MATLAB EXPO 2019

Despliegue de Inteligencia Artificial para decisiones de fabricación cercanas al tiempo real

Lucas García
The Need for Large-Scale Streaming

Predictive Maintenance

*Increase Operational Efficiency*
*Reduce Unplanned Downtime*

More applications require near real-time analytics

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

Medical Devices

*Patient Safety*
*Better Treatment Outcomes*

Connected Cars

*Safety, Maintenance*
*Advanced Driving Features*

Car: ~25 GB per hour
Example Problem: Develop a machine learning model to predict failures in industrial pumps

- We did this for the customer
- We wanted to go further:
  - Create a streaming application based on this real customer request
  - Develop application in a 3-4 week sprint
- We believe this represents a realistic customer situation
Our Project: Develop and operationalize a machine learning model to predict failures in industrial pumps

- **Process Engineer**: Develops models in MATLAB and Simulink
- **System Architect**: Deploys and operationalizes model on Azure cloud
- **Operator**: Makes operational decisions based on model output

Current system requires Operator to manually monitor operational metrics for anomalies. Their expertise is required to detect and take preventative action.
**Project statement**: Develop end-to-end predictive maintenance system and demo in one 3-4 week sprint

1. Monitor *flow, pressure*, and *current* of each pump so I always know their *operational state*

2. Need *alert* when fault parameters drift outside an acceptable range so I can take *immediate corrective action*

3. Continuous estimate of each pump’s *remaining useful life (RUL)* so I can *schedule maintenance or replace* the asset
Challenges of AI Deployment

We don’t have a large set of failure data, and it’s too costly to generate real failures in our plant for this project.

Solution: Use an accurate physics-based software model for the pump to develop synthetic training sets.
Challenges of AI Deployment

We don’t have a large IT/hardware budget, and we need to see results before committing to a particular platform or technology.

Solution: Leverage cloud platform to quickly configure and provision the services needed to build the solution, while minimizing lock-in to a particular provider.
Challenges of AI Deployment

Need software for multidisciplinary problem across teams, plus integration w/ IT

Solution: Use MATLAB and integrate with OSS
Predictive Maintenance Architecture on Azure

**Edge**
- Generate telemetry

**Production System**
- Azure
- MATLAB Production Server
  - Worker processes
  - Request Broker
- Apache Kafka
- State Persistence
- Storage Layer

**Analytics Development**
- MATLAB Compiler SDK
- MATLAB
  - Debug
  - Package & Deploy
  - Model

**Business Decisions**
- kibana
  - Presentation Layer

**System Architect**
- Operator

**Process Engineer**
Modeling approach

Process Engineer

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
Review model requirements

- Continuous predictions of type of fault
  - “Blocking”
  - “Leaking”
  - “Bearing”
  - Combination of above
- Continuous predictions of Remaining Useful Life [RUL]

Requirements From Operator

Requirements From System Architect

- Define window for streaming
- Define format of results, intermediate values
- Test code
- Scale code
Physics of Triplex Pump

- Crankshaft drives three plungers
  - Each 120 degrees out of phase
  - One chamber always discharging
  - Three types of failures

![Diagram of a triplex pump](image)

- Algorithm
- Failure Diagnosis

**Physiological Flow Rate at Output**

- Plunger 1
- Plunger 2
- Plunger 3
- Pump

- Outlet
- Leak Area
- Inlet

**Components**
- Crankshaft
- Bearing Friction
- Blocking Fault
Use sensor data from pump to identify levels of failure

Simulate faults

Pump sensor data
Build digital twin and generate sensor data
Simulate data with many failure conditions

Leak Area = [1e-9  0.036]

Bearing Friction = [0  6e-4]

Blocking Fault = [0.5  0.8]
Simulate data with many failure conditions

**Access Data**

```python
ens = simulationEnsembleDatastore(location)
```

- `ens = simulationEnsembleDatastore` with properties:
  - `DataVariables`: [25x1 string]
  - `IndependentVariables`: [0x0 string]
  - `ConditionVariables`: [0x0 string]
  - `SelectedVariables`: [25x1 string]
  - `ReadSize`: 1
  - `NumMembers`: 702
  - `LastMemberRead`: [0x0 string]
  - `Files`: [702x1 string]

**Process Engineer**

Run parallel simulations

Store data on HDFS
Represent signal information

Signal processing

```matlab
[Spectrum, Frequencies] = psspectrum(data.Flow);
[plow, pH] = bounds(Spectrum);
fPeak = Frequencies(Spectrum==pH);
qPeak2Peak = peak2peak(data.Flow);
qCrest = peak2rms(data.Flow);
qRMS = rms(data.Flow);
qMAD = mad(data.Flow);
```
Develop Predictive Models in MATLAB

Process Engineer

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>LeakFault</th>
<th>BlockingFault</th>
<th>BearingFault</th>
<th>FaultType</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.8472</td>
<td>-0.1477</td>
<td>1.8000</td>
<td>All</td>
</tr>
<tr>
<td>2</td>
<td>-0.1498</td>
<td>-0.4207</td>
<td>1.3103</td>
<td>Bearing &amp; Blocking</td>
</tr>
<tr>
<td>3</td>
<td>0.6511</td>
<td>1.6521</td>
<td>-0.5357</td>
<td>Leak</td>
</tr>
<tr>
<td>4</td>
<td>0.1469</td>
<td>-0.2775</td>
<td>1.0074</td>
<td>All</td>
</tr>
<tr>
<td>5</td>
<td>-0.6480</td>
<td>0.7065</td>
<td>-0.8876</td>
<td>Blocking</td>
</tr>
<tr>
<td>6</td>
<td>-0.8165</td>
<td>-0.5434</td>
<td>-0.3079</td>
<td>Blocking</td>
</tr>
<tr>
<td>7</td>
<td>-1.0061</td>
<td>1.2083</td>
<td>0.0661</td>
<td>Bearing</td>
</tr>
<tr>
<td>8</td>
<td>1.0125</td>
<td>-1.9098</td>
<td>-0.7027</td>
<td>Leak &amp; Blocking</td>
</tr>
</tbody>
</table>

Label Faults

Scale

```
tt = tall(ds);
tt = preprocessData(tt);
model = TreeBagger(50,tt,'Event');
```

Validating 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
Develop Predictive Models in MATLAB

3. Develop Predictive Models

Type of Fault (Classification)

Remaining Useful Life (Regression)

Fault Classification

Estimated Remaining Useful Life ~ 18 hrs

Real Data

Failure Threshold

Forecast Data

Plant Operator

Process Engineer
Develop Machine Learning Models
Estimate Remaining Useful Life

\[ S(t) = \phi + \theta(t) e^{(\beta(t)t+\epsilon(t) - \frac{\sigma}{2})} \]
Develop a Stream Processing Function

**Batch Processing:** Build and test model on simulated data

**Stream Processing:** Apply model to sensor data in near real-time
Develop a Stream Processing Function

Streaming Function

```matlab
function new_state = streamingFunction(data, old_state)

Preprocess signals
[data, features] = preprocessData(data);

Predict faults
[Leak, Blocking, Bearing] = predictFaultValues(features);
FaultType = predictFault(features);
[RUL, Model] = predictUpdateRUL(data.Timestamp, data.Flow, 500);

Update state
new_state = updateState(data, old_state);

Write results
writeResults(Leak, Blocking, Bearing, FaultType, RUL, Model)
end
```

Process each window of data as it arrives

Previous state

Current window of data to be processed
Test Stream Processing Function

```
results = runtests('predictFaults_tests')

Running predictFaults_tests
....
Done predictFaults_tests

results =
1x4 TestResult array with properties:
Name  Passed  Failed  Incomplete  Duration  Details
Totals: 4 Passed, 0 Failed, 0 Incomplete. 0.01614 seconds testing time.
```
Share with the team

Process Engineer

4 Integrate with Production Systems

Review results with Operator

Share code with System Architect

Source Control

.pdf, html, LaTeX
Package Stream Processing Function

Integrate with Production Systems

Process Engineer
Review System Requirements

- Requirements from the Process Engineer
  - Every millisecond, each pump generates a time-stamped record of flow, pressure, and current
  - Model expects 1 sec. window of data per pump
  - Initially, 1’s – 10’s of devices, but quickly scale to 100’s

- Requirements from the Operator
  - Alerts when parameters drift outside the expected ranges
  - Continuous estimating of RUL for each pump
Integrate Analytics with Production Systems

Production System

- Apache Kafka
- Connector
- MATLAB Production Server
  - Worker processes
  - Request Broker
- State Persistence
- Storage Layer

Analytics Development

- MATLAB Compiler SDK
- Debug
- Package & Deploy
- Model

Business Decisions

- Presentation Layer
- kibana

System Architect

Generate telemetry

Edge

4

Integrate with Production Systems

MATLAB Production Server
MATLAB Production Server on Azure

Integrate with Production Systems

Production System

Virtual Network

Management Server

MATLAB Production Server(s) scaling group

Application Gateway Load Balancer

State Persistence

Connectors for Storage & Databases

Connectors for Streaming/Event Data

Enterprise Applications

https management endpoint

Integrate with Production Systems
Connecting MATLAB Production Server to Kafka

- Connector feeds single Kafka topic to a MATLAB function
- Publisher library for MATLAB for writing to a results stream

**Connector Features:**
- Deploy as a micro-service with Docker
- Drive everything through config
- Group data into time windows and pass to MATLAB as a timetable
- Use Kafka’s check-pointing (i.e. at-least-once)
Messaging adapter for Production Server

- Bridges streaming data and Production Server Async Java Client
- Batches incoming messages and sends them via HTTP request/response
  - Time windows, event time processing, and out-of-order data
- Uses Asynchronous pipeline model with back-pressure
  - Kafka consumers are automatically paused when server is busy
- Supports sequential (stateful) and unordered (stateless) processing
  - Provide unique stream ID/topic/partition info for persistence layer
- Pass data as MATLAB timetables
- Partition aware – enables full exploitation of partition-based parallelism
Kafka connector architecture

**Message State (Offsets, Timestamps, Watermarks)**

**Active Windows**

- $P_0$
- $C_0$
- $W_n$, $W_1$, $W_0$
- $P_1$
- $C_1$
- $W_n$, $W_1$, $W_0$
- $P_n$
- $C_n$
- $W_n$, $W_1$, $W_0$

**Async Request Handler**

- $P_0$
- $P_1$
- $P_n$
- $r_0$
- $r_1$

**Kafka**

- Topic
- $P_0$
- $P_1$
- $P_n$

**Consumer Thread Pool**

- Consumer Thread Pool
- $C_0$
- $C_1$
- $C_n$

**Committed Offsets**

- $P_0$
- $P_1$
- $P_n$

**Async HTTP to Server**

**Production Server Java Client**

**Networking Threads**
Streaming data is treated as an unbounded Timetable

**Input Stream**

<table>
<thead>
<tr>
<th>Event Time</th>
<th>Pump Id</th>
<th>Flow</th>
<th>Pressure</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:01:10</td>
<td>Pump1</td>
<td>1975</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>18:10:10</td>
<td>Pump3</td>
<td>2000</td>
<td>109</td>
<td>115</td>
</tr>
<tr>
<td>18:05:20</td>
<td>Pump1</td>
<td>1980</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>18:10:45</td>
<td>Pump2</td>
<td>2100</td>
<td>110</td>
<td>100</td>
</tr>
<tr>
<td>18:30:10</td>
<td>Pump4</td>
<td>2000</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>18:35:20</td>
<td>Pump4</td>
<td>1960</td>
<td>103</td>
<td>105</td>
</tr>
<tr>
<td>18:20:40</td>
<td>Pump3</td>
<td>1970</td>
<td>112</td>
<td>104</td>
</tr>
<tr>
<td>18:39:30</td>
<td>Pump4</td>
<td>2100</td>
<td>105</td>
<td>110</td>
</tr>
<tr>
<td>18:30:00</td>
<td>Pump3</td>
<td>1980</td>
<td>110</td>
<td>113</td>
</tr>
<tr>
<td>18:30:50</td>
<td>Pump3</td>
<td>2000</td>
<td>100</td>
<td>110</td>
</tr>
</tbody>
</table>

**Output Stream**

<table>
<thead>
<tr>
<th>Time window</th>
<th>Pump Id</th>
<th>Bearing Friction</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:00:00</td>
<td>Pump1</td>
<td>5</td>
</tr>
<tr>
<td>18:10:00</td>
<td>Pump3</td>
<td>...</td>
</tr>
<tr>
<td>18:10:00</td>
<td>Pump4</td>
<td>...</td>
</tr>
<tr>
<td>18:20:00</td>
<td>Pump2</td>
<td>7</td>
</tr>
<tr>
<td>18:20:00</td>
<td>Pump3</td>
<td>3</td>
</tr>
<tr>
<td>18:20:00</td>
<td>Pump4</td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>Pump1</td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>Pump3</td>
<td>4</td>
</tr>
<tr>
<td>18:30:00</td>
<td>Pump4</td>
<td>...</td>
</tr>
</tbody>
</table>

**System Architect**

Integrate with Production Systems

MATLAB Function

State

Input Stream

Output Stream

Streaming data is treated as an unbounded Timetable

**MATLAB EXPO 2019**

35
Debug your streaming function on live data
Debug a Stream Processing Function in MATLAB
Complete your application
Complete Your Application

Plant Operator

Visualize Results

Pump Summary

MathWorks®

Pump Selection

Pump List

Add a filter

Clear form

Cancel changes

Apply changes

Fault events

46 Fault events

Fault events

Leakage Analysis

- 0.05
- 0.66
- 0.67
- 0.70
- 0.69
- 0.71
- 0.72
- 0.73

Pump Average results

Max Bearing

Average Spacing

Average Leak

Timestamp date ranges

October 28th 2018 17:00:00:000 to November 8th 2018 22:52:59.697

Current / Energy consumption

Pressure

Leakage per Day

Remaining Useful Life

MATLAB EXPO 2019
Team Retrospective

- Completed demo of full system in 3 week sprint
- Successfully used digital twin to generate faults and train models
- Fast prototyping of physical and AI models with MATLAB and Simulink. Easy integration with OSS
- Cloud platform enabled faster IT setup

Next steps:
- Make model adjustments
- Test against real pump
- Customize dashboard for Operator’s needs
Resources to learn and get started

- GitHub: MathWorks Reference Architectures
- Working with Enterprise IT Systems
- Data Analytics with MATLAB
- Simulink