Prognostics and Health Management in the Cloud: An Introduction

José Celaya and Indranil Roychoudhury

Machine Learning, Software Technology Innovation Center, Schlumberger

September 26, 2019
Outline

- Why you would like to learn about PHM in the Cloud
- Introduction to Google cloud and its benefits
- Walkthrough of GCP related tools
- Demonstration of a model-based PHM Example on the cloud
- Closing remarks
Motivation for PHM

- All engineered systems will eventually degrade or fail.
- Maintenance is key to increase uptime and safety, arrange for spare parts, reduce loss of life and property, and minimize maintenance costs.
- Types of Maintenance
  - Reactive maintenance
  - Scheduled maintenance
  - Predictive maintenance, a.k.a. prognostics and health management (PHM)
Motivation for this Tutorial

1. Share our experience using Cloud as a development environment
2. Invite PHM Society audience to see Cloud as a tool that is here to stay and key for all to know how to use to remain relevant
3. Expand your engineering and scientific ambitions by seeing what is possible using cloud today and what is coming in the future
4. Share an example of the many possibilities available to you to deploy a production ready system in a cloud environment
5. Give you an idea of what a modern data analytics development environment looks like and relate to:
   a. PHM data driven development
   b. PHM physics modeling development
   c. Hybrid use of data, physics, operational and other subject matter expert knowledge
Some Things That We Will Not Cover

- Cloud cyber-security concerns the audience might have
  - Not the intention of this talk but we can discuss offline
  - Advising you on how to work with your company policies related data security for PHM

- Topics related to IIoT/Edge in the cloud from the pure IIoT side
  - Our demo works with IIoT aspects but we will not cover here
  - There is a world of new developments at industrial grade and lots of promise for PHM

- We care about science and engineering in PHM and all the information here is towards that
  - Cannot advise you on how to architect your company or individual solution in this tutorial (Software Engineering side)
  - But we can certainly discuss offline or at our discretion if it enriches the tutorial experience
Google Cloud Walkthrough

Sources of Information:

- https://cloud.google.com/
- Console: https://console.cloud.google.com
- Docs: https://cloud.google.com/docs/
Everything You Need To Build And Scale

**Compute**
From virtual machines with proven price/performance advantages to a fully managed app development platform.
- Compute Engine
- App Engine
- Container Engine
- Container Registry
- Cloud Functions

**Storage and Databases**
Scalable, resilient, high performance object storage and databases for your applications.
- Cloud Storage
- Cloud Bigtable
- Cloud Datastore
- Cloud SQL
- Cloud Spanner

**Networking**
State-of-the-art software-defined networking products on Google's private fiber network.
- Cloud Virtual Network
- Cloud Load Balancing
- Cloud CDN
- Cloud Interconnect
- Cloud DNS

**Management Tools**
Monitoring, logging, and diagnostics and more, all a easy to use web management console or mobile app.
- Stackdriver Overview
- Monitoring
- Logging
- Error Reporting
- Debugger
- Deployment Manager & More

**Big Data**
Fully managed data warehousing, batch and stream processing, data exploration, Hadoop/Spark, and reliable messaging.
- BigQuery
- Cloud Dataflow
- Cloud Dataproc
- Cloud Dataprep
- Cloud Datalab
- Cloud Pub/Sub
- Genomics

**Machine Learning**
Fast, scalable, easy to use ML services. Use our pre-trained models or train custom models on your data.
- Cloud Machine Learning Platform
- Vision API
- Video Intelligence API
- Speech API
- Translate API
- NLP API

**Developer Tools**
Develop and deploy your applications using our command-line interface and other developer tools.
- Cloud SDK
- Deployment Manager
- Cloud Source Repositories
- Cloud Endpoints
- Cloud Tools for Android Studio
- Cloud Tools for IntelliJ
- Google Plugin for Eclipse
- Cloud Test Lab
- Cloud Container Builder

**Identity & Security**
Control access and visibility to resources running on a platform protected by Google's security model.
- Cloud IAM
- Cloud IAP
- Cloud KMS
- Cloud Resource Manager
- Cloud Security Scanner
- Cloud Platform Security Overview
How to unlock the value of data

1. Rehost existing databases
2. Fully managed databases
3. Serverless data & analytics
4. AI and machine learning
Range of fully managed databases

<table>
<thead>
<tr>
<th></th>
<th>Cloud SQL</th>
<th>Datatstore</th>
<th>Bigtable</th>
<th>Spanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full managed</td>
<td>MySQL, PostgreSQL</td>
<td>NoSQL document database for mobile &amp; web apps</td>
<td>Wide-column database with HBase API</td>
<td>Mission-critical relational database with transactional consistency, global scale, and high availability</td>
</tr>
<tr>
<td>Coming soon with</td>
<td>Microsoft SQL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A comprehensive platform

**Ingestion**
- Cloud Pub/Sub
- Data Transfer Service
- Cloud IoT Core
- Storage Transfer Service

**Pipeline**
- Cloud Dataflow
- Cloud Dataproc
- Cloud Dataprep
- Apache Beam

**Warehousing**
- BigQuery
- Cloud Storage

**Analytics**
- Cloud AI Services
- Google Data Studio
- Tensorflow
- Sheets

**Tools**
- Data Fusion
- Data Catalog
- Cloud Composer
Google BigQuery

Google Cloud Platform’s **enterprise data warehouse** for analytics

- **Exabyte-scale** storage and petabyte-scale SQL queries
- **Encrypted**, durable, and highly available

- Fully managed and **serverless**
- **Real-time** analytics on streaming data
- Built-in **ML and GIS**
- High-speed, **in-memory BI Engine**
Cloud AI Solutions

Ready-to-deploy AI solutions to plug into your existing technology and workflows

Cloud AI Builder Tools

Tools, services, and APIs that make it easy for developers to build AI-enabled systems
Cloud AI builder tools

High Level of ML Expertise

ML Infrastructure

AI Platform

Cloud AutoML

API - Pre-trained models

Low Level of ML Expertise

AI Hub
Powered by Open source
Cloud TPUs
Hardware acceleration for AI

Cloud TPU v3
Now generally available (GA)

Cloud TPU v2 Pod
100 petaflops, now in early Alpha

Most Accessible Scale for ML
27X faster and 38% cheaper as measured by MLPerf benchmark

Growing Software Ecosystem
PyTorch, TF 2.0, Kubernetes Engine, Deep Learning VMs, reference models, and more
AutoML and APIs
Making ML accessible to all developers

Sight
- Cloud Vision
- Cloud Video Intelligence
- AutoML Vision
- AutoML Video Intelligence

Language
- Cloud Translation
- Cloud Natural Language
- AutoML Translation
- AutoML Natural Language

Conversation
- Dialogflow Enterprise Edition
- Cloud Text-to-Speech
- Cloud Speech-to-Text

Structured Data
- AutoML Tables
- Recommendation AI
What makes Google Cloud different

- Best-in-class Security: Protect systems, data, and users
- Hybrid & Multi-Cloud: Enables choice
- Fully Managed No Ops: Ease of use with serverless
- Embedded AI & ML: Intelligence in everything
- Best of Google: Bringing culture of innovation to customers and partners
Physics-Based PHM in the Cloud: An Example
Facts About Our Tutorial

Our intention is NOT to give you code that you can copy and paste and the run your own cloud PHM solution

Our intention is NOT to make you an expert on cloud development

Our intention is to start opening the eyes to this community, that cloud is real, it is here to stay and more importantly is the not only the way of the future, it is the “now”.

- If you are student, this is what you want to learn
- If you are a mid-career person, and you want to expand your expertise, this is what you would like to do
- Project manager - the same
Designing/Transitioning Custom PHM Applications in the Cloud

● Design the application as a collection of cloud services, or APIs
  ○ Expose underlying functions as services that can be leveraged independently
  ○ Combine services into composite services or applications
  ○ These services are “stateless”
  ○ Read in and return information in JSON format

● Decouple the data from the application
  ○ Can store and process data on any public or private cloud instance
  ○ Helps with performance, as database reads/writes have latency

● Consider communications between application components
  ○ Optimize communications between application components as communication over internet introduces latency
  ○ E.g., combine communications into a single stream of data or a group of messages, rather than constantly communicating as if the application components reside on a single platform

● Model and design for performance and scaling
  ○ In some instances, cloud services provide auto scaling capabilities
  ○ Orchestration platforms such as Kubernetes can help with this as well
    ■ Automatic deployment, scaling, and management of containerized applications

Ref: https://techbeacon.com/enterprise-it/5-steps-building-cloud-ready-application-architecture
PHM Basics

- PHM consists of 5 steps:

  - Anomaly Detection: Is something wrong with my system?
  - Fault Isolation: What is wrong with my system?
  - Degradation Quantification: How bad is the damage?
  - Remaining Useful Life Prediction: How much time do I have to do something about the fault before the system loses functionality?
  - Decision Making: What can be done to mitigate the effects of the fault?

- Data-driven and physics-based approaches can be used to answer each of the 5 questions
## State-of-the-art in PHM (Notional)

<table>
<thead>
<tr>
<th></th>
<th>Physics-based Approaches</th>
<th>Data-driven Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomaly Detection</td>
<td>Anomaly Detection (e.g., Z-test)</td>
<td>Anomaly Detection (e.g., single-class classifiers)</td>
</tr>
<tr>
<td>Fault Isolation</td>
<td>Fault Isolation (e.g., filtering or state-estimation + search)</td>
<td>Fault Isolation (e.g., multi-class classification)</td>
</tr>
<tr>
<td>Degradation Quantification</td>
<td>Degradation Quantification (e.g., filtering or state-estimation)</td>
<td>Degradation Quantification (e.g., regression, object detection)</td>
</tr>
<tr>
<td>RUL Prediction</td>
<td>RUL Prediction (e.g., simulation, open-loop filtering - only predict, no update)</td>
<td>RUL Prediction (e.g., future predictions, CNNs)</td>
</tr>
<tr>
<td>Decision Making</td>
<td>Decision Making (e.g., Partially observable Markov Decision Process)</td>
<td>Decision Making (e.g., Reinforcement Learning)</td>
</tr>
</tbody>
</table>

### Data Requirement
- **Low**
- **High**

### Algorithm Complexity
- **Low**
- **High**

### Maturity
- **High**
- **Low**

---

Schlumberger Private
Case Study — A 3-Pump IIoT Testbed

- **The System**
  - 3 DC Motor Pumps
  - 3 Flow meters
- **Inputs:**
  - Controlled Pump Speed for each pump
- **Outputs:**
  - Flow rate out of each pump
- **Faults:**
  - Loss of efficiency in each of the three pumps
  - Faults are single and persistent
- **End-of-life condition**
  - When the output flow of any pump dips below 0.15 units
Our Physics-Based PHM Architecture

- **Fault Detection**
- **Fault Isolation**
- **Fault Identification**
- **RUL Prediction**

Diagram:
- Input: $u(k)$
- Output: $y(k)$, $\hat{y}(k)$
- System
- Faulty Model Observer

Mathematical Notations:
- $p(x_f(k), \theta_f(k)|y(0:k)) : f \in F(k)$
- $p(\text{EOL}_f(k)|y(0:k)) : f \in F(k)$
- $p(\text{RUL}_f(k)|y(0:k)) : f \in F(k)$
Residual Generation for Fault Detection

- Observer based on nominal model estimates nominal behavior as a reference
  - E.g., Kalman filter, particle filter
  - Uses state-space model of nominal system
  - Predict and update steps

**Nominal System**

### State transition model

```python
def state_transition_nominal(x, dt, params, inputs):
    dt = dt * 1.15e-2
    i1, w1, i2, w2, i3, w3 = x
    L1, R1, k1, J1, B1, L2, R2, k2, J2, B2, L3, R3, k3, J3, B3 = params
    Vs1, Vs2, Vs3 = inputs
    di1 = (1/L1) * (Vs1 - R1 * i1 - k1 * w1) * dt + i1
    dw1 = (1/J1) * (k1 * i1 - B1 * w1) * dt + w1
    di2 = (1/L2) * (Vs2 - R2 * i2 - k2 * w2) * dt + i2
    dw2 = (1/J2) * (k2 * i2 - B2 * w2) * dt + w2
    di3 = (1/L3) * (Vs3 - R3 * i3 - k3 * w3) * dt + i3
    dw3 = (1/J3) * (k3 * i3 - B3 * w3) * dt + w3
    return [di1, dw1, di2, dw2, di3, dw3]
```

### Observation model

```python
def observation_model_nominal(x, dt, params, inputs):
    return [x[1], x[3], x[5]]
```
Fault Detection and Symbol Generation

- Residual = observed sensor value - estimated sensor values
  - Nominally residual is approximately zero
- Fault detected when residual deviation from zero to statistically significant
- Usually there is a delay between fault occurrence and fault detection
  - This delay is typically not avoidable

Once fault detected, measurements $\Rightarrow$ symbols
  - 0 (at nominal), + (above nominal), - (below nominal)
Fault Detection and Symbol Generation — 3-Pump IIoT Testbed
Fault Isolation - Matching Residual Symbols with Fault Signatures

- Fault signatures are **qualitative** predictions of how residuals will change in response to a fault
  - SMEs can help generate fault signatures
  - Information also captured in nominal system model
  - Can be generated using simulation

<table>
<thead>
<tr>
<th>Fault \ Measurement</th>
<th>Pump 1 Flow</th>
<th>Pump 2 Flow</th>
<th>Pump 3 Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pump 1 Degrading</td>
<td>-0</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>Pump 2 Degrading</td>
<td>00</td>
<td>-0</td>
<td>00</td>
</tr>
<tr>
<td>Pump 3 Degrading</td>
<td>00</td>
<td>00</td>
<td>-0</td>
</tr>
</tbody>
</table>

**Fault Signature Matrix**

- Pump 1 Flow Symbol: \(-0\)
Degradation Modeling and Fault Identification

- Unexpected change in system components
  - Modeled as parameter changes
  - E.g., $\frac{dk_1}{dt} = 0$, when nominal $= \Delta k_1$, otherwise
- Faults assumed to be
  - Single faults
  - Incipient and persistent

Degraded System - Pump 1 Faulty

# State transition model with faulty param included as additional state

```python
def state_transition_k1_fault(x, dt, params, inputs):
    dt = dt * 1.15e-2
    i1, w1, i2, w2, i3, w3, k1 = x
    L1, R1, J1, B1, L2, R2, k2, J2, B2, L3, R3, k3, J3, B3 = params
    Vs1, Vs2, Vs3 = inputs
    di1 = (1/L1) * (Vs1 - R1 * i1 - k1 * w1) * dt + i1
    dw1 = (1/J1) * (k1 * i1 - B1 * w1) * dt + w1
    di2 = (1/L2) * (Vs2 - R2 * i2 - k2 * w2) * dt + i2
    dw2 = (1/J2) * (k2 * i2 - B2 * w2) * dt + w2
    di3 = (1/L3) * (Vs3 - R3 * i3 - k3 * w3) * dt + i3
    dw3 = (1/J3) * (k3 * i3 - B3 * w3) * dt + w3
    dk1 = 0 * dt + k1
    return [di1, dw1, di2, dw2, di3, dw3, dk1]
```

# Observation model

```python
def observation_model_k1_fault(x, dt, params, inputs):
    return [x[1], x[3], x[5]]
```
Degradation Modeling and Fault Identification — 3-Pump IIoT Testbed
We are specifically interested in predicting failure states
- EOL = end of life (time to failure)
- RUL = remaining useful life (time until failure)

Define a threshold function that partitions state-space into non-failure and failure-states
- $T_f : \mathbb{R}^n_x \rightarrow \{\text{true, false}\}$
- That is, $T_f(x(k))$ returns true when it is a failure state, false otherwise
Remaining Useful Life Prediction

- Prediction involves simulating degraded system model with hypothesized or known future inputs
- Sample from the state and parameter and simulate each sample forward till RUL criteria is fulfilled
- Initial conditions are very important
- Weighted mean of EOLs give mean EOL

**Algorithm 1 EOL Prediction**

**Inputs:** \{(x_f^i(k_P), \theta_f^i(k_P), w^i(k_P))\}_{i=1}^N

**Outputs:** \{EOL_f^i(k_P), w^i(k_P)\}_{i=1}^N

for \(i = 1\) to \(N\) do

- \(k \leftarrow t_p\)
- \(x_f^i(k) \leftarrow x_f^i(k_P)\)
- \(\theta_f^i(k) \leftarrow \theta_f^i(k_P)\)

while \(T_{EOL}(x_f^i(k), \theta_f^i(k)) = 0\) do

- Predict \(\hat{u}(k)\)
- \(\theta_f^i(k+1) \sim p(\theta_f(k+1) | \theta_f^i(k))\)
- \(x_f^i(k+1) \sim p(x_f(k+1) | x_f^i(k), \theta_f^i(k), \hat{u}(k))\)
- \(k \leftarrow k + 1\)
- \(x_f^i(k) \leftarrow x_f^i(k+1)\)
- \(\theta_f^i(k) \leftarrow \theta_f^i(k + 1)\)

end while

- \(EOL_f^i(k_P) \leftarrow k\)

end for
RUL Prediction — 3-Pump IIoT Testbed

- **Pump 1**
  - $t_P = 160$ s
  - $EOL = 162$ s
  - $RUL = 2$ s

- **Pump 2**
  - $t_P = 180$ s
  - $EOL = 537$ s
  - $RUL = 357$ s

- **Pump 3**
  - $t_P = 260$ s
  - $EOL = 545$ s
  - $RUL = 285$ s
RUL Prediction — 3-Pump IIoT Testbed

- **Pump 1**
  - t\(P\) = 280 s
  - EOL = 492 s
  - RUL = 212 s

- **Pump 2**
  - t\(P\) = 240 s
  - EOL = 458 s
  - RUL = 118 s

- **Pump 3**
  - t\(P\) = 380 s
  - EOL = 425 s
  - RUL = 45 s
RUL Prediction — 3-Pump IIoT Testbed

t_P = 400 s  
EOL = 451 s  
RUL = 51 s
Wrapping Up...
Closing Remarks

- What is Cloud
- Benefits of Cloud
- How Cloud can help PHM
- What is available in GCP
- Custom algorithms in GCP
- Example of a PHM Solution on the Cloud
**PHM in the Cloud Status (Notional)**

<table>
<thead>
<tr>
<th><strong>PHM in the Cloud Status (Notional)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Physics-based Approaches</strong></td>
</tr>
<tr>
<td><strong>Data Requirement</strong></td>
</tr>
<tr>
<td><strong>Algorithm Complexity</strong></td>
</tr>
<tr>
<td><strong>Maturity</strong></td>
</tr>
<tr>
<td><strong>Data-driven Approaches</strong></td>
</tr>
</tbody>
</table>

Data Requirement: Low to High

Algorithm Complexity: Low to High

Maturity: High to Low

Data-driven Approaches:
- Anomaly Detection (e.g., single-class classifiers)
- Fault Isolation (e.g., multi-class classification)
- Degradation Quantification (e.g., regression, object detection)
- RUL Prediction (e.g., future predictions, CNNs)
- Decision Making (e.g., Reinforcement Learning)
Acknowledgements

- STIC Team
  - Prasad Bhagwat, Machine Learning Software Engineer
  - Bernard Van Haecke, IIoT Lead Technologist
  - Andrey Konchenko, E&P Production Digital Platform Subject Matter Expert

- Google Cloud Platform Team
  - Ricardo Payan
  - Ferris Argyle

Questions?