Big Data / ETL Automation Testing & Robot Framework

Robin Li

robinli@live.ca

Agenda

1. Big data brief introduction
2. ETL Automation Test
3. Robot Framework
4. Python Language
5. Q &A

Data --> Big Data

Why: BI / AI

Data is resource, ...

Challenges : Volume, Variety, and Velocity

Quantity: KB(kilo), MB(Mega), GB(Giga), TB(Tera), PB(Peta),EB(Exa),ZB(Zetta),YB(Yotta)

Data Types: structure / no-structure

Quality: QA / Testing, Automation

DB -> DW -> Data Mart -> Data Lake

Big Data Technology

Apache Hadoop：

Storage: HDFS + Computing: MapReduce

Main Platforms: Cloudera, Horonworks, ...

Tech: MPP(Massive Parallel Processing)

Some popular tools:

Data Management: HDFS, YARN

Operations: Zookeeper, Cloudbreak, Oozie

Data Access: Pig, Hive, Storm, Hbase, Spark,

Integration: Falcon, WebHDFS, Sqoop,Kafka

E T L / Data Flow
(Extract, Transform, Load)

1. Data sources (SQL, GFF, CSV, PSV....)

Ingestion Parser/Mapper

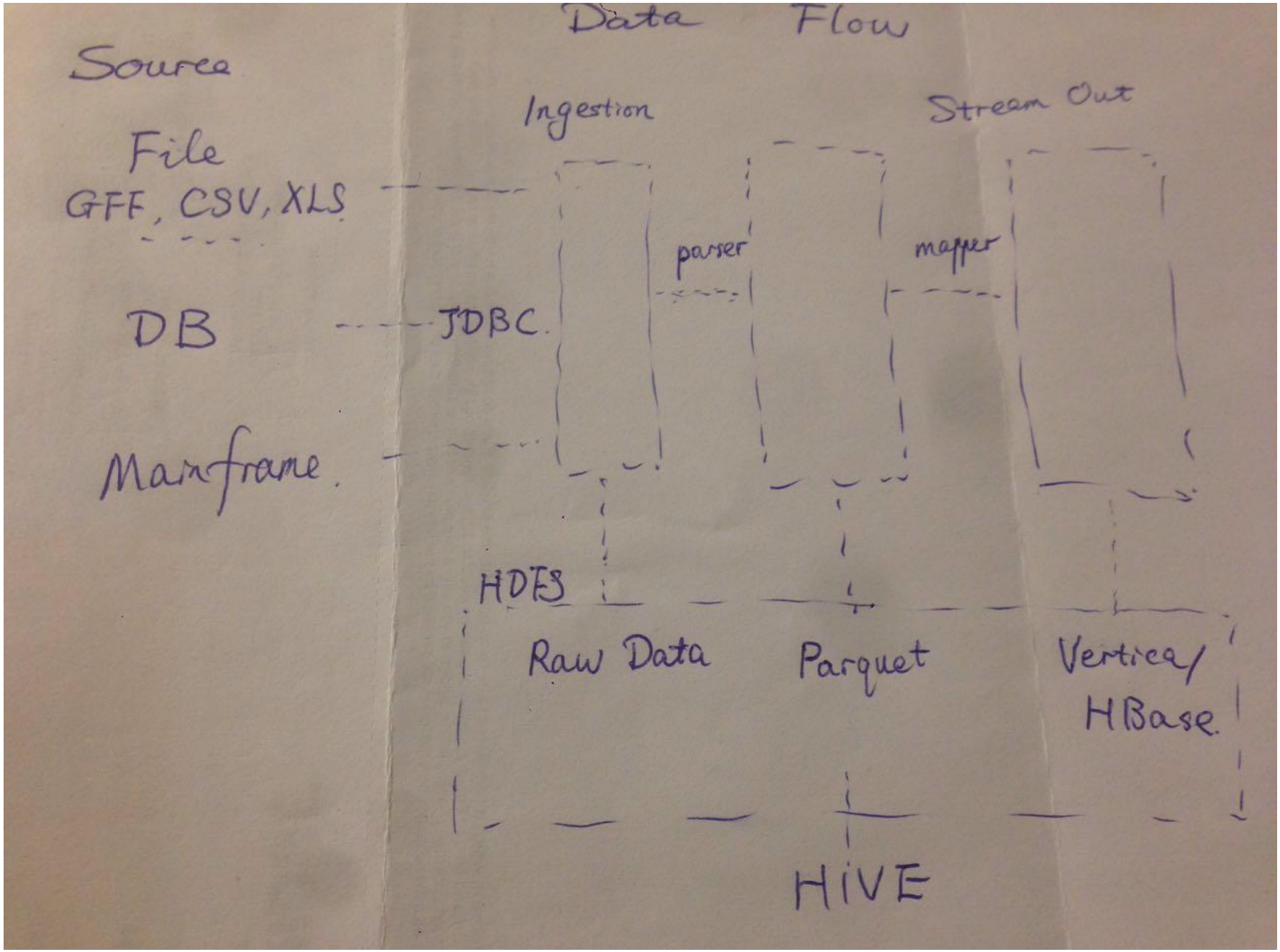
2. Hadoop HDFS

Parquet file -- column-oriented

(Apache HIVE, Cloudera Impala, Pig)

Configuration files

3. Stream out for applications


Data Flow
Source
File
GFF, CSV, XLS
DB
JDBC
Mainframe
Ingestion
parser
mapper
Stream Out
HDFS
Raw Data
Parquet
Vertica/
HBase
HIVE

Data Quality

1. Source quality – Extract Transform Load

   Bulk history: Sampling by order

   Delta data: Fully verification, Timestamp

2. Data Quality (data governance)

Accuracy, Correctness, Integrity, Completeness, Consistency, Currency, ...

(Row Counts, Null rate, MD5,Orphan records, log check.....)

RIDE
(Robot framwork IDE)

1. Python 2.6 above --- Python 2.7.13
2. wxPython 2.8.12.1 with Unicode support
3. Robot framework
4. RIDE

https://github.com/robotframework/RIDE/wiki/How-To#starting-ride

Example: No nested loop

Python Language

Easy, flexible Scripting Languages

Indent: 4 spaces to define block

Convention:

- Constant --UPPER
- camel case: welcomeBigdataEvent
- __function__, _function_, self.

official vs irregular

print-out vs hand-writing

Example :github.com sudoku game

Python common mistakes

1. Silly things: Forgetting to return a value; Misspelling; Mixing up Def and class
2. Styles: Bad naming; PEP-8 violations; Inscrutable lambdas;…
3. Structures: Pathological If/elif blocks; overusing private attributes;…
4. Surprises: Importing everything; Reinventing the wheel;….

From “How to Make Mistakes in Python”

Q & A

THANK YOU !