Integrating MATLAB Analytics into Enterprise Applications

Arvind Hosagrahara
Principal Engineer
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

- AWS
- Kinesis
- kafka
- OSIsoft
- Tableau
- Qlik
- Power BI
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
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

Production System

- MATLAB Production Server
- MATLAB Compiler SDK
- Storage Layer

Analytics Development

- MATLAB
- Algorithm Developers

Business Decisions

- Power BI
- Qlik Sense
- Spotfire
- Tableau
- Business Systems
- End Users
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

Raw Data

<table>
<thead>
<tr>
<th>timestamp</th>
<th>1 value</th>
<th>2 key</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-Jan-2015 22:12:23</td>
<td>'{ &quot;id&quot;: &quot;55a41cb069702d115b059ee0&quot;, &quot;trip_id&quot;: &quot;55a41cb069702d115b059ede&quot;}'</td>
<td></td>
</tr>
<tr>
<td>15-Jan-2015 22:12:24</td>
<td>'{ &quot;id&quot;: &quot;55a41cb069702d115b059ee1&quot;, &quot;trip_id&quot;: &quot;55a41cb069702d115b059ede&quot;}'</td>
<td></td>
</tr>
<tr>
<td>15-Jan-2015 22:12:25</td>
<td>'{ &quot;id&quot;: &quot;55a41cb069702d115b059ee2&quot;, &quot;trip_id&quot;: &quot;55a41cb069702d115b059ede&quot;}'</td>
<td></td>
</tr>
<tr>
<td>15-Jan-2015 22:12:26</td>
<td>'{ &quot;id&quot;: &quot;55a41cb069702d115b059ee3&quot;, &quot;trip_id&quot;: &quot;55a41cb069702d115b059ede&quot;}'</td>
<td></td>
</tr>
</tbody>
</table>

Timetable

<table>
<thead>
<tr>
<th>trip_id</th>
<th>VIN</th>
<th>ktt1001</th>
<th>ktt1005</th>
<th>ktt1006</th>
<th>ktt120</th>
<th>ktt1221</th>
<th>ktt1222</th>
<th>ktt1223</th>
<th>ktt125a</th>
</tr>
</thead>
<tbody>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>17.1000</td>
<td>-84.9323</td>
<td>45.4704</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>59.0434</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>17.1000</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>45.4705</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>18.9000</td>
<td>-84.9322</td>
<td>45.4705</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>18.0000</td>
<td>-84.9321</td>
<td>45.4706</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>49.1095</td>
</tr>
<tr>
<td>55a3fe356...</td>
<td>55a3fe356...</td>
<td>58.5000</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>45.4885</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>45.4883</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>45.4882</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>62.8340</td>
</tr>
</tbody>
</table>
Develop a Preprocessing Function

2 Preprocess Data

Timetable

<table>
<thead>
<tr>
<th>trip_id</th>
<th>VIN</th>
<th>HFT1001</th>
<th>HFT1005</th>
<th>HFT1006</th>
<th>HFT1220</th>
<th>HFT1221</th>
<th>HFT1222</th>
<th>HFT1230</th>
<th>HFT125a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Sun Jul 12</td>
<td>16:18:41</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>17.1000</td>
<td>-84.9323</td>
<td>45.4704</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>2 Sun Jul 12</td>
<td>16:18:42</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>17.1000</td>
<td>-84.9322</td>
<td>45.4704</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>3 Sun Jul 12</td>
<td>16:18:44</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>18.9000</td>
<td>-84.9322</td>
<td>45.4705</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>4 Sun Jul 12</td>
<td>16:18:45</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>18.9000</td>
<td>-84.9322</td>
<td>45.4705</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>5 Sun Jul 12</td>
<td>16:18:45</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>18.9000</td>
<td>-84.9321</td>
<td>45.4706</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>6 Sun Jul 12</td>
<td>16:18:45</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>18.9000</td>
<td>-84.9305</td>
<td>45.46**</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>7 Sun Jul 12</td>
<td>16:18:45</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>56.7000</td>
<td>-84.9304</td>
<td>45.46</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>8 Sun Jul 12</td>
<td>16:18:45</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>56.7000</td>
<td>-84.9304</td>
<td>45.46</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>9 Sun Jul 12</td>
<td>16:18:45</td>
<td>UTC 2015</td>
<td>55a3e356</td>
<td>55a3e356</td>
<td>66.7000</td>
<td>-84.9303</td>
<td>45.46</td>
<td>NaN</td>
<td>NaN</td>
</tr>
</tbody>
</table>

Preprocess data

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

Perform windowed calculations

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

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

Access and Explore Data

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

Data from Edge Devices flows through API Gateway and AWS Lambda to Kafka. Kafka then connects to the MATLAB Distributed Computing Server. Results are generated and passed to Business Systems via tools like Power BI, Qlik, and Tableau. End Users can access the insights generated by the system.
Everything you need to develop a predictive model is found in MATLAB

Label Events

Scale Up

Represent Signals

Train Model

Validate Model

Evaluate tall expression using the Spark Cluster:
- Pass 1 of 2: Completed in 11 sec
- Pass 2 of 2: Completed in 2.3333 min
Evaluation completed in 2.6167 min

Scale up:
```
tt = tall(data); % test tall array
model = TreeBagger(50,tt,'Event');
```

Scale to out of memory data:
```
tt = tall(ds);
tt = preprocessData(tt);
model = TreeBagger(50,tt,'Event');
save machineLearningModel model
```
Develop a Predictive Model in MATLAB
Integrate Analytics with Production Systems

4 Integrate with Production Systems

Production System

Kafka Connector

MATLAB Production Server

MATLAB Analytics

Storage Layer

Analytics Development

MATLAB Compiler SDK

Algorithm Developers

Business Decisions

Power BI
Qlik
Tableau

End Users

Business Systems

API Gateway
AWS Lambda
kafka

Edge Devices

AWS

Integrate Analytics with Production Systems
A quick Intro to Stream Processing

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

- **Stream Processing** applies computation to an unbounded data set that is produced continuously
Why stream processing?

- Why stream processing?
- MATLAB Distributed Computing Server, MATLAB Compiler
- Stream Processing with MATLAB Production Server
- Edge Processing with MATLAB Coder
- C/C++
- Near Real time decisions
- Time critical decisions
- Big Data processing on historical data
- Time
  - Time critical decisions
  - Big Data processing on historical data
  - Reactive
  - Preventive / Predictive
  - Actionable
  - Historical
  - Minutes
  - Hours
  - Days
  - Months
  - Seconds
  - Real-Time
- Today’s example focuses here
- MATLAB Distributed Computing Server, MATLAB Compiler
- Kinesis
- Event Hub
- Kafka
- Integrate with Production Systems
- Value of data to decision making
- Today’s example focuses here
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>

**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>18:10:00</td>
<td>55a3fe</td>
<td>...</td>
</tr>
<tr>
<td>18:10:00</td>
<td>55a419</td>
<td>...</td>
</tr>
<tr>
<td>18:10:00</td>
<td>55a419</td>
<td>...</td>
</tr>
<tr>
<td>18:20:00</td>
<td>55a3fd</td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>55a419</td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>55a3fe</td>
<td>4</td>
</tr>
<tr>
<td>18:30:00</td>
<td>55a419</td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>55a3fd</td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>55a3fd</td>
<td>5</td>
</tr>
<tr>
<td>18:30:00</td>
<td>55a419</td>
<td>8</td>
</tr>
</tbody>
</table>
Introducing MATLAB Production Server

4 Integrate with Production Systems

Data

- Databases
  - DynamoDB
  - Microsoft SQL Server
  - mongoDB
  - Cassandra
  - Cosmos DB

- Cloud Storage
  - Azure Blob

- Streaming
  - AWS Kinesis
  - Azure IoT Hub

Analytics

- MATLAB Production Server

- Request Broker

Business System

- Dashboards
  - Qlik
  - Tableau
  - Microsoft Power BI
  - Spotfire

- Web
  - Microsoft IIS
  - Apache Tomcat
  - WebSphere

- Custom Apps

Platform

- Google Cloud Platform
- Azure
- Amazon web services
- Rackspace
- openstack
- VMware
MATLAB Production Server is an application server that publishes MATLAB code as APIs.
Connecting MATLAB Production Server to Kafka

- Kafka client for MATLAB Production Server feeds topics to functions deployed on the server

- Configurable batch of messages passed as a MATLAB Timetable

- Each consumer process feeds one topic to a specified function

- Drive everything from a simple config file
  - No programming outside of MATLAB!
Develop and Deploy a Stream Processing Function

Production System

Analytics Development

Integrate with Production Systems

Edge Devices

API Gateway
AWS Lambda
Kafka
kafka
Storage Layer

MATLAB Production Server
MATLAB Compiler SDK
MATLAB
Algorithm Developers

Business Decisions

End Users

Power BI
Qlik
Spotfire

Business Systems

4
Develop a Stream Processing Function in MATLAB

Process each window of data as it arrives

Current score

Previous state

Current window of data to be processed
Develop a Stream Processing Function in MATLAB

Apply your pre-processing algorithm
Integrate with Production Systems

Develop a Stream Processing Function in MATLAB

Use the model you created with Classification Learner App
Develop a Stream Processing Function in MATLAB

```matlab
function new_state = calculateScores(car_id, current_data, old_state, resultsStore)

Preprocess and perform calculations
current_data = preprocessData(current_data);

Predict driving events
current_data = predictEvents(current_data);

Count events for each ten second window
countsByTime = countEvents(current_data);

Write discrete data to mongodb
updateResultsStore(car_id, countsByTime, resultsStore);

Update new state
new_state = updateState(countsByTime, old_state);
end
```

Update Mongo database
- Count of events by type and location
- Results of driver scoring
Debug a Stream Processing Function in MATLAB
Debug a Stream Processing Function in MATLAB
4. Integrate with Production Systems

Tie in your Dashboard Application

Production System

Kafka Connector

MATLAB Production Server

MATLAB Analytics

Storage Layer

Analytics Development

MATLAB Compiler SDK

Algorithm Developers

Business Decisions

End Users

API Gateway

AWS Lambda

kafka

Storage Layer

Business Systems

Power BI

Qlik

Spotfire

Tableau

Edge Devices

API Gateway

AWS Lambda

kafka

Production System

Analytics Development

Business Decisions

End Users
Complete Your Application

Acceleration/Deceleration Events, 2014 - 2017

Aggressive acceleration: 94293 Events
Spreading: 2034 Events
Moderate deceleration: 1129 Events
Safe deceleration: 1109 Events
Moderate acceleration: 550 Events
Safe acceleration: 515 Events
Aggressive deceleration: 2903 Events

Fleet Summary

Automatic update:

Enabled

Fleet Statistics

Total Events:
1323351
Scalable Analytics with Enterprise BI Tools

TIBCO Spotfire

Tableau
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