Distributed Time Travel for Feature Generation

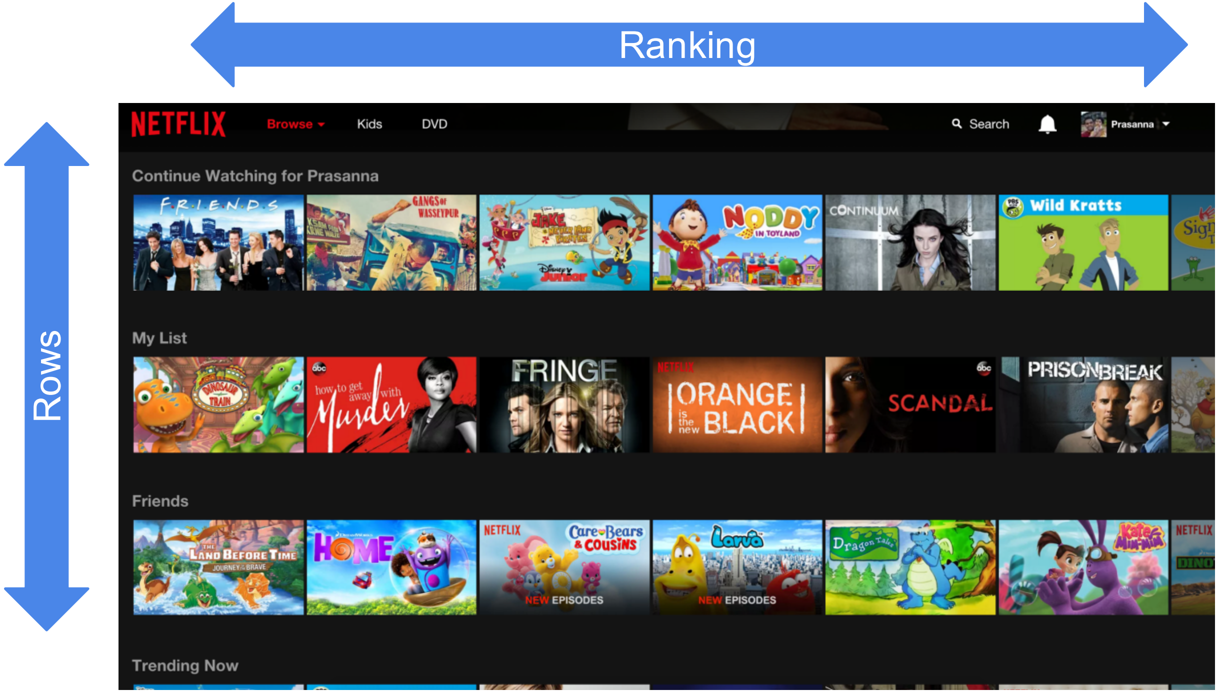
Prasanna Padmanabhan
DB Tsai
Mohammad H. Taghavi

NETFLIX

SPARK SUMMIT EAST
DATA SCIENCE AND ENGINEERING AT SCALE
FEBRUARY 16-18, 2016 NEW YORK CITY

Turn on Netflix, and the **absolute best content for you** would **automatically** start **playing**

Everything is a Recommendation



Over 80% of what members watch comes from our recommendations

Recommendations are driven by **Machine Learning Algorithms**

Data Driven

- Try an **idea offline using historical data** to see if it would have made better recommendations

- If it did, deploy a live **A/B test** to see if it performs well in Production

Why build a Time Machine?

Quickly try ideas on **historical data** and **transition** to online A/B test

The Past

- Generate features based on event data logged in Hive
  - Need to reimplement features for online A/B test
  - Data discrepancies between offline and online sources

- Log features online where the model will be used
  - Need to deploy each idea into production

- Feature generation calls online services and filters data past a certain time
  - Works only when a service records a log of historical events
  - Additional load on online services

DeLorean image by JMortonPhoto.com & OtoGodfrey.com

Time Travel using Snapshots

- Snapshot online services and use the snapshot data offline to generate features

- Share facts and features between experiments without calling live systems

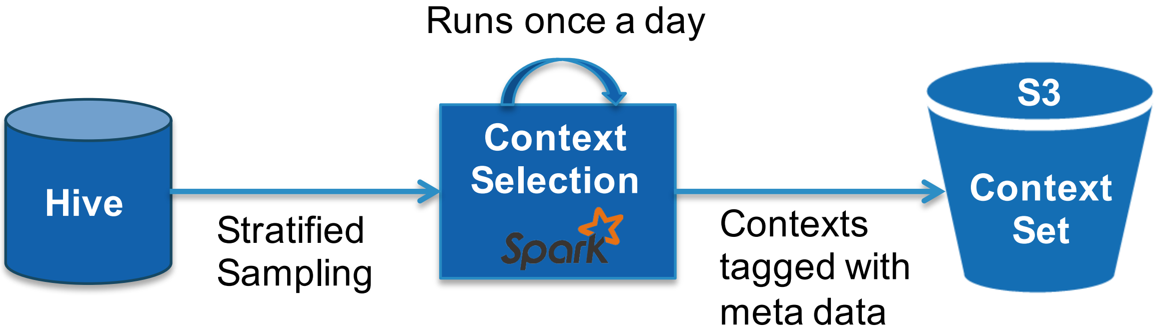
How to build a Time Machine

Context Selection

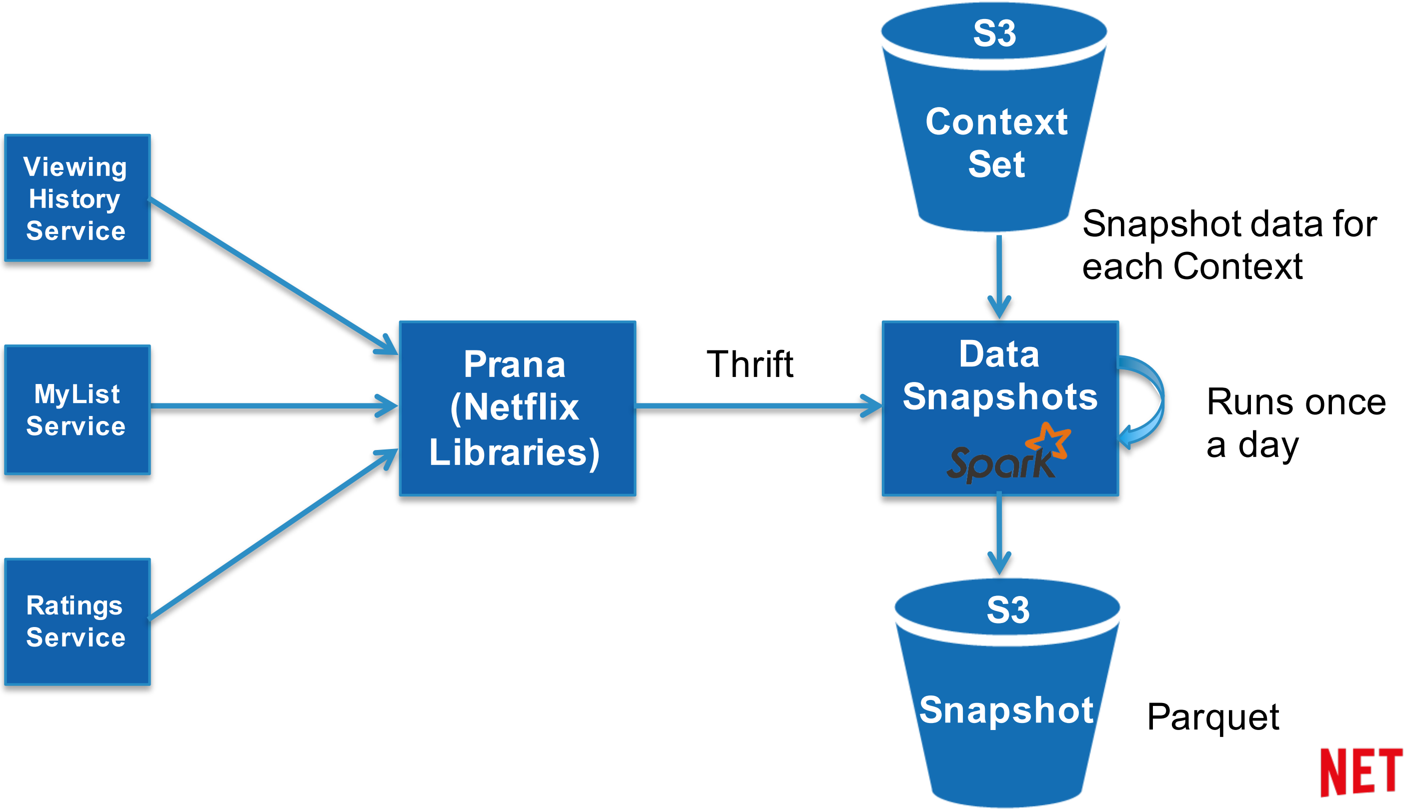
Data Snapshots

APIs for Time Travel

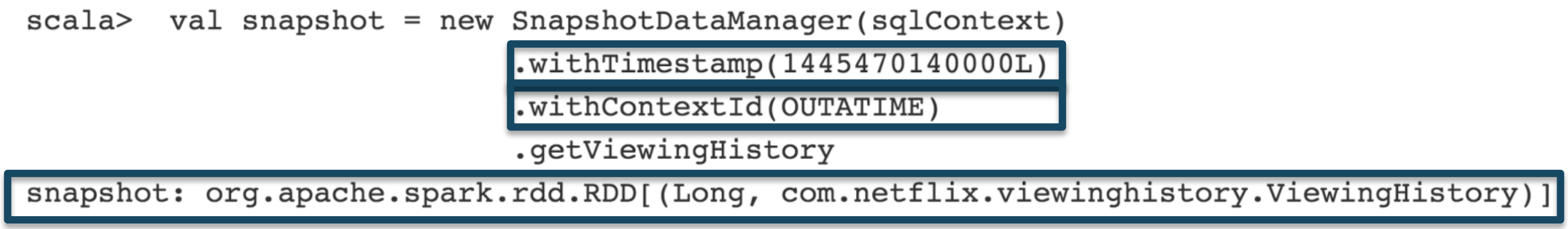
Context Selection



Data Snapshots

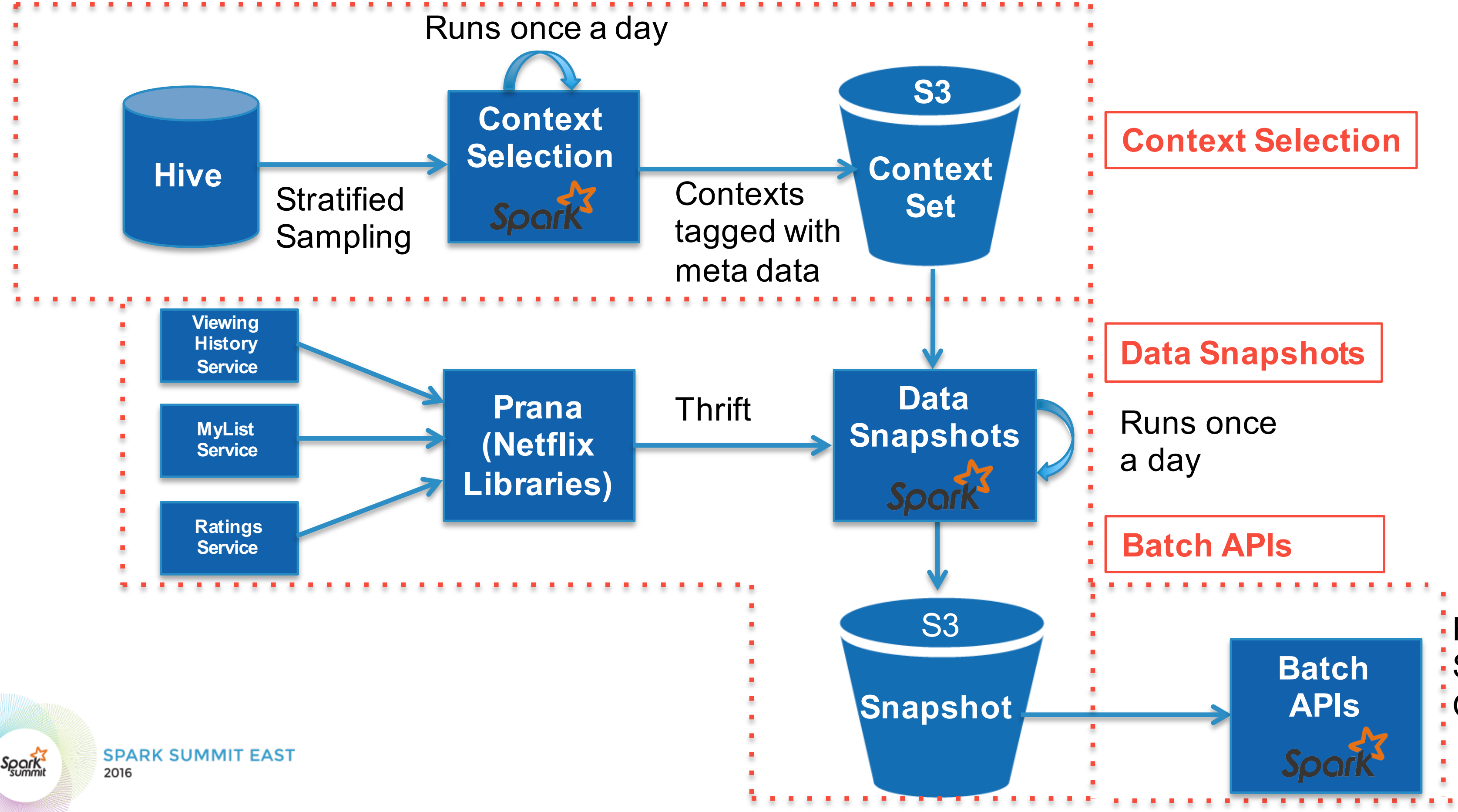


APIs for Time Travel



```
scala>  val snapshot = new SnapshotDataManager(sqlContext)
                       .withTimestamp(1445470140000L)
                       .withContextId(OUTATIME)
                       .getViewingHistory
snapshot: org.apache.spark.rdd.RDD[(Long, com.netflix.viewinghistory.ViewingHistory)]
```

Data Architecture



RDD of Snapshot Objects

NETFLIX

Generating **Features** via **Time Travel**

Great Scott! There's the DeLorean!

- DeLorean: A time-traveling vehicle
  - uses data snapshots to travel in time
  - scales with Apache Spark
  - prototypes new ideas with Zeppelin
  - requires minimal code changes from experimentation to A/B test to production

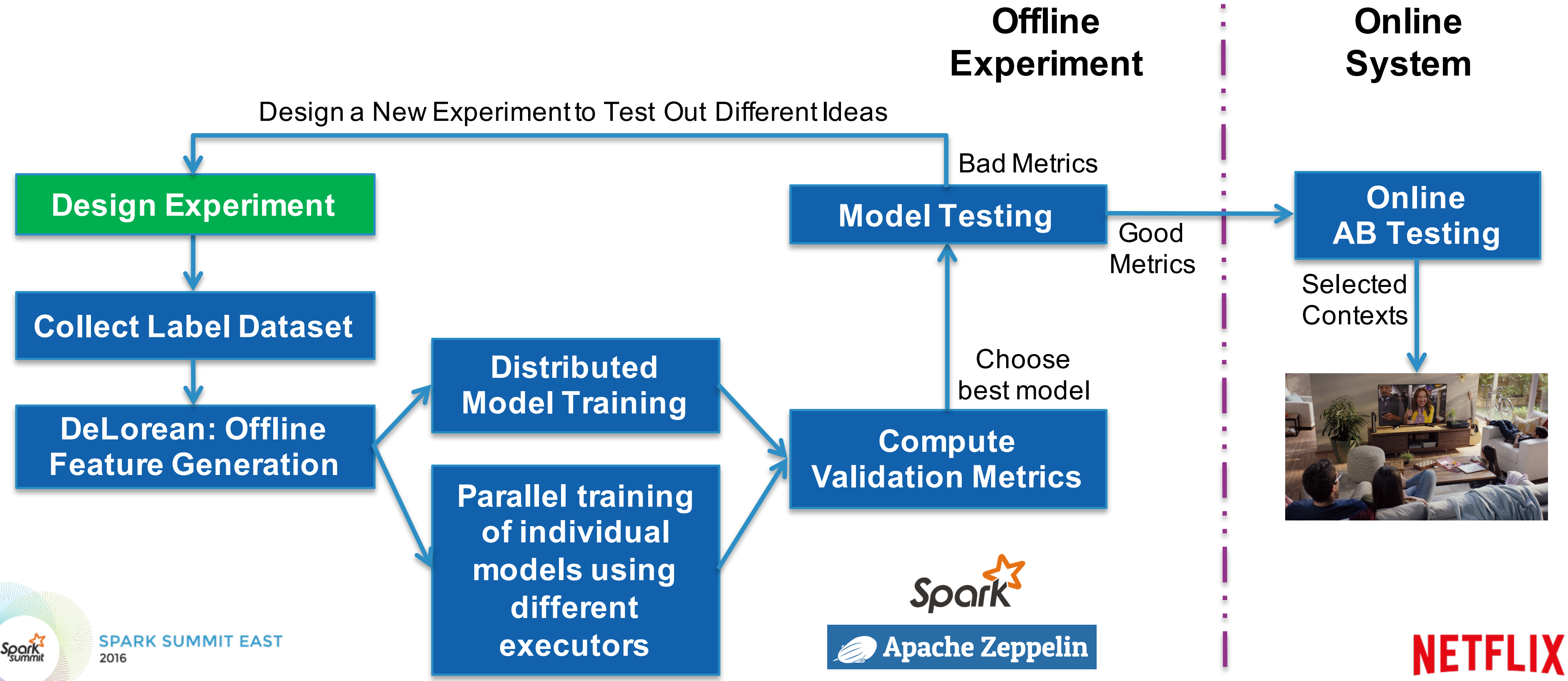
https://en.wikipedia.org/wiki/Emmett_Brown

Running Time Travel Experiment

Select the **destination time**

Bring it up to **88 miles** per hour!

Running Time Travel Experiment



Offline Experiment

Online System

Design a New Experiment to Test Out Different Ideas

Bad Metrics

Design Experiment

Collect Label Dataset

DeLorean: Offline Feature Generation

Distributed Model Training

Parallel training of individual models using different executors

Compute Validation Metrics

Choose best model

Model Testing

Good Metrics

Online AB Testing

Selected Contexts

Spark

Apache Zeppelin

DeLorean Input Data

- Contexts: The setting for evaluating a set of items (e.g. tuples of member profiles, country, time, device, etc.)

- Items: The elements to be trained on, scored, and/or ranked (e.g. videos, rows, search entities).

- Labels: For supervised learning, this will be the label (target) for each item.

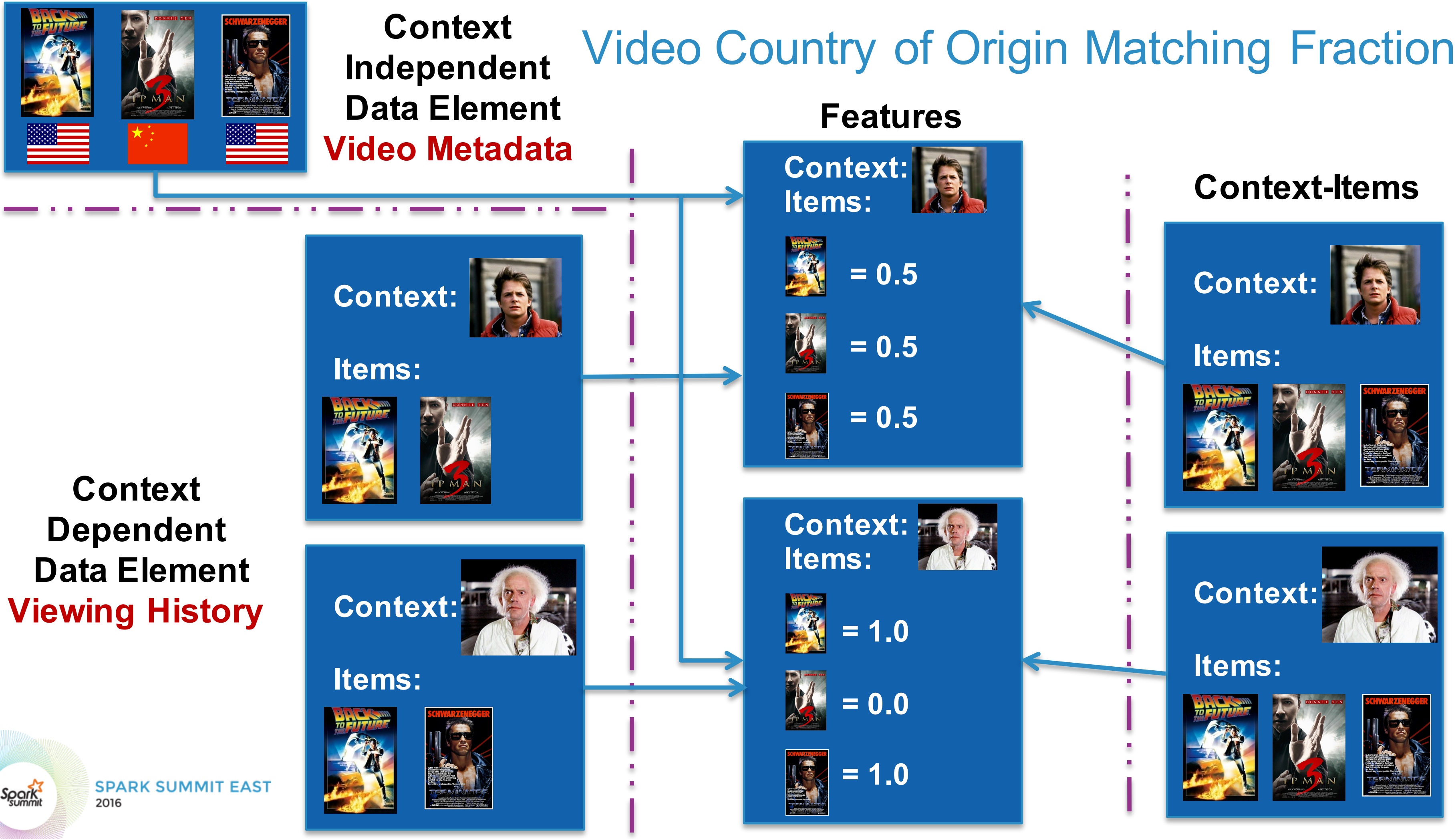
Feature Encoders

- Compute features for each item in a given context
- Each type of raw data element has its own data key
- Data map is a map from data keys to data objects in a given context
- Data map is consumed by feature encoder to compute features

Two type of Data Elements

- Context-dependent data elements
  - Viewing History
  - Mylist
  - ...
- Context-independent data elements
  - Video Metadata
  - Genre Metadata
  - ...

Video Country of Origin Matching Fraction



Context Independent Data Element

Video Metadata

Context Dependent Data Element

Viewing History

Context:

Items:

Context:

Items:

Features

Context:
Items:

= 0.5

= 0.5

= 0.5

Context:
Items:

= 1.0

= 0.0

= 1.0

Context-Items

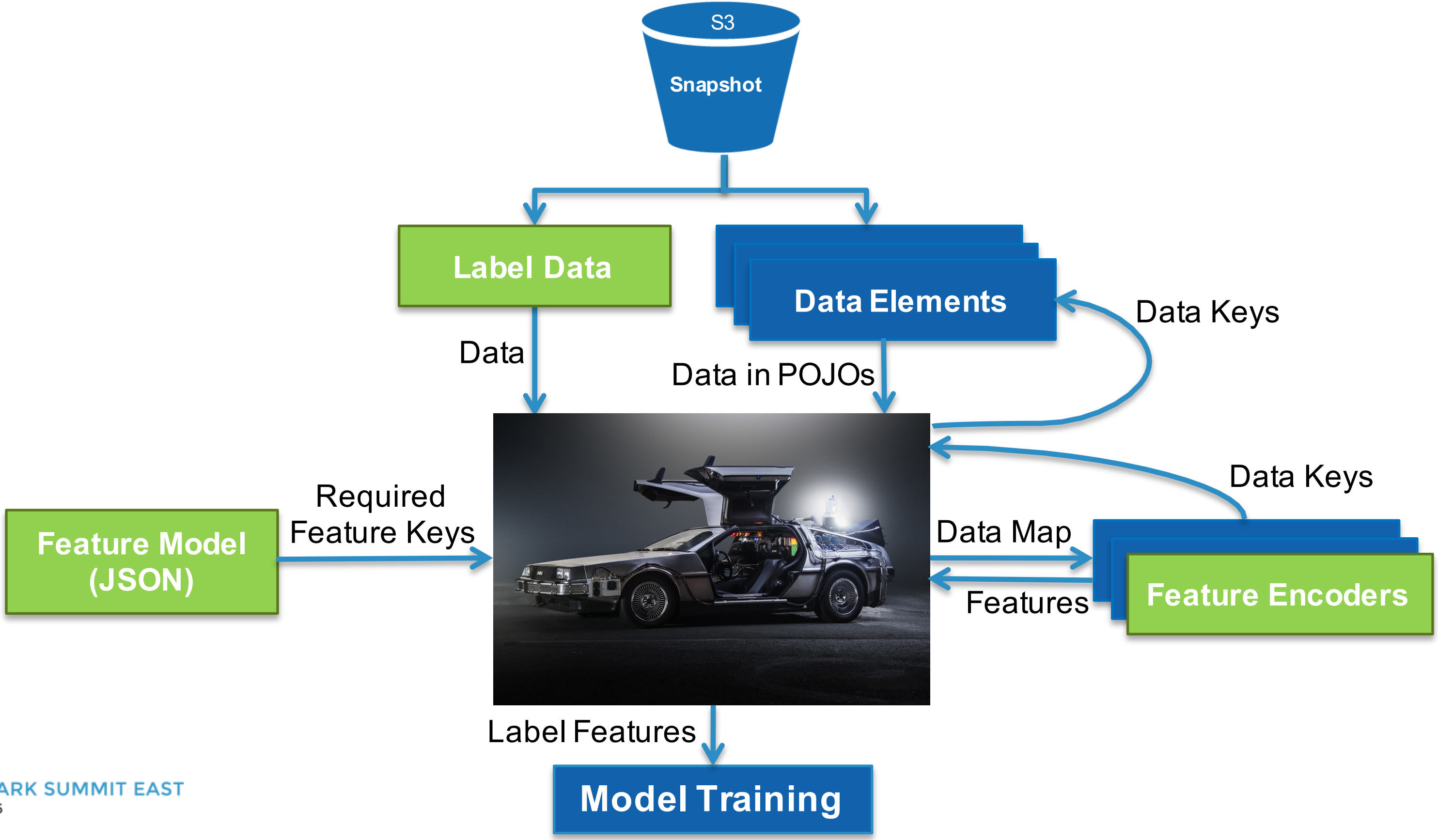
Context:

Items:

Context:

Items:

Feature Generation



S3

Snapshot

Label Data

Data Elements

Data Keys

Data

Data in POJOs

Data Keys

Required Feature Keys

Feature Model (JSON)

Data Map

Features

Feature Encoders

Label Features

Model Training

Features

- Represented in Spark's DataFrames
- In nested structure to avoid data shuffling in ranking process
- Stored with Parquet format in S3

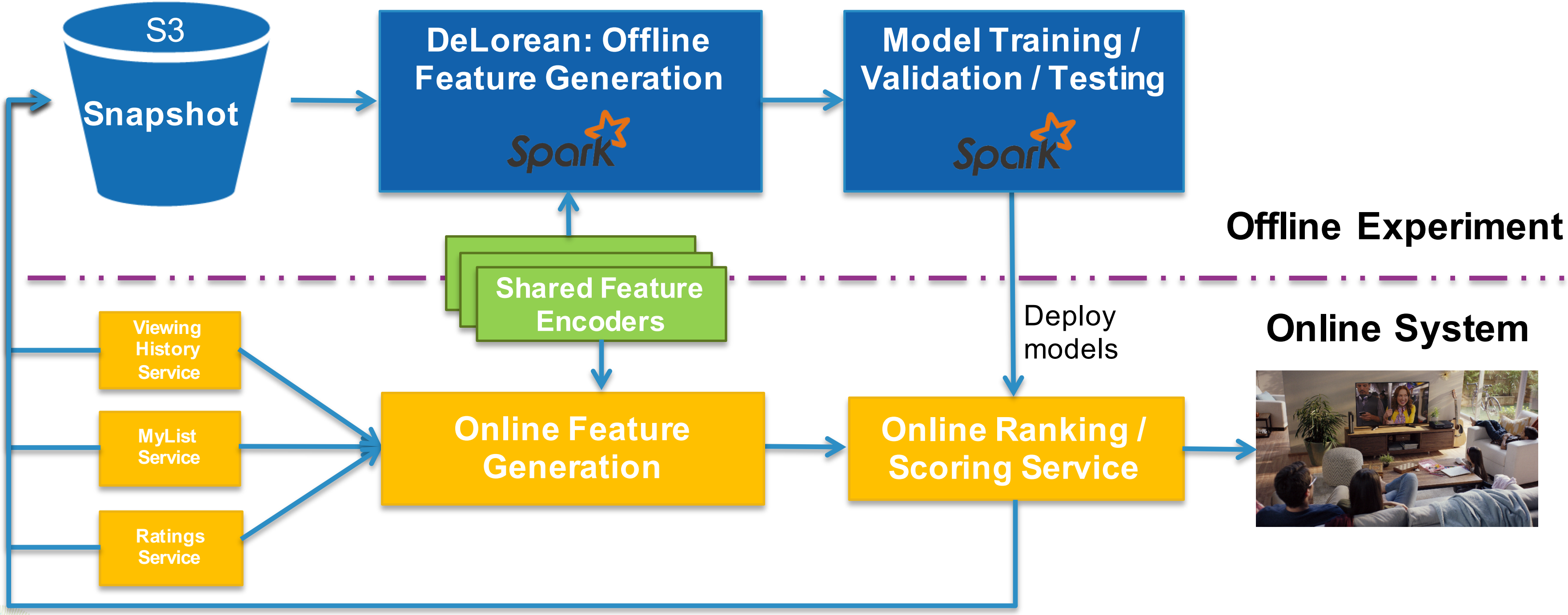
Features

Context

```
root
 |-- Visitor: long (nullable = false)
 |-- Country: string (nullable = false)
 |-- data: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- videoId: long (nullable = false)
 |    |    |-- weight: double (nullable = false)
 |    |    |-- label: double (nullable = false)
 |    |    |-- features: struct (nullable = false)
 |    |    |    |-- rating: double (nullable = false)
 |    |    |    |-- unpersonalizedPopularity: double (nullable = false)
 |    |    |    |-- ...
 |    |    |    |-- ...
 |    |    |    |-- ...
```

Item, label,
and features

Going Online



Conclusion

Spark helped us significantly reduce the time from an idea to an AB Test

Future work

Event Driven Data Snapshots

Time Travel to the Future!!

We’re hiring!

(come talk to us)

https://jobs.netflix.com/

Tech Blog: http://bit.ly/sparktimetravel