Deep Dive
into threat detection in the cloud
with Spark and Python

David Melamed, PhD
Senior Research Engineer, CloudLock
dmelamed@cloudlock.com     @dvdmelamed

PyCon Israel 2016, May 2-3 2016
Who is this guy?
Where is he working?

CloudLock

- Founded: 2011
- Corporate Headquarters: Waltham, Mass. (U.S.A.)
- R&D Headquarters: Tel Aviv
- Employees: 140 (30 in TLV)
- Trusted by major brands:

- GSA: 20,000 Data Privacy Federal
- Woolworths: 250,000 PCI-DSS Retail
- BBVA: 140,000 Reg Compliance Financial Services
- Akamai: 5,000 Trade Secrets Technology
- Dominion: 4,300 PCI-DSS Retail
- Sigma-Aldrich: 10,000 PHI/IP Life Sciences
- ASU: 750,000 PII / FERPA Education
- Yahoo!: 12,000 Data Privacy High Tech
- ACI: 4,800 Reg Compliance Financial Services
- Enterprise: 80,000 PII / PCI Transportation
- Whirlpool: 27,000 Data Privacy Manufacturing
- Cloud Vendor: 72,000
CyberSecurity in the cloud - multiple providers
Data Collection

- OAuth
- webhook

Celery worker

Normalization
Celery - tips when dealing with scale

- If you don’t use the celery task result, use `@task(ignore_result=True)`
- `max_child_tasks_per_node` is important to determine the number of tasks before killing the interpreter (impacts cached data)
- Limitation: Celery can only work with 1 RabbitMQ
- Celery heartbeats
- Batching mode (experimental)
Data Cleaning, Normalization and Enrichment

- Multiple sources: fields mapping
- Schema enforcement: use versioning
- Data enrichment for IP address: geolocation data, IP Reputation data
- Other possible enrichment: early sensitive activity detection
- Use `gevent` since we do a lot of I/O
Data encryption & issues we encountered

- S3 input (official) - API call for each bucket object
- S3 input (official) - latency between files to process
- S3 input (non official) - only initial latency but stops when no new file
- Encryption filter - no multi-core support
Batch data processing & serving architecture overview
Spark Python and JVM
Spark Performance - Python vs. Scala

![Bar chart comparing Spark Python DF, Spark Scala DF, RDD Python, and RDD Scala performance. RDD Python has the longest bar, indicating it takes the longest to aggregate 10 million int pairs.]

Source: Databricks
Parquet file format

- Open-source format
- Supported natively by Spark
- Loads only the columns needed
- Data is compressed
- Supports nested data structure
- Schema merge support
- Implementation in Java (MapReduce) and C++

```json
message root {
  optional binary @timestamp (UTF8);
  optional binary @version (UTF8);
  optional binary asset_asset_id (UTF8);
  ...
  optional binary principal_user_user_email (UTF8);
  optional binary principal_user_vendor_id (UTF8);
  optional binary timestamp (UTF8);
  optional int64 version;
}
```

**Row-based storage**

```
<table>
<thead>
<tr>
<th>title</th>
<th>author</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>title1</td>
<td>author1</td>
<td>year1</td>
</tr>
<tr>
<td>title2</td>
<td>author2</td>
<td>year2</td>
</tr>
<tr>
<td>title3</td>
<td>author3</td>
<td>year3</td>
</tr>
</tbody>
</table>
```

**Columna-based storage**

```
title1 | title2 | title3 | author1 | author2 | author3 | year1 | year2 | year3
```
Spark / S3 - tips when dealing with scale

- Prefer lzio over gzip for input files
- S3 over HDFS adds latency
- Use the Kryo object serializer over Java if possible
- Use data partitioning
- Disable summary metadata for better performance
- Use parquet direct committer
EMR / Spark on YARN tips

- Configuration file on S3
- Make sure to turn on dynamic resource allocation or use EMR Release 4.4+
- Bootstrap file on S3, installs required packages
- Use DevPI to store packages
- Use EMR “steps” for easy access to the logs
df = sqlContext.read.parquet("s3://bucket-name/*.gz")
df.registerTempTable("events")
um_events = df.sql("SELECT COUNT(*) FROM events").show()
print "Num events: {}".format(num_events)
Spark for Machine Learning

- Spark MLLib (works with RDDs)
  - Binary classification
    - SVM
    - Logistic Regression
  - Linear regression
  - Clustering
    - k-means
  - Collaborative Filtering
  - Gradient descent

- Spark ML (works with DF)
from pyspark.mllib.clustering import KMeans
from numpy import array
from math import sqrt

# Load and parse the data
data = sc.textFile("kmeans_data.txt")
parsedData = data.map(lambda line: array([float(x) for x in line.split(' ')])

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, 2, maxIterations=10,
    runs=30, initialization_mode="random")

# Evaluate clustering by computing Within Set Sum of Squared Errors
def error(point):
    center = clusters.centers[clusters.predict(point)]
    return sqrt(sum([x**2 for x in (point - center)]))

WSSSE = parsedData.map(lambda point: error(point)).reduce(lambda x, y: x + y)
print("Within Set Sum of Squared Error = " + str(WSSSE))
Spark MLLib - Geo based Anomaly Detection

- 5K employee company
- Running an ensemble of clustering algorithms
- Using proprietary distance calculation to locate outliers
- Filter out noise
- Discovered access from 2 abnormal locations (Nigeria and Vietnam) within 4M events for this organization in a specific month
Microservice using Docker

- AWS ECR to store the Docker images
- AWS ECS for the containers (supports auto-scaling and ELB)
- Ansible to deploy

Ansible at Scale
Monday, May 9, 2016

http://www.meetup.com/Ansible-Israel/events/230313714/
<table>
<thead>
<tr>
<th>Threat</th>
<th>Platform</th>
<th>Date</th>
<th>City</th>
<th>State</th>
<th>Country</th>
<th>IP Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location Anomaly</td>
<td>Google</td>
<td>February 28th 2016, 15:06 pm</td>
<td>Abu Dhabi</td>
<td></td>
<td>United Arab Emirates</td>
<td></td>
<td>Anomalous activity from Abu Dhabi (United Arab Emirates)</td>
</tr>
<tr>
<td>Location Anomaly</td>
<td>Google</td>
<td>February 28th 2016, 15:03 pm</td>
<td>Buenos Aires</td>
<td></td>
<td>Argentina</td>
<td></td>
<td>2 Anomalous activities from Buenos Aires (Argentina)</td>
</tr>
<tr>
<td>Location Anomaly</td>
<td>Google</td>
<td>February 28th 2016, 15:03 pm</td>
<td></td>
<td></td>
<td>India</td>
<td></td>
<td>17 Anomalous activities from India</td>
</tr>
</tbody>
</table>