Apache Sqoop

iXAT Solution – ixatsolutions@gmail.com

What is Sqoop

- Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases.
- Sqoop imports data from external structured datastores into HDFS or related systems like Hive and HBase.
- Sqoop can also be used to export data from Hadoop and export it to external structured datastores such as relational databases and enterprise data warehouses.
- Sqoop works with relational databases such as: Teradata, Netezza, Oracle, MySQL, Postgres, and HSQLDB.

Why Sqoop?

- As more organizations deploy Hadoop to analyse vast streams of information, they may find they need to transfer large amount of data between Hadoop and their existing databases, data warehouses and other data sources
- Loading bulk data into Hadoop from production systems or accessing it from map-reduce applications running on a large cluster is a challenging task since transferring data using scripts is a inefficient and time-consuming task

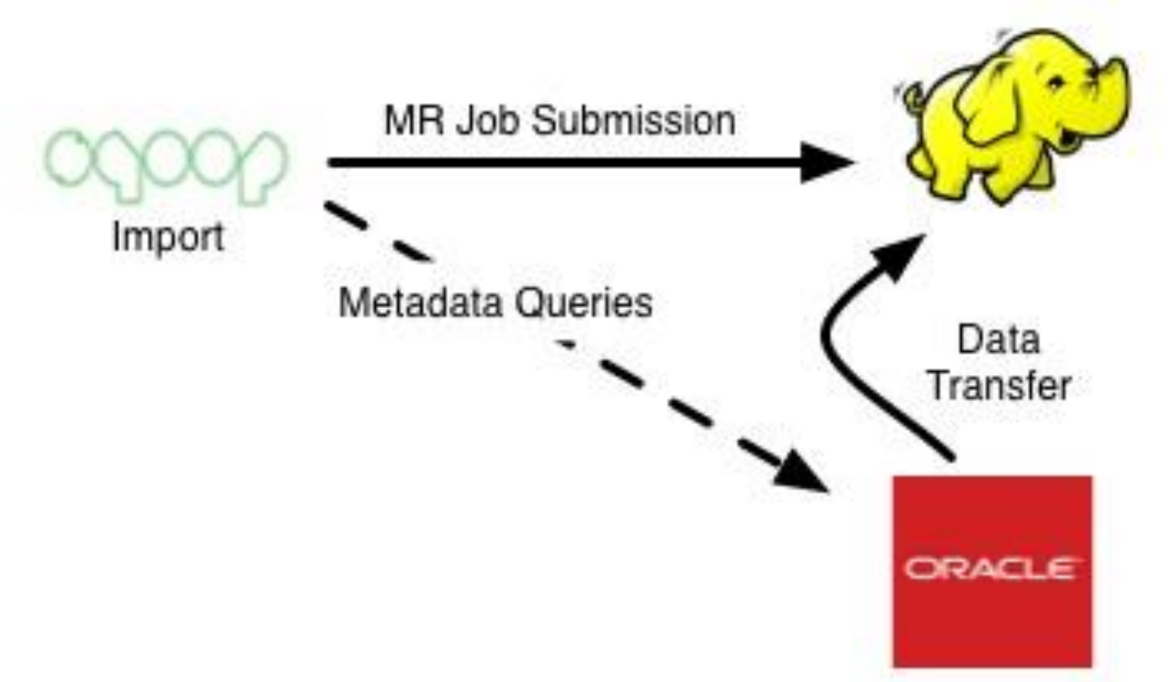
Hadoop-Sqoop?

- Hadoop is great for storing massive data in terms of volume using HDFS
- It Provides a scalable processing environment for structured and unstructured data
- But it's Batch-Oriented and thus not suitable for low latency interactive query operations
- Sqoop is basically an ETL Tool used to copy data between HDFS and SQL databases
    - Import SQL data to HDFS for archival or analysis
    - Export HDFS to SQL (e.g : summarized data used in a warehouse fact table)

What Sqoop Does

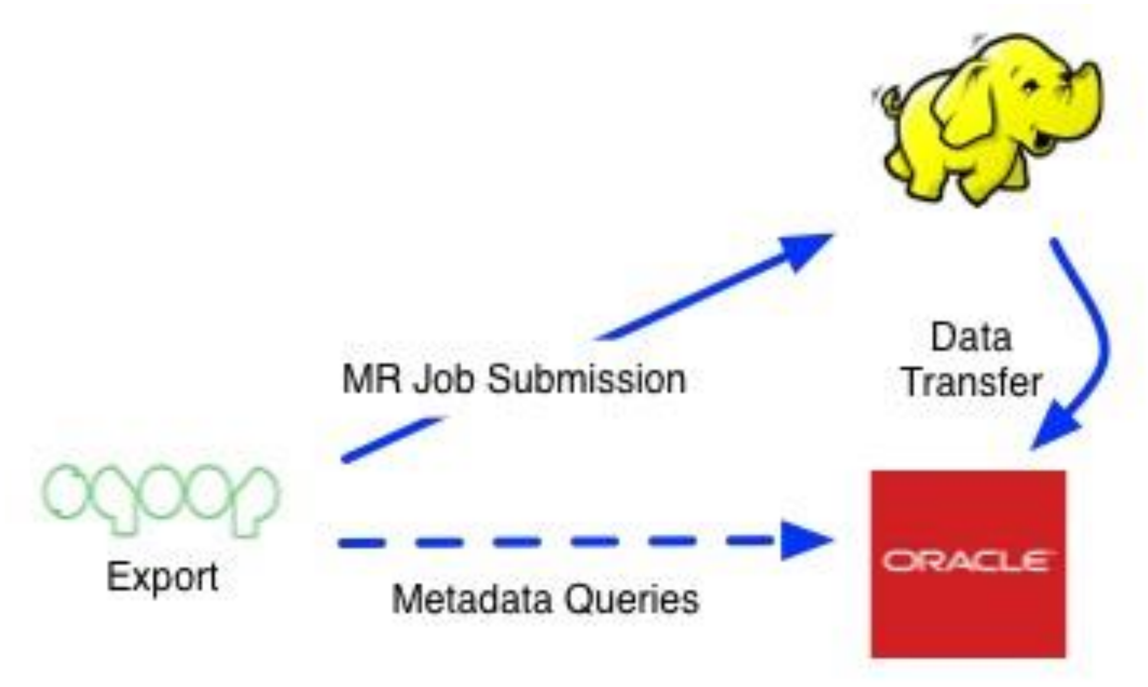
- Designed to efficiently transfer bulk data between Apache Hadoop and structured datastores such as relational databases, Apache Sqoop:
- **Allows data imports** from external datastores and enterprise data warehouses into Hadoop
- **Parallelizes data transfer** for fast performance and optimal system utilization
- **Copies data quickly** from external systems to Hadoop
- **Makes data analysis more efficient**
- **Mitigates excessive loads** to external systems.

Sqoop Import Architecture



```
sqoop import \
  --connect <URL>\
  --username <user> --password <pass> \
  --table <table>
```

Sqoop Export Architecture



```
sqoop export\

   --connect <URL>\

   --username <user> --password <pass> \

   --table <table>

   --export-dir <directory>
```

Examples….

- Copy the appropriate JDBC Driver of the choosen DB to the lib folder of sqoop, sqoop uses JDBC when it connects to an RDBMS and hence the appropriate DriverManager should be available for sqoop.
- We would be using MySQL for our examples, hence copy the mysql jdbc jar to your Sqoop's lib dir.
- Have HDFS+YARN up and running.
- lets say you have a table stocks_mysql in your MySQL, the table has a primary key column TID.
- You want to import the data to HDFS in CSV format.

Import options

- Import using two mappers (parallelism of 2)

```
sqoop import  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --
username lab --password killer --table stocks_mysql -m 2
```

- Import the table, to a specific dir in HDFS

```
sqoop import  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --
username lab --password killer --table stocks_mysql --target-dir
"/stocksnew"
```

- Import the table, to a specific dir in HDFS, delete the target dir if it already exists

```
sqoop import  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --
username lab --password killer --table stocks_mysql --target-dir
"/stocksnew" --delete-target-dir
```

Import options

- Import with Projection

```
sqoop import  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --username lab --
password killer --target-dir "/stocksnew" --query "select
TID,STOCK_NAME,TRADED_DATE from stocks_mysql WHERE \$CONDITIONS "   --delete-
target-dir -m 1
```

- Import the table with a query filter, to a specific dir in HDFS, but in multiple mappers.

```
sqoop import  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --username lab --
password killer --target-dir "/stocksnew" --query "select
TID,STOCK_NAME,TRADED_DATE from stocks_mysql WHERE \$CONDITIONS AND
STOCK_NAME='MSFT' "   --delete-target-dir -m 2 --split-by TID
```

- Do an incremental import from those rows which carry TID value > 800. Note the set of files created after the import

```
sqoop import  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --username lab --
password killer --target-dir "/stocksnew" --query "select TID,STOCK_NAME,TRADED_DATE from
stocks_mysql WHERE \$CONDITIONS AND STOCK_NAME='MSFT' "   --delete-target-dir --
incremental append --check-column TID --last-value 800 -m 1
```

Import Jobs

- An easy way to run imports multiple times, save an import specification with a name and run it when required.

```
 sqoop job --create J1 -- import --connect
jdbc:mysql://ixatlabmaster:3306/hdptests --username lab --password killer --
target-dir "/stocksnew" --query "select TID,STOCK_NAME,TRADED_DATE from
stocks_mysql WHERE \$CONDITIONS AND STOCK_NAME='MSFT' " --incremental append
--check-column TID --last-value 800 -m 1
```

- Run the job

```
sqoop job --exec J1
```

- Get all the jobs that were created

```
sqoop job --list
```

- Get specifics of a Job

```
sqoop job --show J1
```

Sqoop Export

- Lets say you have some data in HDFS and you have run the analytics, you may want to get this data to an RDBMS which is used by a Viz tool for interpretation, in such cases you do an export.
- Run the following Pig script on the stocks data.

```
S1 = Load '/stocksdata/part-m*' USING PigStorage(',') AS  (SYMBOL:chararray, TID:int,
DATE:chararray, HI:double, LO:double, OPEN:double, CLOSE:double, TVTRADED:long) ;

S2 = FOREACH S1 Generate SYMBOL, DATE, CLOSE, $7;

S3 = GROUP S2 BY SYMBOL;

S4 = FOREACH S3 GENERATE group, SUM( S2.TVTRADED) AS SUMOFTVTRADED;

STORE S4 into '/stocksanalyzed/out1' USING PigStorage(',');
```

- The objective now is to get the analyzed data from /stocksanalyzed/out1 to MySql DB.

Sqoop Export

- Sqoop Export requires the table to be created explicit in the target DB. For a MySQL DB create a table similar to structure shown below–

```
USE hdptests;

CREATE TABLE stocks_trade (

        id int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,

        stock_name VARCHAR(20),

        trade_volume BIGINT);
```

- Do the export

```
sqoop export  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --username lab --
password killer --table stocks_trade  --columns stock_name,trade_volume --export-dir
/stocksanalyzed/out1 -m 1
```

Other options

- Eval – to quick check if a given query hold good on the source DB

```
sqoop eval  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --username lab --password killer --query "select TID,STOCK_NAME,TRADED_DATE from stocks_mysql WHERE STOCK_NAME='MSFT' limit 2"
```

- To examine the generated code for debugging needs

```
sqoop codegen  --connect jdbc:mysql://ixatlabmaster:3306/hdptests --username lab --password killer --table stocks_mysql
```

Alternatives

- Apache Manifold
- Apache NiFi
- HIHO
- Cloudera Morphline
- Custom Code