Distribution Transformer Condition Monitoring based on Edge Intelligence for Industrial IoT

Leny Thangiah
Siemens
Singapore
leny.thangiah@siemens.com

Prof. Dr. Chandrashekar Ramanathan
International Institute of Information Technology
Bangalore, India
rc@iiitb.ac.in

Lakshmi Sirisha Chodisetty
Siemens
Bangalore, India
lakshmi.chodisetty@siemens.com

***Abstract*— Adoption of IoT in industrial applications results in huge volumes of data to be processed. By leveraging edge computing and agent based system architecture, autonomous decisions can be made at edge with locally available data without relying on the cloud. An important aspect to consider while designing smart edge systems is the architecture that enables local intelligence and real-time analytics. This paper proposes an architectural approach that combines the key aspects of edge computing and intelligent agents and presents experiment results using a Proof of Concept (PoC) on condition monitoring of distribution transformers in an industrial setting.**

***Keywords*— Intelligent Agents, Edge Intelligence, Edge Computing, IIoT, Condition Monitoring**

I. INTRODUCTION

With Industry 4.0 and digitalization initiatives sweeping the industrial world, assets are increasingly equipped with in-built or retrofit sensors to communicate a variety of parameters to applications that calculate KPIs beyond simple operating conditions and status.

Over the past few years, overriding the security apprehensions, cloud has emerged to be an inevitable layer of the Industrial Internet of Things (IIoT) architecture (e.g. Mindsphere, Predix. Many cloud based industrial analytics applications are successfully launched on such platforms. As a parallel emergence, powerful industrial edge devices (e.g. MindConnect, Dell PC 5000) have made their way to the market. However, in most scenarios, the role of an edge device is restricted to work as a data acquisition system only. As evident in the existing literature in this field [1][2][3][7][9], edge/fog computing offers additional features to the IIoT ecosystem, including data privacy, reliability, performance and availability via edge computing. The problem of processing the data thus can be divided across multiple levels (Fig. 1.), which helps relieve the load off the backend [3][9]. However, bringing Artificial Intelligence (AI) to the edge has its own challenges such as resource availability, network issues, local optima and security concerns.

In this paper, we have proposed an architecture for AI on the edge, and implemented and evaluated the software stack based on it. Using the Open Fog Reference Architecture [9], EdgeX Foundry [10] as the base, the architecture proposed here is a combination of key aspects of edge computing and autonomous or semi-autonomous agents working independently or in cooperative modes [4][5]. In the proposed architecture, without a central controller, the agents rely on the local data to offer machine intelligence ecosystem at the edge. Autonomy is one of the pillars of the reference architectures of OpenFog and Edge X. Although there are many other features available as part of such reference architectures, we have considered autonomy as the scope for this paper and the Proof of Concept (PoC) implementation. The primary objective of the work presented in this paper is to evaluate how IoT applications for industrial use cases deployed on edge/fog layer can be autonomous in their decision making without compromising the quality of the functionality. Evaluation and/or identification of machine learning algorithms or proving the accuracy of decisions on the edge is not in the scope of this paper.

The remainder of the paper is organized as follows: Section II introduces the architecture and explains key components of the edge stack developed based on the architecture, Section III describes the use case and components of the PoC with experiments and results and Section IV states the future work.

II. ARCHITECTURE FOR EDGE INTELLIGENCE

Decentralized IoT with agents/cooperating smart objects (CSO) is not a novel concept. Several studies and results have been published in this field [11][12]. Lack of access to all data sources is one of the key challenges faced by industrial analytics, which might result in ineffective decisions at the edge compared to a centralized deployment, where it is possible to receive data from the entire system. Conventionally, such use cases advocate for a centralized approach, mainly due to partial or missing knowledge of the process and data points to make decisions at the edge level. The proposed architecture addresses this shortfall of decentralized deployments by enabling data exchange between the agents in a peer-peer manner. By leveraging edge computing features, the proposed architecture enables a group of intelligent agents to be deployed at the edge for data acquisition, processing and federation of information amongst the peers. Since agents have close proximity to the field, they can have uninterrupted access to the data to evolve in their learning models to make decisions on the process data in real-time [6]. Such an architecture can support the deployment of data-oriented, context-aware, intelligent decision support systems.

Key components of the architecture and their responsibilities are explained in Fig. 1. and TABLE I.

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

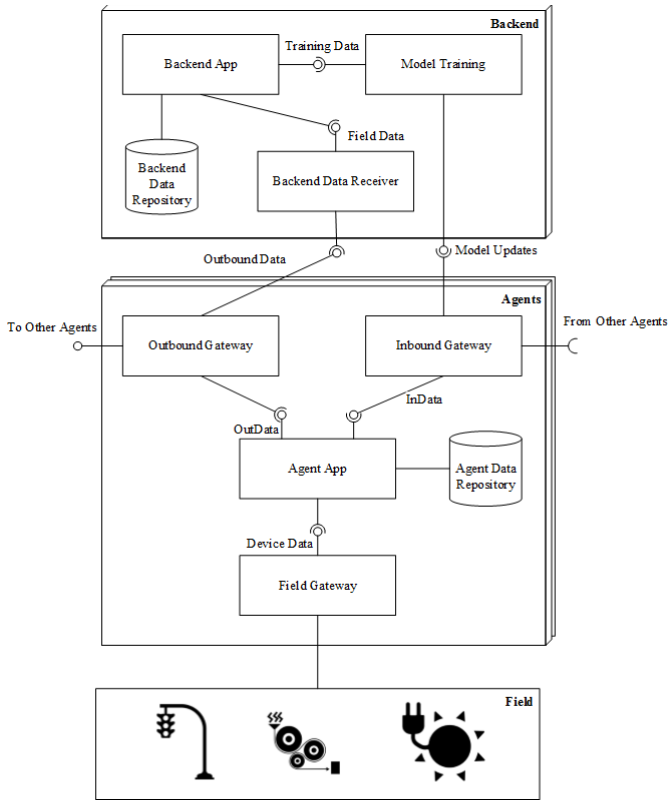

Fig. 1. Component view of the architecture

TABLE I. COMPONENTS AND RESPONSIBILITIES

Component	Responsibility
Field Gateway	Collects field data using domain specific protocols (e.g. IEC 61850). Exposes an internal interface to provide access to the field data.
Agent Data Repository	Used as a data storage for storing the data collected from the field and other data points such as intermittent results and meta data. Exposes an internal interface for data to other components to retrieve the data.
Inbound Gateway	Receives data from other agents and model updates from Backend and provides access to data and models to the Agent App.
Outbound Gateway	Provides external interfaces for the Backend applications to access the data from agents and sending updates to the agent applications.
Agent App	Responsible for collecting the data from the field and making decisions out of locally available knowledge. If the local knowledge is not adequate to make decisions, agent applications communicate with their peers (configurable) to get global views.
Backend Data Receiver	Responsible for collecting the data from the field and storing the data in Data Repository.
Backend Data Repository	Similar to the Agent Data Repository, Backend Data Repository stores the data collected from all agents. It is used by the Backend App.
Backend App	Responsible for processing the data collected from the field to draw global insights on the entire system. It can also be used for improving the models deployed in edge by learning continuously based on the actions by individual agents and their payoffs.

As decentralization is the key enabler for edge intelligence, individual agents are responsible for acting on the data received from corresponding field devices. Each Agent App is connected to a Field Gateway in order in order to receive the data from the field. Each agent is equipped with a Data Repository to store the data locally and Inbound/Outbound Gateways to communicate to the external world.

Agent App acts as a container for the machine learning models to process the field data against predefined KPIs. Initial models are developed based on the domain knowledge, available historical data and the context of the application. For instance, in a smart factory, such a model is developed using the domain knowledge to analyze the vibrations of a motor based on the normal and anomalous behavior of the motor under various loads and physics based rules. While secured REST APIs connect the agents with backend, components within agents are connected with their dependencies using dependency injection [15]. Inter-agents communication is realized with distributed MQTT brokers (e.g. Eclipse Paho), which was not implemented for the experiment covered in this paper.

III. PROOF OF CONCEPT AND EXPERIMENTATION RESULTS

A. Scope and Objectives

The scope of the PoC is to demonstrate an industrial use case on Edge Intelligence using the architecture proposed in this paper. We have chosen condition monitoring of distribution transformers using surface level temperature sensing for this purpose. Heat produced as part of the step-down process needs to be dissipated using a coolant and in contact with the air. Typically vegetable oil is used as coolant in distribution transformers, especially in countries like India. In some places there is a chance of oil loss due to factors like leakages or illegal tapping. This results in overheating of the transformers and deterioration of transformer components. In a few extreme cases it can result in a meltdown and catastrophic failures or accidents. Also, the oil level has a direct impact on the load factor of a transformer, which results in revenue loss if the transformer is not loaded optimally. Though there is an indicator for the oil level in a transformer, it's impractical to manually inspect it on a regular basis.

Remote condition monitoring and maintenance system will be highly useful to the utility companies in such scenarios. For this PoC, temperature at various parts of a transformer has been considered as the input to infer the corresponding oil level inside the transformer tank and conservator [13]. Following are the reasons why this use case is considered appropriate for demonstrating Edge Intelligence:

- Real-time monitoring of oil level and condition of a transformer is important to protect its components from quick deterioration and major failure, which typically happens within three hours in case of a total oil drain.
- Transformers are installed at remote places without sophisticated network capabilities to continuously transmit the readings to a central infrastructure.
- The analytics task can be distributed to edge hardware instead of loading the central infrastructure by streaming sensor readings from hundreds of transformers in a city.

B. *Deployment of the Proof of Concept*

The test infrastructure (Fig. 3.) was constructed based on a standard Oil Natural Air Natural (ONAN) type distribution transformer. By referring the existing literature in transformer condition monitoring using temperature profiling [13][14], sensors were placed on the surface of the transfer on the following locations: conservator top, conservator rod, conservator bottom, fin top, tank mid, and tank bottom of the transformer (Fig. 2.). Field gateway receives the data from these sensors using a proprietary protocol. The temperature data is sampled at a regular frequency, which is in the form of a data stream with time-series values. The input tuples are fed through a MQTT broker for real-time subscriptions on the data stream.

Agent App receives the temperature readings and feeds this data through a deep learning model based on autoencoders (AE). The construction of AE model is based on the normal and abnormal operating conditions, which is explained further in this section. AE model detects the anomalous behavior of the transformer based on temperature inputs from sensors. In parallel, the data is sent to the backend part, which includes a data store (InfluxDB) and visualization components (Fig. 4.).

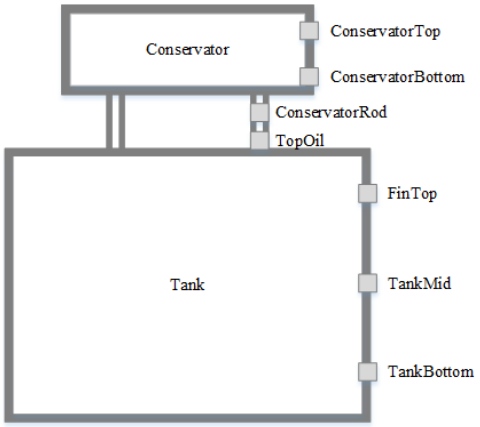

Fig. 2. Distribution transformer with placement of sensors

Fig. 3. Test bed for the PoC (Edge device highlighted)

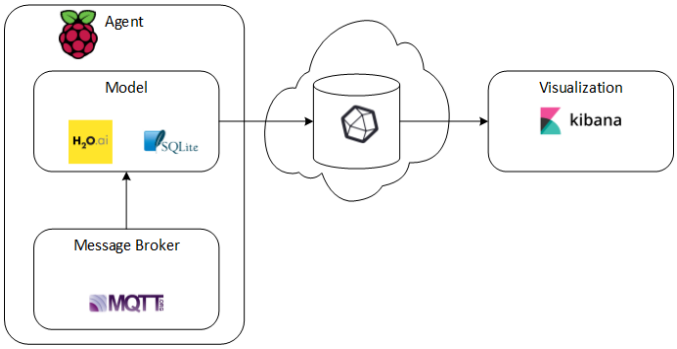

Fig. 4. Data flow and technology stack

C. *Test Bed and Technology Stack*

The Test Bed is constructed using a Siemens distribution transformer with standard analog to digital convertors and USB based data acquisition into Raspberry Pi 3 Model B board where the edge stack and the anomaly detection model are deployed (Fig. 3.). The test infrastructure and choice of technology are made based on the proposed architecture and deployment (Fig. 1.). H2O.AI is used for training the autoencoders deep learning model [16]. SQLite works as the agent data repository to buffer and store input values and intermittent results. Data stored in the backend (Influx DB) is primarily used for the visualization in current scope of the PoC (Fig. 4.).

D. *Experimentation results of the PoC*

Experiments were conducted on test infrastructure explained in previous sections of this paper. Temperature readings are collected from sensors placed at various locations of the transformer on a continuous basis. Raw values are collected as current inputs of 4-20 mA range and converted to corresponding degree centigrade values. All values are received with associated timestamps and stored in Influx DB at the backend. The time series data points that record the temperature dynamics under normal and oil drained operating conditions were used for constructing the model. Trained model was deployed on the edge device after integrating with other components of the software stack.

Two sets of measurements were collected with full oil level and oil drained conditions. As depicted in Fig. 5., values observed from all sensors exhibit an upward trend until the transformer reaches a stabilized state. While there is no significant change in ambient temperature, it is evident that the rest of sensors record a significant increase.

TABLE II describes a snippet of the input feature vectors:

TABLE II. INPUT FEATURES

timestamp	fin_bottom	fin_top	ambient	conservator_top	conservator_bottom	conservator_rod	top_oil
0:00:00	26.824375	27.53438	27.44	30.02875	27.67125	27.53125	27.903125
0:00:11	26.824375	27.53125	27.42563	30.02875	27.69125	27.525625	27.8975
0:00:21	26.83	27.53688	27.44	30.051875	27.674375	27.534375	27.90625
0:00:31	26.83625	27.54813	27.44813	30.040625	27.70875	27.53125	27.86625
0:00:41	26.83875	27.56563	27.4425	30.043125	27.7	27.514375	27.8975
0:00:51	26.83875	27.5425	27.48813	30.051875	27.7	27.548125	27.90625
0:01:01	26.833125	27.55688	27.5825	30.04625	27.694375	27.53125	27.883125
0:01:11	26.844375	27.52813	27.60563	30.02875	27.68875	27.5425	27.911875
0:01:21	26.8475	27.5425	27.61688	30.0575	27.694375	27.54	27.88875

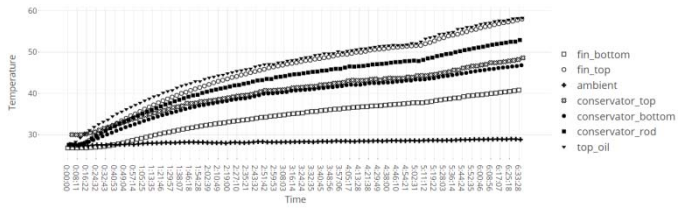

Fig. 5. Temperature modeling under normal operating conditions

Fig. 6. depicts anomaly detection when the oil is drained using the ball valve located at the bottom of the transformer.

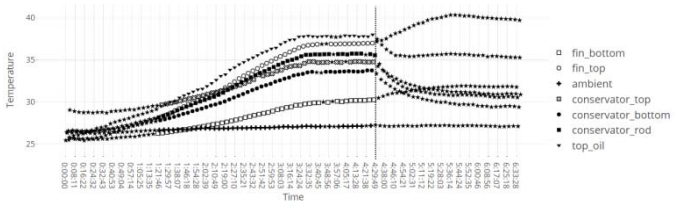

Fig. 6. Detection of oil drain on edge device using AutoEncoders model

While the star symbol indicates the detection of anomalous temperature values in Fig. 6. , the vertical dotted line indicates the point of detection of anomalies with all values. Temperature values are continuously recorded during the experiment that last for more than 6 hours. The values indicate an inclining trend at the beginning. After 4.5 hours, values start falling, mainly due to lack of oil inside the transformer body to dissipate the heat. Our observation conforms to existing studies in predicting the condition of the ONAN transformers [14] based on the temperature of the transformer body. The autoencoders model was trained outside the edge for 5000 epochs using values recorded under normal operating conditions with a configuration of 50 neurons, 50 hidden layers with uniform adaptive weight distribution. Threshold for detecting anomalies was calculated based on training error in reconstructing the output by feeding the temperature values recording during the oil drain condition. As evident in Fig. 6., anomaly is detected at the same time for all feature vectors with a reconstruction mean square error (MSE) in the range of 0.000076 and 0.135415. Distribution of MSE was observed as 25% values within 0.016162, 50% values within 0.067012 and 75% values within 0.086203.

IV. Conclusion and Future Work

In this paper, we have proposed an architecture for Industrial IoT by combining the concepts of edge/fog computing and agents. The envisaged architecture not only supports a wide range of Industrial IoT use cases but also enables intelligent decision making at the field by autonomous or semi-autonomous agents deployed on edge. The proposed architecture and the results of PoC proves that edge intelligence using advanced learning based on complex neural network is feasible in addressing some of the key challenges faced by industrial IoT applications like low-latency decisions and handling voluminous data.

In future, we plan to carry out comprehensive experiments involving cooperative learning by enabling peer-peer communication between the agents and backend analytics to evaluate the implications of various nonfunctional requirements such as scalability, fault-tolerance and security [1][4][7][8]. Further, we believe such architectures can enable distributed AI deployments and decentralized autonomous decision making in industrial IoT applications.

References

[1] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P. and Riviere, E., 2015. Edge-centric computing: Vision and challenges. ACM SIGCOMM Computer Communication Review, 45(5), pp.37-42.

[2] Bonomi, F., Milito, R., Natarajan, P. and Zhu, J., 2014. Fog computing: A platform for internet of things and analytics. In Big Data and Internet of Things: A Roadmap for Smart Environments (pp. 169-186). Springer International Publishing.

[3] Tang, B., Chen, Z., Hefferman, G., Wei, T., He, H. and Yang, Q., 2015, October. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In Proceedings of the ASE BigData & SocialInformatics 2015 (p. 28). ACM.

[4] Panait, L. and Luke, S., 2005. Cooperative multi-agent learning: The state of the art. Autonomous agents and multi-agent systems, 11(3), pp.387-434.

[5] Giordano, A., Spezzano, G. and Vinci, A., 2016, June. Smart Agents and Fog Computing for Smart City Applications. In International Conference on Smart Cities (pp. 137-146). Springer International Publishing.

[6] El-Tantawy, S., Abdulhai, B. and Abdelgawad, H., 2013. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown Toronto. IEEE Transactions on Intelligent Transportation Systems, 14(3), pp.1140-1150.

[7] Cerrada, M., Cardillo, J., Aguilar, J. and Faneite, R., 2007. Agents-based design for fault management systems in industrial processes. Computers in Industry, 58(4), pp.313-328.

[8] Vermesan, O. and Friess, P. eds., 2014. Internet of things-from research and innovation to market deployment (pp. 74-75). Aalborg: River Publishers.

[9] OpenFog Reference Architecture for Fog Computing Version 2.09.17 https://www.openfogconsortium.org

[10] EdgeX Foundry https://www.edgexfoundry.org/

[11] Fortino, G., Guerrieri, A., Russo, W. and Savaglio, C., 2014, May. Integration of agent-based and cloud computing for the smart objects-oriented IoT. In Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on (pp. 493-498). IEEE.

[12] Marik, V. and McFarlane, D., 2005. Industrial adoption of agent-based technologies. IEEE Intelligent Systems, 20(1), pp.27-35.

[13] Santos, E.P., 2013. System for monitoring oil level and detecting leaks in power transformers, reactors, current and potential transformers, high voltage bushings and the like. U.S. Patent 8,400,320.

[14] Susa, D., Lehtonen, M. and Nordman, H., 2005. Dynamic thermal modeling of distribution transformers. IEEE Transactions on Power Delivery, 20(3), pp.1919-1929.

[15] Fowler, M., 2004. Inversion of control containers and the dependency injection pattern, Jan. 2004. URL: http://martinfowler. com/articles/injection.html

[16] Candel, A., Parmar, V., LeDell, E. and Arora, A., 2016. Deep learning with H2O. H2O. ai Inc.