ML through Streaming at

Sherin Thomas
@doodlesmt
Stopping a Phishing Attack
Hello Alex, I'm Tracy calling from Lyft HQ. This month we're awarding $200 to all 4.7+ star drivers. Congratulations!

Hey Tracy, thanks!

Np! And because we see that you’re in a ride, we’ll dispatch another driver so you can park at a safe location....

....Alright your passenger will be taken care of by another driver
Before we can credit you the award, we just need to quickly verify your identity.

We'll now send you a verification text. Can you please tell us what those numbers are……

12345
Fingerprinting
Fraudulent
Behaviour
Sequence of User Actions

- Request Ride
- ...
- Driver Contact
- Cancel Ride
- .....
Sequence of User Actions

- Request Ride
- ...
- Driver Contact
- Cancel Ride
- ..... 
- Something

Red Flag

Reference: Fingerprinting Fraudulent Behaviour
SELECT
    user_id,
    TOP(2056, action) OVER (PARTITION BY user_id
    ORDER BY event_time
    RANGE INTERVAL '90' DAYS PRECEDING) AS client_action_sequence
FROM event_user_action
SELECT
  user_id,
  TOP(2056, action) OVER (
    PARTITION BY user_id
    ORDER BY event_time
    RANGE INTERVAL '90' DAYS PRECEDING
  ) AS client_action_sequence
FROM event_user_action
SELECT
  user_id,
  TOP(2056, action) OVER (PARTITION BY user_id ORDER BY event_time RANGE INTERVAL '90' DAYS PRECEDING)
AS client_action_sequence
FROM event_user_action

Historic context is also important (large lookback)
SELECT
    user_id,
    TOP(2056, action) OVER (PARTITION BY user_id
                             ORDER BY event_time
                             RANGE INTERVAL '90' DAYS PRECEDING
                        ) AS client_action_sequence
FROM event_user_action
Make streaming features accessible for ML use cases
THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
Flink
Apache Flink

- Low latency stateful operations on streaming data - in the order or milliseconds
- Event time processing - replayability, correctness
- Exactly once processing
- Failure recovery
- SQL Api
Event Ingestion Pipeline

{ "ride_req", "user_id": 123, "event_time": t0 }
Processing Time vs Event Time

**Processing time**
System time when the event is processed -> determined by processor

**Event time**
Logical time when the event occurred -> part of event metadata

Credit: The Beam Model by Tyler Akidau and Frances Perry
Example: integer sum over 2 min window

Credit: [The Beam Model](#) by Tyler Akidau and Frances Perry
Example: integer sum over 2 min window

Credit: The Beam Model by Tyler Akidau and Frances Perry
Example: integer sum over 2 min window

Credit: The Beam Model by Tyler Akidau and Frances Perry
Usability
What Data Scientists care about

1. Model Development
2. Feature Engineering
3. Data Quality
4. Scheduling, Execution, Data Collection
5. Compute Resources
ML Workflow

Data Input
- Data Discovery
- Normalize and Clean Up Data

Data Prep
- Extract & Transform Features
- Label Data
- Maintain External Feature Sets

Modeling
- Train Models
- Evaluate and Optimize

Deployment
- Deploy
- Monitor & Visualize Performance
ML Workflow

Data Input
- Data Discovery
- Normalize and Clean Up Data

Data Prep
- Extract & Transform Features
- Label Data
- Maintain External Feature Sets

Modeling
- Train Models
- Evaluate and Optimize

Deployment
- Deploy
- Monitor & Visualize Performance
Dryft! - Self Service Streaming Framework

User Plane
- Dryft UI

Control Plane
- Data Discovery
- Query Analysis
- Job Cluster

Data Plane
- Kafka
- DynamoDB
- Druid
- Hive
- Elastic Search
Declarative Job Definition

Flink SQL

```
SELECT
    geohash,
    COUNT(*) AS total_events,
    TUMBLE_END(rowtime, INTERVAL '1' hour)
FROM event_user_action
GROUP BY TUMBLE(rowtime, INTERVAL '1' hour)
```

Job Config

```
{
    "retention": {},
    "lookback": {},
    "stream": {
        "kinesis": user_activities
    },
    "features": {
        "user_activity_per_geohash": {
            "type": "int",
            "version": 1,
            "description": "user activities per geohash"
        }
    }
}
```
Eating our own dogfood 🐶
Feature Fanout App - also uses Dryft

```
SELECT
    -- this will be used in keyBy
    CONCAT_WS('_', feature_name, version, id),
    feature_data,
    CONCAT_WS('_', feature_name, version)
    AS feature_definition,
    occurred_at
FROM features

{  
    “stream”: {  
        “kinesis”: feature_stream  
    },  
    “sink”: {  
        “feature_service_dynamodb”: {  
            “write_rate”: 1000,  
            “retry_count”: 5  
        }  
    }  
}
```
Deployment
Previously...

- Ran on AWS EC2 using custom deployment
- Separate autoscaling groups for JobManager and Taskmanagers
- Instance provisioning done during deployment
- Multiple jobs(60+) running on the same cluster
Multi tenancy hell!!

Apache Flink Dashboard

Overview | Version: 1.4.2 | Commit: 04e4c85

- **Task Managers**: 1058
- **Task Slots**: 4232
- **Available Task Slots**: 1222

**Total Jobs**
- Running: 65
- Finished: 0
- Canceled: 99
- Failed: 0

**Running Jobs**

<table>
<thead>
<tr>
<th>Start Time</th>
<th>End Time</th>
<th>Duration</th>
<th>Job Name</th>
<th>Job ID</th>
<th>Tasks</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-06-14, 2:51:47</td>
<td>2019-06-14, 23:00:33</td>
<td>20h 8m</td>
<td>marketplace.ride_requested_1_minute.4</td>
<td>880cd1c5e2144dbbcf35e481d7b615ef</td>
<td></td>
<td>RUNNING</td>
</tr>
<tr>
<td>2019-06-14, 9:19:15</td>
<td>2019-06-14, 23:00:33</td>
<td>13h</td>
<td>marketplace.ride_requested_4_hours.3</td>
<td>da2250b2800a9909b03eadb640088e638</td>
<td></td>
<td>RUNNING</td>
</tr>
<tr>
<td>2019-06-14, 11:06:55</td>
<td>2019-06-14, 23:00:33</td>
<td>11h</td>
<td>marketplace.ride_canceled_pax_1_minute.3</td>
<td>1a2b53a887f0cc4790429b86c111bd85</td>
<td></td>
<td>RUNNING</td>
</tr>
<tr>
<td>2019-06-14, 11:39:32</td>
<td>2019-06-14, 23:00:33</td>
<td>11h</td>
<td>marketplace.ride_canceled_pax_5_seconds.4</td>
<td>2302424953b9ae728f73a464dada188b8ea</td>
<td></td>
<td>RUNNING</td>
</tr>
<tr>
<td>2019-06-14, 13:00:33</td>
<td>2019-06-14, 23:00:33</td>
<td>9h 59m</td>
<td>marketplace.ride_canceled_pax_24_hours.3</td>
<td>18688c933904b4bbf47a61d431d9fb3d</td>
<td></td>
<td>RUNNING</td>
</tr>
<tr>
<td>2019-06-14, 13:00:42</td>
<td>2019-06-14, 23:00:33</td>
<td>9h 59m</td>
<td>marketplace.ride_requested_1_hour.4</td>
<td>3e1bb25908ab632f90919c22e1b70b</td>
<td></td>
<td>RUNNING</td>
</tr>
</tbody>
</table>
Kubernetes Based Deployment

App 1

App 2

App 3
Flink-K8s-Operator

Managing Flink on Kubernetes
Custom Resource Descriptor

```yaml
apiVersion: flink.k8s.io/v1alpha
kind: FlinkApplication
metadata:
  name: flink-speeds-working-stats
  namespace: flink
spec:
  image: '100,dkr.ecr.us-east-1.amazonaws.com/abc:xyz'
  flinkJob:
    jarName: name.jar
    parallelism: 10
  taskManagerConfig: {
    resources: {
      limits: {
        memory: 15Gi,
        cpu: 4
      }
    },
    replicas: num_task_managers,
    taskSlots: NUM_SLOTS_PER_TASK_MANAGER,
    envConfig: {...},
  }
```

- Custom resource represents Flink application
- Docker Image contains all dependencies
- CRD modifications trigger update (includes parallelism and other Flink configuration properties)
Flink on Kubernetes

- Separate Flink cluster for each application
- Resource allocation customized per job - at job creation time
- Scales to 100s of Flink applications
- Automatic application updates
Bootstrapping
What is bootstrapping?

```sql
SELECT
    passenger_id,
    COUNT(ride_id)
FROM event_ride_completed
GROUP BY
    passenger_id,
    HOP(rowtime,
        INTERVAL '30' DAY,
        INTERVAL '1' HOUR)
```
Read historic data to ‘bootstrap’ the program with 30 days worth of data. Now your program returns results on day 1. But what if the source does not have all 30 days worth of data?
Solution - Consume from two sources

Read historic data from persistent store (AWS S3) and streaming data from Kafka/Kinesis

Bootstrapping state in Apache Flink - Hadoop Summit
Job starts
Bootstrapping over
Start Job
With a higher parallelism for fast bootstrapping

Detect Bootstrap Completion
Job sends a signal to the control plane once watermark has progressed beyond a point where we no longer need historic data

“Update” Job with lower parallelism but same job graph
Control plane cancels job with savepoint and starts it again from savepoint but with a much lower parallelism
Output volume spike during bootstrapping

Bootstrapping
Output volume spike during bootstrapping

- Features need to be fresh but eventually complete
- Smooth out data writes during bootstrap to match throughput
- Write features produced during bootstrapping separately
What about skew between historic and real-time data?
Skew

Watermark =

Kinesis

S3
Solution: Source synchronization

FLINK-10887, FLINK-10921, FLIP-27
Now...
• 120+ features
• Features available in DynamoDB (real time point lookup), Hive (offline analysis), Druid (real time analysis) and more...
• Time to write, test and deploy a feature is $\leq \frac{1}{2}$ day
• p99 latency <5 seconds
• Coming Up - Python Support!
Thank you!
Later
Backfill

- What if one implementation could provide the training time and scoring time feature values?
  - Batch processing mode to backfill historic values for training
  - Stream processing mode to generate values in real-time for model scoring
- Enable delivery of consistent features between training and scoring
- Green/Blue deploy - zero downtime deploys
- “Auto” scaling of Flink cluster and/or job parallelism
- Feature library