Siemens and EdgeX IIoT Platforms: A Functional and Performance Evaluation

R. Venanzi, M. Solimando, M. Patrali, L. Foschini and P. Chatzimisios, "Siemens and EdgeX IIoT Platforms: A Functional and Performance Evaluation," *ICC 2023 - IEEE International Conference on Communications* keywords: {Performance evaluation;Companies;Foundries;Fourth Industrial Revolution;Industrial Internet of Things;Business;IIoT platforms;IoT Solutions;Industry 4.0;Edge Computing;Siemens Industrial Edge;EdgeX Foundry}, , Rome, Italy, 2023, pp. 834-839.

The final published version is available online at: https://doi.org/10.1109/ICC45041.2023.10278750

Siemens and EdgeX IIoT Platforms: A Functional and Performance Evaluation

Riccardo Venanzi*, Michele Solimando*, Marina Patrali†, Luca Foschini* and Periklis Chatzimisios†‡

*DISI - Dipartimento di Informatica Scienze e Ingegneria, University of Bologna
Viale Risorgimento, 2, 40136 Bologna, Italy
Email: {riccardo.venanzi, michele.solimando, luca.foschini}@unibo.it

†Department of Information and Electronic Engineering, International Hellenic University,
57400, Thessaloniki, Greece
Email: marina.patrali@ieee.org, pchatzimisios@ihu.gr

‡Department of Electrical and Computer Engineering, University of New Mexico,
Albuquerque, NM 87106, USA

***Abstract*—Recently, the Industrial Internet of Things (IIoT) and Industry 4.0 paradigms have attracted considerable relevance by leading all manufacturing companies to invest in IIoT solutions in order to be competitive on the market. Despite the huge number of IIoT solutions on the market, all platforms refer to a common architectural model and share the same functionalities. Among these functionalities, the most relevant is the capability of deploy and run custom containerized applications. This feature has a pivotal role of customizing the IIoT platform and tailoring it to each business scenario. In this paper, we define the general architectural model and common functionalities of the IIoT platforms. Then, we analyze and compare two of the most used IIoT solutions on the market, Siemens Industrial Edge, and EdgeX Foundry. We also define three metrics for evaluating the functionality of deploying custom services on the edge, and we test these two platforms. Finally, we present their comparative performance, advantages, and limitations.**

***Index Terms*—IIoT platforms, IoT Solutions, Industry 4.0, Edge Computing, Siemens Industrial Edge, EdgeX Foundry**

I. Introduction

In recent years, the Industrial Internet of Things (IIoT) and the Industry 4.0 paradigms have gotten considerable relevance and they became cornerstone concepts for the industry digitalization [1], [2], by also paving the way for new industrial research horizons [3], [4]. Cloud and edge computing are crucial enablers of this new scenario by providing solutions capable of exploiting the large amount of data generated by production lines and industrial machines and transforming it into value for manufacturing companies. These cloud-edge solutions are based on the interposition of edge nodes between the industrial shop floor, Operational Technology (OT) level, and the IT/cloud applications (IT level). The edge nodes gather data from the industrial plants and enable the collection, and analysis of data on-premises [5]. In addition, edge solutions enhance data security, and relegation, by keeping and processing sensitive data within the company instead of uploading them to the cloud. These features bring to the companies a significant increment of economic benefits with lightening production costs.

Recently, the advent edge solutions with their benefits lead all manufacturing companies to invest in IIoT solutions in order to be competitive on the market. This provisioning rush has paved the way for the birth of many IIoT platforms [6]. In spite of the huge number of Industrial IoT platforms on the market, all these platforms refer to a unique general model that carries out the Industry 4.0 principles and answers to companies' needs.

The much sought-after feature of IoT platform users is the ability to run custom services at the edge of the network. The openness of an IoT platform to run third-party, and self-developed applications in a plug-and-play manner is the key enabler for tailoring the IIoT solution to the company's business goals. This feature is pivotal for the companies that exploit it for building complex applications on the edge that transform the data coming from the industrial plant into actual value. In this work, we first introduce and detail the general model to which each IIoT platform refers, then, we present all the common functionalities and features these IoT solutions share and implement. Among the several IIoT solutions, we selected two of the most widely used in the industry, one proprietary and one open-source, namely Siemens Industrial Edge (IE) and EdgeX Foundry, respectively [7], and [8]. The contribution of this work is twofold, on the one hand, it proves that we can have the same functions and services deployed on different IIoT edge ecosystems, and on the other hand, it covers the gap in the literature about the direct comparison of functional requirements and performance between the different edge platforms. In addition, by focusing on the key functionality of deploying and running custom services, we defined three crucial metrics under which to test this functionality, *service deploy time*, *number of dropped requests*, and *faulted service recovery*. Finally, we tested the target IIoT platforms under the defined metrics and we have shown the experimental results along with a detailed comparison of the two solutions.

II. IoT Platforms and Edge Functionalities

This section presents the general reference model of IIoT platforms, its common functionalities, and how the two target platforms implement such model and functionalities.

A. IoT Platforms: Model and Functionalities

In spite of the huge number of Industrial IoT solutions on the market, all platforms refer to a general model that shares common functionalities. This model, depicted in Figure1, is composed by the typical key modules on which every IoT platform builds its solution. This model represents the functional architecture of the edge nodes which are typically deployed on the OT level of the factories, on the shop floor. The first two bottom layers depicted in Figure 1, are strictly dependent by the node itself not by the IoT platform running on it. It is worth to notice that the Specific OS layer is a Linux-based OS most of the time. This choice is relevant for two main reasons, the first one is the high lightweight of some Linux distributions perfectly suitable for running on resource-constrained devices, secondly, and more relevant, Linux is the native OS for running and supporting Docker containers. On top of the first two layers, there are four layers strictly dependent by the IoT platform running on the edge nodes. These four layers share the same functionalities but they have different implementations depending on the manufacturer. **Southbound Communication & Field Connectors** layer is the closest to the shop floor and it is responsible to bridge the machinery/production line with the edge node and the whole IIoT solution. This contains all those components that enable the communication with the industrial machines through many field protocols, OPC-UA, MODBUS, MQTT, Fanuc, just to name a few. IIoT platforms usually provide some connectors ready to use, an SDK for developing custom connectors, and/or a kind of marketplace on which the final user can buy field connectors.

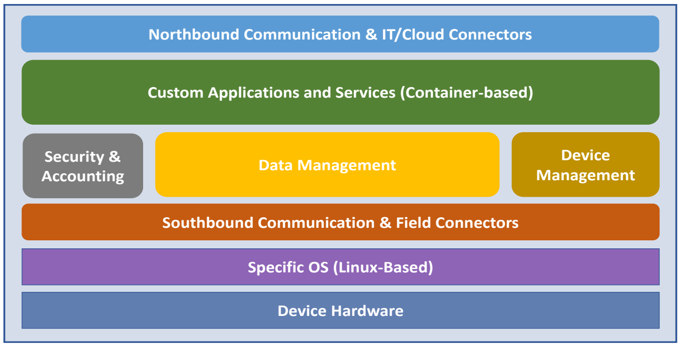
Northbound Communication & IT/Cloud Connectors
Custom Applications and Services (Container-based)
Security & Accounting
Data Management
Device Management
Southbound Communication & Field Connectors
Specific OS (Linux-Based)
Device Hardware

Fig. 1. Architectural Model of IIoT Platforms

Platform Services are strictly related to the IIoT platform and they include all the functionalities for managing the platform itself. Inside this category, there are three modules responsible of three crucial aspects, **Data Management**, **Device Management**, and **Security & Accounting**. The first module includes all the platform services responsible for managing data coming from the field connectors. Among these services, there is always a message bus, typically implemented with a pub/sub paradigm, that delivers commands and data to the other services running on the platform (both custom or native). The device Management module includes all those services in charge to start/stop platform services, monitoring their health and status, getting system metrics (CPU/MEM), and configuring the platform. Finally, the Security & Accounting module provides all the functionalities responsible for access regulation, data privacy, and role management.

Custom Application and Services is a layer that contains third-party and custom user applications. Every IIoT platform provides a mechanism for running custom applications on the platform. This layer is the most important and interesting because it represents the major entry point for user customization of the platform. By running its own application services, end users can tailor the IIoT solution on their own industrial use cases. The services running within this layer are containerized application running on the edge node, and in most of IIoT platforms, those services are Docker containers. The application services running at this layer contain the pure business logic of companies' use cases, they receive data from underneath layers and then they analyze, process, and work on these data in order to produce value. Since it is not trivial deploying, manage and run complex custom containerized applications on the edge node, each platform adopts its own policies, procedures, and methodologies. Similarly to the first layer, **Northbound Communication & IT/Cloud Connectors** contains all the elements and connectors that enable the communication towards IT layer or cloud. This layer bridges the OT layer with IT layer by enabling the forwarding of processed data coming from the previous layer to IT or cloud applications for long-term storage, machine learning, and/or analytics. Typically, IIoT platforms provide cloud connectors ready to use (off-the-shelf component) for the major cloud providers, or alternatively, they provide tools for developing custom connectors.

Once we have detailed the general architectural model of the IIoT platforms and all their common shared functionaries, we consider a proprietary and a open source platform (Siemens IE and EdgeX Foundry, respectively) and we study how these two solutions implement the reference model.

B. Siemens Industrial Edge

Siemens IE is one of the proprietary most adopted edge solution on the market for Industry 4.0 scenarios. Siemens IE is a multi-layer solution that involves both of IT and OT levels. The Siemens platform, indeed, is composed by two entities, the Industrial Edge Management (IEM) and the Industrial Edge Device (IED). The IED is the actual edge node and it thought to be deployed on OT level close to the industrial machines. On the other hand, the IEM is a component deployed on IT layer that is responsible of managing and controlling the fleet of IEDs. The IEM is thought to be deployed anywhere in the IT layer typically inside the industrial site. In a real industrial deployment, there can be multiple IEDs and a IEM that manages and controls all the devices. All the components and functionalities described in subsection II-A are diffused between IEM and IED. More in detail, the user defines all the connectors, docker applications, and policies (i.e. security, role definitions, and software updates) on the IEM and then they are deployed on IEDs. The architecture of IED implements the model described in the previous subsection and it is depicted in Figure 2.

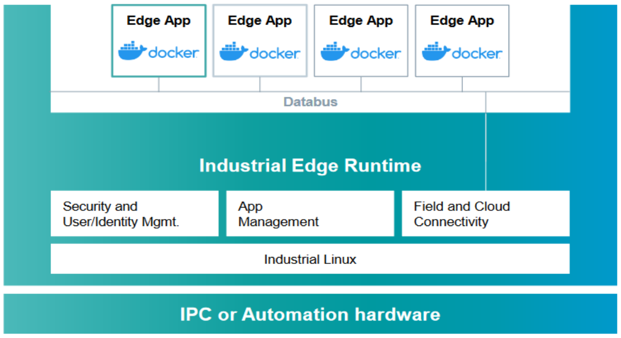

Fig. 2. Siemens Industrial Edge Device Architecture

Figure 2 depicts how the Siemens IED architecture implements the general model of the IIoT platforms. More in detail, the system, called Industrial Edge Runtime, runs on an industrial Linux version that includes a Docker engine for running containerized custom, third-party, and native applications. Siemens IE provides the **Field and Cloud Connectivity** module as implementation of Southbound Communication & Field Connectors layer presented in II-A. It contains field connectors to communicate with the industrial machines using the major protocols. These connectors have to be created and configured on IEM and then they can be deployed on the IEDs. Siemens provides a marketplace, named IE Catalog, from which it is possible to buy and download both of field and Cloud connectors. The IED provides the **App Management** and **Security and User/Identity Mgmt** modules as part of the implementation of Platform Services. These services along with a pub/sub system based on MQTT, called IE Databus, are responsible for data management and delivery. The first module implements functionalities for app managing and node configuration, such as application start/stop, status monitoring, and device general settings. The module of Security and User/Identity management regulates the access control based on the user roles, and it decides who can do what. The security on Siemens IED is regulated by certificates and user/password access. IED is capable of running custom Applications and Services called **Edge App**. They are docker-compose applications loaded on the IEM and then deployed on the IED. The docker-compose file has to respect certain criteria in order to be compliant with Industrial Edge Runtime. The custom Edge App can communicate to each other or with other Siemens applications through the IE Databus.

C. EdgeX Foundry Platform

EdgeX Foundry, briefly EdgeX, is an open-source, vendor-neutral IIoT platform recently developed by Dell in its own edge gateway project and now hosted by the Linux Foundation. EdgeX acts as a data gateway at the edge and its main goal is to bridge field devices (i.e. industrial machines) to IT applications and systems. This solution is composed by a single entity, typically deployed on the edge, OT level. EdgeX architecture is composed by microservices which make it deployable both on a single node at the edge level and on multiple nodes between IT and/or OT levels. The EdgeX architectural model is depicted in Figure3 and it is composed by four service layers that implement the model described in sub-section II-A. The first layer, **Device Services** (**DS**), implements the Southbound Communication & Field Connectors of the model. It is the layer responsible of communicating with the shop floor machinery and running connectors. A DS is the EdgeX entity capable of reading data from the field and actuating commands on industrial machines. Each DS can implement a field protocol and an EdgeX deployment can run multiple DSs at the same time. The **Core Services** (**CS**) layer contains the four essential services for running EdgeX platform, they are responsible of managing and storing data, piloting DSs, keeping metadata and configurations of each registered microservice, and a service registry. The functionalities of Security & Accounting, Device Mmarketplaced data delivery are implemented by the system's services called **Security**, **Management**, and **Support Services**. More in detail, Security implements all the policies to protect the data and to control devices, Management implements all the functionalities for handling and monitoring services and the Support Services are responsible of delivering EdgeX messages and events to the upper layers. Core, Security, Management, and Support implement the Platform Services described in the model. **Application Services** (**AS**) layer is placed on top of CSs and it implements the Custom Application and Services module. This layer hosts the services developed by the users. EdgeX includes a pure Docker engine on which users can deploy and

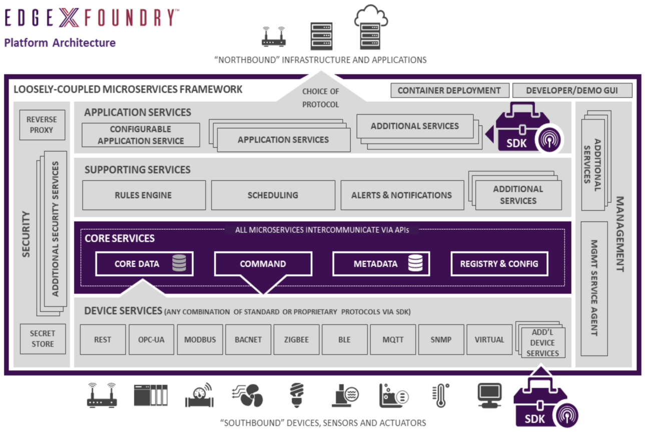

Fig. 3. EdgeX Foundry Platform Architecture

run their own services. This layer contains another type of service, Additional Services that implement the communication with the IT or cloud providers. EdgeX Foundry provides two SDK to allow the user to develop both field and cloud connectors (DSs and Additional Services).

III. Performance Metrics

The two considered platforms provide very similar functionalities but they have different implementations. The only comparable feature is how the platforms allow to deploy and run custom user services. For the sake of that, we focus our study on this aspect by analyzing the performance of the two subject platforms under three metrics, the service deploy time, the number of dropped requests, and a failed service recovery time.

We define the **service deploy time** as the time a platform takes to successfully deploy a service on the edge. More precisely, this time is considered from when the request arrives on the edge platform and when the service is up and running. In real industrial use cases, requests for service deployment might occur in an asynchronous and uncontrolled manner, and how fast the deployment of new services can be achieved is paramount to allow a smooth roll-out of features, upgrades, and to manage correctly the scalability of the services. For this reason, we plan to measure the service deploy time under different request loads to test how the platform's performance scales in response of the increasing requests frequency. This metric takes into account the latency of a service deployment request and if this request succeeds or fails (number of dropped requests), i.e. due to the congestion of the platform's job queues and/or a network packet loss. It is worth to mention that Siemens IE involves two entities, the IEM that receives the requests, and IED where the service is actually deployed.

The other key metric of our work is **service recovery time** after a service failure. Fault recovery plays a central role in IIoT real deployment. The minimization of unplanned service downtime is pivotal to avoid data and value loss.

To calculate the service downtime, we need to understand when the target service fails. We define the failure detection time as the time needed for our tool to detect the failure of the target service on the edge platform. The tool introduces its own latency since it requests the service status to the edge platform by polling (polling time, T_{pol}). We formally define the failure detection time (T_{fd}) as:

$$T_{fd} = T_{pol} + T_{exec}$$

where T_{exec} is the time the platform takes for executing the service status checking request. Once we have defined the failure detection time we can define the service downtime time (T_{down}) as the sum of the time needed for detecting a target service failure (T_{fd}), plus the time to restore the target service on the platform (T_{res}). Formally, the downtime time is defined as:

$$T_{down} = T_{fd} + T_{res}$$

T_{res} is the elapsed time since the platform receives the service restore request until the target service is up and running again. The time for executing a deployment or restoration of a service can be broken down into *pending* and *executing* time. Where *pending* is the time since the request enters in the job queue of the platform up to it is carried out, while the *executing* time is the duration of the process that fulfills the request, i.e. the service is up and running. In the Siemens solution, these times are more emphasized because the job queue is on the IEM, while the execution is on the IED. The IED checks for available jobs in queue with a certain polling time, and it adds delay to the process.

IV. Tests and Experimental Results

This section describes the test environments, the experimental results, and the platforms' performance comparison.

A. Test Environment

The testbed deployment consists of four elements, a IEM and a IED, for Siemens IE, an edge node running EdgeX, and a Server from which we run scripts and calls. More precisely, the IEM and Server are virtual machines (VMs) deployed in IT layer at one hop distance from edge nodes. The IED and EdgeX nodes are deployed on OT layer at a hop distance from the shop floor. The IED is a Siemens *SIMATIC IPC227E* device, while the EdgeX platform is deployed on a VM, with the same resources of the IED. Table I shows the resources of each entity in our deployment.

TABLE I
Testbed deployment

	IEM	**IED**	**EdgeX**	**Server**
OS Version	IEM OS 1.5.6	IED OS 1.5.0-21	Ubuntu 20.04.4	Ubuntu 20.04.4
RAM(GB)	12	8	8	16
CPU(cores)	4	4	4	8
HD(GB)	200	200	200	150

B. Tests and results

Here, we present the tests on the two platforms run under the metrics described in section III and their performance comparison.

1) Deploy time and Dropped Requests: The deploy time is the time elapsed from when we sent the deployment request up to the new service is up and running on the platform. Siemens IE requires a preliminary login action before enabling the user to deploy a service. This procedure has not taken into account for the latency measurements. We vary the delay among the calls to simulate different loads on platforms. We increase the request loads on the platforms by linearly varying the inter-request time among these values: 0.25, 0.5, 1, 2, 3, and 5 seconds. From now on we call this time delay, which meant the delay between two consecutive requests. We sent 20 requests for each delay time and we repeat each run 10 times in order to have more stable and reliable results. Under these service deployment request loads, we calculate the average deploy time and the dropped requests that the platforms were unable to manage. Figure 4 depicts the trends of the dropped requests in Siemens IE and EdgeX solutions. From Figure 4 emerges that Siemens solution has an increasing number of dropped requests as the load increase, with a delay time between requests under the second, latency 0.25, 0.5, while all the requests are perfectly managed when the latency is more than a second. More precisely, the Siemens platform shows a number of requests dropped of 1, 8, and 15, with a drop rate of 5%, 40%, and 75%, for delays of 1, 0.5, and 0.25 seconds, respectively. On the other hand, EdgeX solution shows a single dropped request out of 20 calls on average only under the heaviest request load, with a drop rate of 5%. Figure 5 depicts the average service deploy time of the target platforms. From Figure 5 emerges that Siemens solution has a quite constant service deploy time on average for each requests load, this time never falls below 30 seconds latency of requests, while, EdgeX performs better the deployment operation with a time that is around 15 seconds in average for each load. The performance of EdgeX improves when the latency among requests is more than a second. The high time that Siemens IE takes to fulfill a service deployment request has a strong influence on the dropped requests. Indeed, when the requests come more frequently, i.e. latency 0.25, and 0.5 seconds, the dropped requests rate is higher, 75%, and 40% respectively. This is because with the time needed to accomplish the single request and those high request loads, the IEM job queue saturates soon and the IEM drops the requests.

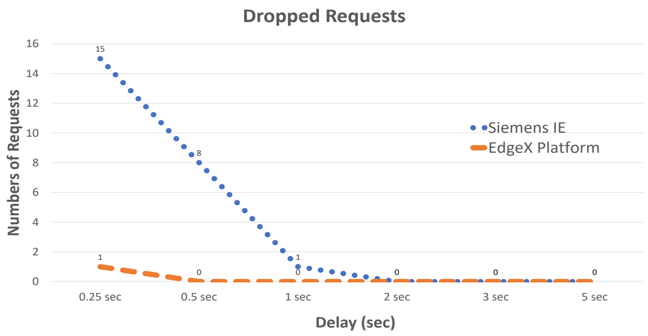

Fig. 4. Dropped Deploy Requests: Siemens IE vs EdgeX

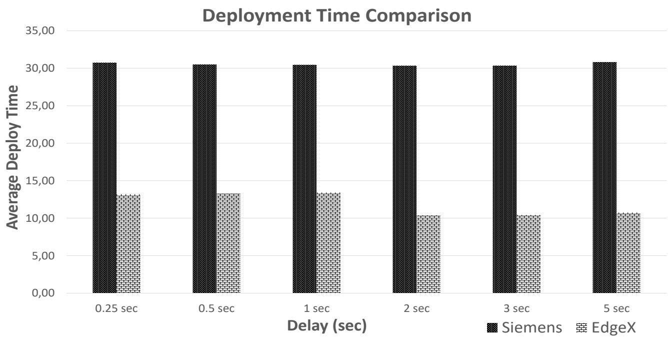

Fig. 5. Average Service Deploy Time: Siemens IE vs EdgeX

2) *Fault Tolerance:* In this subsection, we show the experimental results of downtime metric meant as we described in section III. In Figure 6, we show the comparison of EdgeX and Siemens solutions for service unplanned downtime metric. Every run on the chart is the average of 10 restorations of failed service. From Figure 6 emerges that EdgeX solution has the best performance in case of recovering the operation of a failed service with a time in average around 15 seconds against a time of around sixty seconds on average of Siemens IE solution.

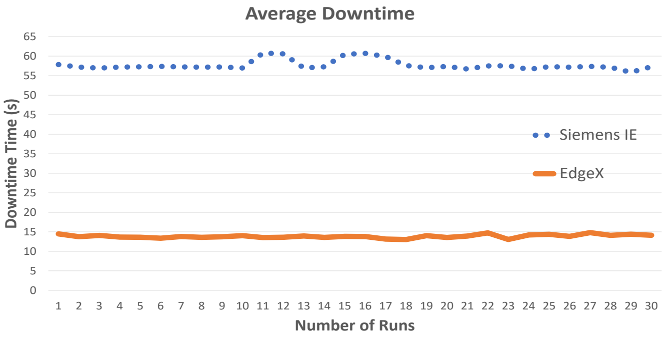

Fig. 6. Average Service Downtime Time: Siemens IE vs EdgeX

From a purely quantitative point of view, by considering the three metrics under test, we can definitely say that the EdgeX Foundry platform performs better than the Siemens IE. It is worth to mention that Siemens IE is a mature ecosystem that involves more entities in its solution, IEM, and IED. In the Siemens solution, the IEM adds an additional layer that provides security, service orchestration, and device management functionalities. This layer adds overhead and accumulates delay in performing operations like service deployment. EdgeX solution, being deployed on a single node, has better performance under the defined metrics, but on the other hand, it does not provide any additional functionalities like multi-device management or service orchestration, that are in charge of the final user.

V. Related Works

While there are several works focused on a single IoT platform [9]–[12], we were not able to find much related research works that discuss the IoT platforms comparison.

In [13], the authors proposed MIINT, a middleware for IIoT platform integration, and they provide a comparison of the field data reading functionality, using MODBUS, of Microsoft Azure IoT and EdgeX Foundry.

A comparison between two proprietary solutions, Amazon AWS Greengrass and Azure IoT, is presented in [19]. Here, the authors compare compute time, time-in-flight, end-to-end latency, payload size, and CPU and memory utilization of the two platforms by concluding that Azure may not be suitable for latency-sensitive applications, while, on the other hand, the AWS Greengrass platform has fewer customization options.

In [14], the authors present a comparison among several IoT platforms, KAA, Thingspeak, Microsoft Azure IoT, Thingsboard, and AWS IoT, but they focused their work in comparing platform-related services such as device management, networking, and data storage, leaving out the performance analysis. In [16] and [15], the authors provide an analysis of the functionalities of IoT platforms. The first focuses on OpenMTC, FIWARE, SiteWhere, and AWS IoT, while the latter focuses on AWS IoT, Azure IoT, Watson IoT, PTC ThingWorx and Google IoT. Both works present architectural similarities and functional features comparison.

Finally, [17] and [18] provide two works in which the authors compare IT big players IoT platforms, such as Azure, AWS, and Google Cloud. The first work presents a comparative analysis of services dedicated to IoT management, by excluding industrial IoT deployment. The latter, instead, focuses the comparison on the performance of a MQTT broker service used as IoT resource connector. Both works leave out open-source and on-premise solutions in terms of IoT platforms for industrial deployments.

Our contribution overcomes the limitations of related work that relies only on proprietary platforms ([19]). In addition, by analyzing platforms suitable specifically for integration with IIoT-kind assets, our work focuses on the area of industrial applications, while some related papers mainly focus on the more general topic of IoT ([17] and [18]). Finally, we conducted a meticulous performance analysis, identifying the bottlenecks between the two platforms under test, and not just the functional requirements they exhibit, as shown in works [14] [15] [16].

VI. Conclusion and Future Directions

This work defined the general architectural model to which every IIoT platform refers along with the common functionalities it provides. We studied Siemens IE and EdgeX Foundry IIoT platforms, and how these two solutions have implemented the presented architectural model and functionalities. In addition, we have defined two metrics for measuring the efficiency with which they provide the functionality of deploying and running custom or third-party services on the edge. We tested the target platforms by measuring the service deploy time, the service deploy dropped requests, and the time for service recovery due to a failure. We presented performance results in comparison by highlighting the advantages and limitations of these two platforms. Finally, in our ongoing efforts, we are extending the comparison of IIoT platforms both in terms of functionalities and different solutions.

References

[1] Okano, Marcelo T. "IOT and industry 4.0: the industrial new revolution." International Conference on Management and Information Systems September. Vol. 25. 2017.

[2] Kagermann, Henning, Wolf-Dieter Lukas, and Wolfgang Wahlster. "Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution." VDI nachrichten 13.1 (2011): 2-3.

[3] Hugh Boyes, Bil Hallaq, Joe Cunningham, Tim Watson, The industrial internet of things (IIoT): An analysis framework, Computers in Industry, Volume 101, 2018, Pages 1-12, ISSN 0166-3615, https://doi.org/10.1016/j.compind.2018.04.015.

[4] Kim, Jin Ho. "A review of cyber-physical system research relevant to the emerging IT trends: industry 4.0, IoT, big data, and cloud computing." Journal of industrial integration and management 2.03 (2017): 1750011.

[5] S. Trinks and C. Felden, "Edge Computing architecture to support Real Time Analytic applications : A State-of-the-art within the application area of Smart Factory and Industry 4.0," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 2930-2939, doi: 10.1109/BigData.2018.8622649.

[6] Sittón-Candanedo, Inés, et al. "Edge computing architectures in industry 4.0: A general survey and comparison." International Workshop on Soft Computing Models in Industrial and Environmental Applications. Springer, Cham, 2019.

[7] https://www.siemens.com/global/en/products/automation/topic-areas/industrial-edge.html

[8] https://www.edgexfoundry.org/

[9] K. Han, Y. Duan, R. Jin, Z. Ma, H. Rong and X. Cai, "Open Framework of Gateway Monitoring System for Internet of Things in Edge Computing," 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), 2020, pp. 1-5, doi: 10.1109/IPCCC50635.2020.9391568.

[10] D. Gupta, S. Bhatt, M. Gupta, O. Kayode and A. S. Tosun, "Access Control Model for Google Cloud IoT," 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE HPSC and IEEE IDS, 2020, pp. 198-208, doi: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00044.

[11] S. Forsström and U. Jennehag, "A performance and cost evaluation of combining OPC-UA and Microsoft Azure IoT Hub into an industrial Internet-of-Things system," 2017 Global Internet of Things Summit (GIoTS), 2017, pp. 1-6, doi: 10.1109/GIOTS.2017.8016265.

[12] J. Waterman, H. Yang and F. Muheidat, "AWS IoT and the Interconnected World – Aging in Place," 2020 International Conference on Computational Science and Computational Intelligence (CSCI), 2020, pp. 1126-1129, doi: 10.1109/CSCI51800.2020.00209.

[13] R. Venanzi, A. Cavalucci, L. Foschini and P. Bellavista, "MIINT: Middleware for IIoT Platforms Integration," 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1-6, doi: 10.1109/GLOBECOM46510.2021.9685653.

[14] Y. Y. Fridelin Panduman, S. Sukaridhoto and A. Tjahjono, "A Survey of IoT Platform Comparison for Building Cyber-Physical System Architecture," 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2019, pp. 238-243, doi: 10.1109/ISRITI48646.2019.9034650.

[15] T. G. F. Barros, E. F. Da Silva Neto, J. A. D. S. Neto, A. G. M. De Souza, V. B. Aquino and E. S. Teixeira, "The Anatomy of IoT Platforms—A Systematic Multivocal Mapping Study," in IEEE Access, vol. 10, pp. 72758-72772, 2022, doi: 10.1109/ACCESS.2022.3189660.

[16] J. Guth, U. Breitenbücher, M. Falkenthal, F. Leymann and L. Reinfurt, "Comparison of IoT platform architectures: A field study based on a reference architecture," 2016 Cloudification of the Internet of Things (CIoT), 2016, pp. 1-6, doi: 10.1109/CIOT.2016.7872918.

[17] A. S. Muhammed and D. Ucuz, "Comparison of the IoT Platform Vendors, Microsoft Azure, Amazon Web Services, and Google Cloud, from Users' Perspectives," 2020 8th International Symposium on Digital Forensics and Security (ISDFS), 2020, pp. 1-4, doi: 10.1109/ISDFS49300.2020.9116254.

[18] P. Pierleoni, R. Concetti, A. Belli and L. Palma, "Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison," in IEEE Access, vol. 8, pp. 5455-5470, 2020, doi: 10.1109/ACCESS.2019.2961511.

[19] A. Das, S. Patterson and M. Wittie, "EdgeBench: Benchmarking Edge Computing Platforms," 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 2018, pp. 175-180, doi: 10.1109/UCC-Companion.2018.00053.