Industrial IoT Edge Architecture for Machine and Deep Learning

Authors:
Chanchal Chatterjee
Senior Director, AI Solutions
Teradata Inc.
chanchal.chatterjee@teradata.com

Salim AbiEzzi
Director R&D, IoT OCTO
VMWare Inc.
salim@vmware.com

OVERVIEW

This article presents an architecture for Machine and Deep Learning at the edge and Platform tiers for Industrial IoT. The architecture extends the classical lambda architecture in which both streaming and batch processing are handled simultaneously. It proposes data aggregation and protocol normalization at the edge with open source software such as Linux Foundation's EdgeX Foundry™. This is followed by Machine or Deep Learning inference at the edge. The data from the edge is cached and distributed to the Platform tier online or offline via open source software. Subsequently, at the Platform tier a data lake is created of edge and historical data, which is used for Deep Learning training. Note that two sets of models are created by training – one for the edge tier inference and one for the Platform tier inference with real-time streaming data.

INTRODUCTION

The Industrial IoT architecture promoted by the Industrial Internet Consortium (IIC) has three tiers: Edge, Platform and Enterprise. This article focusses on the edge and Platform tiers with stress on the computation and storage at these tiers for Machine and Deep Learning applications. The table below shows Machine and Deep Learning use cases and challenges for two Industrial IoT market verticals – Smart Manufacturing and Transportation:

Use Cases	Drivers	Challenges
Smart Manufacturing		
• Predictive maintenance • Process optimization • Supply chain optimization • Remote asset management • Product lifecycle monitoring • Integrated plant management • Product-as-a-service	• Increase yield and asset utilization • New revenue streams • Operational efficiencies • Increased worker satisfaction/safety • Eco-sustainability • Risk avoidance	• Low latencies • Data security • Interoperability between diverse sets of equipment • Rapid interpretation of large volumes of data • Indoor/outdoor reliability in harsh environments • Connectivity across access technologies
Transportation		
• Remote asset tracking • Predictive asset maintenance • Warehouse capacity optimization • Real-time fleet management • Route optimization	• Better fleet utilization/ opex reduction • Reduce emergency response times • Monitor driver behavior • Maintenance cost reduction	• Constant connectivity • Local regulation compliance for international fleets • Low latency

Table 1: Smart Manufacturing and Transportation Use cases for Machine and Deep Learning (Adapted from SDx Central IoT Infrastructure Report 2017)

In a typical Industrial IoT implementation we have the following components:

1. Edge Data Aggregation and Streaming Framework
2. Edge Cache
3. Platform Streaming framework
4. Platform Cache

In addition, for Machine Learning (ML) and Deep Learning (DL) we have the following items:

1. Edge ML/DL Inference Framework using a CPU and/or a GPU if low latency is required
2. Platform DL Inference Framework using CPU and/or GPU
3. Platform DL Training framework using CPU and/or GPU

In classical implementations, all Machine/Deep Learning is done on the cloud hosted platform or enterprise tiers, and all real-time data is sent to the platform/cloud for training. All inference is also traditionally done in the cloud. This methodology has the following drawbacks:

1. Excessive data transport cost to the cloud.
2. High latency in obtaining results creating a non-real-time inference on real-time data.

New Ideas

Three major innovations proposed are:

1. Move computation to the edge creating a low latency, distributed solution.
2. Implement the lambda architecture at the edge; i.e., handle both real-time and batch data.
3. Use two inference engines – one at the edge and one at the platform to get two different views of data – local and global.

Benefits

The new ideas have the following benefits. Some benefit details are obtained from SDxCentral-Innovations-in-Edge-Computing-and-MEC-2017-Rev-A.

1. Latency: The edge can provide latency in milliseconds while multiple hops and long transmission distances to the Platform tier is in the 50-150 ms range. Latency to centralized data centers and the public cloud is even greater.
2. High throughput: The throughput available to the user from the edge, served

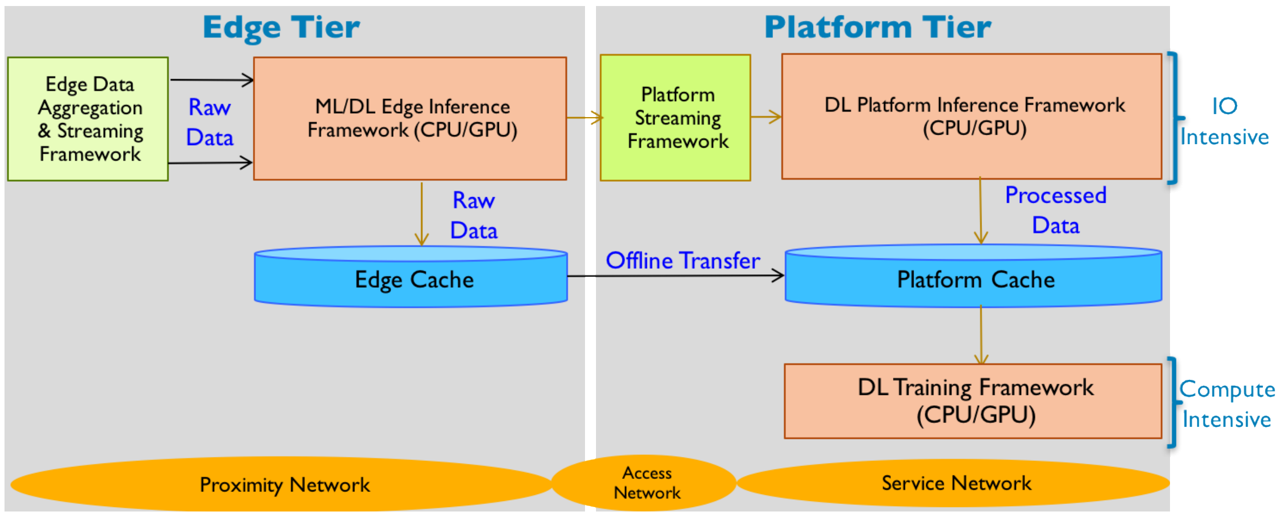

Edge Tier
Platform Tier
Edge Data Aggregation & Streaming Framework
Raw Data
ML/DL Edge Inference Framework (CPU/GPU)
Platform Streaming Framework
DL Platform Inference Framework (CPU/GPU)
IO Intensive
Raw Data
Processed Data
Edge Cache
Offline Transfer
Platform Cache
DL Training Framework (CPU/GPU)
Compute Intensive
Proximity Network
Access Network
Service Network

Figure 1: Industrial IoT Implementation with Machine and Deep Learning

via cached or locally generated content, can be orders of magnitude greater than from a core data center.

3. Data reduction: By running applications such as ML/DL at the edge, operators and application vendors can substantially cut down the amount of data that has to be sent upstream. This cuts costs and allows for other applications to transfer data.
4. Isolation: A number of environments are not always connected to the Internet over high speed links. The edge is able to provide services during periods of degraded or lost connections.
5. Compliance: Edge applications can help with privacy or data location laws.
6. Accuracy: Combine local results at the edge with global results at the platform to get a more comprehensive view of the data in real-time.

Challenges

The new ideas have the following challenges.

1. Most edge gateways do not have the computation capability to perform Deep Learning. In those cases, we need to fit the computation possible at the edge. These can be simple rule-based methods and auto thresholding algorithms or more complex feature engineering and Machine Learning algorithms such as decision tree or logistic regression.
2. This can increase the cost of edge hardware by requiring more computation at the edge and drive processing away from platform based computation. This will increase investments that may not have sufficient monetization.
3. This will require a new infrastructure for processing, storing and securing data.
4. There does not exist many mature orchestration and management solutions for applications that involve the edge and Platform tiers.
5. There are not enough ML/DL application vendors for the edge and especially those that cover both edge and Platform tiers. The software platforms and ecosystems are immature. There is a lack of expertise and skills in this area.

Edge Computing Cost-Benefit

1. Third Party: Edge computing allows third party applications to coexist with operator applications at the edge. Third party applications will unleash new innovations and services like Machine and Deep Learning at the edge.
2. Real-time: Some applications simply cannot tolerate latency more than in the order of 10s of milliseconds. Additionally, these applications are also sensitive to jitter (the variation in latency). Industry 4.0, which in turn depends on Machine Learning will only be possible through low-latency connectivity to compute resources.
3. Cost: ML/DL applications could run in the cloud, but the upstream bandwidth required to transmit the data to the cloud to power them is not economical. Video surveillance, face recognition, vehicle recognition, IoT gateway are ML/DL applications that belong to the edge. In an IoT gateway, where even though the bandwidth may not be high, sending billions of events to the cloud would be expensive and inefficient vs. handling them at the edge with an IoT gateway, especially considering that most of these

events are going to be routine without any anomaly.

Figure 2 shows cost of compute vs data transfer cost benefit of edge-based ML/DL applications.

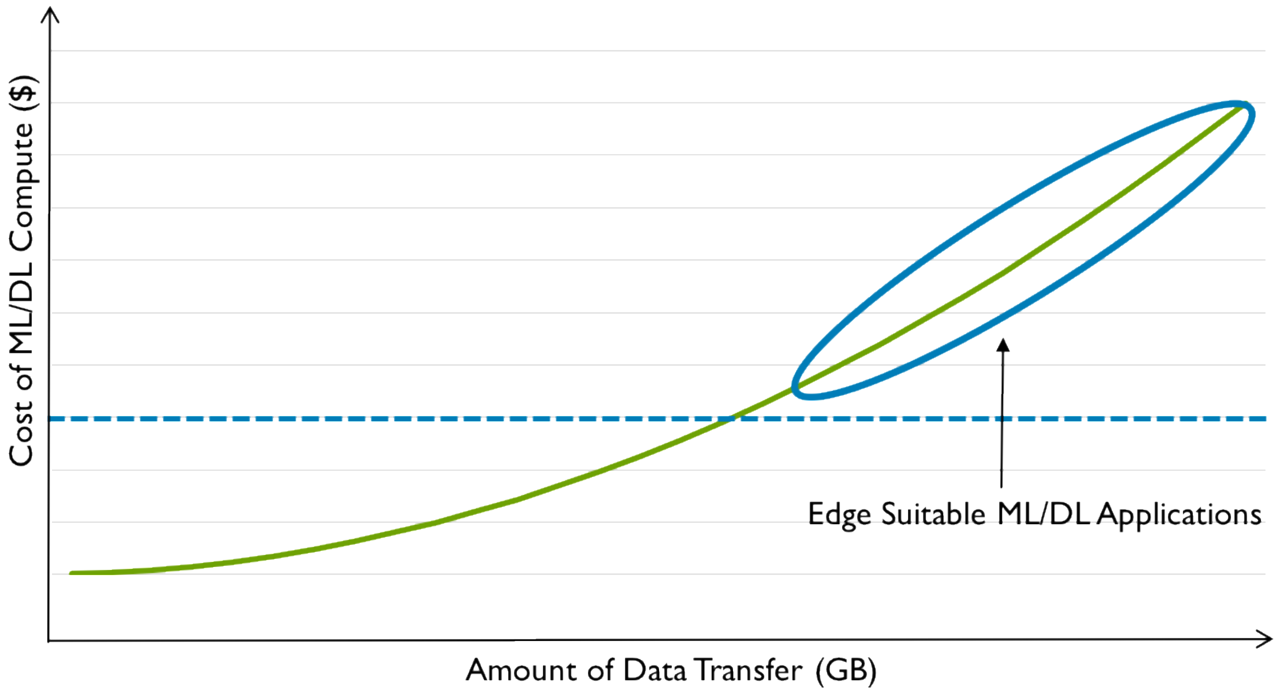


Figure 2: Cost of compute vs. data transfer cost benefit of edge-based ML/DL applications

In the following sections, we shall describe the following – a short summary of Deep Learning, edge tier innovations for DL, Platform tier innovations for DL and a use case of sensor based trucking.

WHY MACHINE AND DEEP LEARNING

The science and practice of artificial intelligence is vast and consists of the following types: (1) Data, (2) Learning and (3) Algorithms. **Data** consists of:

1. Structured data such as time series data, events and graphs and
2. Unstructured data such as video, images, speech and text.

Learning consists of the following types:

1. Unsupervised where we do not require training data and assume that normal instances are far more frequent than anomalies.
2. Semi-Supervised where training data has labeled instances of only one class. We also assume that normal instances are more frequent than anomaly classes.
3. Supervised where training data has labeled instances for normal and anomaly classes. We also assume accurate representation of the labels for the classes.
4. Reinforcement where rewards are provided to guide the learning through policy.

We next consider the types of **algorithms**. Here we diverge into Machine Learning algorithms which consist of anomaly detection, trends, prediction and forecasting algorithms, and associations and grouping algorithms. Machine Learning evolved out of the sciences of statistics and probability theory. Deep Learning evolved out of artificial neural networks and has many types including:

1. Convolutional networks, which have been very successful in classifying images and videos.
2. Recurrent networks, which are known to perform well on text, speech and natural language.
3. Deep belief networks, which have performed well on un-supervised and semi-supervised cases.

The Asimov Institute (www.asimovinstitute.org/neural-network-zoo) shows descriptions and figures for various types of Deep Learning networks.

In the context of IIoT we have all these types of data and learning. We have unstructured text/log data as well as speech and sound. Structured data consists of images, spectrographs and time series data. Traditional methods of Machine Learning have the following issues:

1. Extensive feature engineering before using the data,
2. Cannot easily deal with high volume and high dimensional data,
3. Cannot decipher interdependency of data and complex high varying non-linear functions.

Hence the performance of these algorithms is limited to smaller volumes of data and their accuracy and predictability has room for improvement. Deep Learning has fundamentally changed this with its well-proven superior accuracy on images and texts as well as high volume time series data. However, Deep Learning has high computational needs and has the following disadvantages:

1. Difficult to select model type and topology
2. Difficult to interpret models for validation
3. Computationally intensive and may require specialized hardware.
4. Overfitting is common due to layers of abstraction

Note that most Deep Learning typically requires specialized hardware such as GPUs to handle high volume of data but the same can be achieved with CPUs at higher density of computational units. Ongoing research such as Tensor2Tensor are continually improving the performance of Deep Learning.

A use case for separating Machine and Deep Learning at the edge and Platform tiers is given in the section below.

EDGE TIER ARCHITECTURE FOR MACHINE AND DEEP LEARNING

The edge consists of the edge devices such as sensors and actuators commonly known as “things,” device aggregators and gateways. Figure 3 shows the typical components of the edge.

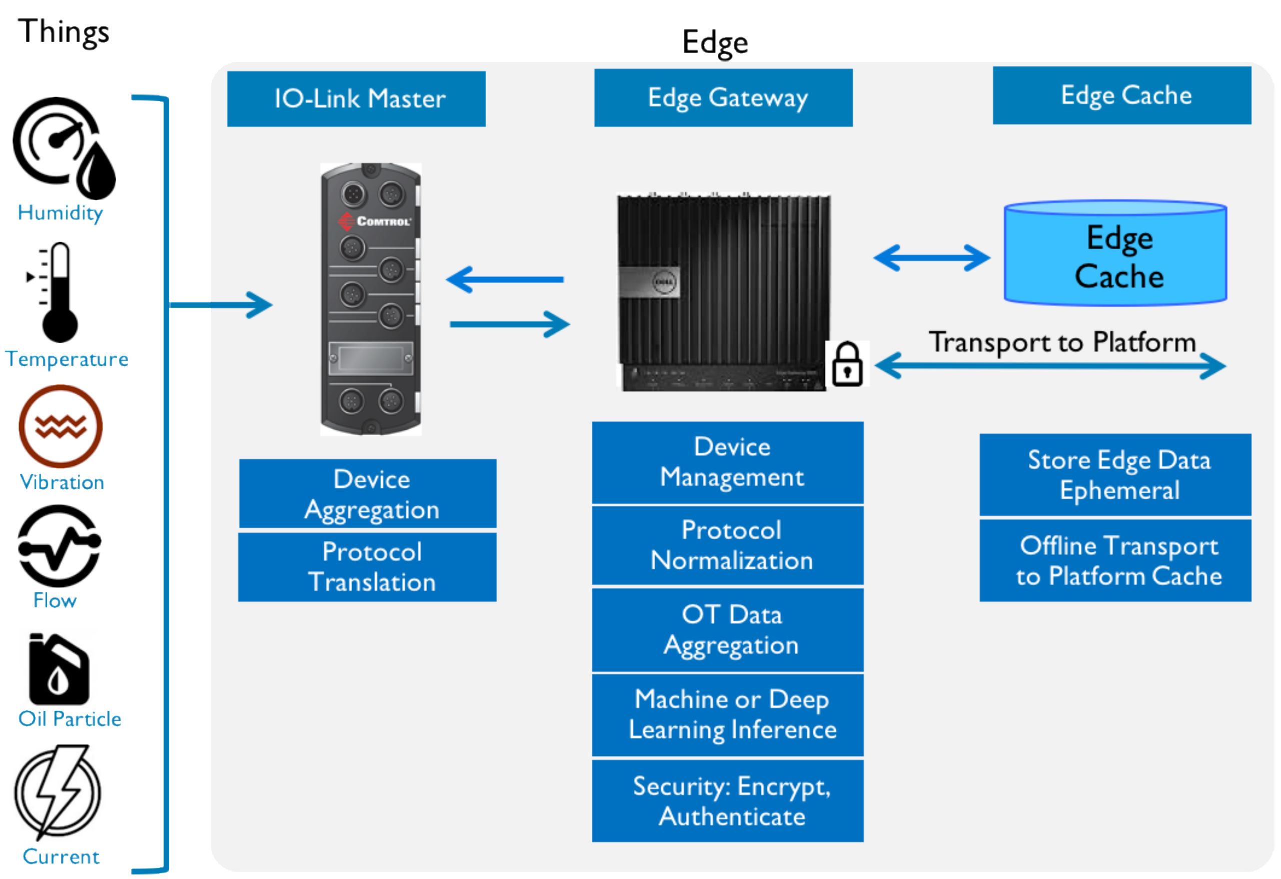


Figure 3: Typical components of the edge

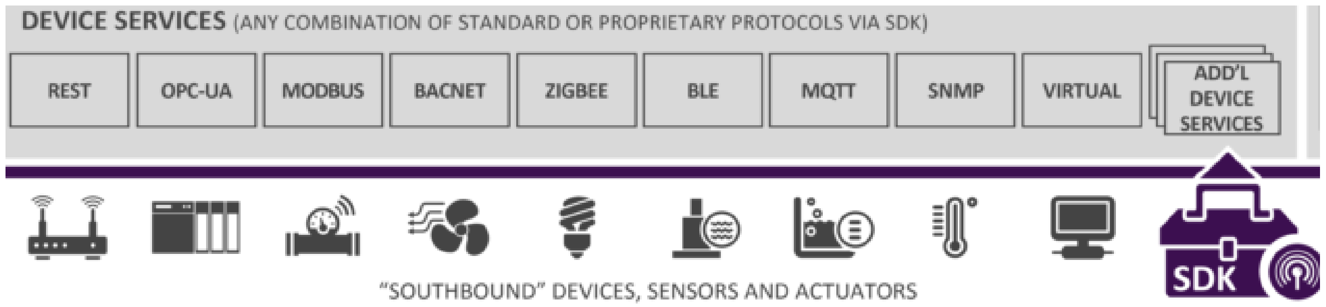
Data from the devices are aggregated and normalized at the gateway. The gateway is also used for Machine or Deep Learning inference. Raw data from the edge are temporarily stored at the edge cache which is offloaded to the Platform tier cache for training. Transfer of data from the edge to the platform can happen in batches when network bandwidth is available. Alternatively, edge data can be transferred to the platform without edge caching on available transport.

The architecture above can be implemented with several frameworks. We discuss implementing it with two open source frameworks: (1) EdgeX Foundry, and (2) Liota™.

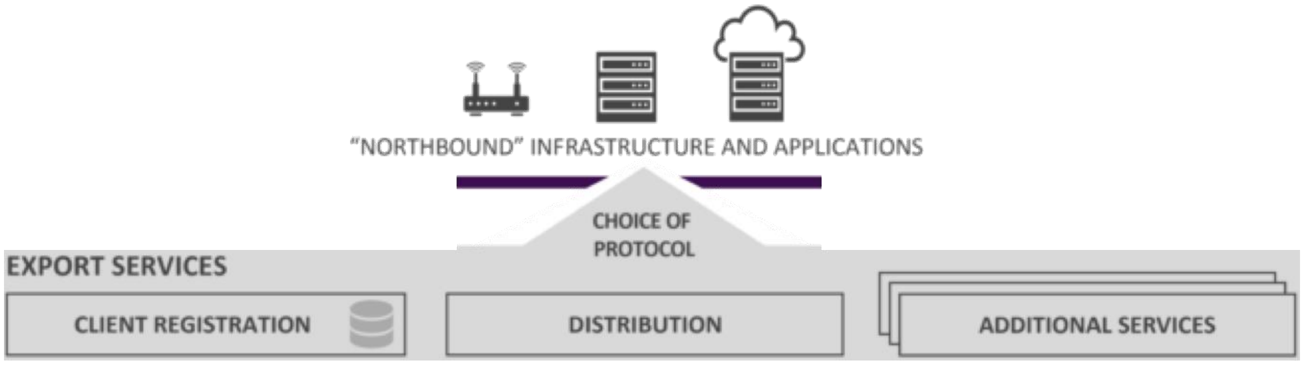
EdgeX Foundry

EdgeX Foundry is an open source project hosted by The Linux Foundation offering a common framework for IoT edge computing. See details of EdgeX™ Foundry at www.edgexfoundry.org. EdgeX offers many services including security, identification, scheduling, logging, deployment and console management. Due to the high volume of data transfer between devices and edge as well as from edge to platform, the services we most use for Deep Learning architecture are:

South Bound Device Services: Bridge a large set of device standard protocols and data formats to common REST APIs.


DEVICE SERVICES (ANY COMBINATION OF STANDARD OR PROPRIETARY PROTOCOLS VIA SDK)
REST
OPC-UA
MODBUS
BACNET
ZIGBEE
BLE
MQTT
SNMP
VIRTUAL
ADD'L DEVICE SERVICES
SDK
"SOUTHBOUND" DEVICES, SENSORS AND ACTUATORS

Northbound Export Services: Northbound distribution of data to the Platform tier from edge devices and cache.



Both these services provide the necessary modules and protocols to support data transport to and from the edge, devices and platform.

Liota

Little IoT Agent (Liota) is an open source software development kit (SDK) for building IoT gateway applications to monitor devices and orchestrate their data to platform or cloud services. Liota operates on generic edge devices or integrates seamlessly with EdgeX. It leverages the EdgeX

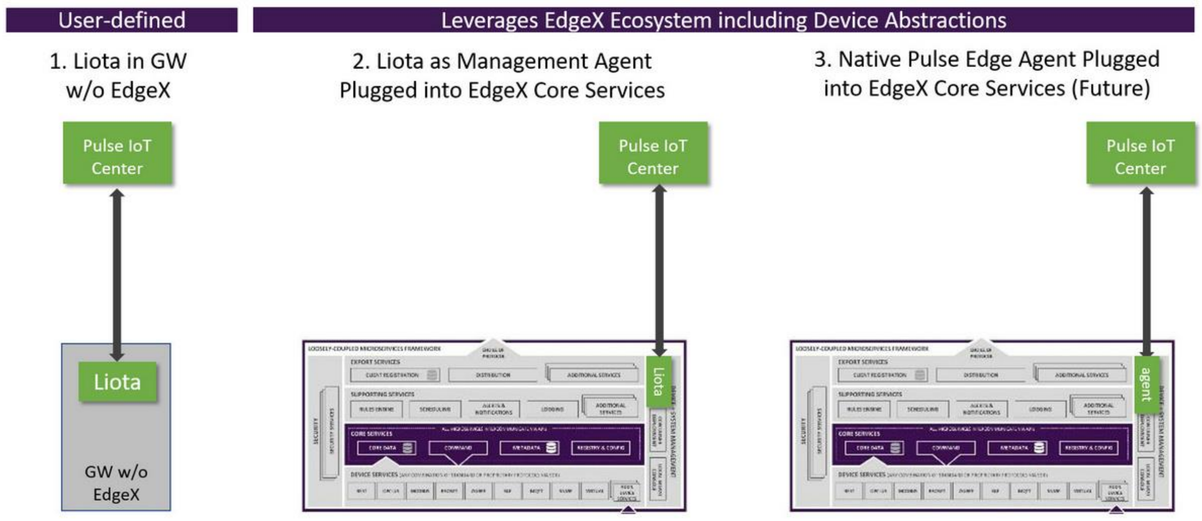

User-defined
Leverages EdgeX Ecosystem including Device Abstractions
1. Liota in GW w/o EdgeX
2. Liota as Management Agent Plugged into EdgeX Core Services
3. Native Pulse Edge Agent Plugged into EdgeX Core Services (Future)
Pulse IoT Center
Pulse IoT Center
Pulse IoT Center
Liota
GW w/o EdgeX
Liota
agent

Figure 4: Liota

ecosystem including device abstraction to transport data to VMware Pulse IoT Center™, which offers an end to end infrastructure management solution for secure and reliable data flow from devices to applications. The figure below shows the different ways that Pulse IoT Center connects with edge devices. See details at: www.vmware.com/products/pulse.html. Liota is part of Pulse IoT Center; however, it can be used separately to orchestrate data from sensors through gateways to other IoT services.

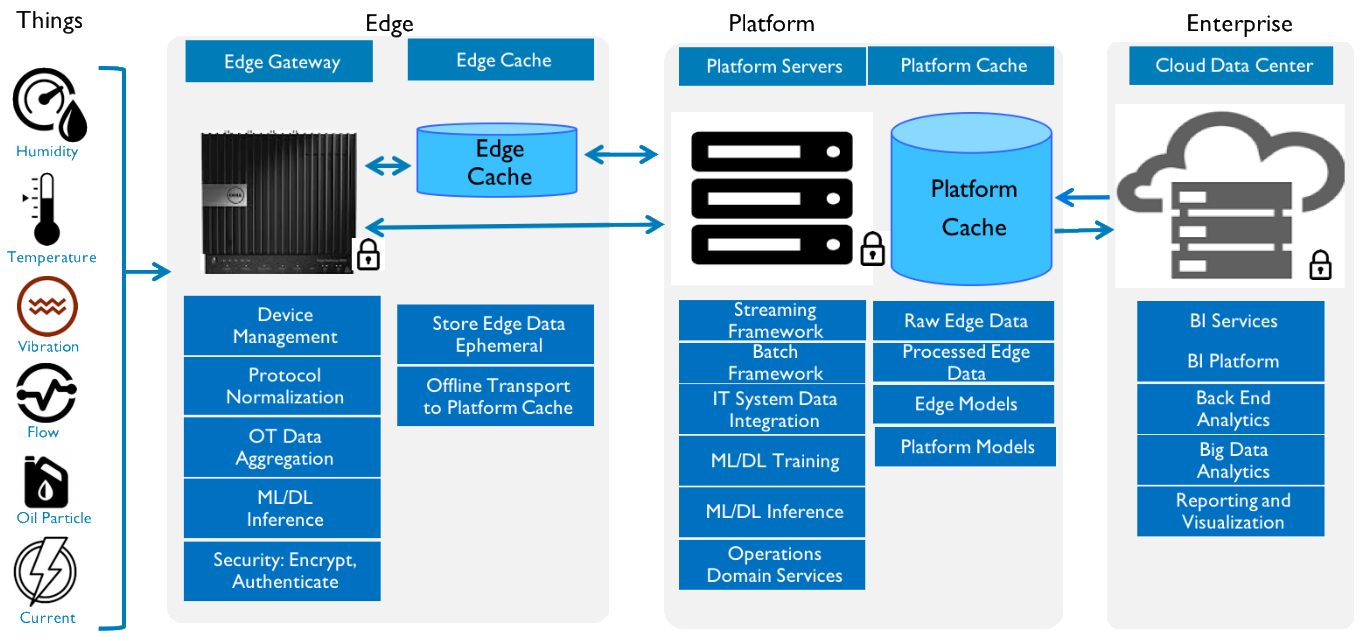


Figure 5: Platform Tier Architecture for Deep Learning

PLATFORM TIER ARCHITECTURE FOR DEEP LEARNING

The Platform tier receives real-time and batch data from the edge and performs Deep Learning model inference and training. It caches raw data as well as processed data after feature engineering. It also transfers the final model inference results to the enterprise tier for Business Intelligence (BI) reporting and visualization.

The Platform tier performs Deep Learning training as well as inference. The platform pulls raw data from the edge cache, as well as performs feature engineering on real-time and stored edge data. The feature data includes information from all edges to create aggregate information such as median temperature of devices at all edges.

Use Case for Edge and Platform Tier Learning

The example below shows a use case for separating Machine and Deep Learning at the edge and Platform tiers.

Figure 6 shows temperatures from four devices. The median temperature of all four devices (in red) is overlaid on the device temperatures in blue. The edge can detect the anomaly in Device 3 due to a sudden drop in temperature but the anomaly in Device 4 is hard to detect at the edge. When Device 4 temperature is compared with median

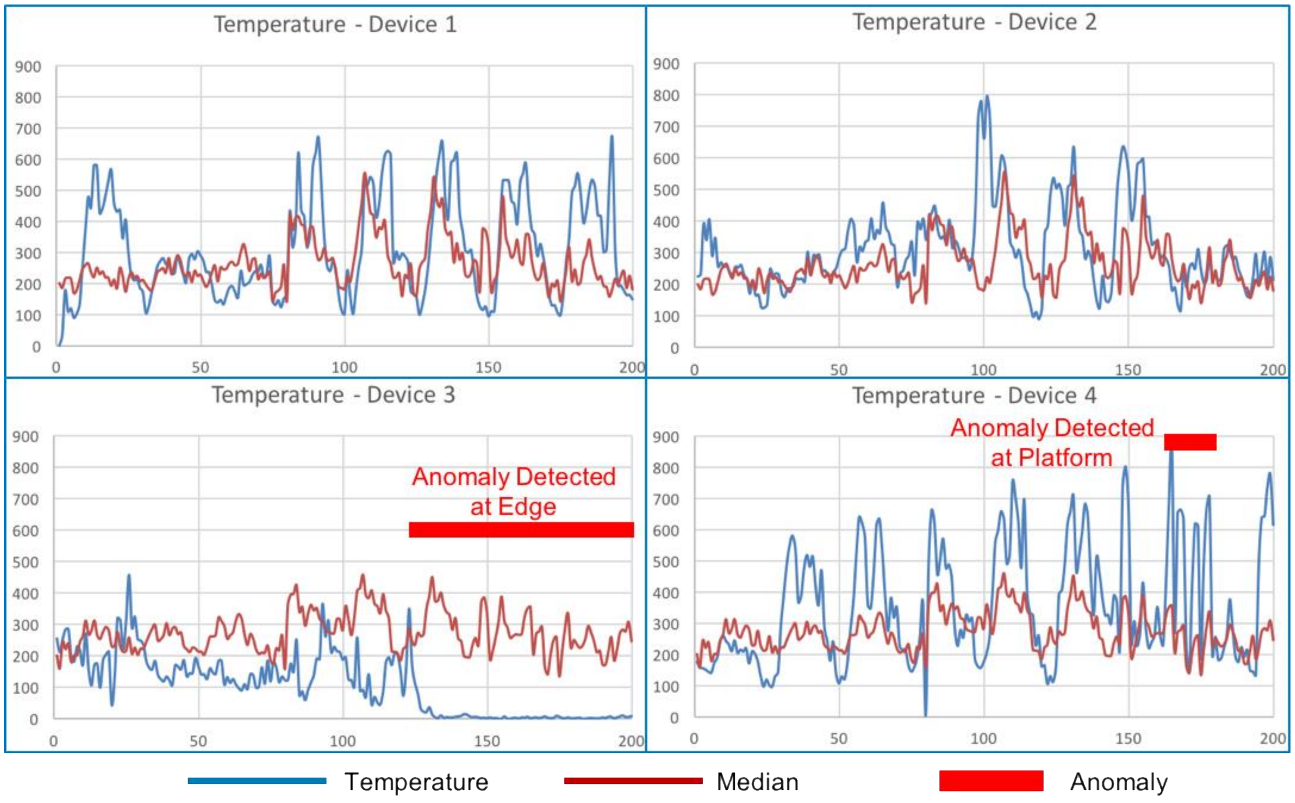


Figure 6: Temperatures from four devices in use case example

temperatures of all devices (in red) the anomaly in Device 4 is detected at the Platform tier.

The edge model can be an anomaly detection model on a univariate time series. The platform model can be an anomaly detection model on a multivariate time series.

The raw data from edge is used to train edge models whereas feature data extracted from the raw edge data are used for platform models. Models are stored in the model cache. Edge models are pushed to the edge inference engine, whereas platform models are pushed to the platform inference engine. Both models are served by the model serving engine. Figure 7 shows the edge and platform architectures for training and inference as well as the data and model caches.

Note that the human domain knowledge mentioned here for the Deep Learning training and inference has several components:

1. It is used to label training samples manually or automatically through an algorithm that applies rules based on human domain knowledge.
2. It is used to choose the models used for training.
3. It is used to provide auxiliary rules in addition to the Deep Learning models.
4. It is used to interpret the inference results and determine if the interpretations match human domain knowledge.

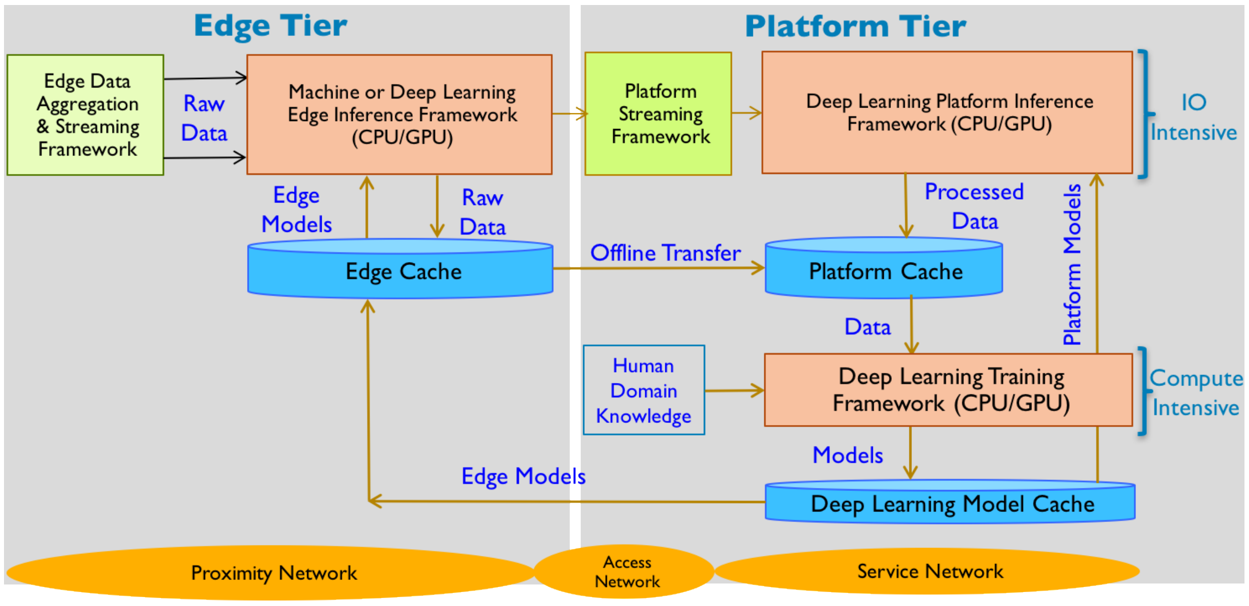

Edge Tier
Platform Tier
Edge Data Aggregation & Streaming Framework
Raw Data
Machine or Deep Learning Edge Inference Framework (CPU/GPU)
Platform Streaming Framework
Deep Learning Platform Inference Framework (CPU/GPU)
IO Intensive
Edge Models
Raw Data
Processed Data
Platform Models
Edge Cache
Offline Transfer
Platform Cache
Data
Human Domain Knowledge
Deep Learning Training Framework (CPU/GPU)
Compute Intensive
Models
Edge Models
Deep Learning Model Cache
Proximity Network
Access Network
Service Network

Figure 7: Edge and platform architectures for training and inference

Connected Trucking Use Case

We now consider an Industrial IoT use case on sensor based trucking involving all the components discussed earlier in this article. The truck consists of several sensors including cameras, radar, GPS and Lane sensors. Figure 8 shows a truck with these sensors.

These sensors are aggregated at the edge gateway which is located within the truck. EdgeX normalizes the protocols from all sensors and caches the data. Liota interfaces

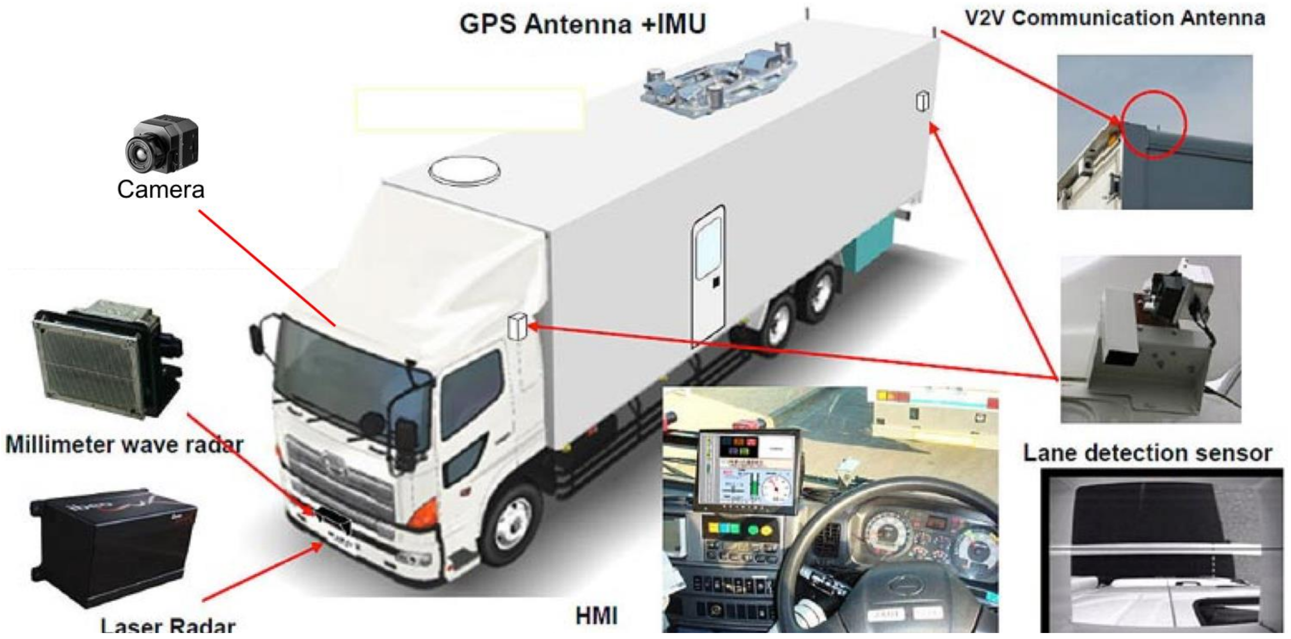


Figure 8:Trucking Sensors

to EdgeX, and orchestrates the data to Pulse IoT Center to manage.

The Platform tier receives real-time edge data as well as stored data from the edge cache. The real-time edge data is processed and aggregated across edges to create features which are stored in the platform cache. The real-time features are used by the platform inference models. Inference results from the edge and platform are aggregated and sent to the Enterprise tier for BI reporting and visualization as well as remediation if necessary.

The Platform tier uses the cached raw and processed data to train edge and Platform tier models, which are served in real-time to the edge and platform inference engines via Pulse IoT Center OTA (over the air) software update capability. Figure 9 summarizes the architecture.

CONCRETE ILLUSTRATION OF APPLYING THIS ARCHITECTURE

Consider a set of surveillance cameras attached to the edge device in the truck. The edge device processes the footage from all cameras and detects events of interest (accident, intrusion, fist fight, pedestrian near misses, a comet falling from the sky, etc.). Thanks to edge computing, the voluminous video feeds are processed locally, thus at low latency (important for quick alerting; e.g., for crash avoidance). Only footage of such significant events is sent OTA to a cloud service for archival and future search, retrieval and learning/training. Therefore, such important footage is protected in the event of vandalism or crash of local equipment. In addition, this approach significantly lowers OTA traffic; we estimate that important footage constitutes

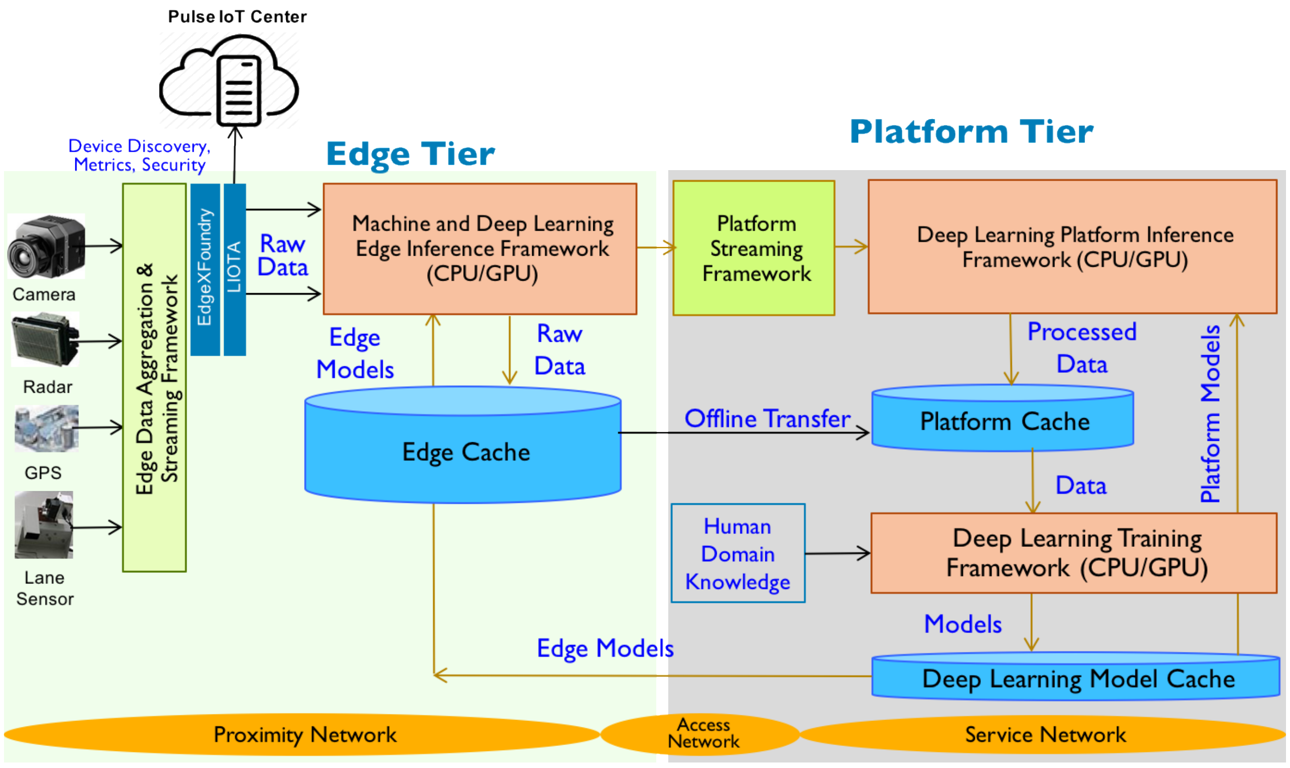

Pulse IoT Center
Device Discovery, Metrics, Security
Edge Tier
Platform Tier
Camera
Radar
GPS
Lane Sensor
Edge Data Aggregation & Streaming Framework
EdgeXFoundry
LIOTA
Raw Data
Machine and Deep Learning Edge Inference Framework (CPU/GPU)
Edge Models
Raw Data
Edge Cache
Platform Streaming Framework
Deep Learning Platform Inference Framework (CPU/GPU)
Processed Data
Offline Transfer
Platform Cache
Data
Platform Models
Human Domain Knowledge
Deep Learning Training Framework (CPU/GPU)
Models
Edge Models
Deep Learning Model Cache
Proximity Network
Access Network
Service Network

Figure 9: Trucking Use Case Architecture

significantly less than 1% of total video footage.

How do you distinguish between normal footage 99+% of the time, vs. interesting events worthy of archival? It is based on image recognition and inference neural nets. Learning (generation and enhancement of the inference model/framework) happens at the Platform tier based on as many camera feeds as possible. Big Data storage and processing in action. Perhaps with human/expert assistance to train on identifying events of interest. So, gradually the system becomes better at detecting more and more important events. Periodically, upon non-trivial improvements to the inference framework, an update is pushed to all the edge devices for better inference (i.e., judgement/harvesting) of “interesting events.” In this truck use case, this manifests as an OTA software update.

What we describe above is the application/data plane, which layers on top of and is enabled by the control/infrastructure plane, provided by Pulse IoT Center. IoT Center insures the secure, reliable and orderly functioning of the different components of the system, and the data flow from end device to platform through the edge device. IoT Center also manages the OTA software updates. Following are more details of the control plane.

EdgeX on-boards the cameras as end devices (the initial version of EdgeX does not support video; however, this could be added in the future – our emphasis in this example is on the architecture), and populates the device metadata. An EdgeX microservice processes the footage from each camera and applies inference model/incidence detection. Another EdgeX microservice stores important event footage. Then an export microservice transmits this footage to a Platform tier service. Liota discovers on-boarded cameras through the EdgeX device metadata microservice and proxies these devices for management by IoT Center. It obtains infrastructure metrics (e.g., heart beat signal) to monitor and ensure proper function. Liota also obtains metrics from the edge device itself (e.g., storage availability and CPU load), for monitoring and managing. In addition, IoT Center is used for OTA software updates of the EdgeX inferencing microservice, and for the latest security patches.

CONCLUSIONS

We presented architectures for the edge and platform for real-time and batch data collection, model training, model management and model inference at the edge and Platform tiers. This is a comprehensive solution for Deep Learning in Industrial IoT which can be used in many use cases. One use case involving sensor-based trucking with open source and commercial solutions is also shown using our architecture.

➢ Return to IIC Journal of Innovation landing page for more articles and past editions.