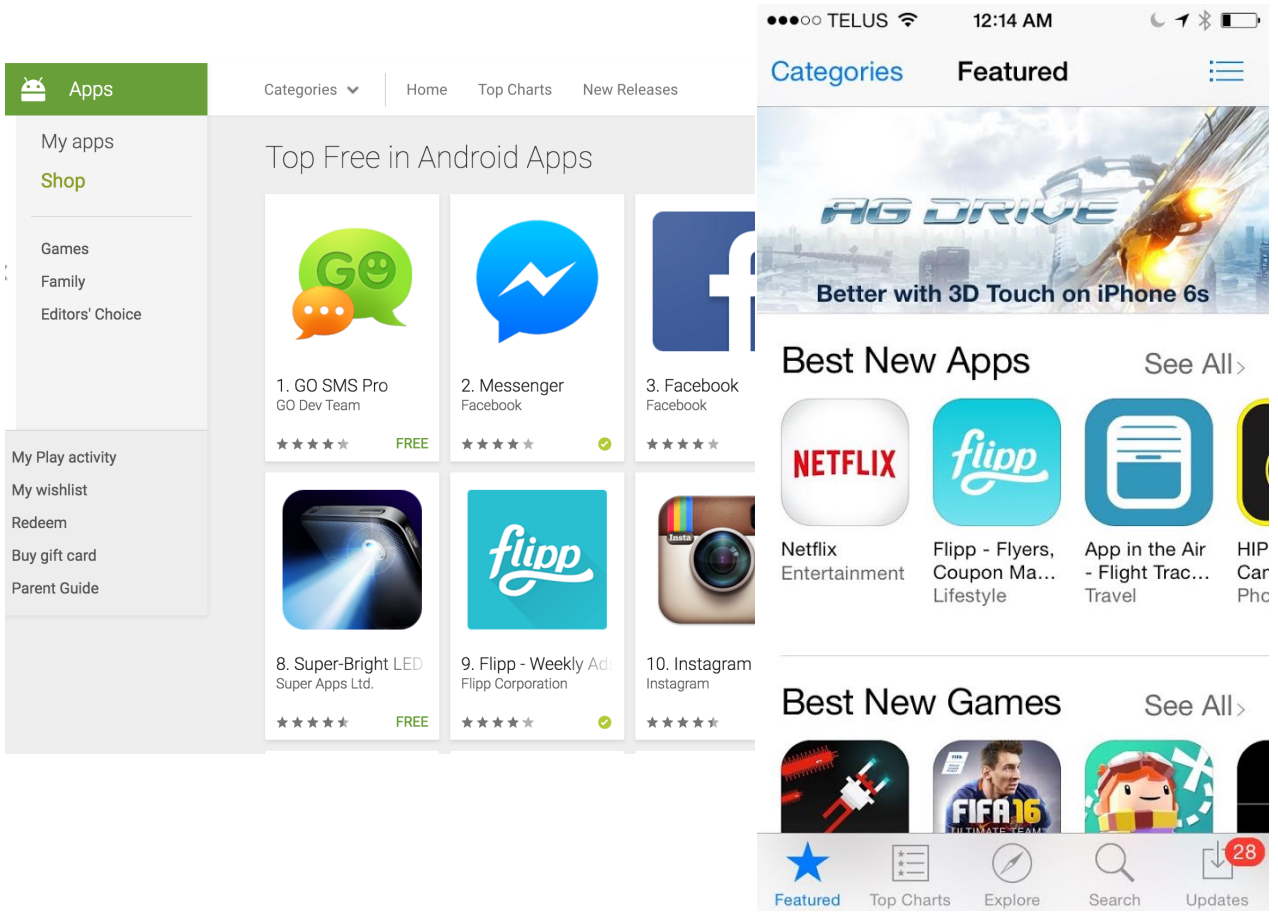
A year of Spark at Flipp

Younes Abouelnagah

November 25th, 2015

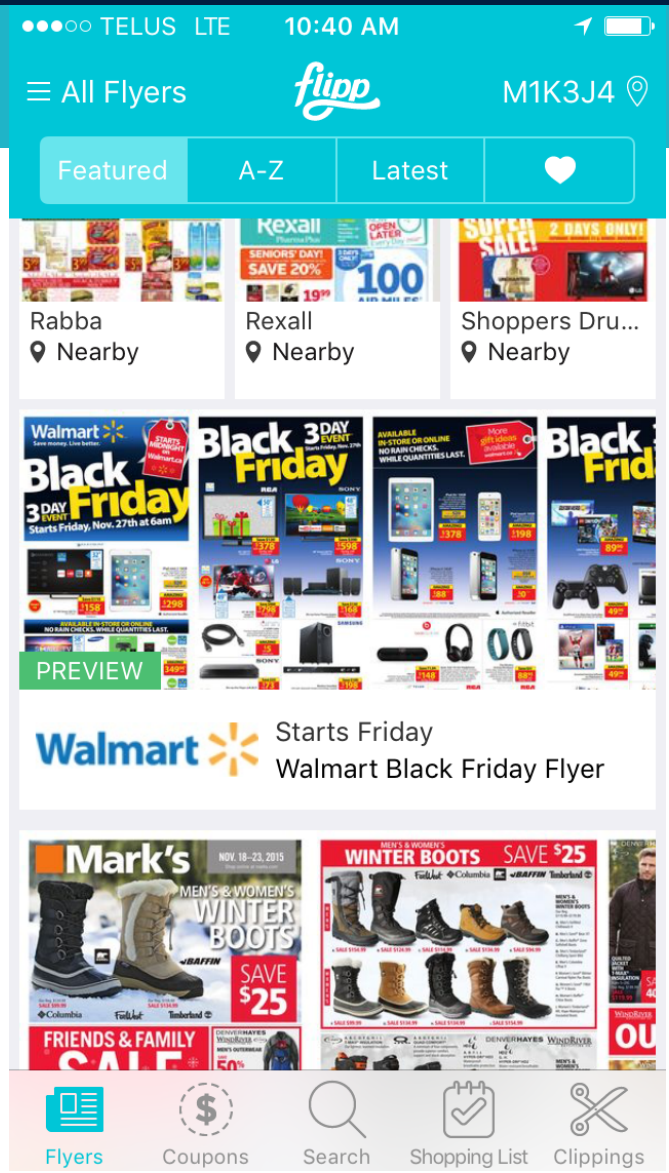
What we do at Flipp

- **Connecting consumers with local deals for their weekly shopping**
  - The Flipp App
    - Flyers
    - Coupons
    - Shopping lists
  - Other distribution channels



Data we deal with at Flipp

- Event data across channels
  - User interactions
    - Millions of daily active users
  - No interactions
    - Impressions
- Budgets and billing
- Entity data
  - Flyers, Items, .. etc



Our big data cluster

- Yarn cluster on AWS
  - Tens of nodes.. not tons
- Spark
- Hive
- Luigi
- One off uses of other tools

Usage of Spark at Flipp

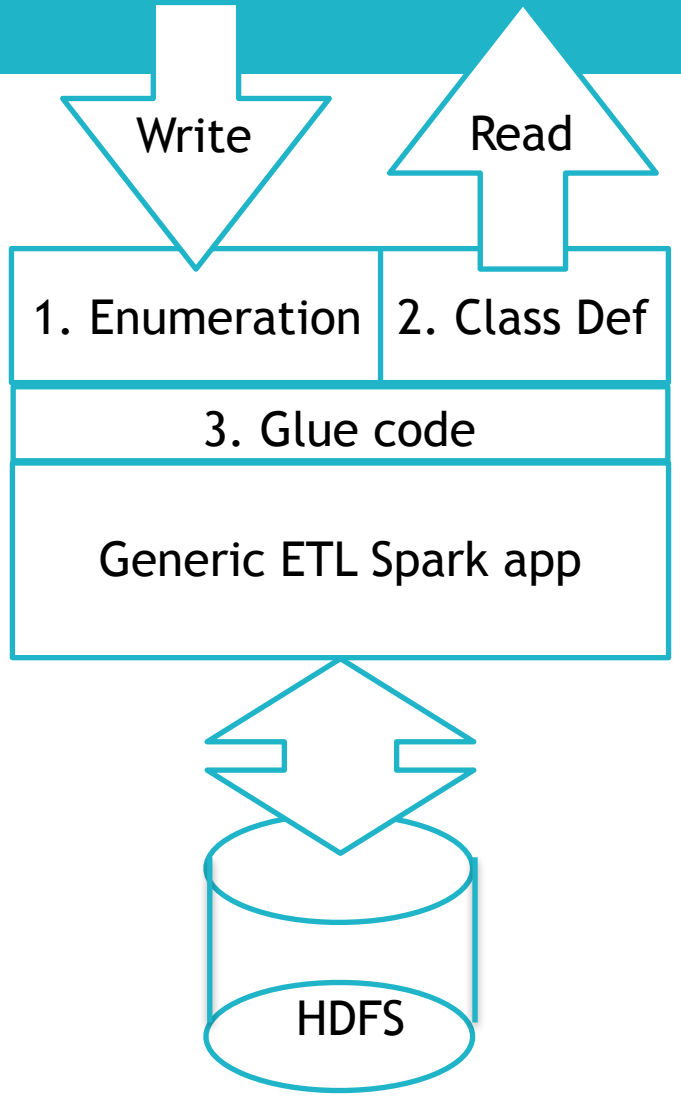
- ETL jobs
- Simple batch jobs (traditionally written in Pig)
- Machine Learning (MLLib)
- Stream processing

ETLs using Spark

- We keep our data clean
  - No surprises at query time
  - Scala's type system helps a lot
- One generic ETL Spark app
  - Don't repeat yourself
  - Really makes use of Generics

Adding a new ETL

- Enumeration to describe the schema
  - Used to do most of the work
  - Documentation that remains true
- Class to hold a deserialized record
- A little bit of glue code



More Merits to writing ETLs in Spark

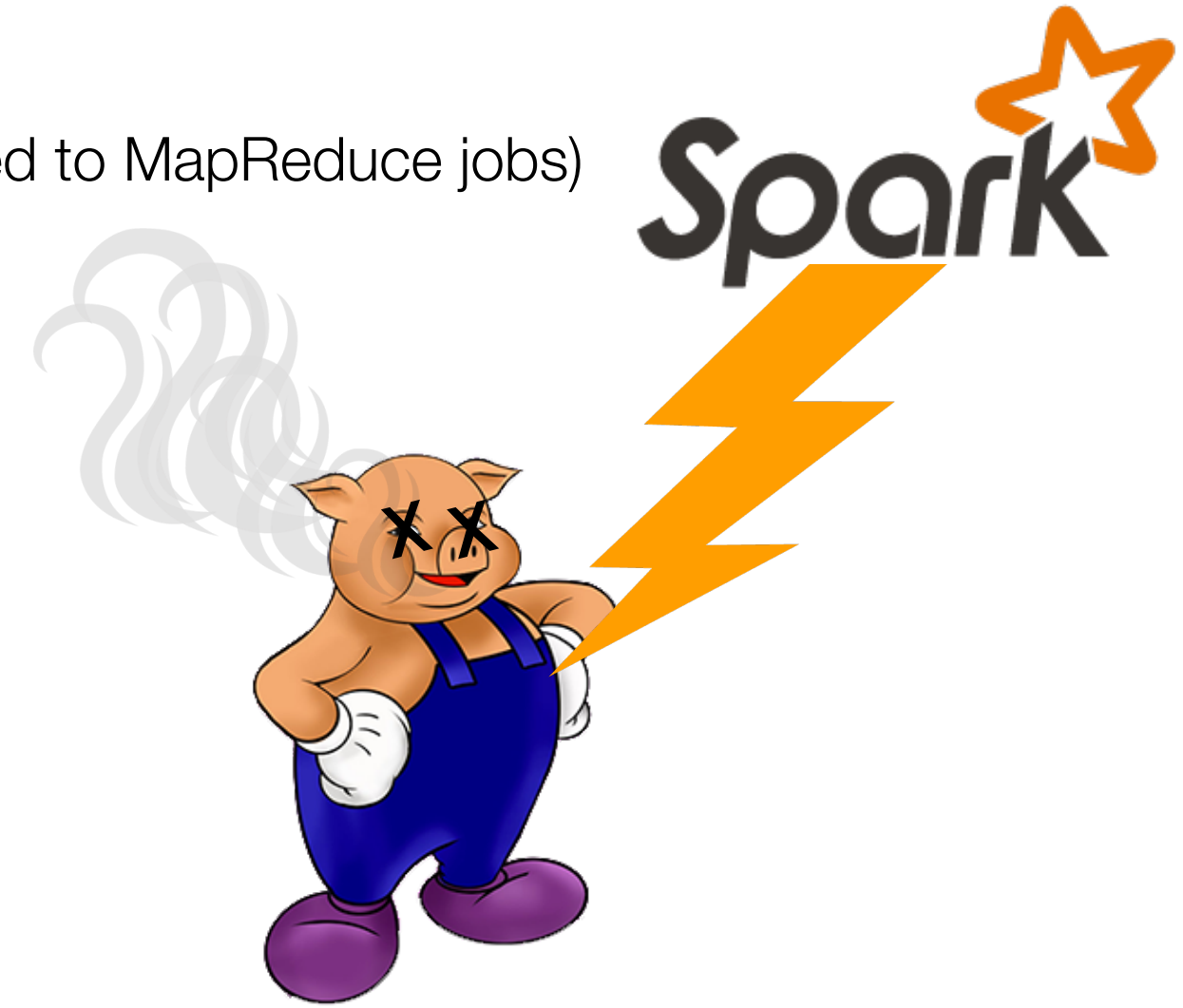
- Coding in Scala
  - As expressive as dynamic languages
  - Powerful type system
  - Access to mature Java libraries

- Joining massive RDDs
  - Remember to "Import spark.SparkContext._ " at the top of the file
  - Broadcast to denormalize ids to corresponding values

Moving from Pig to Spark

- Pig was a good tool at its time
  - Pig jobs are easy to write (compared to MapReduce jobs)

- Compared to Spark
  - Pig jobs can be very slow
  - Scala vs Pig Latin
    - Expressiveness
    - Reusability and modularity
    - Testability



Machine Learning with MLLib

- Fastest possible on-boarding
  - Learning curve next to none
  - No installation required
- Spark for preparing the data
- MLLib usage attempts at Flipp
  - Classification of products into categories
  - Prediction of the usage of the app (and other channels)
  - Grouping similar entities

Classification of products

- MLLib's SVM classifier performs very well
  - In terms of scalability
  - In terms of accuracy

- Caveat: Necessary tools are marked experimental up to v1.5
  - Evaluation (org.apache.spark.mllib.evaluation)
  - Tuning (org.apache.spark.mllib.tuning)

- Pain point: Saving and loading models prior to v1.3

Prediction of the usage of the app

- MLLib's regression is ok
  - Overhead of distributed implementation

- PredictionIO (http://prediction.io/)
  - Cuts down boiler plate drastically
    - Packaging a Spark app into a Jar with dependencies
    - Saving a model, and loading it into a web service
  - Caveat: You will need to dive into the code of the framework

Grouping Similar Entities

- Clustering
  - MLLib K-Means doesn't allow changing the distance measure
    - Euclidean distance works with very few things

- Community detection on a GraphX graph
  - A very strong data structure for representing graphs
  - Exposes various high level graph abstractions and operations
  - Not supported by good documentation
    - Even diving into the code wasn't very helpful

Spark Streaming

- Not much to say.. it just works!
  - Regardless of the plumbing
- Jobs written in Python
  - Very simple
- Heard warning about the Spark Streaming job halting but it never happened

Wrap up

- Writing ETLs in Scala is great
  - Certainly beats Pig Latin
  - Testability of Spark jobs
- MLLib
  - Classification and Regression are good
  - Consider PredictionIO to cut down boiler plate
  - Clustering and Graph algorithms are not the best (yet)
- Spark Streaming works like a charm

Thank you!

Questions?

younes.abouelnagah@flipp.com

+

= <3