Sqoop

Niko Mäkitalo

Outline

- Introduction
- List of SubTools
- An Example
- Importing from Relational Databases
- Exporting from HDFS
- Conclusions

Introduction

- Tool for extracting data from relational databases to Hadoop for further processing
- Can export data from Hadoop to relational databases
- Uses MapReduce
- Open-source
- Written in Java and uses JDBC

List of Sqoop's SubTools

- **sqoop-import**
- sqoop-import-all-tables
- **sqoop-export**
- sqoop-job
- sqoop-metastore
- sqoop-merge
- **sqoop-codegen**
- **sqoop-create-hive-table**
- sqoop-eval
- sqoop-list-databases
- sqoop-list-tables
- sqoop-help
- sqoop-version

An example

- MySQL table with some entries:

```
+----+-------------+-------+-------------+---------+----------------------+
| id | widget_name | price | design_date | version | design_comment       |
+----+-------------+-------+-------------+---------+----------------------+
|  1 | sprocket    |  0.25 | 2010-02-10  |       1 | Connects two gizmos  |
|  2 | gizmo       |  4.00 | 2009-11-30  |       4 | NULL                 |
|  3 | gadget      | 99.99 | 1983-08-13  |      13 | Our flagship product |
+----+-------------+-------+-------------+---------+----------------------+
```

- Sqoop import command:

```
sqoop import --connect jdbc:mysql://localhost/hadoopguide --table widgets -m 1
```

- Results:

```
hadoop fs -cat /user/root/widgets/part-m-00000
1,sprocket,0.25,2010-02-10,1,Connects two gizmos
2,gizmo,4.00,2009-11-30,4,null
3,gadget,99.99,1983-08-13,13,Our flagship product
```

Code Generation

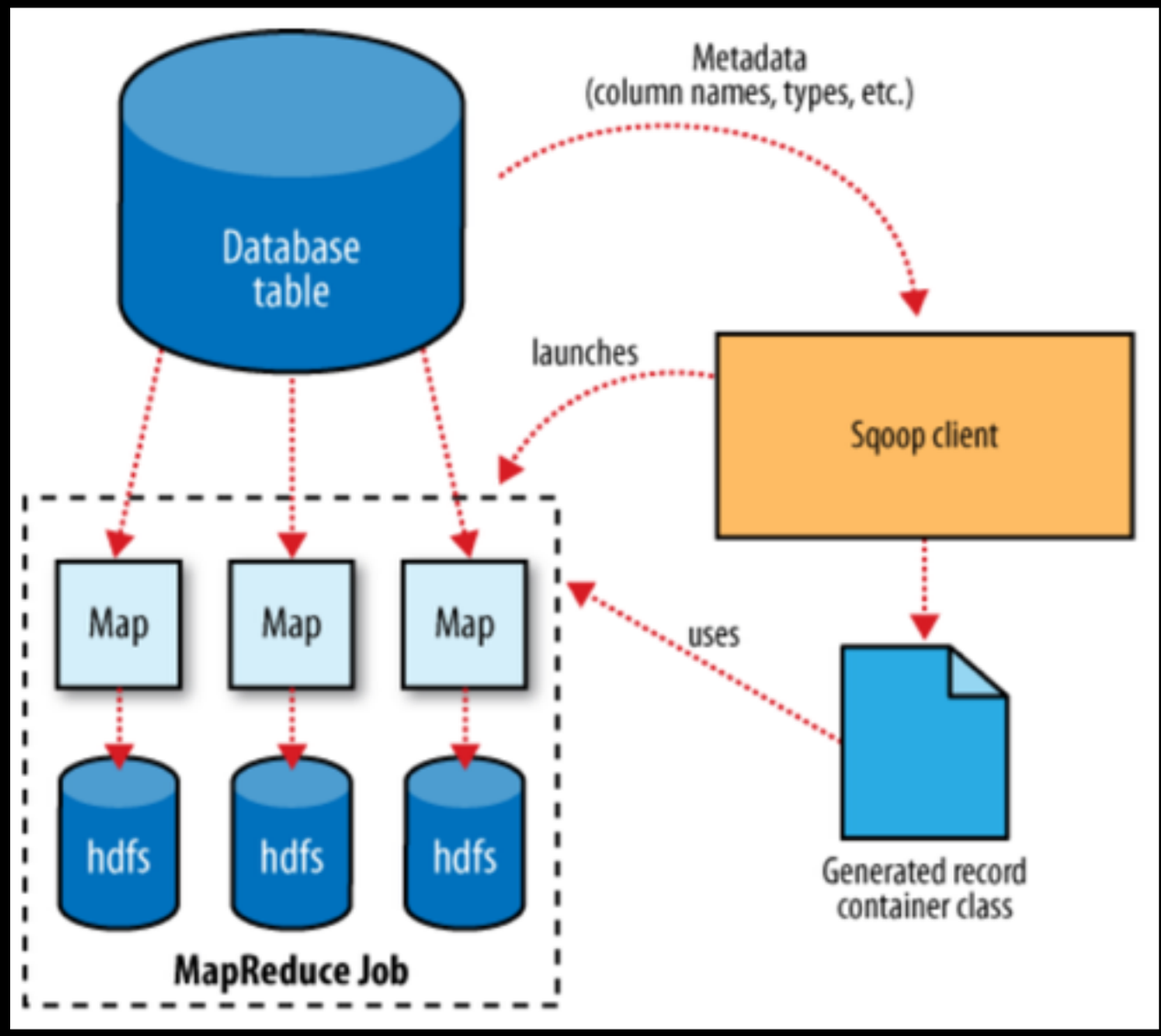
sqoop-codegen

- Uses JDBC to examine the table it is to import: Retrieves a list of all the columns and their SQL data types
- Then maps the database data types to Java data types, like VARCHAR -> String
- Sqoop's code generator creates a class based on the retrieved information to hold a single record from the exerted table

```
public Integer get_id();
public String get_widget_name();
public java.math.BigDecimal get_price();
public java.sql.Date get_design_date();
public Integer get_version();
public String get_design_comment();
```

MapReduce

- Sqoop imports a table from a database by running a MapReduce job that extracts rows from the table, and writes the records to HDFS
- Default amount of nodes is 4
- By default, uses the primary key for dividing the query across multiple nodes



Example 2: Larger data set

- 300 000 lines of information about cars
  - First try -> FAILED

```
15/03/03 23:15:23 INFO mapreduce.Job: Task Id : attempt_1425281013897_0013_m_000000_0, Status : FAILED
Error: GC overhead limit exceeded
```

  - Second try:

```
sqoop import --connect jdbc:mysql://localhost/hadoopguide?dontTrackOpenResources=true
\&defaultFetchSize=1000\&useCursorFetch=true --table fords -m 1
15/03/04 08:22:01 INFO mapreduce.ImportJobBase: Transferred 44.1487 MB in 35.5109 seconds (1.2432 MB/sec)
15/03/04 08:22:01 INFO mapreduce.ImportJobBase: Retrieved 293927 records.
```

  - Third, fourth, and fifth try:

```
2 Nodes:
15/03/04 08:19:55 INFO mapreduce.ImportJobBase: Transferred 44.1487 MB in 44.4097 seconds (1,017.9815 KB/sec)
15/03/04 08:19:55 INFO mapreduce.ImportJobBase: Retrieved 293927 records.

4 Nodes:
15/03/04 07:42:41 INFO mapreduce.ImportJobBase: Transferred 44.1487 MB in 79.3971 seconds (569.3946 KB/sec)
15/03/04 07:42:41 INFO mapreduce.ImportJobBase: Retrieved 293927 records.

8 Nodes:
15/03/04 08:17:31 INFO mapreduce.ImportJobBase: Transferred 44.1487 MB in 104.9651 seconds (430.6982 KB/sec)
15/03/04 08:17:31 INFO mapreduce.ImportJobBase: Retrieved 293927 records.
```

Importing Data to Hive

- To import the data to Hive couple of steps:
- First use create-hive-table command:

```
sqoop create-hive-table --connect jdbc:mysql://localhost/hadoopguide
--table fords --fields-terminated-by ','
```

- Then launch Hive and load the data:

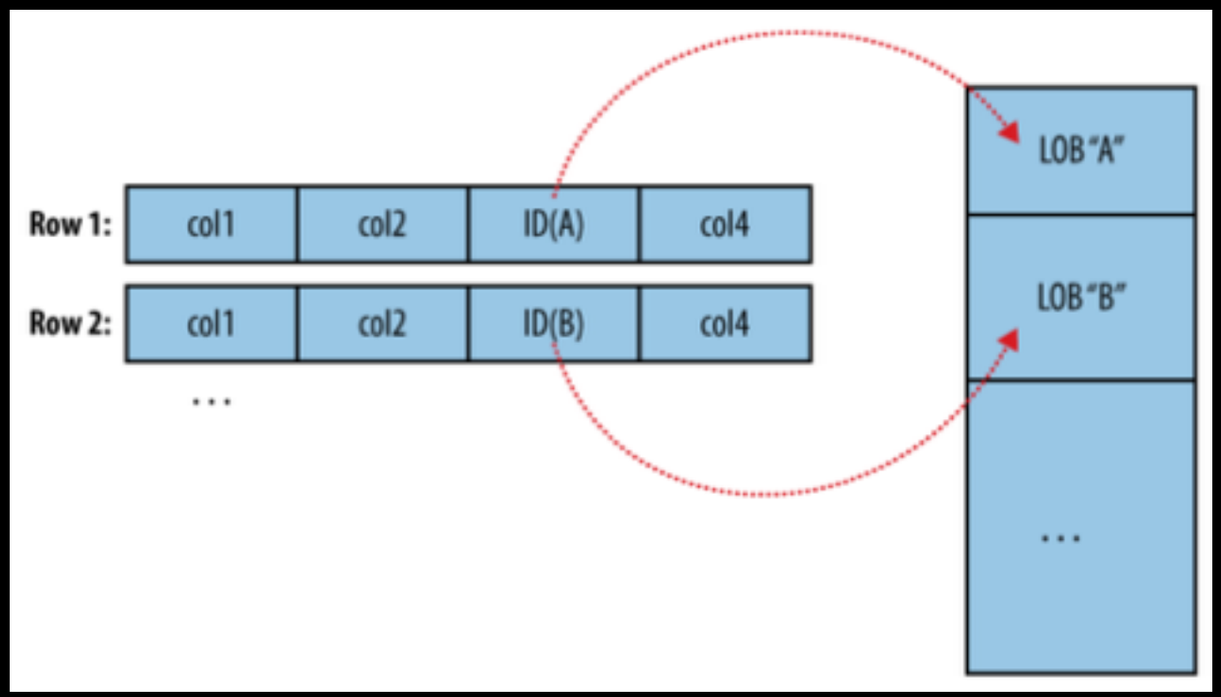
```
hive> LOAD DATA INPATH "fords" INTO TABLE fords;
```

- Make queries:

```
hive> select * from fords where kayttoonottopvm == '20140819';
```

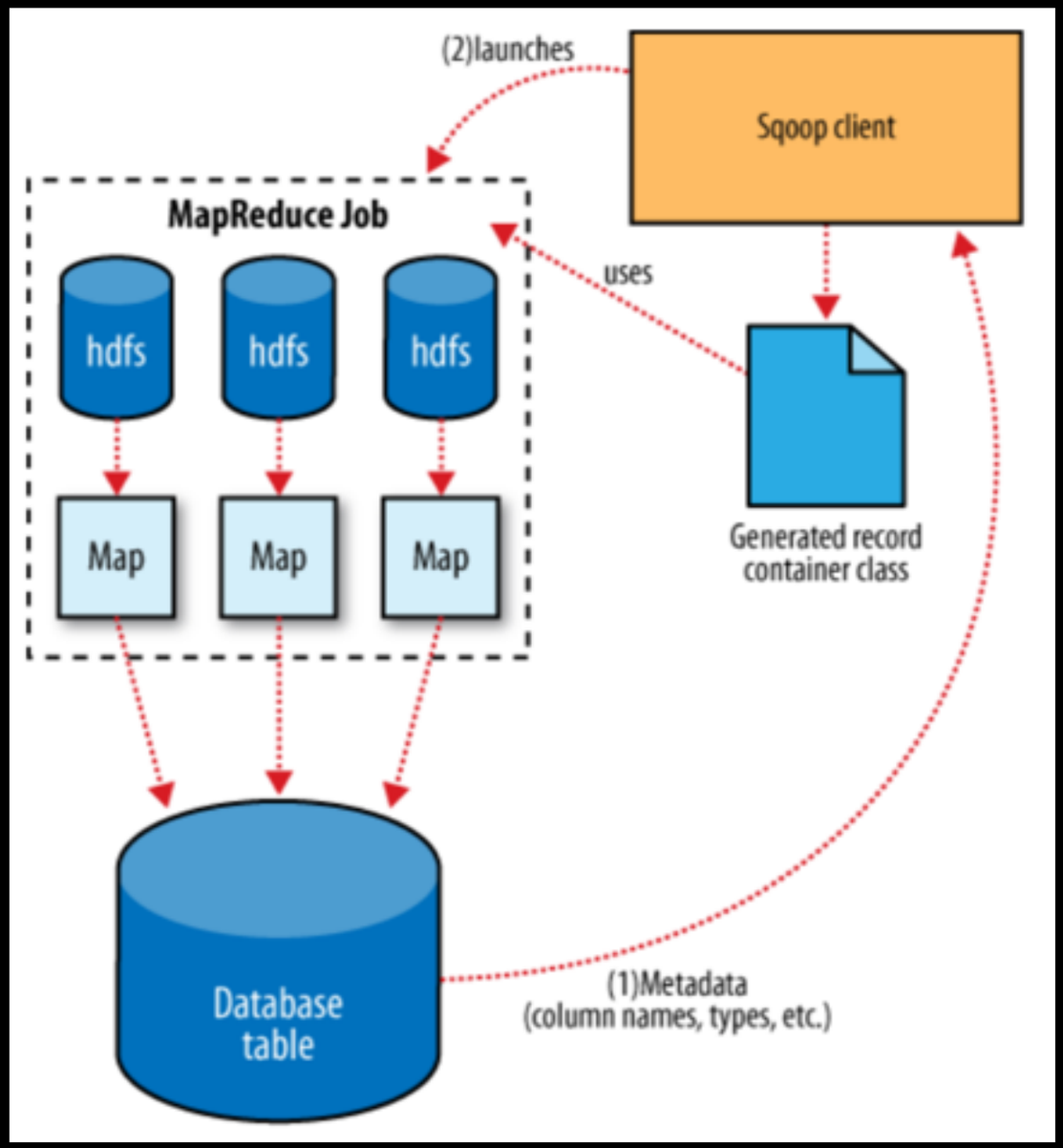
Importing Large Files

- Sqoop supports importing large files as LobFiles
- When a record is imported, the “normal” fields will be materialized together in a text file, along with a reference to the LobFile where a CLOB or BLOB column is stored:
  - 2,gizmo,4.00,2009-11-30,4,null,externalLob(lf,lobfile0,100,5011714)
  - BlobRef bR = Widget.get_schematic()
  - bR.getDataStream() method actually opens the LobFile and re- turns an InputStream allowing you to access the schematic field’s contents



Exporting Data

- Picks a strategy based on the database connect string
- Sqoop generates Java class based on the target table definition
    - Class can parse records from text files and insert values into a DB table
- Launches a MapReduce job that reads the source data files from HDFS, parses the records using the generated class, and executes the chosen export strategy.
- Uses separate threads for reading data from HDFS and for writing data to to DB ensure that I/O operations involving different systems are overlapped as much as possible



An Example

- Exporting Fords:

```
sqoop export --connect jdbc:mysql://localhost/hadoopguide?dontTrackOpenResources=true
\&defaultFetchSize=1000\&useCursorFetch=true --table fords2 -m 1 --export-dir /apps/
hive/warehouse/fords

15/03/04 12:42:17 INFO mapreduce.ExportJobBase: Transferred 44.1489 MB in 109.1143
seconds (414.3225 KB/sec)
15/03/04 12:42:17 INFO mapreduce.ExportJobBase: Exported 293927 records.
```

Conclusions

- Seems to do its job
- Not necessarily very efficient while importing small data sets,
  - But may pays back when importing larger amounts of data?
- Support for many databases (due to JDBC)
- Generated Java classes can be used from Java-based mappers

References

[1] Hadoop The Definitive Guide, 3rd Edition

[2] http://archive.cloudera.com/cdh/3/sqoop/SqoopUserGuide.html#_example_invocations_4