Hadoop_relational data store

NoSQL

- 대용량의 비정형화된 데이터를 다루기에 적합하다.
- 비관계형 데이터베이스
- 수평 확장이 가능하여 매우 빠르고 탄력적이다.

(1) HBase

- HDFS 위에 구축된다.
- 수평 분할 된 HDFS 파일 시스템에 저장된 데이터를 요청하는 데 매우 빠르고 확장성이 뛰어난 트랜잭션 시스템을 사용할 수 있다.
- 비관계형 데이터베이스이다.
- HDFS를 기반으로 하는 확장 가능한 데이터베이스이다.
- query는 없지만 신속하게 대응할 수 있는 API를 가지고 있다.
- Google사의 big data 처리 개발 프로그램인 BigTable이 기초 아이디어가 되었다.

CRUD API를 갖고 있다.

- Create
- Read
- Update
- Delete

- HDFS 위에서 키 범위에 대하여 분할된 region으로 나뉘어진다.
  - 데이터가 추가되면 자동으로 적용된다.
  - master node를 다루는 것이 아니라 region을 다룬다.

HBase data model

- 각 행에 빠르게 접근할 수 있다.
- 각 행은 unique key에 의해 조회할 수 있다.
- 각 행은 적은 수의 column family를 갖고 있다.
- column family는 무작위의 column을 포함한다.
- column familly는 많은 수의 column을 가지고 있을 수 있다.
- 각 셀은 timestamp가 주어진 많은 버전을 가질 수 있다.
- sparse data - 누락된 셀은 저장공간을 소모하지 않는다.

HBase 접근 방법

- HBase shell
- JAVA API - Python, Scalar 등을 통해
  - HBase 자체는 JAVA로 작성되었다.
- Spark, Hive, Pig
- REST service
  - HTTP request
- Thrift service - Facebook이 만들었다.
  - binary 형식이기 때문에 결과가 빽빽하게 표현될 수 있다.
  - 결과를 효율적으로 저장하고 빠르고 전송할 수 있다는 장점이 있다.
- Avro service
  - binary 형식이기 때문에 결과가 빽빽하게 표현될 수 있다.
  - 결과를 효율적으로 저장하고 빠르고 전송할 수 있다는 장점이 있다.

HBase Exercise - import movie ratings

- Python => Rest => HBase/HDFS

- HBase와 연결 가능 확인하기
  - virtual box의 Hortonworks setting
  - 네트워크 => 고급 => 포트 포워딩
  - 호스트 포트 8000 확인
  - 없을 경우 추가

- Ambari 통하여 HBase 시작
- HDP 통하여 superuser 로그인 : su root

```
$ /usr/hdp/current/hbase-master/bin/hbase-daemon.sh start rest -p 8000 --infoport 8001
```

In [ ]:

```
from starbase import Connection

c = Connection("127.0.0.1", "8000") # localhost, virtual box

ratings = c.table('ratings')

if (ratings.exists()):
    print("Dropping existing ratings table₩n")
    ratings.drop()

ratings.create('rating')

print("Parsing the ml-100k ratings data ...₩n")
ratingFile = open("C:/Users/JI SEONG MIN/Desktop/epopcon/ml-100k/u.data", "r")

batch = ratings.batch() # batch는 1개의 row를 추가한다.

for line in ratingFile:
    (userID, movieID, rating, timestamp) = line.split()
    batch.update(userID, {'rating': {movieID: rating}})

ratingFile.close()

print("Committing ratings data to HBase via REST service₩n")
batch.commit(finalize=True)

print("Get back ratings for some users...₩n")
print("Ratings for user ID 1:₩n")
print(ratings.fetch("1"))
print("Ratings for user ID 33:₩n")
print(ratings.fetch("33"))

ratings.drop()
```

- shut down 명령어

```
$ usr/hdp/current/hbase-master/bin/hbase-daemon.sh stop rest
```

(2) Pig

- 사전에 HBase 테이블을 생성해야 한다.
- HBase에 저장된 열이 있어야 하며 unique key가 첫번째 행이여야한다.
- "USING"과 "STORE" 구문을 사용하여 HBase 테이블에 저장할 수 있다.
- 행에서 변환이 가능하다.

(3) Cassandra

- 단일 장애 지점이 없다. - 마스터노드가 없다.
  - 모든 노드는 정확하게 같은 노드로서 같은 기능으로 같은 일을 한다.
- 데이터 모델은 BigTable이나 HBase와 유사하다.
- 비관계형이지만 자체 인터페이스로 제한된 CQL 쿼리를 가진다.
- 일관성보다 가용성을 선호한다.(CAP 이론)
  - 결국은 일관성이 된다. 조정 가능한 일관성

(4) mongoDB

- 가용성보다 일관성을 유지한다.
  - 단일 마스터 데이터베이스를 사용한다.
  - 그러나 데이터베이스의 백업 복사본을 유지 관리 합니다.
    - 메인이 다운되면 보조 노드로 대체된다.
    - 하지만 작업 로그가 실행할 때 기본 로그를 복구 할 시간이 충분해야한다.
- 자동으로 추가되는 "_id" column을 제공한다.
- 원하는 모든 문서에서 다른 필드를 구성할 수 있다.
- 다른 데이터베이스와 마찬가지로 단일 'key'가 없지만 원하는 필드에 인덱스를 만들거나 필드 조합을 만들 수 있다.
  - 분리하고 싶다면 일부 index를 사용해야 한다.
- 유연성이 매우 높다.

데이터베이스 선택 시 고려해야할 점

- 통합 시스템
  - 관계형 데이터베이스
  - 비관계형 데이터베이스
- 데이터의 확장성
- 지원 사항 - 보안 등
- CAP - 일관성, 가용성, 분리 가능 중 두가지만 선택 가능(분리 가능에 대한 항목은 고정)
- 간단하게 유지되는지 여부