Impala for the busy analyst

Goal

- Provide some background and coding techniques that will lead us to high performance and scalability.

Impala SQL Language

- **It's just SQL!** (SQL-92 Standard).
- Supported features:
  - Joins
  - Views
  - Aggregation (`ORDER BY`, `GROUP BY`)
  - Casts, aliases, `WITH` clause
  - Analytic functions: `OVER`

Some differences: Handling dates

- No native support for `date` time in many versions.
  - `2020-12-01` (YYYY-MM-DD) can be read natively in Impala
  - `select cast('2020-12-01' as timestamp);` OK
  - `select cast('2020/12/01' as timestamp);` **Not** OK

Some differences: Handling dates (cont.)

- **Workaround**
  - `select from_unixtime(unix_timestamp('2020/12/01', 'yyyy/MM/dd'));`
- Support to parse dates/system date with special functions:
  - `select extract('2020-12-01', 'month');`
  - `select extract(now(), 'minute');`
- Operations with time intervals:
  - `select now() + interval 2 weeks;`
  - `select now() + interval 2 weeks - interval 6 hours;`

Supported types

- `STRING` : all things text.
- `INT` / `TINYINT` / `BIGINT` : integer types.
- `FLOAT` / `DOUBLE` : real numbers.
- `BOOLEAN` : true/false values.
- `TIMESTAMP` : time and date.
- More recent versions: `DECIMAL` , `VARCHAR` , `CHAR` .

Limited DML

- There is no `DELETE` or `UPDATE` (maybe in Kudu?).
- No indexes, constraints, foreign keys.
- To replace data, the only posibility is `INSERT OVERWRITE`.
- Instead of using `DELETE`, you can use `DROP TABLE`, `ALTER TABLE`, `DROP PARTITION` statements.

Big Data Considerations

Similarities to data warehouse

- Optimized for fast bulk read and data load operations.
  - Good for data warehouse-type queries: "max visitors/sales"
  - No overhead from creating and mantaining indices.
  - Less need for normalization.
- **Partitioning:** physically divide the data in one or more criteria.
  - Reduce disk I/O and improve scalability of queries.

Storage

- HDFS
  - Default blocksize of 128 MB.
  - Usually 3 copies per each block across the cluster.
- Parquet
  - 256 MB default blocksize, which helps handle big data.
  - Compromise between blocksize and clustersize:
    - Too many small files vs too few large files.
  - **Columnar format:** each query only reads the columns that are referenced there.

File formats

- You can store Impala data files in different formats.
- Each file format supported is open-source and documented.
- Two opposites:
  - **Text:** Convenient and flexible.
  - **Parquet:** Compact and query-optimized.

Text

- Bulkiest format, less efficient for Big Data.
- 1234567 takes up 7 bytes on disk; 1234567 takes up 8 bytes; 1234.567 takes up 9 bytes.
- You can specify how the table will be stored (defaults to text) on the `CREATE TABLE` statement:

```
CREATE TABLE csv (c1 STRING, c2 STRING, c3 STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ","
STORED AS TEXTFILE;
```

Parquet

- Binary file format.
- Numeric values are packed in either 4 or 8 bytes.
- Boolean values are 1 bit, instead of a true/false text.
- Automatic dictionary encoding (up to 16k values) for columns with repeated values.

Parquet (cont.)

- Each Parquet data file has all the columns for a subset of rows in the table, but values in the same column are adjacent in disk.
- This speeds up queries that need to examine all the values of the same columns:
  - Sums, averages, etc across time periods or geographic zones.
- `SELECT *` is way too costly.

File format information

- `SHOW TABLE STATS my_table;`
- `DESCRIBE FORMATTED my_table;`

Switching file format

- You can change file format at any time:

```
CREATE TABLE t2 LIKE t1;
-- Copy the data, preserving the original file format.
INSERT INTO t2 SELECT * FROM t1;
ALTER TABLE t2 SET FILEFORMAT = PARQUET;
-- Now reload the data, this time converting to Parquet.
INSERT OVERWRITE t2 SELECT * FROM t1;
```

Switching file format (cont.)

- Or mix file formats for the same table:

```
CREATE TABLE t3 (c1 INT, c2 STRING, c3 TIMESTAMP)
PARTITIONED BY (state STRING, city STRING);
ALTER TABLE t3 ADD PARTITION
(state = 'CA', city = 'San Francisco');
-- Load some text data into this partition...
ALTER TABLE t3 ADD PARTITION
(state = 'CA', city = 'Berkeley');
ALTER TABLE t3 PARTITION
(state = 'CA', city = 'Berkeley')
SET FILEFORMAT = PARQUET;
-- Load some Parquet data into this partition...
```

Exercise

- Create a `dummy` table in a new database, `tutorial`, using the provided `dummy.csv` file.

```
CREATE TABLE dummy
(id bigint, val int, zerofill string, name string,
assertion boolean, city string, state string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ","
LOCATION '/user/username/dummy/'
;
```

By default, this table will be stored as textfile.

Exercise (cont.)

- Now recreate this table in Parquet format:

```
CREATE TABLE dummy_parquet STORED AS PARQUET
AS SELECT * FROM dummy;
```

- Compare different queries in these two tables:
  - `SHOW TABLE STATS my_table;`
  - `SELECT COUNT(*) FROM my_table;`
  - `SELECT MAX(name) FROM my_table;`
  - `SELECT AVG(val), MIN(name), MAX(name) FROM my_table;`

Use the `impala-shell` to see information on the time taken by the query.