

A Cloud-Edge Orchestration Platform for the Innovative Industrial Scenarios of the IoTwins Project

Alessandro Costantini[1(✉)], Doina Cristina Duma[1], Barbara Martelli[1], Marica Antonacci[2], Matteo Galletti[1], Simone Rossi Tisbeni[1], Paolo Bellavista[3], Giuseppe Di Modica[3], Daniel Nehls[4], Jean-Christian Ahouangonou[5], Cedric Delamarre[5], and Daniele Cesini[1]

[1] INFN -CNAF, Bologna, Italy
alessandro.costantini@cnaf.infn.it
[2] INFN-Bari, Bari, Italy
[3] Università di Bologna, Bologna, Italy
[4] Fraunhofer FOKUS, TU, Berlin, Germany
[5] ESI GROUP, Rungis, France

Abstract. The concept of digital twins has growing more and more interest not only in the academic field but also among industrial environments thanks to the fact that the Internet of Things has enabled its cost-effective implementation. Digital twins (or digital models) refer to a virtual representation of a physical product or process that integrate data from various sources such as data APIs, historical data, embedded sensors and open data, giving to the manufacturers an unprecedented view into how their products are performing. The EU-funded IoTwins project plans to build testbeds for digital twins in order to run real-time computation as close to the data origin as possible (e.g., IoT Gateway or Edge nodes), and whilst batch-wise tasks such as Big Data analytics and Machine Learning model training are advised to run on the Cloud, where computing resources are abundant. In this paper, the basic concepts of the IoTwins project, its reference architecture, functionalities and components have been presented and discussed.

Keywords: Digital twins · Digital models · IoT · Edge · Cloud

1 Introduction

Many projects have been developed under the European Commission financial project Horizon 2020, to sustain and promote research in IOT, edge and cloud technologies in the scientific and industrial fields. The European Open Science Cloud Hub [1] contributes to the development of a Hub for European researchers and innovators to discover, access, use and reuse a broad spectrum of resources for advanced data-driven research. The above investments, which could be perceived as cost-prohibitive, are now becoming a business necessity for innovative

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12950, pp. 533–543, 2021.
https://doi.org/10.1007/978-3-030-86960-1_37

companies, in both the manufacturing and facility management domains, with particular attention to the rich ecosystem of EU small and medium enterprises (SMEs).

This is also the case of the H2020 IoTwins project that, by designing a framework for a seamless, straightforward and loose integration of already developed and deployed industrial software components running in typical long-lived industrial test-beds, it aims to lower the barriers for building edge-enabled and cloud-assisted intelligent systems and services based on big data for the domains of manufacturing and facility management.

The IoTwins project propose a highly distributed and hybrid digital twin model, which provides for an integration of simulative and data-driven models to feed AI services. This will be tackled by harmonising standards to enable interoperability, and by developing an easy-to-use service layer that facilitates and decreases the cost of integration and deployment.

IoTwins leverage on the experience and the outcomes from past EU funded projects such as DEEP-Hybrid-DataCloud (DEEP-HDC) [2] aimed at bridging cloud and intensive computing resources to explore large datasets for artificial intelligence, deep, and machine learning, eXtreme-DataCloud (XDC) [3] aimed at developing scalable technologies for federating storage resources and managing data in highly distributed computing environments and INDIGO-DataCloud [4] aimed at developing a computing platform, deployable on multiple hardware, and provisioned over hybrid infrastructures.

In this paper the IoTwins project platform is presented and discussed in Sect. 2, where the proposed architecture and the platform capabilities derived form the use case driven analysis are described. In Sect. 3 the IoTwins software stack and the related IoT, Edge and Cloud service components are treated, including the interaction among them. Section 4 summarises the activity performed and draft the related conclusions.

2 IoTwins Project Platform

2.1 Proposed Architecture

IoTwins will provide during its activity the needed support to the users for building and running their services by deploying a platform providing a single point of access to heterogeneous computing, high capacity storage and interconnected resources.

The platform will deliver interfaces to data analytics and AI techniques, physical simulation, optimizations and virtual lab services. In addition, the IoTwins platform will apply well established and emerging standards to enable communication- and data-level interoperability to 12 different use cases coming from (but not limited to) the industrial IoT domain.

The proposed computing architecture is drafted in Fig. 1.

The figure shows the different computing layer of the platform and the different interactions among them, including the interfaces between IoT and Edge layers and between Edge and Cloud layers. In particular we have:

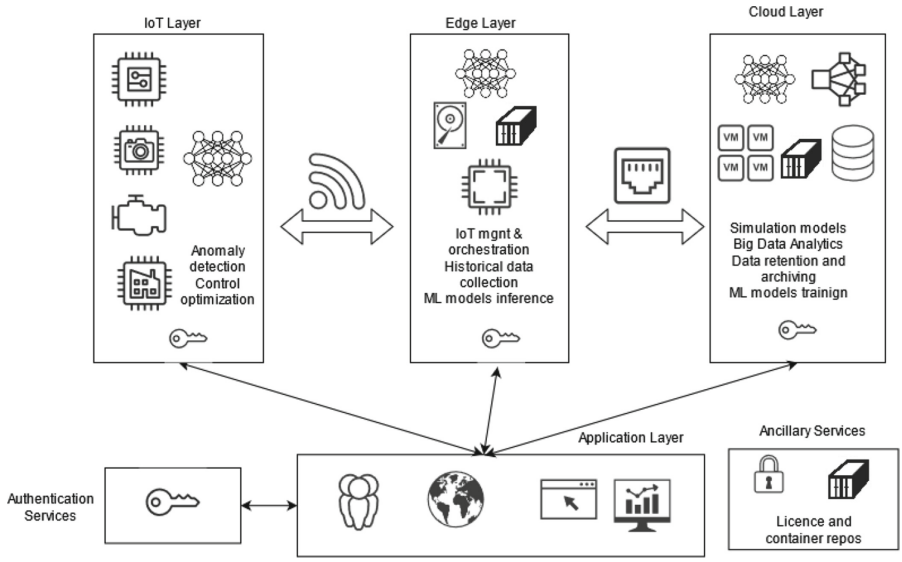

Fig. 1. IoTwins high level architecture

- Internet of Things (IoT) layer
- Interface between IoT and Edge layers
- Edge Layer
- Interface between Edge and Cloud layers
- Cloud Layer
- Application Layer

2.2 Use Case Requirements

The IoTwins User is the main stakeholder that will benefit from the services offered by the IoTwins platform. In the scope of the project, the IoTwins User role is played by Testbeds, who have contributed with their use cases to identify the requirements of the platform.

In Table 1, the different use cases are categorized according to the infrastructure where computing tasks are run. Furthermore, the platform will provide the User with services to build "data processing chains", i.e., a collection multiples services, distributed on at least two different infrastructures, chained together to serve the specific computing needs.

The "IoT-related use cases" refer to the action of configuring and running computing tasks on data sensed by sensors that the IoT device is equipped with. Depending on the need and the capabilities of the device, computing tasks may consume data on-the-fly or data at rest.

"Edge-related use cases" configure data processing allows the User to set-up an environment for data elaboration. Configuration involves actions like a) setting up data sources address, type, format, b) choosing and setting up the

Table 1. Mapping among IoTwins infrastructure, use case and their computational tasks

Reference infrastr.	Use case flow	Computational task
IoT-related	Configure data processing and run data processing	Data on-the-fly or data at rest
Edge-related	Configure data processing, Run bulk-data processing, Run data stream processing and Run ML Model	Data stream processing
Cloud-related	Configure data processing, Run bulk-data processing, Run data stream processing, Run ML training and run simulation	Data streams from either IoT or Edge or data-at-rest stored in the cloud

parameters of a specific computing task, c) selecting the destination of the data output, etc. Run bulk-data is the action of invoking tasks that elaborate data at rest, i.e., data locally stored, while Run data stream processing refers to the elaboration of live data coming from IoT devices. Finally, the Run ML model is a sample data stream processing use case.

In "Cloud-related use cases" instead, most of use cases available in the Edge are available in the Cloud too. The User can configure and run computing tasks on data streams coming from either IoT or Edge, or on data-at-rest stored in the Cloud. For instance, the User can request to run Simulations (Run Simulation use case) or to train ML models (Run ML training case) on bulk data residing in the Cloud. Trained ML models can then be moved to the Edge where, fed by data streams coming from IoT devices, will execute (see Edge's Run ML Model use case).

Out of the project scope, potential Users will be players of the manufacturing and facility/infrastructure management sectors that intend to benefit of the IoTwins platform services to implement a digital twin-based management of their business processes.

2.3 Platform Capabilities

The IoTwins platform is proposed as an open framework enabling Users to configure and run data processing tasks on IoT, Edge and Cloud respectively. It manages digital twins in the Cloud-to-Things continuum, built on top of open-source software and tools, allowing third-party software to integrate by way of open application programming interfaces (API).

The main principles that drive the design of the IoTwins platform are openness and software re-use respectively.

Starting from this principle, in fact, the platform will provide different services listed hereafter.

Data Handling. As one of the Digital Twins' cornerstones, Data Handling becomes extremely important in order to represent (or even forecast) the monitored asset. It is divided into three different consecutive steps: Data gathering, Data transportation and Data storage.

Data Gathering is performed by the sensing devices of the testbeds. The variety of sensors involved in the testbeds is quite wide. Thus, the platform has to be able to adapt itself, in order to enable the gathering of different data types and sources.

Data Transportation provides the protocols used to implement communication in the IoTwins architecture depend on the level (IoT, edge, cloud/backend). Currently, data transportation is performed by adopting standard (or de facto standard) technologies to enable all required data transportation and, at the same time, minimize the number of different technologies used by finding common particularities.

Data storage solutions (Data Models (D.M) and database) will be used either as a preliminary storage or as long terms storage for the testbeds.

Computation. Computation is requested to be done at all levels: IoT, Edge and Cloud.

At the IoT level few computation tools may be needed. The tools may be provided either by the data collection tool or as a service of the platform. Light and quick calculation may be done at the IoT level for some types of devices

At the Edge level operation like filtering, data aggregation or Big Data Analytics may be needed.

Cloud level will be mostly dedicated to off-line computations. Simulations and ML training will be performed at this level. Obviously, all the above listed computations (IoT & EDGE levels) may be performed here.

Anonimization. Anonymisation consists of replacing/encrypting/deleting elements directly or almost directly identifying individuals, generally does not protect individuals. Anonymisation of data is a compromise between data and individual protection on the one hand, and the possibility to process data in an interesting way on the other.

Due to the fact that the most of the testbed comes from industries, those data need to be kept under strict confidentiality ruled. So a process of anonymization that prevents from whatsoever association between data, their elaboration and inferenced conclusions is needed before sending the data out.

Encryption. Encryption converts a plain text into a ciphertext using a secret key. Encryption may be used as an alternative to anonymization for those testbeds that operates on sensitive data. Encrypted channels for data transfers from the IoT level to the cloud is, in general, requested by all testbeds. Obviously, further investigation is needed in order to clarify if data should be encrypted also during access and whether it should be encrypted in the Edge before being uploaded to the Cloud.

Moreover, two obstacles have to be tackled to applying encryption:

1. The computational cost of the encryption algorithms leading to a performance slow down and a higher energy consumption
2. the difficulty of processing over encrypted data.

If efficient solutions exist to deal with the first obstacle, enabling efficient processing over encrypted data is one the biggest challenge in the security field that the project will try to address.

3 IoTwins Orchestration Platform

3.1 Authentication

The authorization and authentication infrastructure service (AAI) in IoTwins is provided among the above mentioned stack layers by the INDIGO Identity and Access Management Service (INDIGO-IAM). INDIGO-IAM is a service developed by the INDIGO-DataCloud EC project and maintained for the foreseeable future by the INFN partner.

INDIGO-IAM is provided through multiple methods (SAML [5], OpenID Connect [6] and X.509 [7]) by leveraging on the credentials provided by the existing Identity Federations (i.e. IDEM [8], eduGAIN [9], etc.). The support to Distributed Authorization Policies and Token Translation Service will guarantee selected access to the resources as well as data protection and privacy.

INDIGO-IAM is based on the OpenIDConnect and OAuth2.0 protocols and supports two types of users:

- Normal users: these users can login with the IAM, register client applications that use the IAM for authentication, link external accounts to their account (when allowed by the configuration),
- Administrators: these are users that have also administration privileges for an IAM organization.

The service exposes the OpenID Connect/OAuth dynamic client registration functionality offered by the MitreID OpenID Connect server libraries. In OAuth terminology, a client is an application or service that can interact with an authorisation server for authentication/authorization purposes and a new client can be registered in the IAM in two ways: (1) using the dynamic client registration API (2) via the IAM dashboard (which simply acts as a client to the API mentioned above). After registration the client can obtain a token using APIs or already available scripts and GUIs.

The INDIGO-IAM service has been successfully integrated with many off-the-shelf components like Openstack, Kubernetes, Atlassian JIRA and Confluence, Grafana and several middleware services that can be used in the IoTwins architecture.

3.2 Cloud Service Components

The Cloud Level Software provides services aimed to orchestrate the full stacks (IoT, Edge and Cloud) and controls which software is running on which hardware.

To properly address the requests coming from the use cases and, at the same time, to be flexible enough to interact with the most popular choices of the computer centers, without interfering in the operation of their facilities, the virtualization of resources becames a key word.

Taking care of such needs, set of PaaS core components that have been deployed (within the activities of the INDIGO-DC project [4] and subsequent evolutions [2,3]) as a suite of small services using the concept of microservice. Referred to a software architecture style, this term identify complex applications composed of small independent processes communicating with each other via lightweight mechanisms like HTTP resource APIs. The modularity of microservices makes the approach highly desirable for architectural design of complex systems, where many developers are involved.

The base components and functionalities of the INDIGO PaaS are briefly described and their interrelations are shown in Fig. 2

- **PaaS Orchestrator** [10] is the core component of the PaaS layer. The requests related to application or service deployment coming from the user are expressed using TOSCA [11], the OASIS [12] standard to specify the topology of services provisioned in IT infrastructures. Such requests are grouped and organized in the TOSCA template that is processed by the INDIGO Orchestrator. The Orchestrator is able to implement a complex workflow aimed at fulfilling a user request using information about the health status and capabilities of underlying IaaS and their resource availability, QoS/SLA constraints, the status of the data files and storage resources needed by the service/application. This process allows to achieve the best allocation of the resources among multiple IaaS sites.
- **Managed Service/Application (MSA) Deployment Service** [13] is in charge of scheduling, spawning, executing and monitoring applications and services on top of one or more Mesos clusters.
- **Infrastructure Manager (IM)** [14] is in charge to deploy complex and customized virtual infrastructures on different IaaS Cloud deployment, providing an abstraction layer to define and provision resources in different clouds and virtualization platforms. IM enables computing resource orchestration using OASIS standard languages (i.e. TOSCA protocol). Moreover, it eases the access and the usability of IaaS clouds by automating the VMI (Virtual Machine Image) selection, deployment, configuration, software installation, monitoring and update of the virtual infrastructure,
- **Data Management Services** is a collection of services that provide an abstraction layer for accessing data storage in a unified and federated way. These services provide a unified view of the storage resources together with the capabilities of importing data, schedule transfers of data from different

sources. The support of high-level storage requirements, such as flexible allocation of disk or tape storage space and support for data life cycle is achieved by the adoption of the INDIGO CDMI Server [15]. The CDMI server has been extended in the INDIGO project to support Quality-of-Service (QoS) and Data Life-cycle (DLC) operations for multiple storage back-ends like dCache [16], Ceph [17], IBM Spectrum Scale [18], TSM [19], StoRM [20] and HPSS [21],

- **Monitoring Service** [22] is in charge of collecting monitoring data from the targeted clouds, analysing and transforming them into information to be consumed by the Orchestrator,
- **CloudProviderRanker** [23] is a rule-based engine that allows to manage the ranking among the resources that are available to fulfil the requested services. The Service uses the list of IaaS instances and their properties provided by the Orchestrator in order to choose the best site that could support the user requirements,
- **Cloud Information Provider** [24] generates a representation of a site's cloud resources, to be published inside INDIGO Configuration Management DataBase (CMDB),
- **Configuration Management DataBase** [25] is a REST service storing all the information about the cloud sites available and providing details such as the images and containers they support. It is used as an authoritative source of information for matchmaking and orchestration of VM and containers.
- **QoS/SLA Management Service** [26] it allows the handshake between a user and a site on a given Service Level Agreement (SLA) and it describes the Quality of Service (QoS) for a group or specific user both in the PaaS as a whole or over a given site.

3.3 Edge Service Components

The Edge Level Software provides services aimed to provide communication with both the IoT level and the Cloud level by providing methods to manage the provided functionality (apps) dynamically. In the IoTwins architecture, the applications are packed into OCI containers. The Edge Service Orchestration thus has to provide functionality to download, start, stop and configure OCI [27] containers.

In such respect, the IoTwins platform adopted the Apache Mesos [28] to host the Edge Service Orchestration. Apache Mesos is an open-source project to manage computer clusters which has two framework: Marathon[29] and Chronos[30]. Marathon is a production-grade container orchestration platform for Mesosphere's Datacenter Operating System (DC/OS) and Apache Mesos. Marathon is a framework (or meta framework) that can launch applications and other frameworks and can also serve as a container orchestration platform which can provide scaling and self-healing for containerized workloads. Chronos, insted, is a fault-tolerant scheduler that runs on top of Apache Mesos that is used for job orchestration.

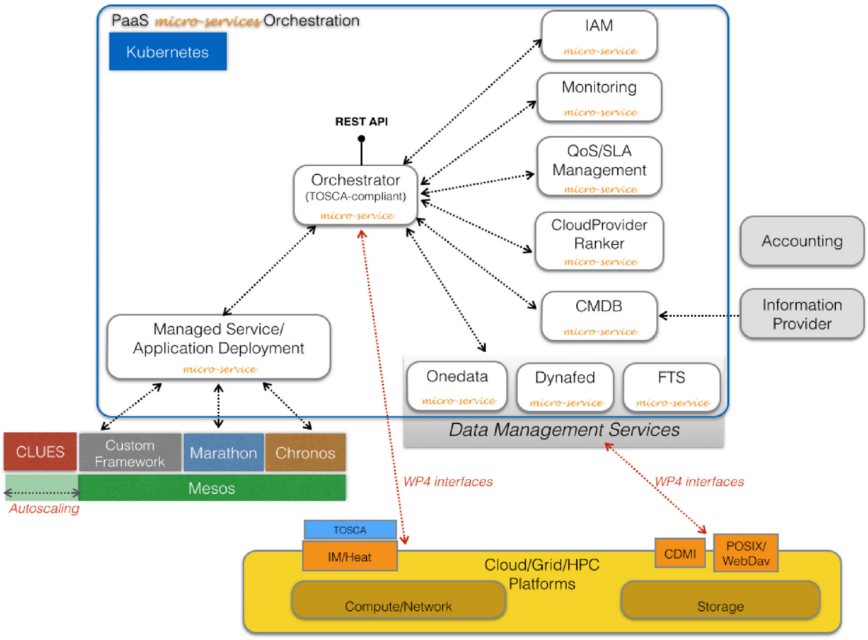

Fig. 2. Graphic schema showing the interrelations among INDIGO PaaS components.

Mesos CLI or UI can be used. Docker containers can be launched using JSON definitions that specify the repository, resources, number of instances, and command to execute. Scaling-up can be done by using the Marathon UI, and the Marathon scheduler will distribute these containers on slave nodes based on specified criteria. Autoscaling is supported. Multi-tiered applications can be deployed using application groups. Containers can be scheduled without constraints on node placement, or each container on a unique node (the number of slave nodes should be at least equal to the number of containers). High availability for Mesos and Marathon is supported using Zookeeper. Zookeeper provides election of Mesos and Marathon leaders and maintains cluster state. Host ports can be mapped to multiple container ports, serving as a front-end for other applications or end users. Marathon continuously monitors the number of instances of a Docker container that are running. If one of the containers fail, Marathon reschedules it on other slave nodes. Auto-scaling using resource metrics is available through community-supported components only. Local persistent volumes (beta) are supported for stateful applications such as MySQL. When needed, tasks can be restarted on the same node using the same volume. The use of external storage, such as Amazon EBS, is also in beta. At the present time, applications that use external volumes can only be scaled to a single instance because a volume can only attach to a single task at a time.

Starting from the available Mesos frameworks: Marathon, which allows to deploy and manage LRS, and Chronos, which allows to execute jobs, the INDIGO project has added new interesting functionalities in Mesos that make them suitable for explitation: the elasticity of a Mesos cluster so that it can automatically shrink or expand depending on the tasks queue; the automatic scaling of the user services running on top of the Mesos cluster and a strong authentication mechanism based on OpenID-Connect.

Thanks to the integration between the Cloud orchestration layer and the Edge orchestration layer provided by the INDIGO-DC components, is it possible, from the PaaS Orchestrator to deploy and made available services on the Edge layer (see Fig. 2).

4 Conclusion

This paper reports on the effort made as an outcome of the IoTwins project to define and deploy a computing and software architecture to be proposed as a highly distributed and hybrid digital twin model, which provides for an integration of simulative and data-driven models to feed AI services.

The use case driven approach permitted to taking into account the heterogeneity of the different use cases and at the same time to defines standard based interfaces for functionality blocks to allow for further development and integration of 3rd party tools.

Furthermore, following a proven software development approach, the requirement collection and analysis phase was followed by applying techniques aimed to define a general high-level platform that has been presented and described.

The present work lays the foundation to build a prototype platform which then can be deployed and tested with the project's use case testbeds. Experiences and discussions resulting from these prototyping will directly influence further development iteratively.

Acknowledgements. This work was partially supported by EU H2020 IoTwins Innovation Action project (g.a. 857191).

References

1. European Open Science Cloud Hub, web site, last view April 2021. https://www.eosc-hub.eu/
2. DEEP-Hybrid-DataCloud, web site, last view April 2021. https://deep-hybrid-datacloud.eu/
3. eXtreme-DataCloud, web site, last view April 2021. http://www.extreme-datacloud.eu/
4. INDIGO-DataCloud, web site, last view April 2021. https://repo.indigo-datacloud.eu/
5. Security Assertion Markup Language, web site, last view April 2021. https://it.wikipedia.org/wiki/Security_Assertion_Markup_Language
6. OpenID Connect, web site, last view April 2021. https://openid.net/connect/

7. X.509, web site, last view April 2021. https://en.wikipedia.org/wiki/X.509
8. IDEM Federation, web site, last view April 2021. https://idem.garr.it/en/federazione-idem-en/idem-federation
9. eduGAIN interfederation service, web site, last view April 2021. https://edugain.org/
10. INDIGO Orchestrator, web site, last view April 2021. https://github.com/indigo-dc/orchestrator
11. TOSCA, web site, last view April 2021. http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html
12. OASIS, web site, last view April 2021. https://en.wikipedia.org/wiki/OASIS_(organization)
13. Managed Service/Application, web site, last view April 2021. https://github.com/indigo-dc/mesos-cluster
14. Infrastructure Manager, web site, last view April 2021. https://indigo-dc.gitbook.io/im/
15. INDIGO Cloud Data Management Interface, web site, last view April 2021. https://www.snia.org/cdmi
16. dCache, web site, last view April 2021. https://www.dcache.org
17. CEPH, web site, last view April 2021. https://ceph.io
18. IBM Spectrum Scale, web site, last view April 2021. https://www.ibm.com/docs/en/spectrum-scale/5.0.4?topic=overview-spectrum-scale
19. IBM Spectrum Protect, web site, last view April 2021. http://www-03.ibm.com/software/products/it/spectrum-protect
20. Storage Resource Manager, web site, last view April 2021. http://italiangrid.github.io/storm/documentation/functional-description/1.11.2/
21. High Performance Storage Systems, web site, last view April 2021. https://www.hpss-collaboration.org
22. High Performance Storage Systems, web site, last view April 2021. https://github.com/indigo-dc/Monitoring
23. CloudProviderRanker, web site, last view April 2021. https://github.com/indigo-dc/CloudProviderRanker
24. Cloud Info Provider, web site, last view April 2021. https://github.com/EGI-Federation/cloud-info-provider
25. Configuration Management DataBase, web site, last view April 2021. https://github.com/indigo-dc/cmdb
26. QoS/SLA Management Service, web site, last view April 2021. https://github.com/indigo-dc/slam
27. Open Container Initiative, web site, last view April 2021. https://opencontainers.org/
28. Mesos, web site, last view April 2021. http://mesos.apache.org/
29. Apache Marathon framework, web site, last view April 2021. https://mesosphere.github.io/marathon/
30. Apache Chronos framework, web site, last view April 2021. https://mesos.github.io/chronos/