

Advanced Analytics as a Service in Smart Factories

Mirco Soderi, Vignesh Kamath, Jeff Morgan, John G. Breslin

Data Science Institute, National University of Ireland Galway, Ireland {firstname.lastname}@nuigalway.ie

Abstract—A key aspect of Intelligent Manufacturing is the interface between the Edge and Fog layers on one side and the Cloud on the other side. Once that some data extraction and cleaning has been performed in the (logical and/or physical) nearby of the production line, it is necessary to move to the Cloud where distributed data and parallel computations make it possible to train and use Machine Learning models. For a wide range of applications, it is also necessary to restructure and reconfigure the computation network over the time, reacting to relevant events at real-time. This can be achieved (i) by making the network and all of its components fully configurable through API calls, (ii) by using containerization technologies, and (iii) by relying on graphical user interfaces for development, data visualization, monitoring, and interaction. In this work, an architecture for a computation network is presented that (i) spreads across the Edge, Fog and Cloud layers, (ii) is fully configurable through API calls, and (iii) is based on Docker, Node-RED, MQTT, Scala, Spark, and Cloud storage technologies such as Hadoop. Two proofs of concept are presented: a clustering-based alerter, and a demo environment plus a Postman collection including over 600 API calls to demonstrate how the proposed architecture enables Big Data stream transformations and analytics as a Service.

Index Terms—Digital Factory, Smart Factory, Cloud Manufacturing, Cloud Computing, Internet of Things (IoT), Software Containers, Graphical Programming Interfaces (GPI), Enterprise Integration, Micro-services, Cyber Devices, Docker, Node-RED, MQTT, Scala, Spark, Hadoop

I. INTRODUCTION

A key aspect of Intelligent Manufacturing is the interface between the Edge and Fog layers on one side and the Cloud on the other side. Once that some data extraction and cleaning has been performed in the (logical and/or physical) nearby of the production line, it is necessary to move to the Cloud where distributed data and parallel computations make it possible to train and use Machine Learning models.

For a wide range of applications, it is also necessary to restructure and reconfigure the computation network over the time, reacting to relevant events in real-time. This can be achieved through the following: (i) let every component of the computation network be accessible through APIs for remote configuration; (ii) the use of containerization technologies; (iii) graphical user interfaces for development, data visualization, monitoring, and interaction. These three corners of what we can call the *Triangle of Smart Manufacturing* (Fig. 1) emerge from a wide review conducted as a part of this work, that (i) provides state-of-the-art insights into the various IIoT technologies available in 2021; (ii) utilizes a bottom-up approach

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number SFI/16/RC/3918 (Confirm). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

to IIoT technology review, from Edge to Fog to Cloud; and (iii) identifies technology trends.

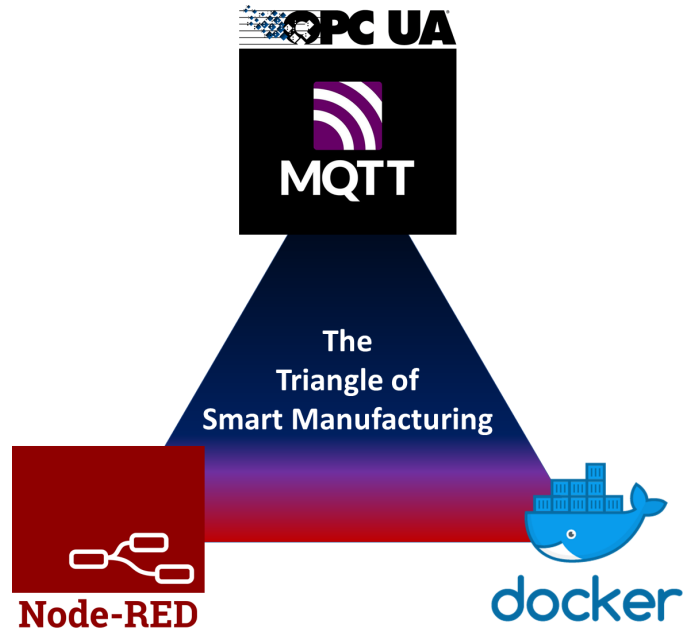





Fig. 1. The Triangle of Smart Manufacturing

A three-layer model is adopted for IIoT systems and devices, including the Edge, the Fog, and the Cloud layer, as shown in Table I.

TABLE I
IT MODEL: EDGE, FOG, AND CLOUD

	Cloud , medium-to-large scale data centers that provide services to devices and systems located externally, typically off the factory premises
	Fog , small-to-medium scale data centers or a group of computers (also known as “cloud-lets”) that provide services to devices and systems located in close proximity
	Edge , all physical devices at the edge of the network and their embedded software, and all software components that directly read from wireless devices

The surveyed devices have been identified by their popularity, references in the scientific literature, and through the authors’ experience. A contrast can be observed between well-established and new industrial technology brands, that allows to identify new feature/technology trends. These trends

imply new skill requirements for digital workers/engineers. Therefore, identifying these trends is paramount for academic/commercial institutions to be able to ensure that they maintain their market competitiveness and technical relevance. As for the Cloud technologies, a comprehensive review is also found in [1].

This paper is structured as follows. A review of emerging IIoT technologies is proposed in Section II. The proposed architecture for a computation network is presented in Section III. The interface between the Edge/Fog layers and the Cloud is described in Section IV, along with a description of the actors involved in the communication. Two proofs of concept are presented in Section VI. Future directions are outlined in Section VII. Conclusions are drawn in Section VIII.

II. REVIEW

A wide range of technologies is surveyed, including IoT devices, ASIC devices, IIoT devices, and other (Fig. 2). In the authors' experience, the selection of the right solution for any specific application should consider a number of criteria, among which: functional capability, performance [33], scaling cost, strategic technology partnerships, and other. Upon reviewing the IIoT devices, a number of trends were identified (columns in Fig. 2).

A. Standard Communication

A diverse range of protocols exist for connecting industrial technologies, all of which may be considered to be quite complex with respect to the enterprise integration requirements [3].

In the past, the OPC standard, now known as OPC classic or OPC DA (Data Access) [5]; sought to unify data under a single open communication protocol. Presently, the OPC classic standard has evolved into the OPC Unified Architecture (OPC UA) Standard [6] [7] [8] [9], that also supports multiple services in a Service Oriented Architecture (SOA). As such, OPC UA enables interoperability among internal/external manufacturing systems, complex data modelling, secure encryption, executable methods, data discovery and events.

The MQTT standard protocol for message brokering was created out of a need for simplicity and scale, and is universally applied across various IoT domains of application [13] [14] [15] [11]. It provides message transport that is agnostic to the payload content. Messages are organized in topics. Presently, Sparkplug [10] [12] has emerged for industrial data modelling and for real-time control, that is based on MQTT.

Both OPC UA and MQTT have overlapping and distinguishing aspects. In the literature, several studies [16] [17] [18] have highlighted how MQTT could be considered the standard of choice for IIoT applications, while OPC UA can be considered the standard of choice for distributed machine control. Both of them are open and supported by well-established institutions [5] [11] [12]. As such, both have a pivotal role in Cloud and Fog systems [19], and are often found together in the middleware servers: OPC UA is used on the factory floor, and MQTT is used to interface to the Cloud. Monolithic and distributed versions are available for both of them.

B. Graphical Programming Interfaces

Graphical User Interfaces (GUI) are a cornerstone of automation engineering, from Programmable Logic Controllers (PLCs), to Human Machine Interfaces (HMIs), and Supervisory Control And Data Acquisition (SCADA) systems [20].

Traditionally, PLCs/HMIs/SCADA systems have been developed through commercial software that translates graphic items into low level commands and functions optimized for deterministic real-time control [21].

Presently, the IIoT technology survey has highlighted the emergence of new GUIs with programming capabilities (GPI), such as Node-RED [22]. Node-RED is today used extensively, since it is open, it has a wide community of developers, it is easily deployed across Edge-Fog-Cloud environments, and it is operated through a Web interface. Used by the IoT hobbyists in the origin, it is now widely used in professional production-ready solutions [23], and in the academy for remote drone control [4], sensor integration for intelligent manufacturing containers [24], smart meter monitoring and security [25], IoT sensor and Augmented Reality (AR) integration [26], and remote energy monitoring [27].

Thanks to Node-RED, a common Web browser can be used to connect to physical devices, collect and process data, and

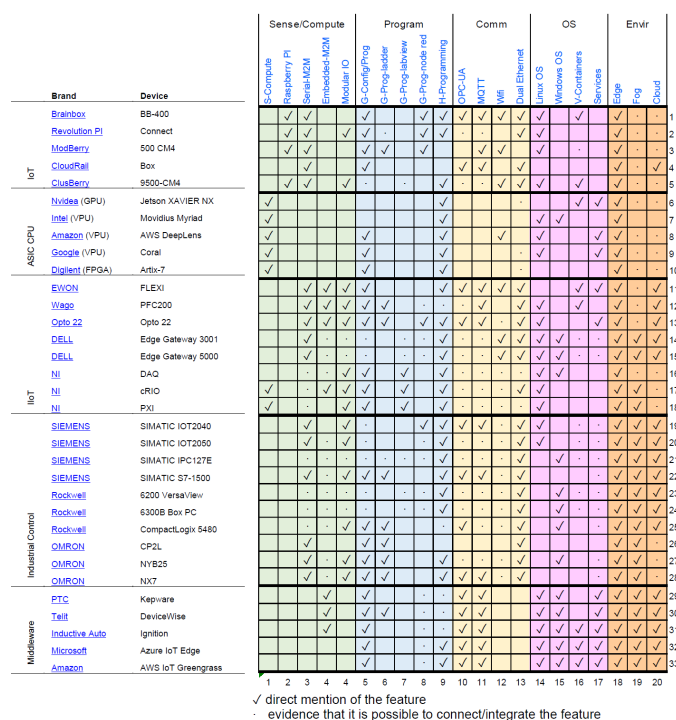


Fig. 2. Industrial Internet of Things (IIoT) Technology Survey

As mentioned, what clearly emerges from the survey is that there are three pillars that underpin Smart Manufacturing at present: (i) Standard Communication, (ii) Graphical Programming Interfaces, and (iii) Virtual Containers.

integrate services. Atomic functionalities are represented as *nodes*, and are organized in *flows*. Furthermore, Node-RED supports modules for the building of user interfaces and data visualizations. In summary, while it does not provide complex programming capabilities, it enables system integration across the Edge, Fog, and Cloud layers.

C. Virtual Containers

Cloud and Fog environments consist of a collection of interconnected hosts that grant scalability and redundancy through the load balancing [28]. A multiplicity of operating systems can be found, also thanks to Virtual Machines (VMs).

More recently, the Containers have appeared on the scene. They enable portability, efficient deployment, scalability, and reduced overhead [29]. Containers are executable packages that are isolated from other systems running on host computers. Applications, commonly referred to as *images*, run in the Containers, and often offer their services via APIs.

As such, a service created to run in a Container can be hosted seamlessly and reliably on any technology layer which provides a Container Engine such as Docker. Therefore, software becomes ubiquitously transportable in Containers across Edge, Fog, and Cloud environments.

The IIoT technology survey has allowed to identify various uses of containerization, from IIoT devices to industrial computers. Containers enable micro-services in smart manufacturing architectures [2], and are prevalent in IIoT architectures [30] and Fog/Cloud computing [31]. They grant a new level of interoperability for systems and services, that are enhanced as for the deployment, scalability, and redundancy. So, centralized systems can be broken down into reusable collaborative components in Edge devices, Fog nodes, and Cloud environments. Remarkably, there are 100,000s of ready-to-use containerized applications at today [32], including MQTT servers and Node-RED, and many others are being moved.

III. ARCHITECTURE

In Fig. 3 an example of a *Smart Manufacturing Network* is represented.

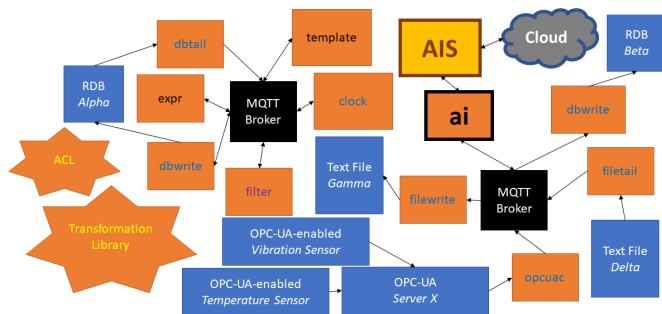


Fig. 3. An example of a Smart Manufacturing Network

A. Devices

The *physical devices* and the other data sources are represented as blue boxes.

The *cyber devices* and the other computation nodes lying in the Edge/Fog layers (*Service Nodes*), are depicted as orange boxes. They are dockerized Node-RED applications, interconnected through MQTT broker(s). Each of them exposes a number of configuration APIs through which they can be fully customized as for their tasks, inputs, main outputs, and secondary outputs (statuses). Task implementations are loaded into the Service Nodes from a specialized Node-RED application, still dockerized, called the *Transformation Library*, that consists of a mere collection of reusable Node-RED subflows (similar to functions in traditional programming languages).

It is possible to distinguish five types of Service Nodes: (i) those that read from an external data source and write to a MQTT broker (orange boxes with blue writings); (ii) those that read from an MQTT broker and write to an external storage (represented the same); (iii) those that read from an MQTT broker/topic, perform a transformation, and write to a different MQTT broker/topic (orange boxes with black writings); (iv) those that perform some sort of flow control (filtering, routing, and other), represented in orange boxes with violet writings; (v) those that interact with an Artificial Intelligence Server (AIS), represented as orange boxes with a large and bold “ai” writing.

The backup of a Docker Volume to be restored to the /data folder of a virgin Node-RED Docker Container to make of it an empty Service Node ready to be configured is available open-source on GitHub¹, named as servicenode.tar (being too large for GitHub, tar files require Git LFS, please find them also in the Google Drive² in case you would experience any issue). Some environment variables must be set, see env.txt for an example. Self-signed SSL certificates must be loaded in the /data folder of each Service Node. See ExampleCerts.zip for ready-to-use certificates for development and test purposes.

The *Service Node ACL API*, still implemented as a dockerized Node-RED application, is called by all Service Node Configuration APIs to verify if the requester has the necessary privileges for configuring that specific aspect of that specific Service Node. The backup of a Docker Volume with an example implementation is available open-source on GitHub, named as servicenodeacl.tar.

B. Broker

As for the MQTT broker, the backup of a Docker Volume to be restored to the /opt/emqx folder of a virgin emqx/emqx:4.3.2 Docker Container to run the examples described in Section VI is available open-source on GitHub, named as broker.tar. The backup of a Docker Volume to be restored to the /data folder of a virgin Node-RED Container

¹<https://github.com/mircosoderi/Advanced-Analytics-as-a-Service-in-Smart-Factories>

²<https://drive.google.com/file/d/15jDdmnRsfp9o6Qw0fkGIGmBeckVN9gzrn>

to have an example of ACL API compatible with the broker up and running is available open-source on GitHub, named as `brokeracl.tar`.

C. Transformation Library

The *Transformation Library* is a dockerized Node-RED application that is not meant to run any code, being instead just a collection of reusable Node-RED subflows meant to be loaded (via API call) to the Service Nodes where they actually execute. The backup of a Docker Volume meant to be restored to the `/data` folder of a virgin Node-RED Container to make of it a Transformation Library is available open-source on GitHub, named as `transformationlibrary.tar`.

D. Artificial Intelligence Server

The *Artificial Intelligence Server (AIS)*, is a dockerized Scala application that relies on Akka HTTP and on a number of Cloud-related dependencies such as Spark, and Hadoop. The “ai” Service Nodes can be seen as clients of the AIS. A detailed description of both, and of how they interact, is provided in Section IV. The backup of a Docker Volume ready to be restored to the `/home` folder of a virgin mozilla/sbt Docker Container to make of it an Artificial Intelligence Server is available open-source on GitHub, named as `dockerized-aiserver.tar`.

E. Interaction

The interaction among the building blocks presented above is represented in Fig. 4. The community version of the EMQ broker has been used so far. Anyway, it can be replaced by any other implementation.

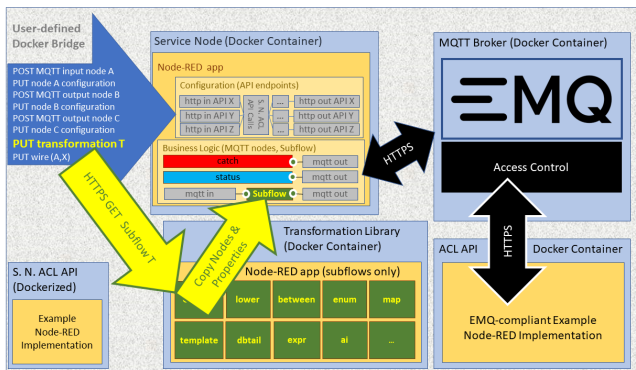


Fig. 4. Interactions among the key components of a Smart Manufacturing Network

IV. FROM EDGE TO CLOUD AND BACK AGAIN

The interaction between each of the *ai* Service Nodes and the AIS is depicted in Fig. 5.

The AIS exposes a set of APIs to the *ai* Service Nodes, meant to be used for configuration purposes, and for the delivery of input values and control signals.

Also, it includes an extendable tasks library, that is made of: (i) Scala classes, that are efficient but require a server

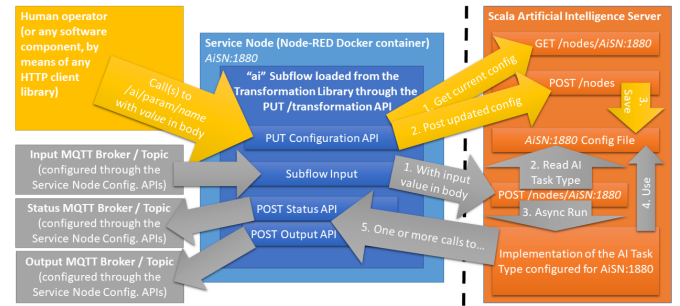


Fig. 5. Interaction between the *ai* Service Nodes, and the AIS

restart to be refreshed; (ii) text files containing appropriate Scala expressions compiled on-the-fly when needed, that are less efficient but can be added while the server is running and are immediately available to the *ai* Service Nodes.

For each *ai* Service Node that interfaces with it, the AIS keeps a configuration file, where it is indicated the task to be executed when a new input arrives from the Service Node, and the configuration parameters that are specific to the task to be executed.

For each *ai* Service Node that interfaces with it, the AIS also reserves some local disk space to accommodate one file in any format plus possible stream folders (see Section V).

As for the *ai* Service Nodes, they are Service Nodes where the *ai* Node-RED subflow has been loaded from the Transformation Library. Thanks to that, each of them exposes APIs for setting: (i) the credentials to be used to make calls to the AIS; (ii) the security level of HTTPS communications with the AIS; (iii) the task configurations, that are forwarded to the AIS.

Also, *ai* Service Nodes are capable to deliver to the AIS those values that they get from their configured input MQTT broker instance and topic. They do that by means of appropriate API calls made to the AIS.

Also, each *ai* Service Node exposes the APIs that are called by the AIS to provide back (asynchronously) the results of the processing of the input values that come from the *ai* Service Node over the time, and/or the possible ongoing status(es) of the execution, including possible errors.

V. BIG DATA STREAM PROCESSING AND ANALYTICS

At the date of today, a number of tasks have been implemented and are natively available in the AIS for (i) the processing of (Kafka) Big Data streams, including datatype casting, computation of statistics and expressions, filtering, comparison and join of multiple streams into one; (ii) the training of clustering, classification, and regression models with libsvm datasets stored on Hadoop and the storage of the trained models still on Hadoop; (iii) the execution of predictions, by getting the trained model from Hadoop and the input data from Kafka streams or API calls; (iv) the feeding of Kafka streams via API calls.

The read/write operations from/to Big Data (Kafka) streams, and the parallel data transformations on Big Data streams, are

performed by peculiar tasks implemented in the AIS library. They rely on the Spark Structured Streaming. They accept control signals (*start*, *stop*) and configuration parameters (including those related to the input and output streams) from the *ai* Service Nodes. They process the data that come from the configured input stream, and they write to the configured output stream.

In Subsection VI-B, it is described how to setup a demo environment, and how to configure and use it, through a Postman collection of over 600 API calls that are meant to demonstrate the Big Data stream processing and analytics functionalities exposed by the AIS.

VI. PROOFS OF CONCEPT

Two proofs of concept are described in this section: (i) a clustering-based alerter; (ii) a comprehensive demonstration of the Big Data stream processing and analytic capabilities of the Artificial Intelligence Server at today.

A. Clustering-based alerter

The computation network built for this proof of concept (Fig. 6) monitors a relational database table, and when a new row is added, it extracts the numeric value from a configured column, and it runs a K-Means clustering with $K=3$ to determine if it is *higher than normal*, in which case it writes the value to a configured *alert* RDB table.

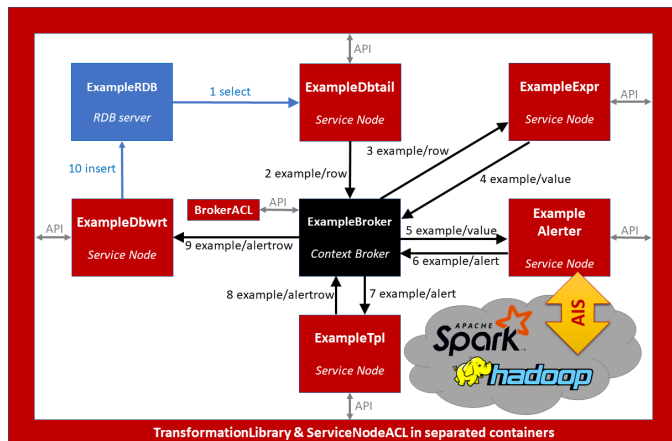


Fig. 6. Proof of concept: anomaly detection through Service Nodes, AIS, Spark, and Hadoop

Each box is a separated Docker Container. All containers are connected to the same user-defined Docker network/bridge, named as ExampleNetwork. The (host)names of Containers are in bold in each box. Docker Volumes are used. Their backups are available for download open-source on GitHub, singularly, or wrapped in the (very large) `example_volumes_backup.tar`. Once that the AIS container is started, the server must be started getting a shell from the container, moving to `/home`, running `sbt`, and then issuing the `reStart sbt` command. The AITask specifically developed for this proof of concept is available open-source on GitHub, named as `MyAlerterV2`. It must be added to the AIS library

via docker cp. The zipped docker-compose project for the single-node Hadoop, based on the Big Data Europe one, is available open-source on GitHub. When starting the Service Nodes, some environment variables must be set. See `env.txt`, and `ExampleAlerter.txt`, both available open-source on GitHub. In the proof of concept, all communications are over HTTPS, and authenticated. To achieve the same, the backup of the Docker Volume for Service Nodes must be uncompressed, the `settings.js` edited, the needed artifacts added, and compressed again. The RDBMS used for the proof of concept is a dockerized MariaDB. The RDB schema is available open-source on GitHub, named as `ExampleRDB.sql`.

After being created and started, the empty Service Nodes must be configured via API calls³⁴. To use the API calls as they are, map the port 1880 to the ports from 1993 onward, clockwise from `ExampleDbtail`. Map the `ExampleAlerter` to the port 2001 anyway.

B. Big Data stream processing and analytics

The demo environment is depicted in Fig. 7. Non-trivial port mappings are indicated. They are mandatory to use the Postman API collection⁵ as is. Ensure that you have created all needed containers and have all of them up and running before that you move to the Postman API collection. You might want to use the `docker-compose.yml` that is available open-source in the GitHub repository to have a single-node dockerized Kafka instance up and running, to be used for local development, and for the purposes of this proof of concept. You also might want to download the `NewServiceNodeQuickly.zip` archive and go through the `quickstart.txt` file inside it, to create empty Service Nodes quickly. Remarkably, you might want to restore `TestMQTTMonitor.tar` to a virgin Node-RED container and use such dockerized Node-RED application to monitor the relevant MQTT topics and have evidence that everything is working as expected while you go through the API calls.

VII. FUTURE DIRECTIONS

This research will proceed along three main directions: (i) extension of the functionalities natively available in the AIS; (ii) Data Visualization as a Service, both in the Edge/Fog layers via the `node-red-dashboard` module, and Cloud-based by adding dedicated functionalities in the AIS; (iii) the development of a new Node-RED palette to hide as far as possible the complexity of Big Data processing and analytics, so that one day, for the developer, working with small data available locally, or with Big Data (streams) flowing on the Cloud, could be the same, or nearly the same.

VIII. CONCLUSIONS

In this work, a review of cutting-edge IIoT technologies has been conducted, and three key trends have been identified: standardization of communication, graphical development environments, and containerization. Based on that, a proposal

³<https://documenter.getpostman.com/view/16531967/Tzm5Gbqy>

⁴<https://documenter.getpostman.com/view/16531967/Tzz4SL7z>

⁵<https://documenter.getpostman.com/view/16531967/UV5agGc6>

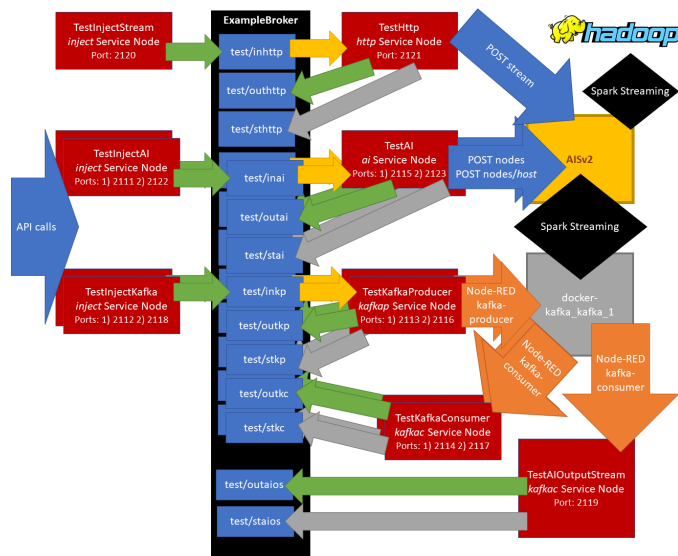


Fig. 7. Proof of concept: Demo environment for Big Data stream processing and analytics

has been presented of a network of dockerized Node-RED applications fully configurable through APIs that interface to Artificial Intelligence Servers to run parallel computations on Big Data in the Cloud. Two proofs of concept have been described: a clustering-based alerter, and a demo environment plus a collection of about 700 API calls to showcase the AIS Big Data stream processing and analytic capabilities. Future directions for this research have been also identified.

REFERENCES

- [1] S. Yangui, "A panorama of cloud platforms for iot applications across industries," *Sensors* (Switzerland), 2020, doi: 10.3390/s20092701.
- [2] M. Moghaddam, M. N. Cadavid, C. R. Kenley, and A. V. Deshmukh, "Reference architectures for smart manufacturing: A critical review," *J. Manuf. Syst.*, 2018, doi: 10.1016/j.jmsy.2018.10.006.
- [3] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Ind. Electron. Mag.*, 2017, doi: 10.1109/MIE.2017.2649104.
- [4] M. Salhaoui, A. Guerrero-González, M. Arioua, F. J. Ortiz, A. El Ouakadi, and C. L. Torregrosa, "Smart industrial iot monitoring and control system based on UAV and cloud computing applied to a concrete plant," *Sensors* (Switzerland), 2019, doi: 10.3390/s19153316.
- [5] OPC Foundation, "OPC Technologies." [Online]. Available: <https://opcfoundation.org/>. [Accessed: 15-Feb-2021].
- [6] M. Schleipen, S. S. Gilani, T. Bischoff, and J. Pfrommer, "OPC UA & Industrie 4.0 - Enabling Technology with High Diversity and Variability," in *Procedia CIRP*, 2016, doi: 10.1016/j.procir.2016.11.055.
- [7] F. Pauker, T. Frühwirth, B. Kittl, and W. Kastner, "A Systematic Approach to OPC UA Information Model Design," in *Procedia CIRP*, 2016, doi: 10.1016/j.procir.2016.11.056.
- [8] P. F. S. De Melo and E. P. Godoy, "Controller Interface for Industry 4.0 based on RAMI 4.0 and OPC UA," in *2019 IEEE International Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2019 - Proceedings*, 2019, doi: 10.1109/METROI4.2019.8792837.
- [9] OPC Foundation, "OPC Unified Architecture Part 1: Overview and Concepts Version 1.02," OPC Foundation, 2012.
- [10] Eclipse Foundation, "Sparkplug Specification Version 2.2," 2019.
- [11] MQTT.org, "MQTT: The Standard for IoT Messaging." [Online]. Available: <https://mqtt.org>. [Accessed: 15-Feb-2021].
- [12] Eclipse Foundation, "SparkPlug." [Online]. Available: <https://sparkplug.eclipse.org/>. [Accessed: 15-Feb-2021].
- [13] E. G. Davis, A. Calveras, and I. Demirkol, "Improving packet delivery performance of publish/subscribe protocols in wireless sensor networks," *Sensors* (Switzerland), 2013, doi: 10.3390/s130100648.
- [14] R. Hassan, F. Qamar, M. K. Hasan, A. Hafizah, M. Aman, and A. S. Ahmed, "Internet of Things and Its Applications: A Comprehensive Survey," pp. 1–29, 2020.
- [15] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, 2015, doi: 10.1109/COMST.2015.2444095.
- [16] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, "OPC UA versus ROS, DDS, and MQTT: Performance evaluation of industry 4.0 protocols," in *Proceedings of the IEEE International Conference on Industrial Technology*, 2019, doi: 10.1109/ICIT.2019.8755050.
- [17] M. S. Rocha, G. S. Sestito, A. L. Dias, A. C. Turcato, D. Brandão, and P. Ferrari, "On the performance of OPC UA and MQTT for data exchange between industrial plants and cloud servers," *Acta IMEKO*, 2019, doi: 10.21014/acta_imeko.v8i2.648.
- [18] M. Silveira Rocha, G. Serpa Sestito, A. Luis Dias, A. Celso Turcato, and D. Brandao, "Performance Comparison between OPC UA and MQTT for Data Exchange," in *2018 Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2018 - Proceedings*, 2018, doi: 10.1109/METROI4.2018.8428342.
- [19] J. Morgan and G. E. O'Donnell, "Enabling a ubiquitous and cloud manufacturing foundation with field-level service-oriented architecture," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 4–5, 2017, doi: 10.1080/0951192X.2015.1032355.
- [20] K. L. S. Sharma, *Overview of Industrial Process Automation: Second Edition*. 2016.
- [21] J. Morgan and G. E. O'Donnell, "The cyber physical implementation of cloud manufacturing monitoring systems," in *Procedia CIRP*, 2015, vol. 33, doi: 10.1016/j.procir.2015.06.007.
- [22] OpenJS Foundation, "Node-Red." [Online]. Available: <https://nodered.org/>. [Accessed: 16-Feb-2021].
- [23] O. Foundation, "2019 Node-RED Community Survey." [Online]. Available: <https://nodered.org/about/community/survey/2019/>. [Accessed: 16-Feb-2021].
- [24] A. D. Neal, R. G. Sharpe, P. P. Conway, and A. A. West, "smARTI—A cyber-physical intelligent container for industry 4.0 manufacturing," *J. Manuf. Syst.*, 2019, doi: 10.1016/j.jmsy.2019.04.011.
- [25] R. P. Díaz Redondo, A. Fernández-Vilas, and G. F. Dos Reis, "Security aspects in smart meters: Analysis and prevention," *Sensors* (Switzerland), 2020, doi: 10.3390/s20143977.
- [26] Ó. Blanco-Novoa, P. Fraga-Lamas, M. A. Vilar-Montesinos, and T. M. Fernández-Caramés, "Creating the internet of augmented things: An open-source framework to make iot devices and augmented and mixed reality systems talk to each other," *Sensors* (Switzerland), 2020, doi: 10.3390/s20113328.
- [27] F. Lima, A. A. Massote, and R. F. Maia, "IoT Energy Retrofit and the Connection of Legacy Machines Inside the Industry 4.0 Concept," in *IECON Proceedings (Industrial Electronics Conference)*, 2019, doi: 10.1109/IECON.2019.8927799.
- [28] P. O'Donovan, C. Gallagher, K. Leahy, and D. T. J. O'Sullivan, "A comparison of fog and cloud computing cyber-physical interfaces for Industry 4.0 real-time embedded machine learning engineering applications," *Comput. Ind.*, 2019, doi: 10.1016/j.compind.2019.04.016.
- [29] A. M. Joy, "Performance comparison between Linux containers and virtual machines," in *Conference Proceeding - 2015 International Conference on Advances in Computer Engineering and Applications, ICACEA 2015*, 2015, doi: 10.1109/ICACEA.2015.7164727.
- [30] S. Trilles, A. González-Pérez, and J. Huerta, "An IoT platform based on microservices and serverless paradigms for smart farming purposes," *Sensors* (Switzerland), 2020, doi: 10.3390/s20082418.
- [31] R. K. Naha et al., "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, 2018, doi: 10.1109/ACCESS.2018.2866491.
- [32] Docker, "Docker Hub is the world's largest library and community for container images." [Online]. Available: <https://hub.docker.com/>. [Accessed: 17-Feb-2021].
- [33] V. Kamath, J. Morgan and M. I. Ali, "Industrial IoT and Digital Twins for a Smart Factory : An open source toolkit for application design and benchmarking," *2020 Global Internet of Things Summit (GIoTS)*, 2020, pp. 1–6, doi: 10.1109/GIoTSS49054.2020.9119497.