Data Playground 7
at 7 p.m. every month with talk & beer
TAKE SOME TIME OFF

하둡과 Kudu를 활용한 Data Lake 활용 사례

발표자 : 지용기

목차

빅데이터 플랫폼

초 간단한 소개

KUDU

간단한 소개

Kudu 와 Impala를 사용한 계층적 저장소

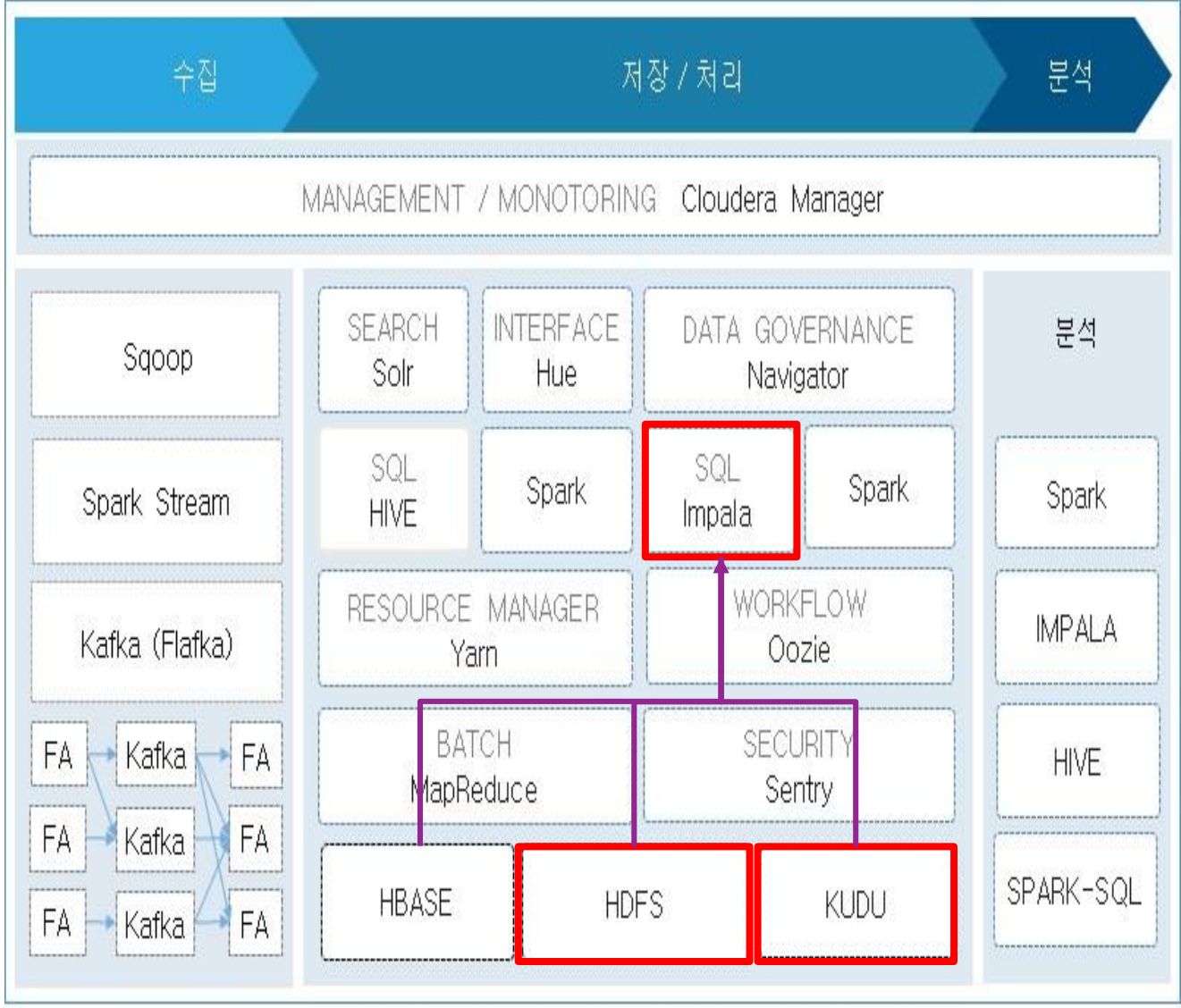
내용 요약

자기 소개

- ✓ 생물학 관련 학과 졸업
- ✓ 모바일(brew, win-ce, wipi, skvm) 개발을 2012년까지
- ✓ 갤럭시A, S, Tab, S2의 메시지 어플 개발
- ✓ 2012년 이후부터 빅데이터 관련 업무~~
- ✓ 2016년 방송통신대학교 통계학과 졸업
- ✓ 커뮤니티 활동은 바이오스핀 그룹에서 유전체 데이터 분석을 위주로~~ (Tensorflow, R, Python)

빅데이터 플랫폼

- ✓ HDFS, KUDU 저장 방식
- ✓ Impala는 저장소로 HBASE, HDFS, KUDU 등을 사용하고 이 데이터를 조회할 수 있는 쿼리엔진.



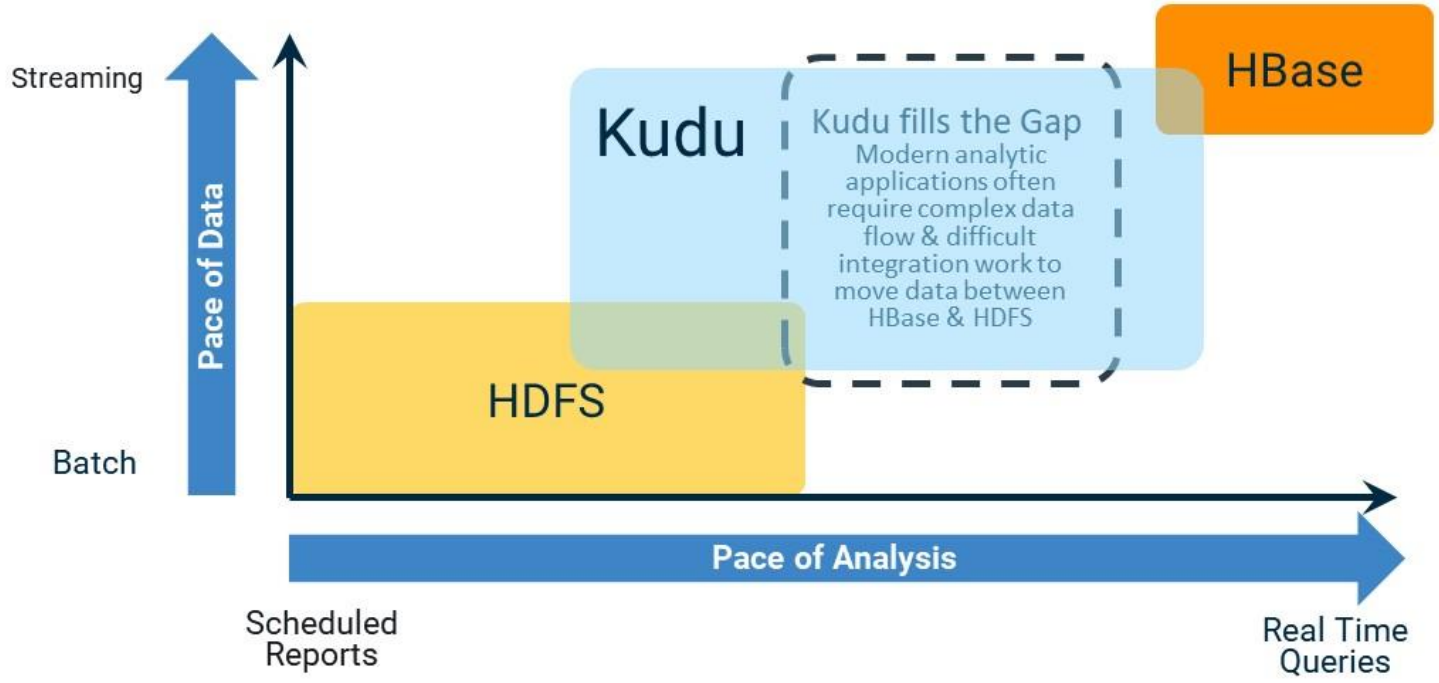
IMPALA

- ✓ 메모리 기반의 SQL on Hadoop 엔진.
- ✓ https://impala.apache.org/
- ✓ ETL은 Hive가 더 빠름

구분	Apache Hive	Apache Impala	Apache Spark SQL
활용대상	ETL 개발자	분석가	데이터 엔지니어 & 사이언티스트
강점	• 장시간 실행되는 ETL 작업, 데이터 준비 작업 , 배치처리 작업에 적합 • 다양한 커스텀 파일 포맷 지원 • 대용량 ETL Sorting 및 Join 오퍼레이션 처리	• High-Concurrency 환경 지원 • 고성능의 Interactive SQL 지원 • 기존 BI Tool 및 스킬 셋과 호환	• Java Scala, Python 어플리케이션 내 손쉽게 SQL 활용 • 어플리케이션 내 SQL과 Spark 코드 함께 활용
새로운 기능	• Hive on Spark • Faster Hive on S3	• KUDU 지원 • 10x faster BI/SQL	• PySpark-SparkSQL 및 DataFrame 지원 • Spark on KUDU

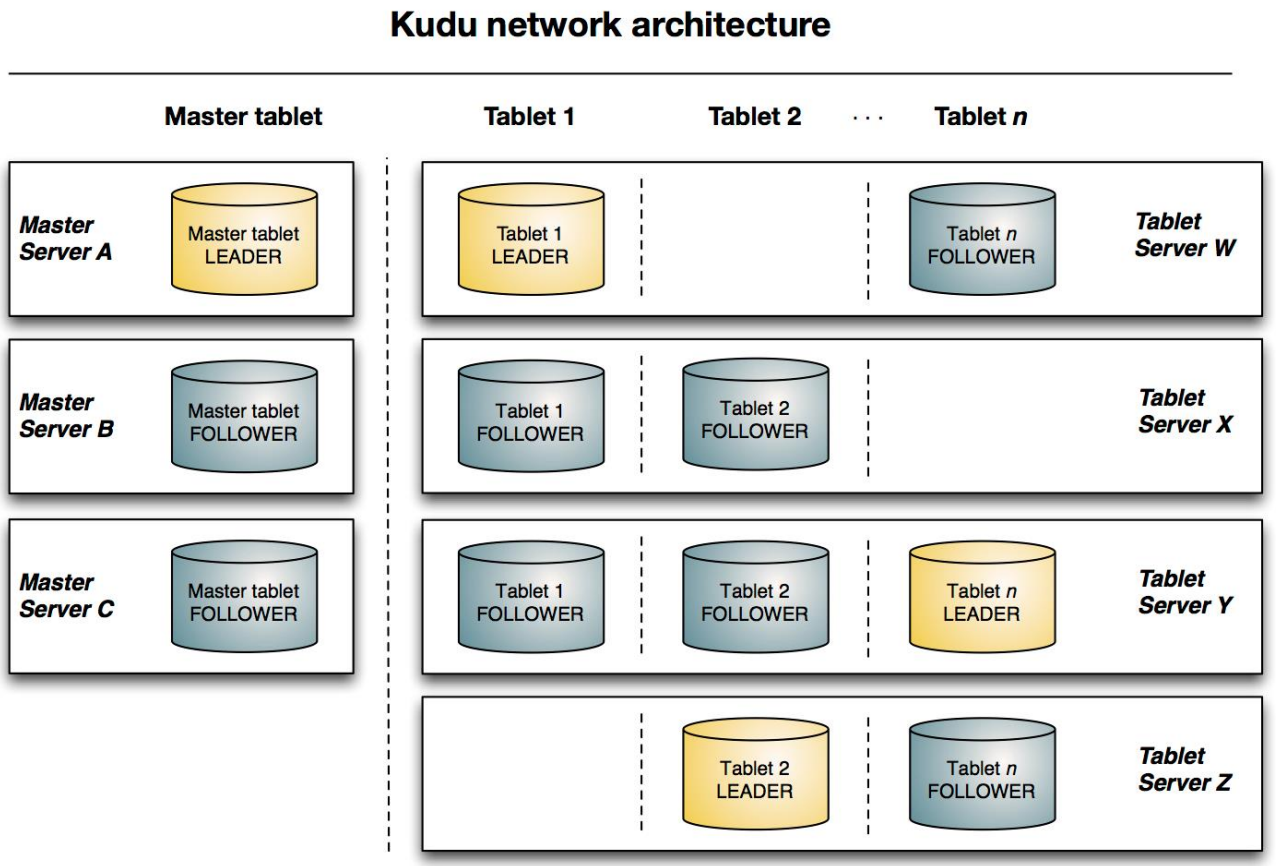
KUDU 소개

- ✓ Cloudera에서 개발한 칼럼 기반 스토리지(columnar storage).
- ✓ primary key를 제공하기 때문에 밀리초(ms) 수준의 랜덤 액세스가 가능.
- ✓ OLAP질의와 OLTP질의를 모두 지원
- ✓ 쿼리 엔진으로 Impala 사용
- ✓ https://d2.naver.com/helloworld/9099561(Kudu를 이용한 빅데이터 다차원 분석 시스템 개발)



KUDU 아키텍쳐

- ✓ Master Server가 카탈로그 역할
- ✓ 각각 Tablet Server가 각자의 Tablet을 저장
- ✓ Tablet이 실제 데이터를 저장하고 있는 파일



KUDU 제약사항

- ✓ Master의 최대 개수 3대.
- ✓ Tablet server 최대 개수 100대.
- ✓ Tablet server 당 최대 8TB.
- ✓ Tablet server 당 최대 tablet의 개수 2000개.
- ✓ Write ahead logs (WALs)는 하나의 디스크에만 저장. (병목지점 , SSD권장)
- ✓ 참조 : https://www.cloudera.com/documentation/enterprise/5-15-x/topics/kudu_limitations.html#kudu_limitations
- ✓ 최대 저장 용량 : 100대 * 8TB / 3복제수 = 267 TB
- ✓ 최대 tablet 개수 : 100대 * 2000개 / 3복제수 = 66,666개 (약 6만6천개)
- ✓ 하둡(HDFS)에 대비 저장 공간이 작고 고비용

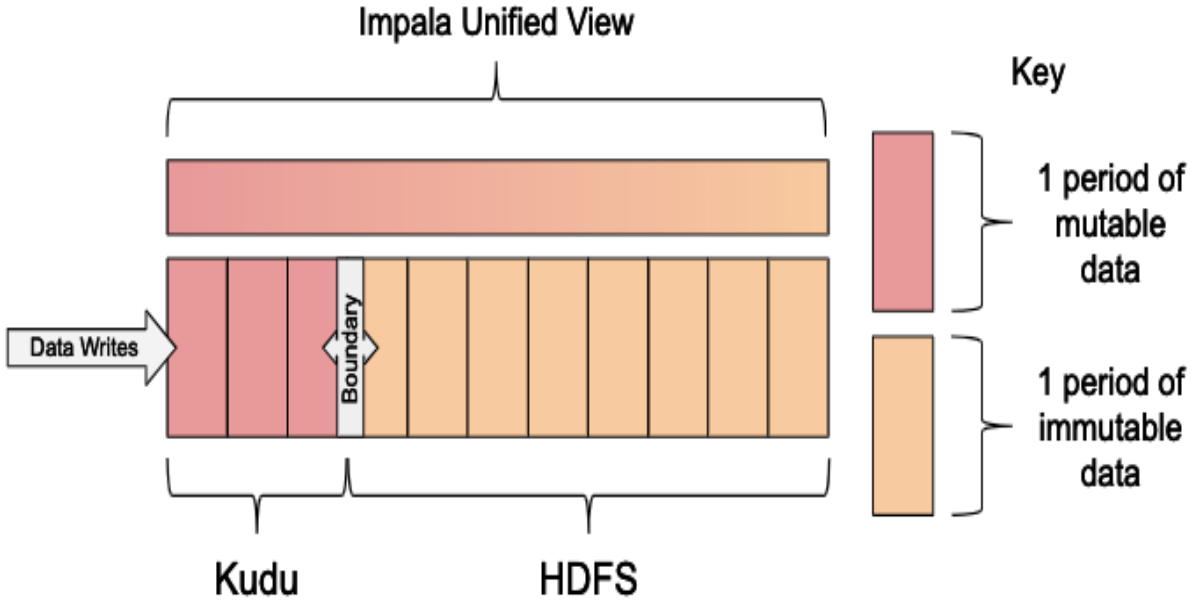
Kudu 와 Impala를 사용한 계층적 저장소 관리

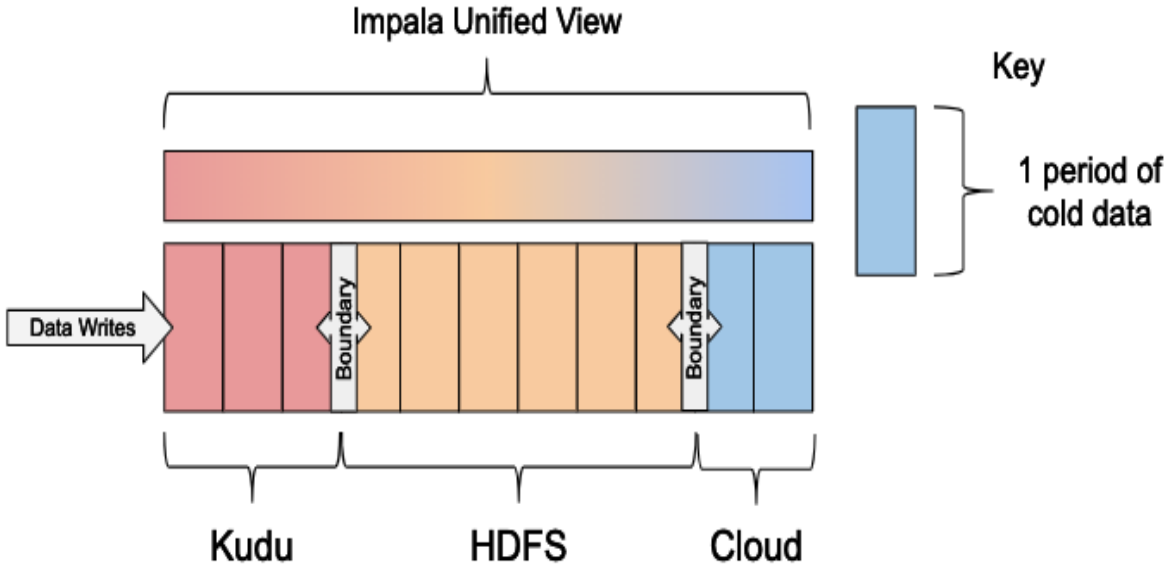
- ✓ HDFS는 낮은 비용으로 무제한의 확장성을 제공
- ✓ 신규 데이터는 Kudu에 오래된 데이터는 HDFS 에 저장하여 관리
- ✓ 슬라이딩 윈도우 패턴 적용
- ✓ https://blog.cloudera.com/blog/2019/03/transparent-hierarchical-storage-management-with-apache-kudu-and-impala/

슬라이딩 윈도우 패턴

- ✓ 일정 시간 동안의 데이터는 Kudu 테이블로 저장
- ✓ 기간이 지난 데이터는 HDFS로 이관
- ✓ Kudu와 HDFS를 동시에 조회할 수 있도록 View 테이블을 제공
- ✓ 데이터의 이동은 ALTER VIEW문을 사용하여 경계를 이동시킴

- ✓ Impala는 S3 및 ADLS 와 같은 클라우드 저장소 옵션 지원
- ✓ 가끔씩만 조회되는 "cold" data는 S3에 저장 관리 가능

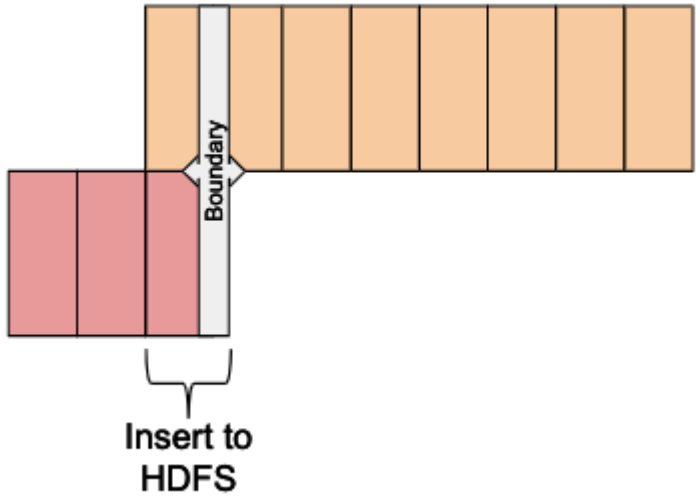




Kudu에서 HDFS로 데이터를 이동

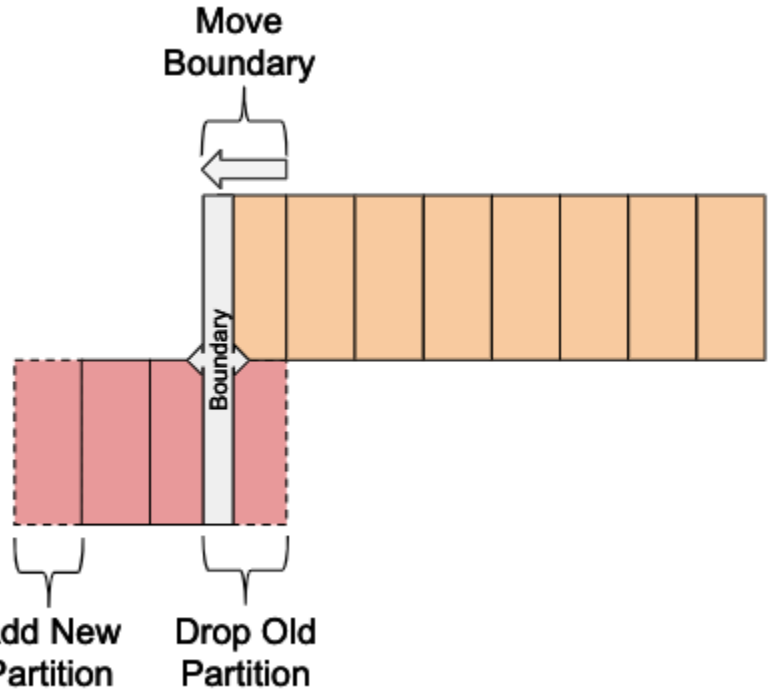
첫번째 단계

- ✓ 데이터가 Kudu에서 HDFS로 복사.
- ✓ 중복 데이터는 View Table에서 중복을 방지함.
- ✓ 데이터 유효성 검사 수행 필요.



두번째 단계

- ✓ View Table 을 수정하여 경계를 앞으로 이동시킴.
- ✓ Kudu의 Old Partition을 삭제



Kudu에서 HDFS로 데이터를 이동

✓ window_data_move.sql

```
INSERT INTO ${var:hdfs_table} PARTITION (year, month, day)
SELECT *, year(time), month(time), day(time)
FROM ${var:kudu_table}
WHERE time >= add_months("${var:new_boundary_time}", -1)
AND time < "${var:new_boundary_time}";
COMPUTE INCREMENTAL STATS ${var:hdfs_table};
```

```
impala-shell -i <impalad:port> -f window_data_move.sql
--var=kudu_table=my_table_kudu
--var=hdfs_table=my_table_parquet
--var=new_boundary_time="2018-02-01"
```

✓ window_view_alter.sql

```
ALTER VIEW ${var:view_name} AS
SELECT name, time, message
FROM ${var:kudu_table}
WHERE time >= "${var:new_boundary_time}"
UNION ALL
SELECT name, time, message
FROM ${var:hdfs_table}
WHERE time < "${var:new_boundary_time}"
AND year = year(time)
AND month = month(time)
AND day = day(time);
```

```
impala-shell -i <impalad:port> -f window_view_alter.sql
--var=view_name=my_table_view
--var=kudu_table=my_table_kudu
--var=hdfs_table=my_table_parquet
--var=new_boundary_time="2018-02-01"
```

Kudu에서 HDFS로 데이터를 이동

✓ window_partition_shift.sql

```
ALTER TABLE ${var:kudu_table}

ADD RANGE PARTITION add_months("${var:new_boundary_time}",
${var:window_length}) <= VALUES < add_months("${var:new_boundary_time}",
${var:window_length} + 1);

ALTER TABLE ${var:kudu_table}

DROP RANGE PARTITION add_months("${var:new_boundary_time}", -1)
<= VALUES < "${var:new_boundary_time}";
```

```
impala-shell -i <impalad:port> -f window_partition_shift.sql
--var=kudu_table=my_table_kudu
--var=new_boundary_time="2018-02-01"
--var=window_length=3
```

이 방법의 단점

- ✓ 테이블이 많을때는 사람의 노력과 데이터 이동 시간이 많이 필요
- ✓ 테이블당 1TB 정도의 크기일때 유용
- ✓ 하둡 또는 KUDU만으로 대부분 처리 가능
- ✓ 제가 알고 있는 국내 사례로 제조사 한 곳.(HDFS : 700TB, KUDU : 30TB 사용중)

감사합니다.

Data Playground @ 7

https://www.facebook.com/groups/databreak/

http://databreak.org/