Apache Phoenix

SQL for HBase

meant for HBase only

Bernard Lee Kok Bang

Apache Phoenix

- Conceptually similar to Drill – offers a SQL interface on top of non-relational database
- Main difference: *only works with HBase*

1.can only run limited SQL-like query
2. can only work with primary key

- A SQL driver for HBase that supports transactions
- Fast, low-latency – support OLTP online transaction process
- Originally developed by salesforce.com (a cloud-based software company), then open-sourced
- Exposes a JDBC connector (driver for Java applications) to HBase
- Support secondary indices and user-defined functions (for complex query)
- Integrates with MapReduce, Spark, Hive, Pig, and Flume (integration of JAR files into Phoenix)

HBase is still a non-relational database - high transaction, not meant for analytical query

Why Phoenix?

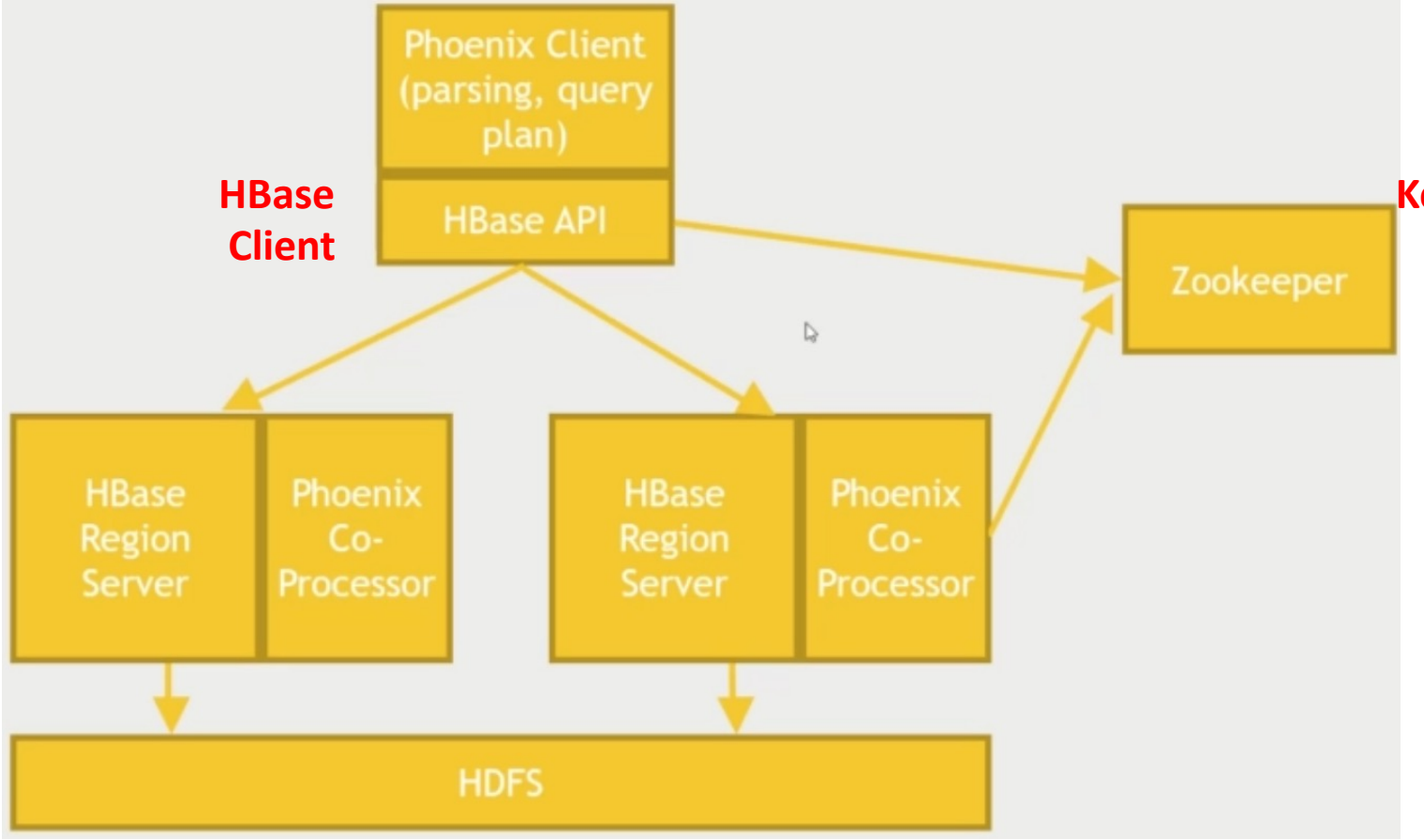
- It is fast [highly optimized, and focused on taking complex queries and making them execute on HBase]
- Why Phoenix and not Drill?

  - *Choose the right tool for the job [if we know we're going to deal with HBase data, and we need SQL, then Phoenix might be a better choice as it focuses exclusively on HBase]*
- Why Phoenix and not HBase native clients?

  - Our apps, and analysts, may find SQL easier to work with
  - Phoenix can do the work of optimizing more complex queries
    - But remember HBase is still fundamentally non-relational!

given user ID 100, give me back all the associated information

Phoenix Architecture



HBase Client

Keep track of what region servers are available

range of keys

Using Phoenix

- Command-Line Interface (CLI)
- Phoenix API for Java -> link to Java program and code directly
- JDBC Driver (thick client)

desktop or PC; whereby we don't need to connect to server; offline

- Phoenix Query Server (PQS) (thin client; virtual desktop)
  - Intended to eventually enable non-JVM access

exclusive softwares; limited license; limited users; online

- JAR's for MapReduce, Hive, Pig, Flume, Spark

Thick vs Thin Clients

- Refer to this YouTube URL for better explanation:

https://www.youtube.com/watch?v=gctocRTgq-U

Play with Phoenix

- Install Phoenix on the Hortonworks Sandbox
- Mess around with the CLI
- Set up a users table for MovieLens dataset into HBase via Phoenix
- Store and load data through Pig integration

Install Phoenix and Query HBase

- Login to Ambari as *admin*
- Start HBase service
- Login to puTTY as root
  - *su root*
- Phoenix has been installed [for HDP 2.6.5]
- Navigate to Phoenix folder to kickstart Phoenix CLI
  - *cd /usr/hdp/current/phoenix-client/*
  - *cd bin*
  - *python sqlline.py* **-> in Phoenix CLI**

Phoenix CLI with simple example

see lists of tables available in HBase

- *!tables*

Create a new table

- *CREATE TABLE IF NOT EXISTS my_population (*
  state char(3) NOT NULL,
  city VARCHAR NOT NULL,
  population BIGINT
  CONSTRAINT my_pk PRIMARY KEY (state,city));

Still in HBase;
need primary key

Phoenix CLI with simple example (cont…)

- # Insert rows of data into the table
  - *UPSERT INTO MY_POPULATION VALUES ('SEL', 'Bangi', 3400289);*
  - *UPSERT INTO MY_POPULATION VALUES ('SWK', 'Kuching', 1987345);*
  - *SELECT * FROM MY_POPULATION;*
  - *SELECT * FROM MY_POPULATION WHERE STATE='SWK';*

Integrate Phoenix with Pig

- # First create *users* table for MovieLens dataset in HBase
  - *CREATE TABLE users (*
  - *USERID INTEGER NOT NULL,*
  - *AGE INTEGER,*
  - *GENDER CHAR(1),*
  - *OCCUPATION VARCHAR,*
  - *ZIP VARCHAR*
  - *CONSTRAINT pk PRIMARY KEY (USERID));*

  - *!tables*
  - *!quit*

Integrate Phoenix with Pig (cont…)

- #Navigate back to maria_dev directory
- cd /home/maria_dev/
- #Make sure you still have *u.user* file in ml-100k folder [in HDFS]

wget http://media.sundog-soft.com/hadoop/ml-100k/u.user

Writing a Pig script that could connect Phoenix to HBase

vi phoenixHBase.pig

```
# Set the zookeeper node
set zookeeper.znode.parent '/hbase-unsecure'

# State where to get the client library for phoenix -> where to find Java class it needs to connect Pig to Phoenix and to HBase
REGISTER /usr/hdp/current/phoenix-client/phoenix-client.jar

# Load raw data for u.user; make sure each data types matched to the column data types defined in HBase [we are still in Pig]
users = LOAD '/user/maria_dev/ml-100k/u.user'
USING PigStorage('|')
AS (USERID:int, AGE:int, GENDER:chararray, OCCUPATION:chararray,
ZIP:chararray);

# Store the relation from Pig into HBase with parameters for hostname and batch size
STORE users into 'hbase://users' using
   org.apache.phoenix.pig.PhoenixHBaseStorage('localhost','-batchSize 5000');
```

batch size - how many rows of data we expect before proceed

Writing a Pig script that could connect Phoenix to HBase (cont…)

 # Load users data back to Pig from HBase; choose only data for userid and occupation, with parameter of hostname

occupations = load 'hbase://table/users/USERID,OCCUPATION' using org.apache.phoenix.pig.PhoenixHBaseLoader('localhost');

 # Create a group by relation

grpd = GROUP occupations BY OCCUPATION;

 # Create a relation that count number of people per occupation

cnt = FOREACH grpd GENERATE group AS OCCUPATION,COUNT(occupations);

DUMP cnt;

 # To submit the script

pig phoenixHBase.pig

```
(none,9)
(other,105)
(artist,28)
(doctor,7)
(lawyer,12)
(writer,45)
(retired,14)
(student,196)
(educator,95)
(engineer,67)
(salesman,12)
(executive,32)
(homemaker,7)
(librarian,51)
(marketing,26)
(scientist,31)
(healthcare,16)
(programmer,66)
(technician,27)
(administrator,79)
(entertainment,18)
```

Count how many members for each occupation; e.g. How many doctors in u.user file

Clean up the mess!!

- Navigate to Phoenix folder to kickstart Phoenix CLI
  - *cd /usr/hdp/current/phoenix-client/*
  - *cd bin*
  - *python sqlline.py*
- # Drop the table to clean up the mess
  - *!tables*
  - *DROP TABLE MY_POPULATION;*
  - *DROP TABLE USERS;*
  - *!table*
  - *!quit*
- Stop HBase as well