제조 생산 현장에서 관리 시스템까지
빠른 인공지능 기반 시스템 구축

엄준상
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

MATLAB EXPO 2019
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

Process Engineer

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
- Connector
- State Persistence
- Storage Layer

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

**Business Decisions**
- kibana
- Presentation Layer

**Other Roles**
- Process Engineer
- System Architect
- Operator
Modeling approach

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

Process Engineer
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
Use sensor data from pump to identify levels of failure

Simulate faults

Pump sensor data
Build digital twin and generate sensor data
Access and explore 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

```
ens = simulationEnsembleDatastore(location)
ens =
    simulationEnsembleDatastore with properties:
    DataVariables: [25x1 string]
    IndependentVariables: [8x0 string]
    ConditionVariables: [8x0 string]
    SelectedVariables: [25x1 string]
    ReadSize: 1
    NumMembers: 702
    LastMemberRead: [8x0 string]
    Files: [702x1 string]
```

Run parallel simulations

Store data on HDFS

MATLAB EXPO 2019
Represent signal information

Signal processing

```matlab
[Spectrum, Frequencies] = pspectrum(data.Flow);
[pLow, pHigh] = bounds(Spectrum);
fPeak = Frequencies(Spectrum == pHigh);
quPeak2Peak = peak2peak(data.Flow);
quCrest = peak2rms(data.Flow);
quRMS = rms(data.Flow);
quMAD = mad(data.Flow);
```
Develop Predictive Models in MATLAB

Process Engineer

Label Faults

Scale

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

Evaluating 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

MATLAB EXPO 2019
Develop Predictive Models in MATLAB

Type of Fault (Classification)

Remaining Useful Life (Regression)
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

  - Historical Data → Train Model → Scale Up → Predictions
    - Files
    - Storage
    - Train Model
    - Scale Up
    - Predictions: Storage, Dashboard

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

  - Continuous Data → Messaging Service → Stream Processing Function
    - Pump Sensor Data
    - Messaging Service
    - Stream Processing Function: $f(x)$
    - Update State
    - Make Decisions: Alerts, Dashboard, Storage
Develop a Stream Processing Function

Streaming Function

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
Test Stream Processing Function

```matlab
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

- Share code with System Architect
- Review results with Operator

Integrate with Production Systems

Process Engineer

.pdf, html, LaTeX

Source Control
Integrate with Production Systems

Process Engineer

Package Stream Processing Function
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

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

Analytics Development

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

Business Decisions

- kibana
- Presentation Layer

System Architect

- Generate telemetry
- Edge
MATLAB Production Server on Azure

Integrate with Production Systems

System Architect

Production System

Azure

MATLAB Production Server(s) scaling group

Virtual Network

Management Server

https management endpoint

Enterprise Applications

Connectors for Streaming/Event Data

Connectors for Storage & Databases

State Persistence

Application Gateway Load Balancer
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

Kafka

Consumer Thread Pool

Active Windows

Message State (Offsets, Timestamps, Watermarks)

Async Request Handler

Production Server

Java Client

Networking Threads

Async HTTP to Server

System Architect

Integrate with Production Systems

MATLAB EXPO 2019

34
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>18:10:30</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>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</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>...</td>
<td>...</td>
</tr>
<tr>
<td>18:10:00</td>
<td>Pump1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Pump3</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Pump4</td>
<td>...</td>
</tr>
<tr>
<td>18:10:00</td>
<td>18:20:00</td>
<td>Pump2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pump3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pump4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>18:20:00</td>
<td>18:30:00</td>
<td>Pump1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pump3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pump4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>18:40:00</td>
<td>Pump5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pump3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pump4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
Debug your streaming function on live data

Integrate with Production Systems

4

System Architect

Production System

Analytics Development

MATLAB Production Server
Worker processes
Broker

MATLAB Compiler SDK
Debug
Package & Deploy
Mode

Apache Kafka
Connector

State Persistence
Storage Layer

Business Decisions

Presentation Layer

kibana
Debug a Stream Processing Function in MATLAB

Integrate with Production Systems

System Architect

```matlab
% function nlearnapp.internal.model.adapterlayer.TrainedClassificationModel
function fnc = trainModel(TrainData)
    % Train the model
    fnc = fit(X, y, 'svm', 'kernel', 'linear', 'boxconstraint', 1);
end

% Model doesn't work well with too few data points.
limit = 100;
if height(data) < limit
    fprintf('Too few rows (\textless \textless limit\textgreater \textgreater) to generate effective model.\n', ... height(data), limit);
    new_state = old_state;
else
    % Get the keys present in the data, use categorical for performance
    data_key = categorical(data_key);
    input = string(categories(data_key));
    % Load models
    persistent leakModel bearingModel blockingModel trainedModel
    if isempty(leakModel)
        x = load('leakModels.mat');
        leakModel = x.leakModel;
        bearingModel = x.bearingModel;
        blockingModel = x.blockingModel;
        trainedModel = x.trainedModel;
    else
```
Complete your application
Complete Your Application
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