In: 2024 International Conference on Cyber-Physical Social Intelligence (ICCSI), 8-12 Nov 2024, Doha, Qatar
DOI:10.1109/iccsi62669.2024.10799418

# An Enabling Architecture for Computational Cost Efficiency in Predictive Maintenance Digital Twins

Ibrahim Abdullahi[1], Stefano longo[1], and Mohammad Samie[1]

[1]School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford, UK, MK43 0AL.

*corresponding author: ibrahim.abdullahi@cranfield.ac.uk

***Abstract*— As digital twins emerge to provide a replication of physical assets in the digital space, the application of predictive maintenance of industrial asset becomes more effective. Developing digital twins for the predictive maintenance case study leverages Internet of Things, cloud computing and machine learning. While these technologies extend the necessary tools for deploying predictive maintenance digital twins, an enabling architecture facilitated by fog computing positions predictive maintenance digital twins for improved computational cost and latency than centralizing in the cloud or running locally at the edge. This work presents the application of a distributed digital twin framework, showing the benefits of better compute utilization and latency by adopting a distributed digital twin framework for predictive maintenance of wind turbine components in a wind farm.**

**Keywords—Digital Twins (DT), Machine Learning (ML), predictive maintenance, computational cost, fog, cloud, edge.**

## I. INTRODUCTION

The emergence of digital twins (DT) has improved asset management and predictive maintenance in industrial internet of things (IIoT) applications [1] [2] [3]. These virtual representations of physical systems enable continuous monitoring, data synchronization, and performance optimization, leading to substantial benefits throughout the system's lifecycle [4].

However, a common challenge in digital twin implementation is the difficulty in accurately modelling every detail of the physical system, particularly for large-scale and complex systems [4]. This often results from computational inefficiencies and bottlenecks, which can hinder the real-time decision-making capabilities essential for effective predictive maintenance.

To address this challenge, this paper proposes an enabling architecture that leverages a hybrid approach, by breaking down the digital twin into layers corresponding to the unit, system, and system of systems digital twin, each with differing levels of model fidelity and computational capacity needs [5] [6]. The core concept is to provide a cost effective architecture by bringing higher compute capacity closer to the data sources, enabling efficient data processing and decision making at the edge, while maintaining a centralized digital twin that synchronizes the full system state and enable high-level optimization with historical data [7].

This builds on the need and benefits of running machine learning algorithms with real time data at the edge, for swift decision making and simulations that can be facilitated by digital twin models. In the Internet of Things (IoT) domain with limited and constrained edge devices, some research outputs have shown that some techniques can balance accuracy trade-off and computational cost thereby enabling both training and inference on edge devices [8] without requiring expensive cloud compute resources. However, for critical industrial infrastructure, such as the wind turbine predictive maintenance in this case study, the proposed architecture combines the strengths of edge computing, cloud computing, and multi-level digital twin models to achieve computational cost efficiency in predictive maintenance applications, for enabling better performance of the DTs.

The key elements of the proposed architecture include decomposing the digital twin instances by modularizing data streaming, processing and inference using the concept of microservices, such as containerization technologies like Docker [9]. The framework used in this work also supports the standardization of the architecture by adopting the ISO23247 standards for digital twin framework for manufacturing [10].

The key contribution of this paper is two-fold, by answering the following questions/hypothesis.

- How can a distributed digital twin framework improve computational efficiency and reduce infrastructure costs for a wind turbine predictive maintenance application?
- What are the enabling technologies and design principles to realize such an architecture that can improve the over solution, from edge to cloud?

The remainder of this paper is organized as follows. Section 2 discusses some related work. Section 3 presents the methodology with a case study on applying the proposed architecture to a wind turbine predictive maintenance use case. Section 4 describes the experiment and results in three scenarios. Section 5 discusses the refinements because of the results and section 6 concludes the paper, mentioning limitations and future research directions.

## II. RELATED WORK

Research on the evaluation of machine learning cost is more prevalent in IoT applications where edge devices usually have limited computational capacity to run machine learning and deep learning applications. Work by [8] has shown that by using model compression techniques, machine learning models can be efficiently deployed on low-power edge devices without compromising accuracy. However, these techniques are yet to be extensively explored in the context of digital twins for industrial applications, where the computational burden is often higher due to the complexity of the physical systems being modeled [11] [12]. The need for a hybrid approach that combines the strengths of edge and

Published by IEEE. This is the Author Accepted Manuscript issued with: Creative Commons Attribution License (CC:BY 4.0).
The final published version (version of record) is available online at DOI:10.1109/iccsi62669.2024.10799418. Please refer to any applicable publisher terms of use.

cloud computing has been highlighted in literature on digital twin architectures [6]. Authors in [5] presented a digital twin architecture that consists of a service system, containing data and algorithms. These two concepts have been shown to enable an important functionality called the prediction feedback loop in [7], where the continuous synchronization between the physical and the digital twin allows utilizing machine learning to generate prediction results used to guide the operations of the physical system.

Providing the necessary compute resources to support the data processing and simulation requirements of industrial digital twins, while maintaining cost efficiency, remains a challenge [4]. The proposed architecture in this paper aims to address this by leveraging edge computing, cloud computing, with a distributed and modularized digital twin framework. Federated learning has been explored as a technique that uses distributed nodes to train machine learning models collaboratively, enabling privacy-preserving and computationally efficient digital twin modeling [13]. With digital twins, predictive maintenance applications can use the feedback loop functionality to optimize the operation of the physical asset by providing an interface for operators to simulate "what-if" scenarios discussed in [6] [7]. Because these scenarios will be running pretrained models with triggers from a global digital twin, a computationally efficient architecture is required.

## III. METHODOLOGY

The methodology adopted for this work involves the deployment of a predictive maintenance digital twin in different scenarios and evaluating the performance by recording some important metrics such, specifically the latency and CPU utilization based on each approach. Earlier work established the framework assessed in this paper is in [7].

The key components of the proposed architecture are:

- **Edge Layer:** This layer consists of sensor nodes that collect and pre-process data from physical wind turbine components. The edge layer is responsible for initial data collection, filtering, and aggregation before transmission to the higher layers. For our experiments, a raspberry pi device and an edge PC were used to represent the edge layer.

- **Fog/Cluster Layer:** This layer receives the pre-processed data from the edge layer and either runs the predictive maintenance models and simulations (feedback) or provides real-time storage and processing in the middle of the edge and cloud. In our experiments, we use a small-scale cluster of 'dockerized' applications [9] running on computers to represent the fog cluster layer. Additionally, the fog layer consists of a set of micro-services that run lightweight real-time data streaming and storage, run models and inference tasks in a distributed manner. As discussed in [7], this enables the digital to physical twin feedback mechanism on receipt of alerts from the cloud layer. With higher compute capacity than the edge layer, it serves the purpose of optimizing resources utilization. The layer uses micro-services such as containerized applications using technologies like Docker [9].

- **Cloud Layer:** This layer hosts the global digital twin model and orchestrates the overall predictive maintenance workflow. It receives the processed data from the fog layer, runs the high-fidelity simulations, send signals to lower layers, and updates the global digital twin accordingly.

The proposed architecture distributes the computational load across the edge, fog, and cloud layers based on the processing requirements and available resources that make the solution cost effective. This methodology uses the architecture in comparing the metrics in order to validate the effectiveness of using such an approach [14] [6] [15] [16]. Figure 1 below depicts the workflow broken down in this validation methodology.

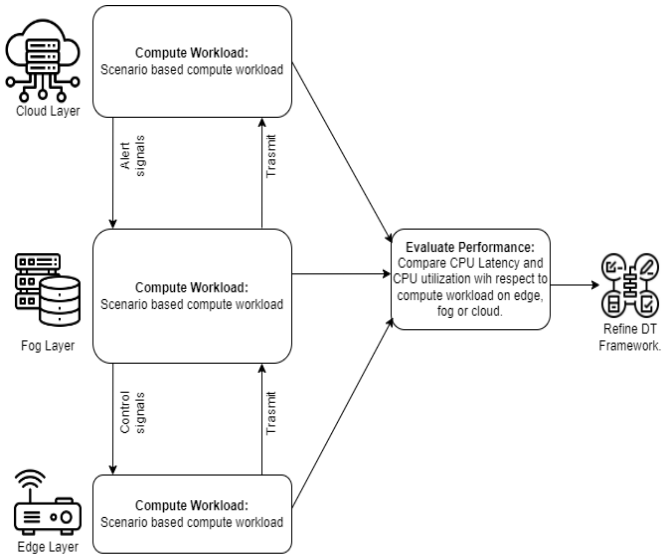



Figure 1: Workflow for methodology/experiments

### A. Case Study

To demonstrate the implementation of the proposed architecture, a case study involving a wind turbine predictive maintenance is presented. Research on the use of digital twin for wind turbine predictive maintenance has been explored in previous works, such as [15] [16], which highlight the benefits, requirements and challenges of using a DT for wind turbines. Work by [7] gives a comprehensive application of a DT for predictive maintenance. However, [7] did not discuss strategies for handling computational bottlenecks. This work therefore addresses the design and implementation constraints of computational latency and CPU Utilization. The following describes the case study in summary.

**DATA:** Real-time operational data from a wind turbine, such as gearbox temperature, vibration, and rotor speed are collected at the edge and transmitted to the fog layer via an MQTT publish/subscribe communication interface.

**MODELS:** The cloud layer does the heavy lifting of training and retraining ML models with historical data, hosting more complex prognostic models, while the fog layer runs lightweight machine learning models to simulate on detected anomalies and predict operational behavior of the system based on impending failures. The models evaluated are

Multiple Linear Regression (MLR), Gradient Boosting (XGBoost) and Long Short-Term Memory (LSTM) based on an Actual run or a feedback run.

- **Actual Run:** Training and Testing of model with full historical data
- **Feedback Run:** Training and testing of simulation based on fault trigger from actual run.

**PLATFORM:** The edge layer is implemented using a Raspberry Pi and an Edge PC, the fog layer is a small cluster of computers running containerized applications, and the cloud layer uses cloud-based infrastructure with GPU.

- **Edge Device:** Raspberry Pi 3: Quad Core 1.2GHz Broadcom BCM2837 64bit CPU 1GB RAM
- **Edge PC:** Intel(R) Core (TM) i5-4590 CPU @ 3.30GHz 3.30 GHz 8GB RAM
- **Fog PC:** 13th Gen Intel(R) Core (TM) i5-1345U 1.60 GHz
- **Cloud PC:** Apple M1 chip (8-core CPU with four performance cores and four efficiency cores, 8-core GPU, 16-core Neural Engine)

**METRICS:** The key metrics evaluated are computational latency and CPU utilization at each layer to assess the cost efficiency of the overall architecture.

- CPU utilization (%) as resource usage efficiency measure.
- Computational Latency (Time (s)) as processing time

### *B. Digital Twin Framework*

Work by [7] discussed a methodology of how a distributed digital twin framework serves the purpose of enabling a prediction feedback loop. The framework simply provides a DT architecture from edge, fog to cloud, and a workflow that supports data gathering, pre-processing, training and inference and a condition monitoring threshold powered by machine learning models. On fault prediction, feedback signals are sent back to the fog layer that regulates the affected component by giving an option of reducing certain parameters in real time. This framework is depicted in figure 2, which forms the basis of the evaluation of the compute workload described in figure 1.

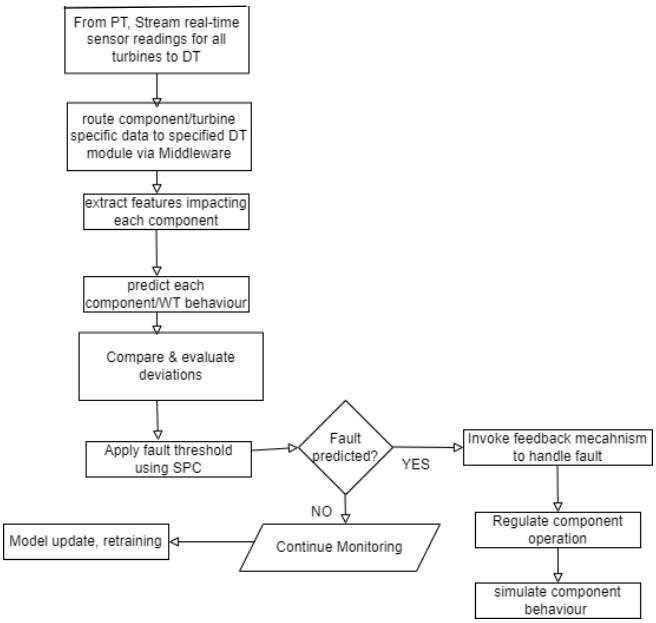



Figure 2: Predictive Digital Twin Workflow with Feedback

## IV. EXPERIMENT AND RESULTS

A series of experiments were conducted to evaluate the performance of the architectures under different load conditions and comparing it to the proposed edge-fog-cloud approach. The following scenarios were used for the evaluation, described in table 1.

| Scenario | Architecture | Description |
|---|---|---|
| A | Edge only | All DT functionalities deployed on raspberry pi device and Edge PC with support from local servers for data storage. |
| B | Edge-cloud only | Global DT functionalities placed in cloud and System and Unit functionalities on edge device and edge PC. |
| C | Edge-fog-cloud | DT functionalities split into a three-layered architecture. |

The scenarios above were used to run experiments on where and how the predictive maintenance DT workload is deployed. From the overview in Figure 1, each of the scenarios were used and the metrics recorded and evaluated.

### *A. Scenario A: Edge Digital Twin Deployment*

This scenario is modelling the traditional legacy systems retrofitted with sensors at the edge. While these sensors may have some compute capability, it is very limited. It relies on local industrial networks and servers top run localized ML models on the devices. The cost of scaling can be substantial.

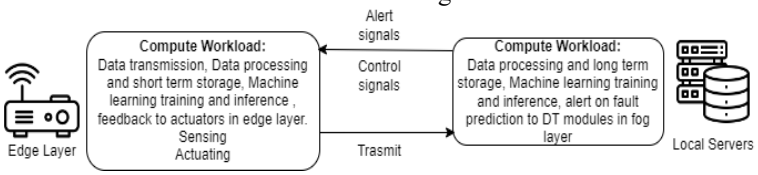



Figure 3: Architecture for Scenario A

The following figure shows the CPU Utilization and computational latency for this scenario by running the three algorithms from our methodology. In Figure 4a, we analyze the CPU utilization for three algorithms during overall training and in figure 4b we show the CPU utilization during feedback simulation on Edge PC. In both instances, both on the edge device and local servers. The hardware capacity could not handle the compute requirements for the DT workload in this scenario. As can be seen in Figure 4, almost 70% of the CPU was occupied running the LSTM model for one turbine component only.

Percentage CPU Utilization by Algorithm/DT Workload (Edge-PC):

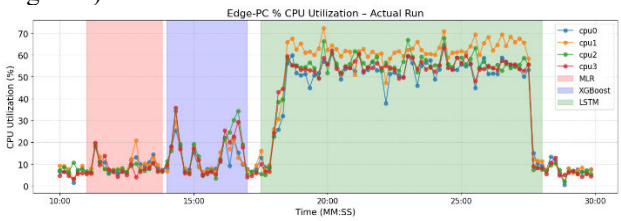



Figure 4a: Edge PC % CPU Utilization – Actual run

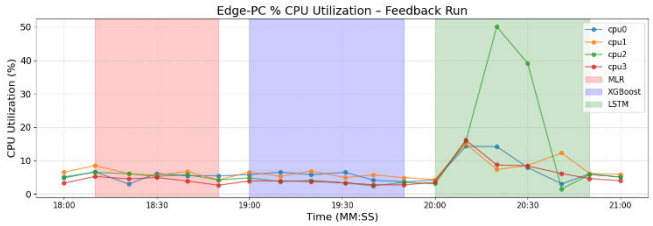



Figure 4b: Edge PC % CPU Utilization – Feedback run

Percentage CPU Utilization by Algorithm/DT Workload (Edge-Rasp):

Figure 5a shows the regular performance by CPU of the raspberry Pi edge device during MQTT publishing of sensor data of some components. Figure 5b shows the performance of running only MLR feedback on Raspberry pi edge.

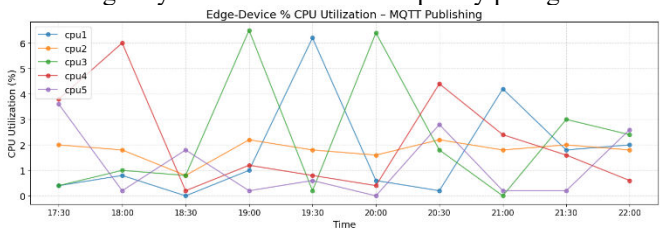



Figure 5a: Edge-Device % CPU Utilization – MQTT Publishing

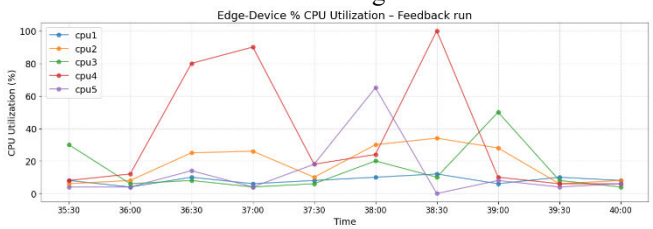



Figure 5b: Edge-Device% CPU Utilization – Feedback run

CPU Latency by Algorithm/DT Workload (Edge-PC):

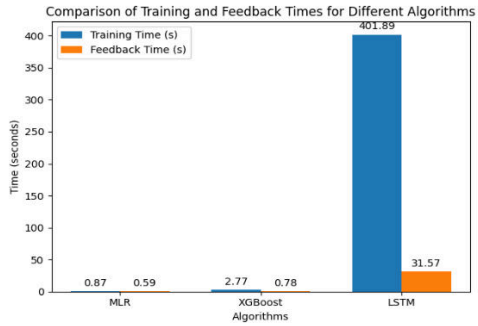



Figure 6a: Edge PC CPU Latency – Actual vs feedback run

CPU Latency by Algorithm/DT Workload (Edge device vs Edge-PC):

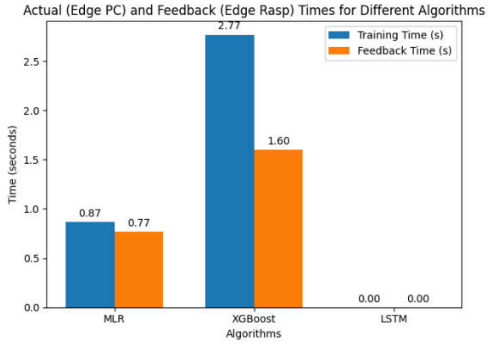



Figure 6b: Edge-PC Actual run vs Edge-Device feedback run CPU Latency

From the comparison in Figure 6b, considering both edge devices show the least latency and CPU utilization, we can see the raspberry PI struggling to handle the feedback algorithms/simulation almost close to what it takes for an actual run in the edge PC. Also, the raspberry pi could not run the LSTM deep learning model for feedback.

### *B. Scenario B: Centralised Cloud-Edge Digital Twin Deployment*

In this scenario, the edge devices are connected directly to cloud systems where the ML solutions are centralized. The edge devices simply transmit the raw sensor data to the cloud, where all the computational and decision making is performed. In this approach, the cloud must process all the data and run all the machine learning models, causing high latency and computational overhead for the edge layer to handle feedback. This scenario represents the traditional cloud-only approach for digital twin implementation.

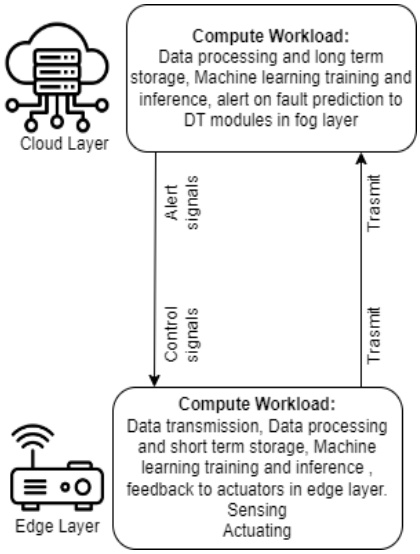



Figure 7: Architecture for Scenario B

From the results in this scenario, we can deduce that the cloud system performs quite well in terms of both latency and CPU utilization percentage for all instances of running the three algorithms in handling the fault of the gearbox component. Figure 8 shows the CPU utilization on the cloud system which is less than 15% in all cases, and figure 9 shows the latency between feedback on edge and actual runs on cloud. LSTM took 149 seconds for an actual run on cloud.

Percentage CPU Utilization by Algorithm/DT Workload:

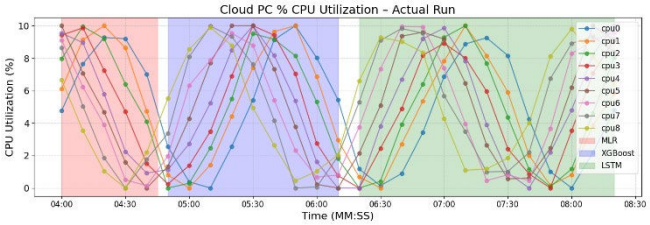



Figure 8: Cloud system % CPU Utilization – Actual run

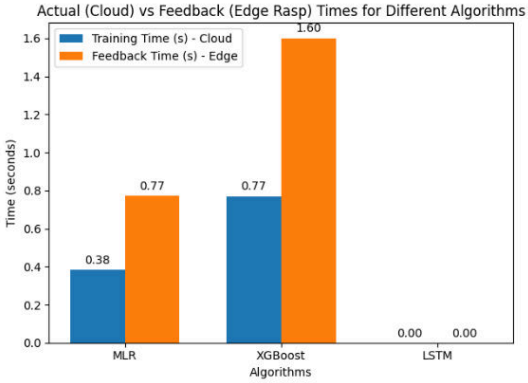



Figure 9: Cloud-Actual vs Edge-Feedback system CPU Latency

### *C. Scenario C: Hybrid/Distributed DT Deployment*

This scenario corresponds to the proposed distributed digital twin framework [6] where an edge-fog-cloud architecture is employed along with a microservices design pattern. In this scenario, the strengths of all layers are utilized to provide computational efficiency leading to a more robust solution. This scenario also adopts microservices for modular DT approach.

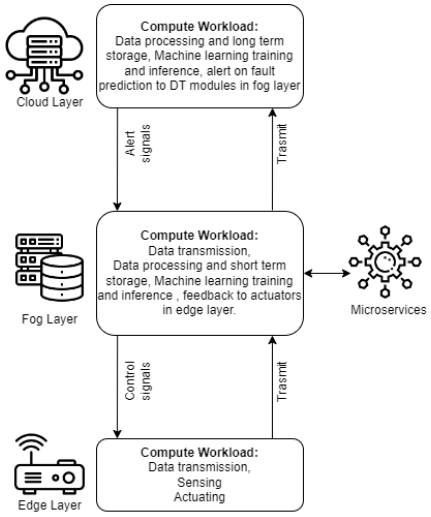



Figure 10: Architecture for Scenario C

This approach involves providing the necessary compute platform to balance the requirements of the DT workload in a middle layer called the fog between the edge layer and the cloud layer. By running the same algorithms as before, we got the following results. Figure 11a and 11b shows the CPU utilization for Actual and Feeback runs, while figure 12 shows the Actual vs Feedback runs in terms of CPU latency, for the gearbox component of 1 wind turbine.

Percentage CPU Utilization by Algorithm/DT Workload:

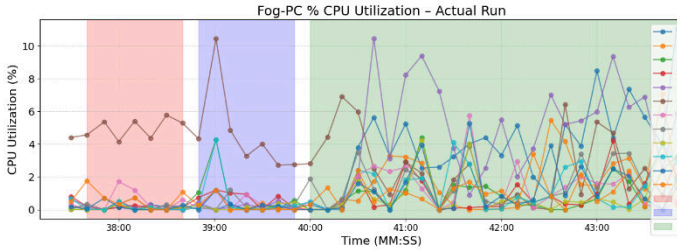



Figure 11a: Fog-PC % CPU Utilization – Actual run

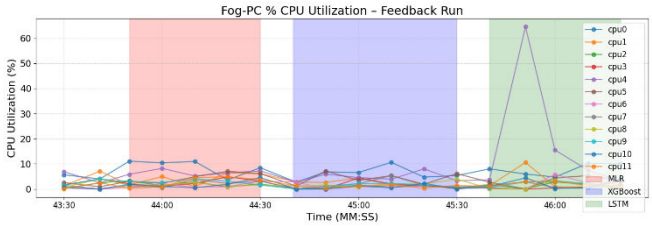



Figure 11b: Fog-PC % CPU Utilization – Feedback run

CPU Latency by Algorithm/DT Workload:

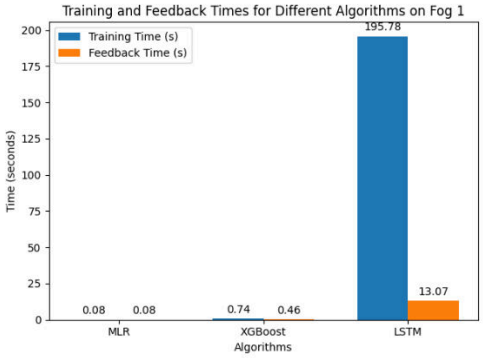



Figure 12: Fog PC CPU Latency – Actual vs feedback run

### *D. SCENARIO C – with Microservices Architecture:*

The fog-based architecture is complemented with a microservices architecture to support the modular DT. When this was implemented using the containerized infrastructure, Docker [9], the following results were achieved. The docker environment virtualizes the host system and takes the 12 CPUs of the fog CPU 1200% available. From the following results in figure 13a for actual run and figure 13b for feedback run, we can deduce that containerizing the DT modules in the fog layer also enhances both CPU utilization and latency as Figure 14 further shows the Actual vs Feedback latency.

Percentage CPU Utilization by Algorithm/DT Workload:

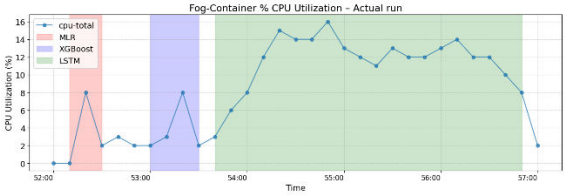



Figure 13a: Fog-Container % CPU Utilization – Actual run

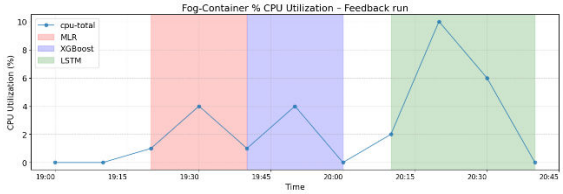



Figure 13b: Fog-Container % CPU Utilization– Feedback run

CPU Latency by Algorithm/DT Workload:

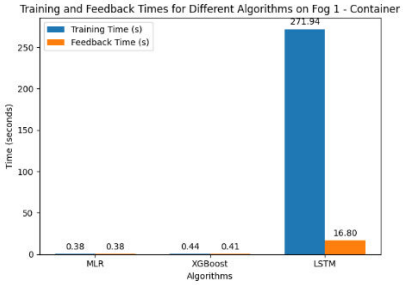



Figure 14: Fog-Container CPU Latency – Actual vs feedback

## V. DIGITAL TWIN FRAMWEWORK REFINEMENT

From the results highlighted in the Scenarios A, B and C. We can see that Scenario C offers the best approach in terms of feedback with a negligible CPU utilization percentage during feedback. The placement of the actual training workload for the DT can be seen to take the best performance in the cloud system. This supports the hypothesis that Scenario C, which balances the DT workload in the fog layer for feedback between the cloud and the edge is valid and stands to improve the prognostics of the wind turbines by allowing operators simulate feedback instantly at the fog with enabling compute. This is also because the edge layer struggles to perform optimally. Using the approach of Scenario C, a simulation of concurrently running multiple algorithms for multiple components and received the same results.

The ISO 23247 framework provides a guiding principle towards standardizing DT development for different use cases [10]. Work by [7], applied this to a predictive maintenance digital twin. In this work, we analyzed the computational performance of a case study for deploying a predictive maintenance digital twin of a wind turbine component. Figure 1 below shows a high-level mapping of the scenario C architecture to the four layers of ISO 23247 [10] [7].

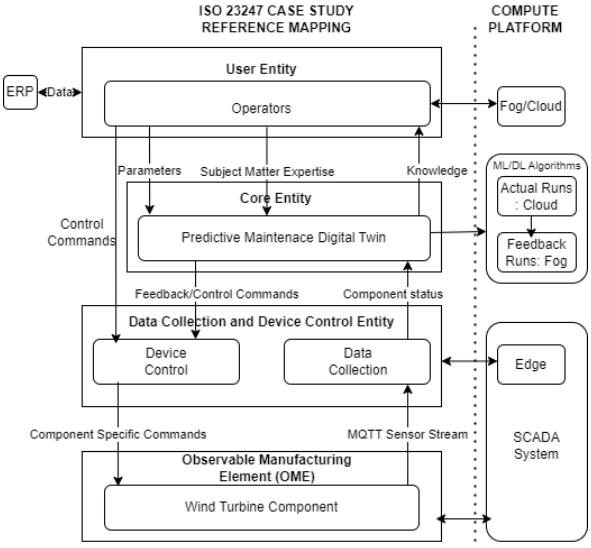



Figure 15: ISO 23247 mapping to Scenario C

## VI. CONCLUSION AND FUTURE WORK

This work extended research on a distributed DT framework by comparing three architectural approaches of how components in an industrial Internet of Things (IIoT) systems collect, process and effect decisions from machine learning and deep learning models. From comparing the metrics, the results answers the research questions by inferring that by splitting the DT work load into unit level to run on Edge layer, system level to run on Fog layer and System of Systems (i.e. Global DT) to run on cloud layer, a more computationally efficient predictive maintenance DT is achieved. The complementing of the architecture with a microservices driven approach further enhances performance. Future work will cover limitations in scalability and integration with common cloud platforms, along with security considerations for integration with a real-world industry setting.


## ACKNOWLEDGMENT

This work acknowledges support form the Petroleum Technology Development Fund (PTDF), Nigeria, and Cranfield University's Digital Aviation Research & Technology Center (DARTEC), UK.


## REFERENCES


[1] I. Rojek, D. Mikołajewski and E. Dostatni, "Digital Twins in Product Lifecycle for Sustainability in Manufacturing and Maintenance," *MDPI - Applied Sciences,* vol. 11, no. 1, 2021.

[2] Q. Lu, X. Xie, A. K. Parlikad, J. M. Schooling and E. Konstantinou, "Moving from building information models to digital twins for operation and maintenance," *Smart Infrastructure and Construction,* vol. 174, no. 2, pp. 46-56, 2021.

[3] S. H. Khajavi, N. H. Motlagh, A. Jaribion, L. C. Werner and J. Holmström, "Digital Twin: Vision, Benefits, Boundaries, and Creation for Buildings," *IEEE Access,* vol. 7, pp. 147406 - 147419, 2021.

[4] A. Thelen, X. Zhang, O. Fink, Y. Lu, S. Ghosh, B. D. Youn, M. D. Todd, S. Mahadevan, C. Hu and Z. Hu, "A comprehensive review of digital twin — part 1: modeling and twinning enabling technologies," *Structural and Multidisciplinary Optimization,* vol. 65, 2022.

[5] H. Park, A. Easwaran and S. Andalam, "TiLA: Twin-in-the-Loop Architecture for Cyber-Physical Production Systems," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, Abu Dhabi, 2019.

[6] I. Abdullahi, S. Perinpanayagam and I. Hamidu, "A Fog Computing Based Approach Towards Improving Asset Management and Performance of Wind Turbine Plants Using Digital Twins," in *2022 27th International Conference on Automation and Computing (ICAC)*, Bristol, UK, 2022.

[7] I. Abdullahi, S. Longo and M. Samie, "Towards a Distributed Digital Twin Framework for Predictive Maintenance in Industrial Internet of Things (IIoT)," *MDPI - Sensors,* vol. 24, no. 8, 2024.

[8] O. Gómez-Carmona, D. Casado-Mansilla, F. A. Kraemer, D. Lopez-de-Ipina and J. Garcia-Zubia, "Exploring the computational cost of machine learning at the edge for human-centric Internet of Things," *Future Generation Computer Systems,* vol. 112, pp. 670-683, 2020.

[9] Docker, "Develop faster. Run anywhere.," Docker, 01 01 2024. [Online]. Available: https://www.docker.com/. [Accessed 2 July 2024].

[10] G. Shao, S. Frechette and V. Srinivasan, "An Analysis of the New ISO 23247 Series of Standards on Digital Twin Framework for Manufacturing," in *ASME 2023 18th International Manufacturing Science and Engineering Conference*, New Brunswick, New Jersey, 2023.

[11] B. A. Talkhestani and M. Weyrich, "Digital Twin of manufacturing systems: a case study on increasing the efficiency of reconfiguration," *Automatisierungstechnik,* vol. 68, no. 6, 2020.

[12] G. Shao and M. Helu, "Framework for a digital twin in manufacturing: Scope and requirements," *Manufacturing Letters,* vol. 24, pp. 105-107, 2020.

[13] O. San, S. Pawar and A. Rasheed, "Decentralized digital twins of complex dynamical systems," *Nature - Scientific Reports,* 2023.

[14] A. R. Al-Ali, R. Gupta, T. Z. Batool, T. Landolsi, F. Aloul and A. A. Nabulsi, "Digital Twin Conceptual Model within the Context of Internet of Things," *MDPI - Future Internet,* vol. 12, no. 10, 2020.

[15] M. Wadhwani, S. Deshmukh and H. S. Dhiman, "Digital Twin Framework For Time To Failure Forecasting Of Wind Turbine Gearbox: A Concept," *TechRxiv,* 2022.

[16] O. O. Olatunji, P. A. Adedeji, N. Madushele and T.-C. Jen, "Overview of Digital Twin Technology in Wind Turbine Fault Diagnosis and Condition Monitoring," in *2021 IEEE 12th International Conference on Mechanical and Intelligent Manufacturing Technologies (ICMIMT)*, Cape Town, 2021.



https://dspace.org/

# An enabling architecture for computational cost efficiency in predictive maintenance digital twins

**Abdullahi, Ibrahim; Longo, Stefano; Samie, Muhammad**

**2024-11-08**
IEEE
https://doi.org/10.1109/iccsi62669.2024.10799418
Abdullahi I, Longo S, Samie M. (2024) An enabling architecture for computational cost efficiency in predictive maintenance digital twins. In: 2024 International Conference on Cyber-Physical Social Intelligence (ICCSI), 8-12 Nov 2024, Doha, Qatar

Downloaded from a DSpace repository. https://dspace.org/