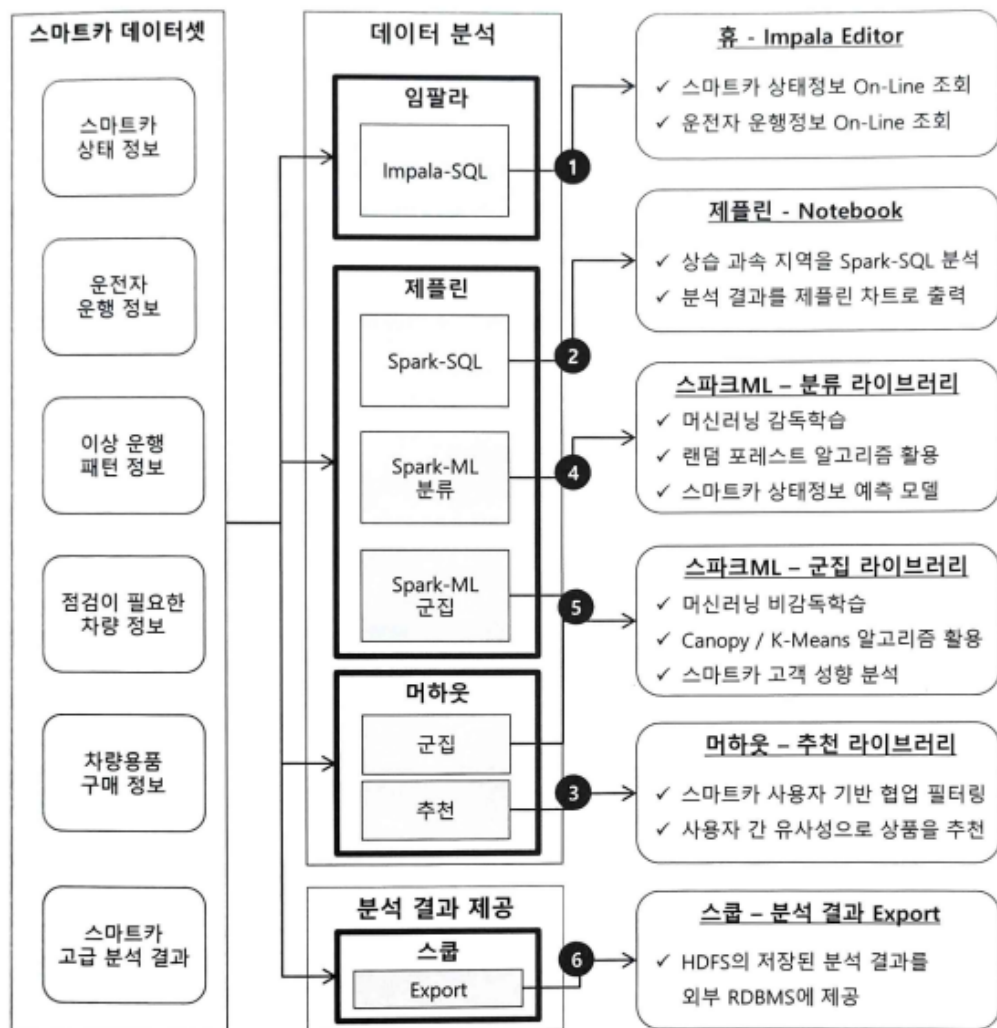


분석 파일럿 실행 1단계 - 분석 아키텍처

- 분석 요구사항
 - 요구사항 1: 차량의 다양한 장치로부터 발생하는 로그 파일을 수집해서 기능별 상태를 점검한다.
 - 요구사항 2: 운전자의 운행 정보가 담긴 로그를 실시간으로 수집해서 주행 패턴을 분석한다.

고급분석 요구사항 구체화	요구사항 분석 및 해결 방안
1. 스마트카 데이터셋을 좀 더 빠르게 탐색 및 분석한다.	임팔라를 이용해 기존 하이브 배치 쿼리를 임팔라의 온라인 쿼리로 실행해서 결과를 확인
2. 스마트카 데이터셋의 탐색 결과를 이해하기 쉽도록 시각화한다.	제플린을 이용해 스파크-SQL로 탐색한 데이터셋을 다양한 차트로 표현
3. 차량용품 구매 이력을 분석해 최적의 상품 추천 목록을 만든다.	머하웃의 추천을 이용해 차량용품 구매 이력을 분석해 성향에 따른 상품 추천 목록을 생성
4. 스마트카의 상태 정보를 분석해 이상 징후를 예측한다.	스파크ML의 머신러닝 기법 중 분류 감독 학습을 통해 이상 징후에 대한 예측 모델을 구성
5. 스마트카 운전자의 마스터 정보를 분석해 고객 군집을 도출한다.	머하웃과 스파크ML로 비감독 학습인 군집 분석을 수행
6. 분석된 결과는 외부 업무 시스템의 RDBMS에 제공되어야 한다.	스킵의 데이터 익스포트 기능을 이용해 HDFS에 저장된 분석 결과를 RDBMS로 전달

- 분석 아키텍처



분석 파일럿 실행 2단계 - 분석 환경 구성

• 임팔라 설치

- 서버2에 역할 할당
- Hue → 구성 → Impala 검색
- **Impala 서비스** 에서 none을 Impala로 설정
- 임팔라 재시작, 휴 재시작
- Hue UI에 들어가 Impala가 있는지 확인

• 스콥 설치

- 서버2에 역할 할당
- 클라이언트 구성 재배포

- 제플린 설치

- xshell server02로 가서 cd /home/pilot-pjt/working으로 경로 이동
- wget <http://archive.apache.org/dist/zeppelin/zeppelin-0.8.2/zeppelin-0.8.2-bin-all.tgz> 입력후 설치
- tar -xvf zeppelin-0.8.2-bin-all.tgz 압축 해제
- ln -s zeppelin-0.8.2-bin-all zeppelin 으로 심볼릭 변경
- 제플린의 환경 정보를 설정
 - cd /home/pilot-pjt/zeppelin/conf 로 경로이동
 - cp zeppelin-env.sh.template zeppelin-env.sh 복사
 - 복사한 파일 열기 vim zeppelin-env.sh
 - o를 눌러 밑으로 내려간다음,
 - export JAVA_HOME=/usr/java/jdk1.8.0_181-cloudera
 - export SPARK_HOME=/opt/cloudera/parcels/CDH/lib/spark
 - export HADOOP_CONF_DIR=/etc/hadoop/conf
 - chmod 777 /tmp/hive로 이동
 - cp /etc/hive/conf/hive-site.xml /home/pilot-pjt/zeppelin/conf 복사
 - cd /home/pilot-pjt/zeppelin/conf 이동
 - cp zeppelin-site.xml.template zeppelin-site.xml 복사
 - vim zeppelin-site.xml 실행하여 127.0.0.1 을 0.0.0.0으로 바꾸고 포트 번호 8080을 8081로 변경
 - vi /root/.bash_profile 패스 설정
 - PATH=\$PATH:/home/pilot-pjt/zeppelin/bin 설정
 - source /root/.bash_profile 한 후에 zeppelin-daemon.sh start 시작
 - server02.hadoop.com:8081로 접속해 확인

- 머하웃 설치

- cd /home/pilot-pjt/ 경로 이동
- wget <http://archive.apache.org/dist/mahout/0.13.0/apache-mahout-distribution-0.13.0.tar.gz> 설치
- tar -xvf apache-mahout-distribution-0.13.0.tar.gz 압축해제

- In -s apache-mahout-distribution-0.13.0 mahout 링크 설정
- vi /root/.bash_profile 를 통한 환경 변수 설정
 - PATH=\$PATH:/home/pilot-pit/mahout/bin
 - export JAVA_HOME=/usr/java/jdk1.8.0_181-cloudera
- source /root/.bash_profile 한 후에 mahout 입력하여 설치 확인

분석 파일럿 실행 3단계 - 임팔라를 이용한 데이터 실시간 분석

▼ 하이브 QL를 임팔라에서 실행하기

- ▼ 하이브의 특성으로 인해 다소 많은 시간이 소요되는 반면 임팔라는 하이브로 실행했을 때의 탐색 시간과는 비교가 안되게 빠른 속도로 데이터가 조회되는 것을 확인할 수 있다.

▼ 인메모리 기반 분석 엔진 사용

- ▼ 임팔라는 빠른 응답 속도를 보여 주지만 많은 리소스를 사용한다.
- ▼ 빅데이터 하둡 환경에서는 임팔라와 하이브를 상황에 맞춰 선택적으로 사용해야 한다,
- ▼ 마트화된 소규모 데이터셋을 대상으로 빠른 에드혹 분석을 수행할 때는 임팔라를 사용하고, 대규모 데이터셋에서 긴 시간에 걸친 가공 및 추출 작업에는 반드시 하이브를 이용한다.
- ▼ 마트 데이터는 집계/요약된 작은 데이터셋으로. 임팔라 작업시 메모리에 대한 부담이 적기 때문이다.

분석 파일럿 실행 4단계 - 제플린을 이용한 실시간 분석

▼ 제플린을 이용한 운행 지역 분석

- ▼ 제플린이 구동됐는지 확인 : `zeppelin-daemon.sh status`
- ▼ `server02.hadoop.com:8081` 로 제플린 접속
- ▼ `hdfs dfs -ls -R /user/hive/warehouse/managed_smartcar_drive_info/` 하위 디렉터리 안에 있는 파일 모두 보여줌
- ▼ `hdfs dfs -cat /user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/* | head` 정보 확인
 - ▼ `val url="hdfs://server01.hadoop.com:8020"`
 - `val`
 - `dPath="/user/hive/warehouse/managed_smartcar_drive_info/biz_date=20220429/*"`

```

val driveData=sc.textFile(url + dPath)
case class DriveInfo(car_num: String, sex: String, age: String,
marriage: String, region: String, job: String,
car_capacity: String, car_year: String,car_model: String,
speed_pedal: String, break_pedal: String,steer_angle: String,
direct_light: String,speed: String,area_num: String,
date: String)
val drive = driveData.map(sd=>sd.split(",")).map(
sd=>DriveInfo(sd(0).toString, sd(1).toString, sd(2).toString, sd(3).toString,
sd(4).toString, sd(5).toString, sd(6).toString, sd(7).toString,
sd(8).toString, sd(9).toString, sd(10).toString,sd(11).toString,
sd(12).toString,sd(13).toString,sd(14).toString,sd(15).toString
)
)
drive.toDF().registerTempTable("DriveInfo")

▼ %spark.sql
select T1.area_num, T1.avg_speed
from(select area_num, avg(speed) as avg_speed
from DriveInfo
group by area_num
) T1
order by T1.avg_speed desc

```