



Hadoop File Formats and Data Ingestion

Prasanth Kothuri, CERN

Files Formats – not just CSV

- Key factor in Big Data processing and query performance
- Schema Evolution
- Compression and Splittability
- Data Processing
 - Write performance
 - Partial read
 - Full read

Available File Formats

- Text / CSV
- JSON
- SequenceFile
 - binary key/value pair format
- RC File
 - row columnar format
- Avro
- Parquet
- ORC
 - optimized row columnar format

AVRO

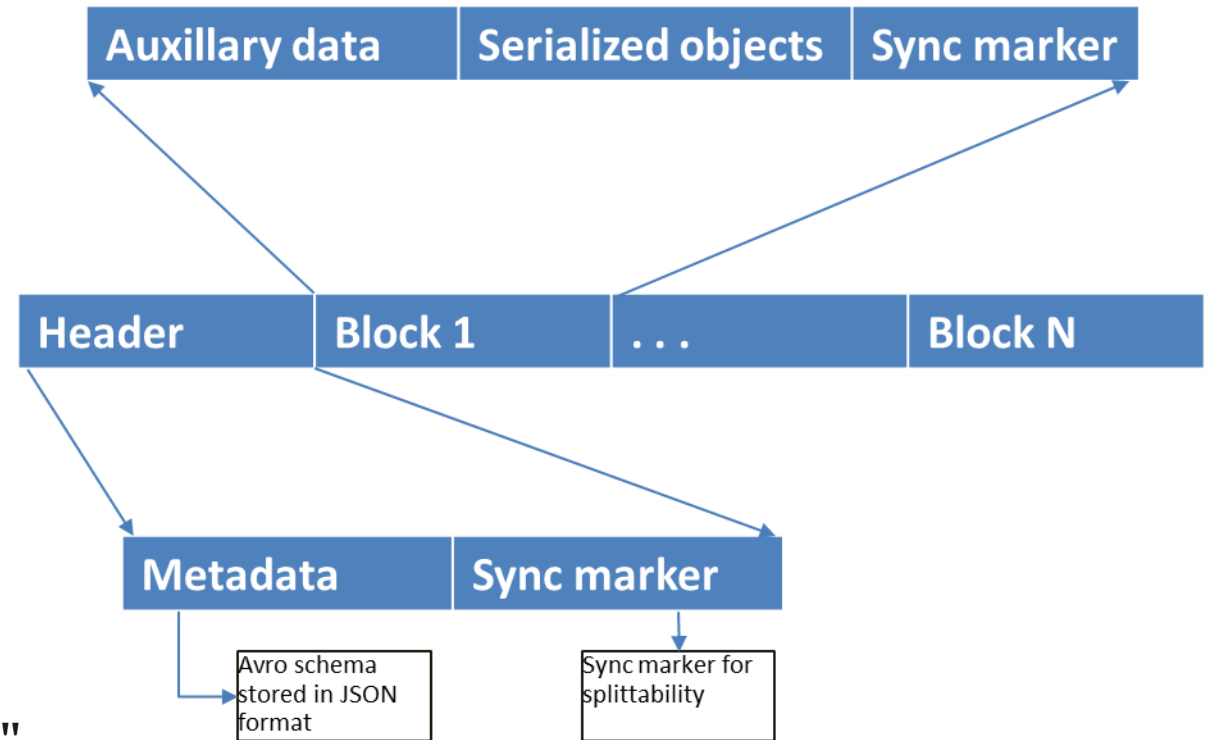
- Language neutral data serialization system
 - Write a file in python and read it in C
- AVRO data is described using language independent schema
- AVRO schemas are usually written in JSON and data is encoded in binary format
- Supports schema evolution
 - producers and consumers at different versions of schema
- Supports compression and are splittable

Avro – File structure and example

Sample AVRO schema in JSON format

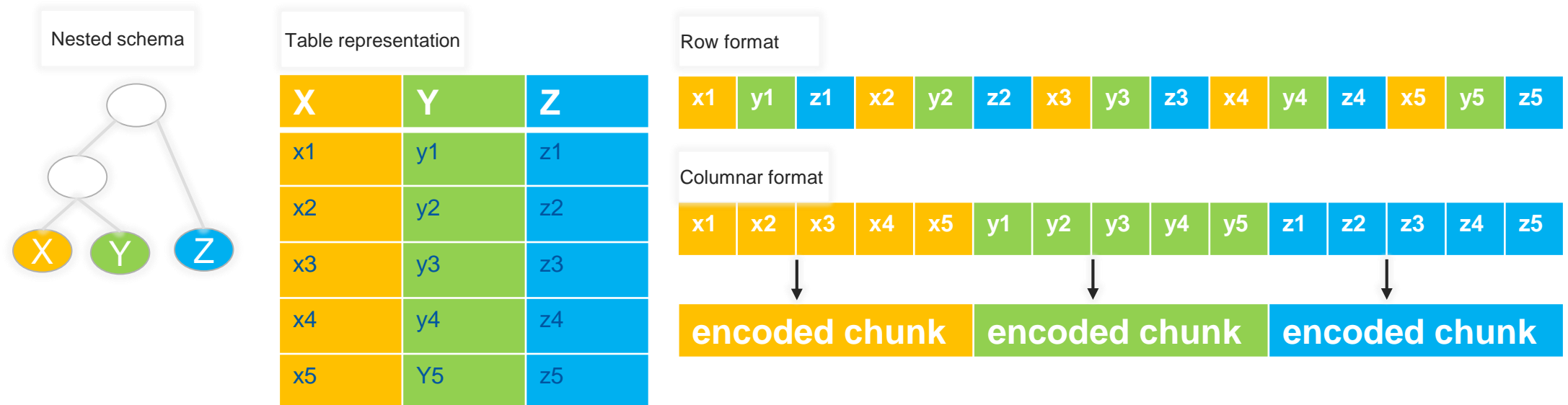
```
{
  "type" : "record",
  "name" : "tweets",
  "fields" : [ {
    "name" : "username",
    "type" : "string",
  }, {
    "name" : "tweet",
    "type" : "string",
  }, {
    "name" : "timestamp",
    "type" : "long",
  } ],
  "doc:" : "schema for storing tweets"
}
```

Avro file structure



Parquet

- columnar storage format
- key strength is to store nested data in truly columnar format using definition and repetition levels¹



(1) Dremel made simple with parquet - <https://blog.twitter.com/2013/dremel-made-simple-with-parquet>

Optimizations – CPU and I/O

Statistics for filtering and query optimization

projection push down

X	Y	Z
x1	y1	z1
x2	y2	z2
x3	y3	z3
x4	y4	z4
x5	y5	z5

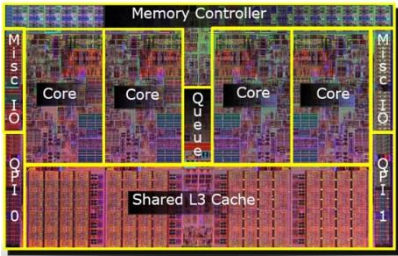
predicate push down

X	Y	Z
x1	y1	z1
x2	y2	z2
x3	y3	z3
x4	y4	z4
x5	y5	z5

read only the data you need

X	Y	Z
x1	y1	z1
x2	y2	z2
x3	y3	z3
x4	y4	z4
x5	y5	z5

Minimizes CPU cache misses



SLOW



cache misses costs cpu cycles

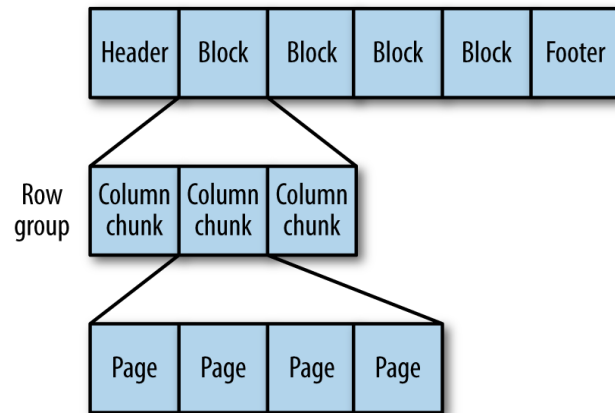


Encoding

- **Delta Encoding:**
 - E.g timestamp can be encoded by storing first value and the delta between subsequent values which tend to be small due to temporal validity
- **Prefix Encoding:**
 - delta encoding for strings
- **Dictionary Encoding:**
 - Small set of values, e.g post code, ip addresses etc
- **Run Length Encoding:**
 - repeating data

Parquet file structure & Configuration

Internal structure of parquet file

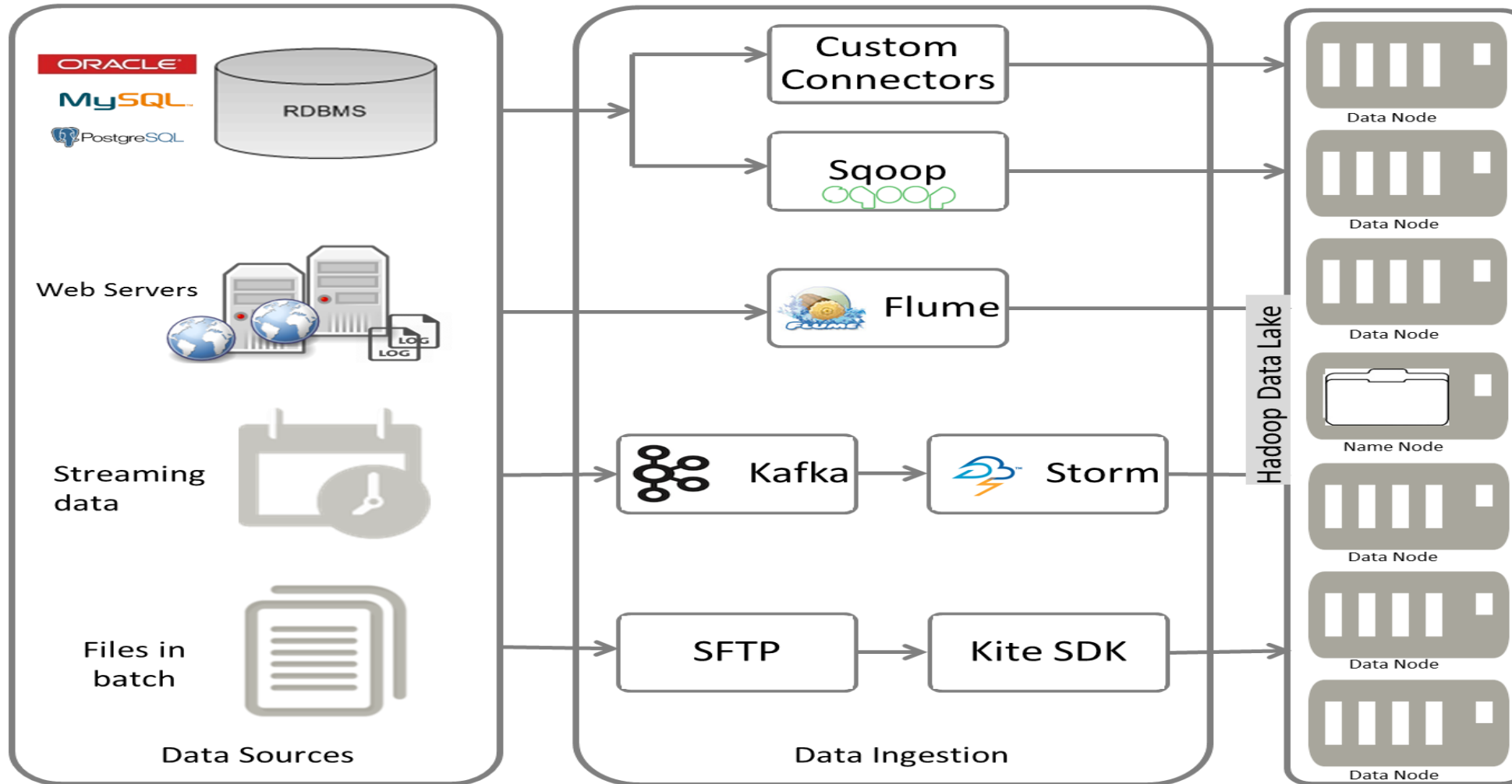


Configurable parquet parameters

Property name	Default value	Description
<code>parquet.block.size</code>	128 MB	The size in bytes of a block (row group).
<code>parquet.page.size</code>	1MB	The size in bytes of a page.
<code>parquet.dictionary.page.size</code>	1MB	The maximum allowed size in bytes of a dictionary before falling back to plain encoding for a page.
<code>parquet.enable.dictionary</code>	true	Whether to use dictionary encoding.
<code>parquet.compression</code>	UNCOMPRESSED	The type of compression: UNCOMPRESSED, SNAPPY, GZIP & LZO

In summation, Parquet is state-of-the-art, open-source columnar format the supports *most* of Hadoop processing frameworks and is optimized for high compression and high scan efficiency

Data Ingestion

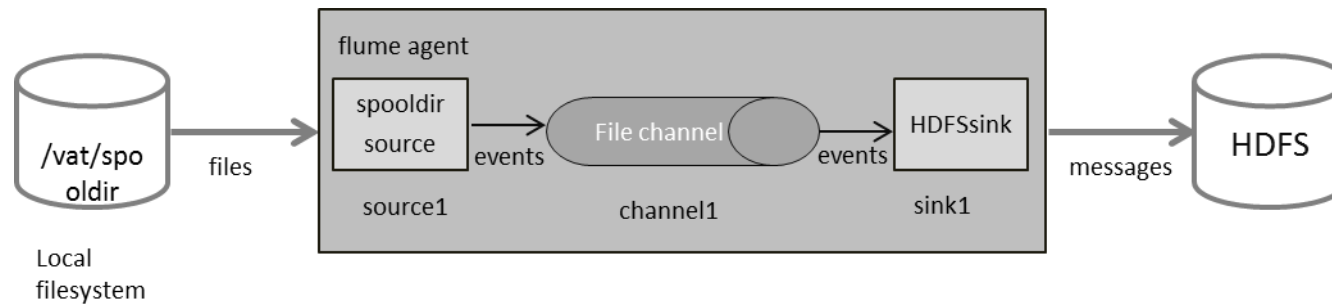


Flume

Flume is for high-volume ingestion into Hadoop of event-based data

e.g collect logfiles from a bank of web servers, then move log events from those files to HDFS (clickstream)

Example



Flume configuration

```
agent1.sources = source1
agent1.sinks = sink1
agent1.channels = channel1
```

```
agent1.sources.source1.channels = channel1
agent1.sinks.sink1.channel = channel1
```

```
agent1.sources.source1.type = spooldir
agent1.sources.source1.spoolDir = /var/spooldir
```

```
agent1.sinks.sink1.type = hdfs
agent1.sinks.sink1.hdfs.path = hdfs://hostname:8020/user/flume/logs
agent1.sinks.sink1.hdfs.filetype = DataStream
```

```
agent1.channels.channel1.type = memory
agent1.channels.channel1.capacity = 10000
agent1.channels.channel1.transactionCapacity = 100
```

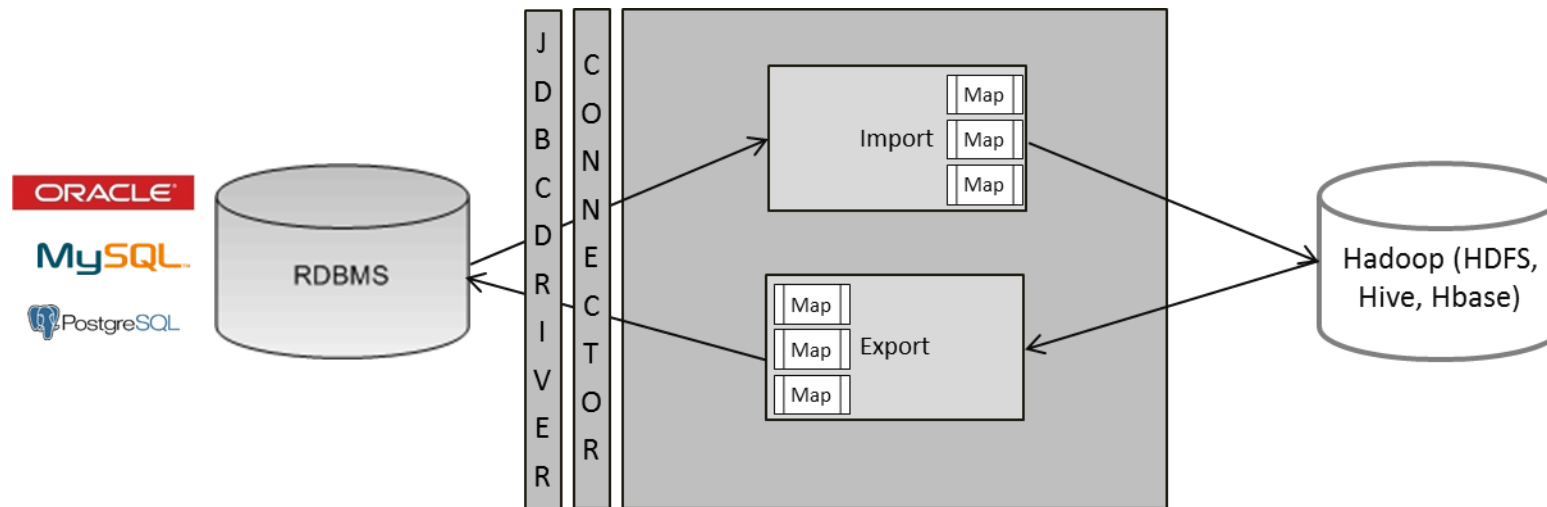
```
flume-ng agent -name agent1 -conf $FLUME_HOME/conf
```

One usecase can be copying of DB audit logs to HDFS

Category	Component
Source	Avro
	Exec
	HTTP
	JMS
	Netcat
	Sequence generator
	Spooling directory
	Syslog
	Thrift
	Twitter
	Sink
Elasticsearch	
File roll	
HBase	
HDFS	
IRC	
Logger	
Morphline (Solr)	
Null	
Thrift	
Channel	
	JDBC
	Memory

Sqoop – SQL to Hadoop

- Open source tool to extract data from structured data store into Hadoop
- Architecture



Sqoop – contd.

- Sqoop schedules map reduce jobs to effect imports and exports
- Sqoop always requires the connector and JDBC driver
- Sqoop requires JDBC drivers for specific database server, these should be copied to /usr/lib/sqoop/lib
- The command-line structure has the following structure

```
Sqoop TOOL PROPERTY_ARGS SQOOP_ARGS
```

TOOL - indicates the operation that you want to perform, e.g import, export etc

PROPERTY_ARGS - are a set of parameters that are entered as Java properties in the format -Dname=value.

SQOOP_ARGS - all the various sqoop parameters.

Sqoop – How to run sqoop

Example:

```
sqoop import \  
--connect jdbc:oracle:thin:@devdb11-s.cern.ch:10121/devdb11_s.cern.ch \  
--username pdbmon \  
-P \  
--num-mappers 1 \  
--target-dir visitcount_rfidlog \  
--table VISITCOUNT.RFIDLOG
```


Sqoop – how to parallelize

```
-- table table_name
```

```
-- query select * from table_name where $CONDITIONS
```

```
-- table table_name
```

```
-- split-by primary key
```

```
-- num-mappers n
```

```
-- table table_name
```

```
-- split-by primary key
```

```
-- boundary-query select range from dual
```

```
-- num-mappers n
```

Hands On – 1

Use Kite SDK to demonstrate copying of various file formats to Hadoop

Step 1) Download the MovieLens Dataset

```
curl http://files.grouplens.org/datasets/movielens/ml-latest-small.zip -o  
movies.zip  
unzip movies.zip  
cd ml-latest-small/
```

Step 2) Load the Dataset into Hadoop in Avro format

-- infer the schema

```
kite-dataset csv-schema ratings.csv --record-name ratings -o ratings.avsc  
cat ratings.avsc
```

-- create the schema

```
kite-dataset create ratings --schema ratings.avsc
```

-- load the data

```
kite-dataset csv-import ratings.csv --delimiter ',' ratings
```

Hands On – 1 contd

Step 3) Load the Dataset into Hadoop in Parquet format

```
kite-dataset csv-schema ratings.csv --record-name ratingsp -o ratingsp.avsc
cat ratingsp.avsc
kite-dataset create ratingsp --schema ratingsp.avsc --format parquet
kite-dataset csv-import ratings.csv --delimiter ',' ratingsp
```

Step 4) Run a sample query to compare the elapsed time between Avro & Parquet

```
hive
select sum(rating)/count(1) from ratings;
select sum(rating)/count(1) from ratingsp;
```

Hands On – 2

Use Sqoop to copy an Oracle table to Hadoop

```
sudo su -  
cd /var/lib/sqoop  
curl -L https://www.dropbox.com/s/z726v1mzbysvq98/ojdbc6.jar?dl=0 -o ojdbc6.jar
```

```
sqoop import \  
--connect jdbc:oracle:thin:@devdb11-s.cern.ch:10121/devdb11_s.cern.ch \  
--username pdbmon \  
-P \  
--num-mappers 1 \  
--target-dir visitcount_rfidlog \  
--table VISITCOUNT.RFIDLOG
```

Check the size and number of files

```
hdfs dfs -ls visitcount_rfidlog/
```

Hands On – 3

Use Sqoop to copy an Oracle table to Hadoop, multiple mapper

```
sqoop import \  
--connect jdbc:oracle:thin:@devdb11-s.cern.ch:10121/devdb11_s.cern.ch \  
--username pdbmon \  
-P \  
--num-mappers 2 \  
--split-by alarm_id \  
--target-dir lemontest_alarms \  
--table LEMONTEST.ALARMS \  
--as-parquetfile
```

Check the size and number of files

```
hdfs dfs -ls lemontest_alarms/
```

Hands On – 4

Use Sqoop to make incremental copy of a Oracle table to Hadoop

```
sqoop job \  
--create alarms \  
-- \  
import \  
--connect jdbc:oracle:thin:@devdb11-s.cern.ch:10121/devdb11_s.cern.ch \  
--username pdbmon \  
-P \  
--num-mappers 1 \  
--target-dir lemontest_alarms_i \  
--table LEMONTEST.ALARMS \  
--incremental append \  
--check-column alarm_id \  
--last-value 0 \  
  
sqoop job --exec alarms
```

Hands On – 4 contd.

```
sqoop import \  
--connect jdbc:oracle:thin:@devdb11-s.cern.ch:10121/devdb11_s.cern.ch \  
--username pdbmon \  
-P \  
--num-mappers 1 \  
--table LEMONTEST.ALARMS \  
--target-dir lemontest_alarms_i \  
--incremental append \  
--check-column alarm_id \  
--last-value 47179 \  
  
hdfs dfs -ls lemontest_alarms_i/
```

Q & A

E-mail: Prasanth.Kothuri@cern.ch

Blog: <http://prasanthkothuri.wordpress.com>