Scaling up MATLAB Analytics with Kafka and Cloud Services

Branko Dijkstra
Agenda

1. Access and Explore Data
   - Files
   - Databases
   - Sensors

2. Preprocess Data
   - Working with Messy Data
   - Data Reduction/Transformation
   - Feature Extraction

3. Develop Predictive Models
   - Model Creation e.g. Machine Learning
   - Parameter Optimization
   - Model Validation

4. Integrate with Production Systems
   - Desktop Apps
   - Enterprise Scale Systems
   - Embedded Devices and Hardware

5. Visualize Results
   - 3rd party dashboards
   - Web apps

MATLAB EXPO 2018
The Need for Large-Scale Streaming

Predictive Maintenance
- Increase Operational Efficiency
- Reduce Unplanned Downtime

Jet engine: ~800TB per day
Turbine: ~ 2 TB per day

More applications require near real-time analytics

Medical Devices
- Patient Safety
- Better Treatment Outcomes

Connected Cars
- Safety, Maintenance
- Advanced Driving Features

Car: ~25 GB per hour
A quick Intro to Stream Processing

- **Batch Processing** applies computation to a finite sized historical data set that was acquired in the past

  - Historical Data
  - Configure Resources
  - Schedule and Run Job
  - Output Data

- **Stream Processing** applies computation to an unbounded data set that is produced continuously

  - Continuous Data
  - Messaging Service
  - Stream Analytics

MATLAB EXPO 2018
Why stream processing?

Near Real time decisions

Stream Processing with MATLAB Production Server

Big Data processing on historical data

Time critical decisions

Value of data to decision making

Preventive / Predictive

Actionable

Reactive

Today's example focuses here

Integrate with Production Systems

MATLAB Distributed Computing Server, MATLAB Compiler

Edge Processing with MATLAB Coder

C/C++

MATLAB EXPO 2018
Example Problem – How’s my driving?

- A group of MathWorks employees installed an OBD dongle in their car that monitors the on-board systems
- Data is streamed to the cloud where it is aggregated and stored
- We would like to use this data to score the driving habits of participants
Example: Fleet Analytics with MATLAB
Fleet Analytics Architecture

Edge Devices

API Gateway
AWS Lambda

Kafka Connector
kafka

Production System

MATLAB Production Server
MATLAB Analytics
Storage Layer

Analytics Development

MATLAB Compiler SDK
MATLAB
Algorithm Developers

Business Decisions

End Users

Business Systems
Power BI
Qlik
Tableau
Spotfire
The first step is to clean up the incoming data.
The Data: Timestamped messages with JSON encoding

```
{ 
  "vehicles_id": {"$oid":"55a3fd0069702d5b41000000"}, 
  "time": {"$date":"2015-07-13T18:01:35.000Z"}, 
  "kc": 1975.0, "kff1225": 100.65293, "kff125a": 110.36619, 
} 
```

```
{ 
  "vehicles_id": {"$oid":"55a3fe3569702d5c5c000020"}, 
  "time": {"$date":"2015-07-13T18:01:53.000Z"}, 
  "kc": 2000.0, "kff1225": 109.65293, "kff125a": 115.36619, 
} 
```

```
{ 
  "vehicles_id": {"$oid":"55a4193569702d115b000001"}, 
  "time": {"$date":"2015-07-12T19:04:04.000Z"}, 
  "kc": 2200.0, "kff1225": 112.65293, "kff125a": 112.36619, 
} 
```
Access a Sample of Data

1. Access and Explore Data

Raw Data

<table>
<thead>
<tr>
<th>timestamp</th>
<th>value</th>
<th>key</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-Jan-2015 22:12:23</td>
<td>&quot;55a41cb069702d115b059ee0&quot;</td>
<td>&quot;trip_id&quot;</td>
</tr>
<tr>
<td>15-Jan-2015 22:12:24</td>
<td>&quot;55a41cb069702d115b059ee1&quot;</td>
<td>&quot;trip_id&quot;</td>
</tr>
<tr>
<td>15-Jan-2015 22:12:25</td>
<td>&quot;55a41cb069702d115b059ee2&quot;</td>
<td>&quot;trip_id&quot;</td>
</tr>
<tr>
<td>15-Jan-2015 22:12:26</td>
<td>&quot;55a41cb069702d115b059ee3&quot;</td>
<td>&quot;trip_id&quot;</td>
</tr>
</tbody>
</table>

Timetable

<table>
<thead>
<tr>
<th>trip_id</th>
<th>VIN</th>
<th>kf1001</th>
<th>kf1005</th>
<th>kf1006</th>
<th>kf1120</th>
<th>kf1221</th>
<th>kf1222</th>
<th>kf1223</th>
<th>kf122a</th>
</tr>
</thead>
<tbody>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>17.100</td>
<td>-84.9323</td>
<td>45.4704</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>56.0434</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>17.100</td>
<td>-84.9322</td>
<td>45.4704</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>57.8609</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>18.9000</td>
<td>-84.9322</td>
<td>45.4704</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>51.1983</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>18.9000</td>
<td>-84.9322</td>
<td>45.4704</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>52.7147</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>58.6000</td>
<td>-84.9305</td>
<td>45.4886</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>73.2005</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>56.7000</td>
<td>-84.9304</td>
<td>45.4865</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>75.3612</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>57.6000</td>
<td>-84.9304</td>
<td>45.4863</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>70.7542</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>56.7000</td>
<td>-84.9303</td>
<td>45.4862</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>62.8340</td>
</tr>
</tbody>
</table>

✓ Decode JSON data
✓ Create Timetable
Develop a Preprocessing Function

Timetable

Preprocess data

\[
t = \text{sortrows}(t); \\
t = \text{rmmissing}(t,'MinNumMissing',\text{width}(t)-2);
\]

Perform windowed calculations

\[
t.\text{Speed} = \text{movmedian}(t.\text{SpeedGPS},3); \\
t.\text{D1} = [0;\text{diff}(t.\text{SpeedGPS})];
\]

\[
[t\text{min},t\text{max}] = \text{bounds}(t.\text{time}); \\
tnew = t\text{min}::\text{seconds}(10):t\text{max}; \\
countsByTime = \text{retime}(t(:,'\text{Event'}),tnew,@\text{histcounts});
\]
Ad Hoc Access to Data from MATLAB

Access the data in S3

Bring up the AthenaClient

```matlab
athenaClient = aws.athena.Client();
athenaClient.Database = 'trainingdata';
athenaClient.initialize();
```

Create a query and submit

```matlab
athenaClient.submitQuery('SELECT * FROM "trainingdata"."sampledata" limit 100', 's3://fleettrainingdata')
```

Fetch data as a table for easy analysis

```matlab
ds = datastore('s3://fleettrainingdata/*.csv');
ds.NumHeaderLines = 2;
data = table(ds);
```

Your usual MATLAB workflow goes here
Develop a Predictive Model

Production System

Analytics Development

Business Decisions

MATLAB EXPO 2018
Develop a Predictive Model in MATLAB
Why stream processing?

- Near Real-time decisions
- Time critical decisions
- Big Data processing on historical data

- Edge Processing with MATLAB Coder
- C/C++
- Stream Processing with MATLAB Production Server
- kafka
- Kinesis
- Today’s example focuses here

- Value of data to decision making
  - Preventive / Predictive
  - Actionable
  - Reactive
  - Historical

- Time
  - Real-Time
  - Seconds
  - Minutes
  - Hours
  - Days
  - Months

MATLAB Distributed Computing Server, MATLAB Compiler

Integrate with Production Systems

MATLAB EXPO 2018
Integrate Analytics with Production Systems

Integrate with Production Systems

Edge Devices

API Gateway

AWS Lambda

Kafka Connector

Production System

MATLAB Production Server

MATLAB Analytics

Storage Layer

Analytics Development

MATLAB Compiler SDK

Algorithm Developers

Business Decisions

Business Systems

End Users

MATLAB EXPO 2018
Streaming data is treated as an unbounded Timetable

Input Table

<table>
<thead>
<tr>
<th>Event Time</th>
<th>Vehicle</th>
<th>RPM</th>
<th>Torque</th>
<th>Fuel Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:01:10</td>
<td>55a3fd</td>
<td>1975</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>18:10:30</td>
<td>55a3fe</td>
<td>2000</td>
<td>109</td>
<td>115</td>
</tr>
<tr>
<td>18:05:20</td>
<td>55a3fd</td>
<td>1980</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>18:10:45</td>
<td>55a3fd</td>
<td>2100</td>
<td>110</td>
<td>100</td>
</tr>
<tr>
<td>18:30:10</td>
<td>55a419</td>
<td>2000</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>18:35:20</td>
<td>55a419</td>
<td>1960</td>
<td>103</td>
<td>105</td>
</tr>
<tr>
<td>18:20:40</td>
<td>55a3fe</td>
<td>1970</td>
<td>112</td>
<td>104</td>
</tr>
<tr>
<td>18:39:30</td>
<td>55a419</td>
<td>2100</td>
<td>105</td>
<td>110</td>
</tr>
<tr>
<td>18:30:00</td>
<td>55a3fe</td>
<td>1980</td>
<td>110</td>
<td>113</td>
</tr>
<tr>
<td>18:30:50</td>
<td>55a3fe</td>
<td>2000</td>
<td>100</td>
<td>110</td>
</tr>
</tbody>
</table>

MATLAB Function

Output Table

<table>
<thead>
<tr>
<th>Time window</th>
<th>Vehicle</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:00:00</td>
<td>55a3fd</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>55a3fe</td>
<td></td>
</tr>
<tr>
<td></td>
<td>55a419</td>
<td></td>
</tr>
<tr>
<td>18:10:00</td>
<td>55a3fd</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>55a3fe</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>55a419</td>
<td></td>
</tr>
<tr>
<td>18:20:00</td>
<td>55a3fd</td>
<td></td>
</tr>
<tr>
<td></td>
<td>55a3fe</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>55a419</td>
<td></td>
</tr>
<tr>
<td>18:30:00</td>
<td>55a3fd</td>
<td></td>
</tr>
<tr>
<td></td>
<td>55a3fe</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>55a419</td>
<td>8</td>
</tr>
</tbody>
</table>
Introducing MATLAB Production Server

Data

Databases
- DynamoDB
- MongoDB
- SQL Server
- Cassandra
- Cosmos DB

Cloud Storage
- Azure Blob
- AWS
- Kafka
- MQTT
- Azure IoT Hub

Streaming
- OSIsoft PI System
- AWS Kinesis

Analytics

MATLAB Production Server

Business System

Dashboards
- Qlik
- Tableau
- Microsoft Power BI
- Spotfire

Web
- Microsoft IIS
- Apache Tomcat
- WebSphere

Custom Apps
- Google Cloud Platform
- Azure
- Amazon Web Services
- Rackspace
- OpenStack
- VMware

Platform

MATLAB EXPO 2018
Develop and Deploy a Stream Processing Function
Debug a Stream Processing Function in MATLAB

Integrate with Production Systems

Edge Devices

Production System

Analytics Development

Kafka Connector

kafka

Storage Layer

MATLAB Compiler SDK

Algorithm Developers

MATLAB

Business Decisions

End Users

MATLAB EXPO 2018
Debug a Stream Processing Function in MATLAB
Tie in your Dashboard Application
Complete Your Application
Scalable Analytics with Enterprise BI Tools

TIBCO Spotfire

Tableau

MATLAB EXPO 2018
Key Takeaways

➢ MATLAB connects directly to your data so you can quickly design and validate algorithms

➢ The MATLAB language and apps enable fast design iterations

➢ MATLAB Production Server enables easy integration of your MATLAB algorithms with enterprise production systems

➢ You to spend your time understanding the data and designing algorithms
Resources to learn and get started

- Data Analytics with MATLAB
- MATLAB Production Server
- MATLAB Compiler SDK
- Statistics and Machine Learning Toolbox
- Database Toolbox
- Mapping Toolbox
- MATLAB with TIBCO Spotfire
- MATLAB with Tableau
- MATLAB with MongoDB