electronics

MDPI

Article

On the Performance of Cloud Services and Databases for Industrial IoT Scalable Applications

Paolo Ferrari *, Emiliano Sisinni *, Alessandro Depari, Alessandra Flammini, Stefano Rinaldi, Paolo Bellagente and Marco Pasetti

Department of Information Engineering, University of Brescia, 25123 Brescia, Italy; alessandro.depari@unibs.it (A.D.); alessandra.flammini@unibs.it (A.F.); stefano.rinaldi@unibs.it (S.R.); paolo.bellagente@unibs.it (P.B.); marco.pasetti@unibs.it (M.P.)

* Correspondence: paolo.ferrari@unibs.it (P.F.); emiliano.sisinni@unibs.it (E.S.)

Received: 28 July 2020; Accepted: 28 August 2020; Published: 3 September 2020

Abstract: In the Industry 4.0 the communication infrastructure is derived from the Internet of Things (IoT), and it is called Industrial IoT or IIoT. Smart objects deployed on the field collect a large amount of data which is stored and processed in the Cloud to create innovative services. However, differently from most of the consumer applications, the industrial scenario is generally constrained by time-related requirements and its needs for real-time behavior (i.e., bounded and possibly short delays). Unfortunately, timeliness is generally ignored by traditional service provider, and the Cloud is treated as a black box. For instance, Cloud databases (generally seen as "Database as a service"—DBaaS) have unknown or hard-to-compare impact on applications. The novelty of this work is to provide an experimental measurement methodology based on an abstract view of IIoT applications, in order to define some easy-to-evaluate metrics focused on DBaaS latency (no matter the actual implementation details are). In particular, the focus is on the impact of DBaaS on the overall communication delays in a typical IIoT scalable context (i.e., from the field to the Cloud and the way back). In order to show the effectiveness of the proposed approach, a real use case is discussed (it is a predictive maintenance application with a Siemens S7 industrial controller transmitting system health status information to a Cloudant DB inside the IBM Bluemix platform). Experiments carried on in this use case provide useful insights about the DBaaS performance: evaluation of delays, effects of involved number of devices (scalability and complexity), constraints of the architecture, and clear information for comparing with other implementations and for optimizing configuration. In other words, the proposed evaluation strategy helps in finding out the peculiarities of Cloud Database service implementations.

Keywords: distributed measurement systems; automation networks; Industry 4.0; cloud computing

1. Introduction

The fourth industrial revolution, commonly addressed as the Industry 4.0 paradigm, leverages on the idea of a digital twin of real-world systems (e.g., machineries, tools, and so on) implemented in the Cloud domain and continuously interacting with the physical twin by means of Internet-based communication infrastructure [1–3]. On the digital side, analytics are executed in order to evaluate the system history and forecast future behavior, thus improving availability, reliability and efficiency of production systems. On the physical side, low-cost smart field devices perform preliminary data processing and transmit refined information about the physical quantities of interest to the Internet.

The previously described scenario leads to the generation and management of very large amounts of data, which pave the way to innovative services aiming at increasing overall performance in terms of cost, lifetime, and efficiency [4–8].

As a matter of fact, communications in the Industry 4.0 scenario rely on a subset of the Internet of Things (IoT) paradigm, which is commonly referred to as the Industrial IoT or IIoT [9]; in particular, the well-known pyramidal arrangement of legacy industrial communications, whose lower level was occupied by fieldbuses, is flattened thanks to the adoption of the Internet as a sort of universal backbone [10]. As a consequence, the time-related constrains of fieldbuses are somehow inherited by such Internet-based communications, which can spread on geographical scale [11,12]. In turn, latency in concluding service requests probably remains the most important key performance indicator (KPI). Unfortunately, there is no universal agreement on how such a performance evaluation must be carried out on new architecture, and the development of metrics univocally defined is still an open research issue [13–15]. Moreover, the impact of remote database accesses in the overall delay is usually not detailed, since the whole Cloud is generally considered as a single black-box entity.

This paper has the following novel contents:

- It provides easy-to-compute metrics to objectively evaluate the time performance of Cloud databases (Database as a service—DBaaS—paradigm), due to the very important role they assume in data analytics services [16].
- It introduces a reference model of an Industry 4.0 scenario (which includes a closed loop acquiring data from the production line and providing back information about the system KPI of interest [17]).
- It provides an abstract view of the above reference model, so that the metrics are independent from the peculiar mechanism implemented to access the database (which only depends on the way the system has been realized, and not on the service it offers).
- It considers real-world use cases for the experimental evaluation of the proposed methodology and of the associated set of metrics. The testbeds confirm the general effectiveness of the proposed approach, especially when the scalability of the system under test must be evaluated and compared.

The paper is arranged as follows. The next Section provides a brief overview of the IIoT paradigm and possible Cloud database scenarios. In Section 3 the proposed approach is detailed, while Section 4 describes the considered real-world use cases. The experimental testbench setup to evaluate the effectiveness of the proposed methodology is in Section 5. Section 6 presents a brief overview of the demo systems used for trade fairs. Finally, the conclusions are drawn.

2. Materials and Methods

Any Industry 4.0 applications exist in a Cyber-Physical System (CPS), consisting of both the real-world product and its digital counterpart [18]. In particular, applications rely on the capability to collect information about the status of a so-called smart thing (or object) during its whole lifetime, from the initial conceiving to the aftermarket, including the disposal and recycle stage. Collected data are saved in distributed data repository built up according to the Cloud computing paradigm; the very same data are used to update the status of the smart object "digital twin", which is nothing but a description (i.e., a model) of the device in the digital domain. The backbone connecting the physical and the cyber domains is based on the Internet: the IIoT is the IoT subset which is sometimes defined as the Internet for the machines.

The very large amount of data generated by means of the IIoT paradigm is the fundament of any Industry 4.0 service. The most common example is furnished by the currently commonly adopted predictive maintenance approach. In the past, maintenance was carried out purely on time basis, i.e., the older the equipment the more frequent maintenance procedures were executed. However, faults very often occur on a random basis, thus vanishing any time-based maintenance scheduling. Nowadays, more accurate failure models are implemented in the cyber space, based on artificial intelligence techniques to learn the actual system/product behavior from the IIoT-derived data, so that more effective maintenance can be scheduled and, based on the results of the prediction, control applications may be configured to provide a real-time feedback via the equipment actuators.

As stated in the introduction, data are saved in databases hosted in the Cloud, generally exploiting the DBaaS approach, often referred to as "Managed Databases" since AWS introduced its Relational Database Service (RDS) back in 2009. In other words, the DBaaS acronym refers to the software infrastructure that permits to setup, use and scale databases by means of a common set of abstractions primitives, thus not caring of the peculiar implementation details [19].

Currently, DBaaS is the fastest growing Cloud service; and the cost of offered services is mainly determined by the writing speed (in terms of transaction per second), the storage size and the available computation capability. On the contrary, the delay in completing writing and reading back operations is usually not taken into account, since it is generally considered a non-critical issue, and Cloud-based applications are generally "slow" [20]. However, such a hypothesis may be false in most industrial applications, where real-time behavior is often mandatory.

The Considered Scenario

The advent of the Industry 4.0 and of the IIoT communications allows programmable logic controllers (PLC), Supervisory Control And Data Acquisition (SCADA) systems, and sensors/actuators to be directly and freely interconnected. Moreover, the Internet backbone extends connectivity to the Cloud, possibly in real-time. As a matter of fact, the communication changes from the client-server approach, e.g., used by human operator for accessing the Web, to the publisher-subscriber approach, where the relationship between the data producer and the data consumer is loosely defined [10]. Consequently, innovative services operating on a geographical scale have been devised, offering many additional benefits with respect to a local scale parameters optimization (e.g., [21]). In such a context, advanced data mining techniques can be activated [22], leading to important savings during production [23].

In this paper, a reference Cloud-based application has been considered, as shown in Figure 1. In particular, without any generality loss, a data source provides information about a generic smart object (e.g., a smart sensor deployed in a production process). Data flow to the Internet by means of an IoT Gateway (which consists of a software interface or may be a hardware peripheral), where are stored and processed in the Cloud. Indeed, user applications reside in the Cloud as well, analyzing data from the field and providing feedback actions, sent to the final destination.

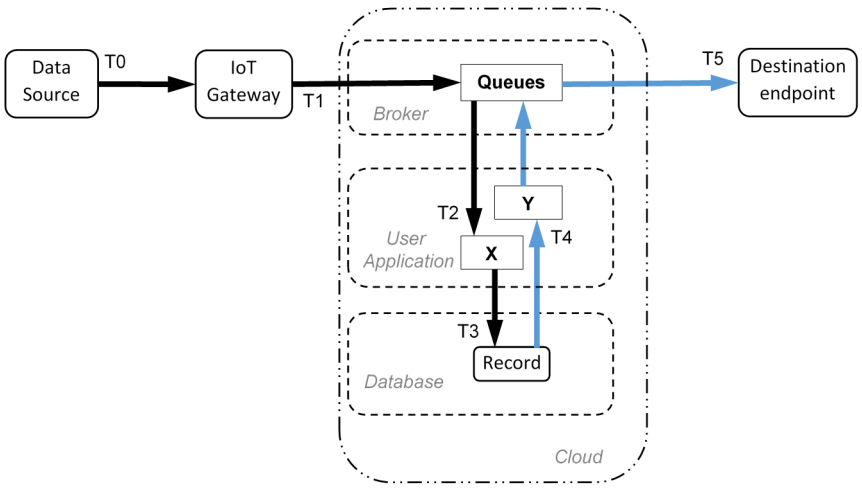

Figure 1. The reference Cloud-based application architecture for definition of delay and latency metrics including the Cloud Database. The black patch shows the data transfer to the Cloud, while the blue path shows the data transfer from the Cloud.

The Cloud environment allows to collect information from very diverse data sources, storing them in the Cloud database (with DBaaS paradigm) and feeding many different applications (e.g., implementing machine learning techniques for predictive maintenance, as previously stated).

Regarding the data dispatching, message-oriented middleware is preferred to client-server REST API (REpresentational State Transfer Application Programming Interface) based on the HTTP protocol. It is well known that messaging protocols are particularly well-suited for IIoT applications, since they natively adopt the publisher/subscriber paradigm and are event-driven. Many of such messaging protocols exist today, each one with its own pros and cons, but all sharing a common architecture. In this work, without any loss of generality, the Message Queuing Telemetry Transport (MQTT) has been adopted.

Regarding the Cloud database access, two different modalities can be considered for the sake of completeness (see Figure 2). The first is traditional polling strategy, in which a DB (Database) request is started by the user, asynchronously with respect to the DB. In such a case, time related performance is also affected by the polling cycle and its uncertainty. Another approach relies on the use of so-called database trigger, that is a purposely defined procedure stored in the DB which is synchronously executed when a specific, predefined action occurs within the DB [24]. Generally, triggers are created to run when data are modified, e.g., after insert, update, or delete operations. The use of triggers thus allows that certain actions are accomplished regardless of the user activity. All the commercially available solutions usually support them.

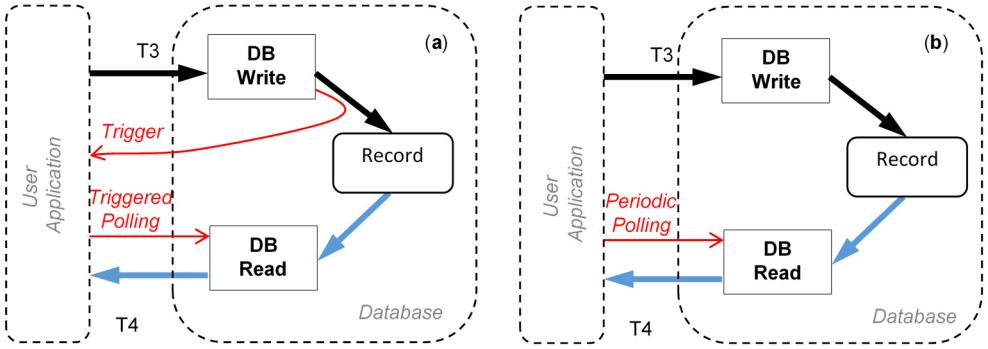

Figure 2. Two methodologies for retrieving data form Databases with a short latency. (**a**) The write action triggers a user function that reads back the recently written data; (**b**) The Database is polled regularly to fetch recently added data.

The trigger approach to synchronously issue a DB request is particularly interesting for IIoT applications, in which, as previously stated, real-time behavior is generally required.

3. How to Evaluate the Performance of the Architecture

The aim of this work is to estimate how the Cloud database services can impact the performance of Industry 4.0 applications. In particular, the reference architecture is a Cloud platform, and the performance are evaluated using time related indicators like latencies and delays.

The latency of an architecture based on data exchange is depending on the data path. In all complex systems, like Cloud-based architectures, there are some parts that cannot be directly controlled by the user. Many times, the use of a black-box approach is preferable. In this paper, instead of using simulations for latency estimation (as in [25,26]), an experimental methodology has been developed. The authors think that, once the metrics are defined, the experimental approach can be easily reproduced on many architectures, as demonstrated in previous works and related literatures [25–28].

3.1. Introduction of Time Related Metrics

The relevant metrics must be defined considering the reference architecture of a classic IoT application for industrial purpose. Figure 1 introduces the architecture highlighting the services in the Cloud. Generalizing the Industry 4.0 process, data are generated in field when the system is producing, they are collected by gateways, and then transported in the Cloud. Data storage and elaboration take

place in the Cloud. Mixing of data coming from various sources and multiple parallel processing by means of several applications is possible. Finally, decisions and parameters calculated online may be sent back to the machines in the field, closing the loop.

The idea of a generally-applicable performance evaluation is tied to the use of time related metrics that highlight the time needed by data to travel from the source to the intended destination. Please, note that the focus is on the Cloud database for evaluating its contribution to latency, therefore the Cloud application is a simple task where the data pass-through without being manipulated. Please note also that a message-based protocol (e.g., MQTT protocol) is used in this work. As required by messaging protocols, there is the "Broker" that handles and dispatches the messages (sent by the "publisher"), forwarding them to the destinations (called "subscribers").

According to the reference architecture shown in Figure 1, at time T0 the source of data provides a new information. At time T1, the IoT gateway sends the collected data into the Cloud using a messaging protocol. In the Cloud, the user application X receives, at time T2, the incoming messages from the gateway. The application X stores the message data payload into the Cloud database, completing at time T3 its task. The data now are uploaded and the black path of Figure 1 is complete. Whatever is the database access implemented, the inverse path (blue color in Figure 1) starts exactly at T3, because at that time the data are available inside the database and further delays are due only to read and send operation (using method shown in Figure 2). The second user application Y is not related to application X, they operate separately. Application Y completes data retrieving from the database at time T4, and it propagates the message to the destination endpoint. Additionally, application Y uses a messaging protocol, and send data though a Broker. The entire information loop ends at time T5.

The authors developed a methodology for collecting timestamps and combining them in previous works [28–30]. In order to be combined, the timestamps must be taken from the local system clock (i.e., inside the device, the machine, or the Cloud) after it is synchronized to UTC (Universal Time Coordinated). In this paper the synchronization is done with NTP (Network Time Protocol) as in [31–33].

The goal is to define metrics for objective comparison of different systems. The timestamps are collected and the following primary metrics are computed:

- The delay, called D_{PC}, between the time the IoT Gateway publishes the data and the time they are received in the Cloud inside the user application, formally

$$D_{PC} = T2 - T1 \tag{1}$$

- The delay, called D_{CD}, for carrying out a Cloud database write operation (i.e., for storing the data), formally

$$D_{CD} = T3 - T2 \tag{2}$$

- The delay, called D_{DC}, for retrieving the data from the Cloud database, formally

$$D_{DC} = T4 - T3 \tag{3}$$

  The D_{DC} can be related to the call of a trigger function that reacts to events inside the database (fastest reaction, since it can send the recently written data to the user application), or it can be related to a polling cycle that periodically read back data from the Cloud database.
- The delay, called D_{CS}, for transmitting the data to the subscriber, formally

$$D_{CS} = T5 - T4 \tag{4}$$

Additionally, the following secondary metrics, used for highlighting particular behaviors, can be calculated as a combination of the previous ones:

- The latency, called L_{CDC}, for storing a data in database and, then, retrieve it back, formally

$$L_{CDC} = D_{CD} + D_{DC} = T4 - T2 \quad (5)$$

  The L_{CDC} can be used for characterizing the global behavior of the Cloud database, considered as a black box, when access parameters are changed (e.g., trigger functions or polling).
- The latency, called L_{PDS}, between publisher and subscriber in the case the data are stored first in the Cloud database and then sent to subscriber, formally

$$L_{PDS} = D_{PC} + L_{CDC} + D_{CS} = T5 - T1 \quad (6)$$

  The L_{PDS} is the reference metric for characterizing the responsiveness of a Cloud service with Cloud Database inside the feedback loop.
- The latency, called L_{PS}, between publisher and subscriber in the case the data are sent immediately to the subscribed before they are stored in the database, formally

$$L_{PS} = L_{PDS} - L_{CDC} = T5 - T1 - T4 + T2 \quad (7)$$

  The L_{PS} is the reference metric for characterizing the responsiveness of a Cloud service with Cloud Database outside the feedback loop (only used for storing data or logging).

Last, it is worth to say that there is another metric related to scalability introduced, for opportunity reasons, in Section 5.4.

4. The Considered Use Case

In this work, a typical predictive maintenance [34,35] application has been considered as the reference use case; in particular, the solution provided by Metalwork for pneumatic valves controller, formally known as EB80, has been adopted. The EB80 solution aims at inferring the health status of electro-pneumatic actuators collecting diagnostic information of actuators, including reaction times, operating temperatures and so on. By the way, each electro-pneumatic actuator forwards such diagnostic data towards a local PLC. The latter leverages on an IoT-gateway to further propagate data into the Cloud, where they are processed according to the Cloud computing paradigm, i.e., they are stored in a Cloud database (according to the aforementioned DBaaS approach) and are analyzed by algorithms remotely executed in the Cloud.

The capability of the proposed methodology to evaluate various solutions no matter the actual implementation details, has been verified considering two different architectures of the predictive maintenance system. In the first one, the IoT-gateway publishes data on a generic MQTT-Broker, which oversees the propagation of acquired information towards an IBM Cloud platform, including an IBM Cloudant Database [36]. Activities carried out in the IBM Cloud are managed via Node.js configured by means of Node-RED dataflows. The very same generic MQTT-Broker is used by "actuator" and/or end users (e.g., technicians) to access results furnished by the prediction system processing. In the second scenario, the generic MQTT-Broker is substituted by an IBM-MQTT-Broker, that also in this case is adopted for both receiving data from the field and furnishing results to end users. Both the considered arrangements are depicted in Figure 3.

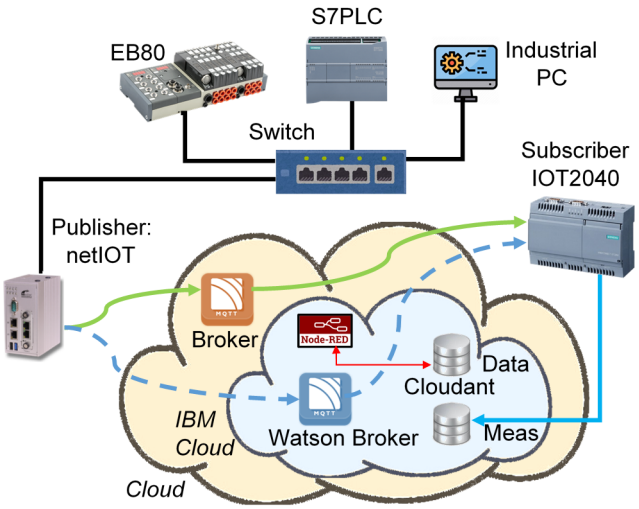

Figure 3. The experimental setup for the considered use case. The database access and the cloud application are the same for both the considered Brokers.

4.1. The Actual Testbench Implementation

The previously described reference scenarios have been implemented by means of industrial devices and software applications, as shown in the previously introduced Figure 3; in particular:

- Data from the field are retrieved by an EB80, the Metalwork electro-pneumatic system including, other than the solenoid valve assembly, digital and analog I/O and power supply modules;
- The local controller is implemented by a S7 PLC from Siemens (the S71215C, supporting Ethernet-based connectivity via both regular TCP/IP stack and PROFINET (an Industrial Real-Time Ethernet protocol));
- The IoT gateway is the Hilscher netIOT device, running a Linux-derived operating system and supporting configuration and management by means of Node-RED interface; the netIOT device embeds an Intel J1900 processor, 4 GB of RAM and 64 GB solid state disk;
- Maintenance related results are collected by an embedded system designed around an IOT2040 from Siemens and running a Yocto Linux operating system distribution; the processor is an Intel Quark X1020, complemented by a 1 GB of RAM;
- The generic MQTT-Broker is an instance of the free, open source iot.eclipse.org broker; as a consequence, Quality of Service (QoS) is not guaranteed, due to the free subscription to the service;
- The IBM Cloud is furnished according to the free subscription reserved for testing proof-of-concept implementations; for this reason, the QoS is not guaranteed as well.

Field devices also include a Siemens industrial workstation, which acts as the local supervisor but has not been involved in the experimental tests discussed in the rest of the paper.

The EB80 system, the netIOT and the IOT2040 devices are all located at the engineering campus of the University of Brescia; local connectivity is provided via the University network, which ensures reliability and high bandwidth. Accordingly, delays in the University local network are ignored.

It must be also highlighted that actual placement of both the iot.eclipse.org and IBM MQTT-Broker is unknown; indeed, due to the virtualization strategies adopted by the providers, location can change time by time.

4.2. Implementation of Timestamp Probes and Measurements Data Collection

All the metrics introduced in the Section 3.1 are based on the availability of as accurate as possible timestamps which allow to track in time the messages.

Authors already demonstrated in previous works that a viable solution is provided by Node-RED; for instance, in [30] they demonstrated that timestamping uncertainty in the order of few milliseconds

can be obtained in typical IoT-like applications. Node-RED is supported by a browser-based frontend which is used to graphically connect and configure the processing nodes implementing the desired application (in agreement with the dataflow programming paradigm). The actual software execution is event-driven and leverages on a Node.js runtime, which is readily available on many different platforms, thus shortening development time and confirming portability.

Once a new timestamp is collected, it is inserted into a purposely defined measurement data structure, which moves along the predictive maintenance data and it is finally populated with the five timestamps Ti, i = 1–5, required to evaluate the proposed metrics. Each new measurement data structure, finalized once it reaches the end user (in this case the IOT2040), is added to the "Meas" Cloudant database shown in Figure 3.

5. Experimental Results

The experiments are divided into three major campaigns: the first one is the longest one and it took ten days; the second and the third campaigns were shorter but more focused on specific aspects related to the database evaluation. Note that the scenarios with two Brokers are simultaneously evaluated, meaning that the involved Cloud functional blocks are duplicated.

5.1. Long Term Reference Experiment

The first experiment is related to the reference situation: data are published every minute and each run ends with the subscriber receiving the data. Exactly 15,000 runs are carried out, simultaneously collecting the measurements from both possible paths shown in Figure 3. Due to the fact that the connection is over Internet, during the days some very long delay situations happened. In order to keep valid measurement data and discard outliers, the outlier detection threshold has been set to exclude the 1% of the measures (i.e., 150 samples are labeled as outliers).

The results in Table 1 show the D_{PC}, which is the delay between the publisher (i.e., IoT gateway) and the user application in Cloud. The advantage of using native direct interface to IBM Cloud offered by the Watson Broker is clear; it is faster (more than 100 ms less in average) than the external MQTT Eclipse Broker. Furthermore, the Watson Broker is less variable than the Eclipse Broker and the maximum delay is one third.

Table 1. Statistics of the D_{PC} (Delay from Publisher to Cloud user application) collected in the experiments. All values are in milliseconds.

Broker Type	Min	Average	P95%	P99%	Max	Std. Dev.
Eclipse	86	145	235	286	338	43
IBM Watson	4	32	55	88	100	15

The results in Table 2 regard the D_{CD} metric. i.e., the delay for writing the data in the Cloud database. The first observation is that the results are independent of the Broker type, since they are only related to the Cloud database that is used. The second observation is that the distribution of the measurements is bimodal (i.e., with measures divided in two groups) as shown in Figure 4. For this specific database (IBM Cloudant) the delay (when a write rate of 1 document per minute is used) can be centered on an average of about 160 ms or centered on an average of 30,160 ms. Figure 4 shows that the probability of experiencing the longer delay is low (5%) but not negligible. This experimental finding is of major importance, and it is the reason why other deeper experimental campaigns have been carried out later on (see in the following).

Table 2. Statistics of the D_{CD} (Delay from Cloud user application to Database) collected in the experiments. All values are in milliseconds.

	Lower Group of Measures		Higher Group of Measures	
Broker Type	**Average**	**Std. Dev.**	**Average**	**Std. Dev.**
Eclipse	150	42	30,152	36
IBM Watson	168	45	30,175	62

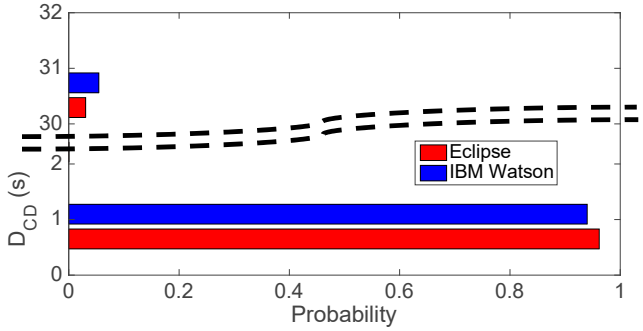

Figure 4. Distribution of the D_{CD} metric during the experiments. There are two groups, the distribution is bimodal.

The results about the D_{DC} metric are in Table 3. In these experiments, the database is accessed using the trigger functions. Hence, as expected, the delay for retrieving a recently written data using the callback function is independent of the Broker type. The experiment results are really very similar in the two situations. In average, only about 50 ms are needed to get back (i.e., into the user application) the new data stored in the Cloud database.

Table 3. Statistics of the D_{DC} (Delay from Database to Cloud user application) collected in the experiments. All values are in milliseconds.

Broker Type	Min	Average	P95%	P99%	Max	Std. Dev.
Eclipse	2	50	73	94	119	13
IBM Watson	2	53	78	99	120	13

The results of the D_{CS} are in the Table 4. As in the case of the $_{DPC}$ the delay for sending data to subscriber is shorter with the Watson Broker with respect to the use of Eclipse Broker (130 ms more in average). Once again, the IBM Watson has a lower variability than the Eclipse Broker.

Table 4. Statistics of the D_{CS} (Delay from Cloud user application to Subscriber) collected in the experiments. All values are in milliseconds.

Broker Type	Min	Average	P95%	P99%	Max	Std. Dev.
Eclipse	102	183	331	364	406	73
IBM Watson	11	27	32	44	56	4

Finally, the results about the L_{PS} and L_{PDS} metrics are reported as they globally resume the goal of this first experiment. The latency between publisher and subscriber describe the entire reaction time of the closed loop architecture. If the database is excluded from the path, the L_{PS} is reported in Table 5. It worth to recall that this metric gives the best possible performance related only to data transfer. As a matter of fact, since cloud elaboration and database operations are excluded, only Internet delays and internal delay of IBM Cloud are considered. The scenario all based on IBM Cloud has a great advantage with respect to the mixed scenario where the Broker is external (Eclipse Broker). The average L_{PS} is 69 ms in the first case and 330 ms (i.e., five time more) in the second one.

Table 5. Statistics of the L_{PS} (Latency between Publisher and Subscriber without passing in the database) collected in the experiments. All values are in milliseconds.

Broker Type	Min	Average	P95%	P99%	Max	Std. Dev.
Eclipse	215	331	487	551	612	89
IBM Watson	30	59	87	124	137	15

On the other hand, the discussion of the L_{PDS} metric is more complicated. The latency between publisher and subscriber when the Database is included is clearly tied to the behavior of the database (and IBM Cloudant seems to introduce a variable delay, i.e., a variable D_{CD}). For this reason, the L_{PDS} is varying from 230 ms to 30,230 ms following the bimodal distribution of D_{CD}.

5.2. Database Access Experiment

The behavior of the database has been studied with another experiments, aimed to determine if the type of software access chosen in the user application could lead to different results. The experiment consists in changing the Node-RED flow in order to use the native IBM Cloudant Node for Node-RED or implementing the Cloudant API REST access using base HTTP Node for Node-RED. Only IBM Watson Broker is used. The experiment took 5 h with a new data published every minute. Table 6 shows the percentage of values of the L_{CDC} metrics that are greater than 5 s. Comparing these results with Figure 4, it can be seen that the solution that minimizes the occurrence of the very long delay is the one that uses API REST for both writing and reading data in the database.

Table 6. Percentage of database access with L_{CDC} greater than 5 s.

	Read Type	
Write Type	**Node-RED Node**	**API REST**
Node-RED node	13.3%	12.3%
API REST	6.3%	5.33%

Referring to Figure 5, and examining more in details the distribution of L_{CDC} (in the four configurations of Table 6 and excluding the $L_{CDC} > 5$ s), it is evident that there are no differences in terms of distribution shape or center. Hence, the different software access method only impacts on the occurrence rate of the long delays, leading to the general suggestion to use API REST for both write and read accesses.

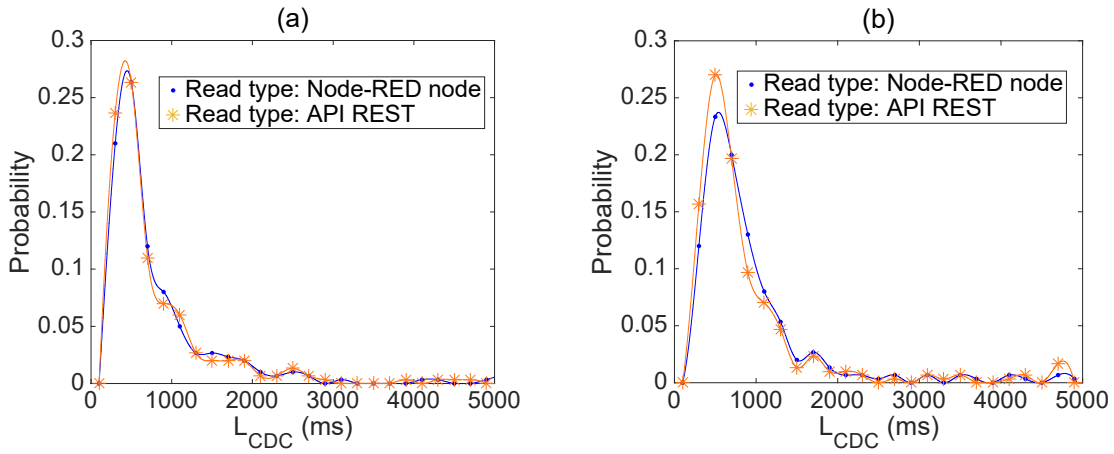

Figure 5. Distribution of the L_{CDC} metric (with $L_{CDC} < 5$ s) in the four configurations of the software access method; (**a**) write type is Node-RED node and (**b**) write type is REST API.

5.3. Database Scalability Experiment

The third part of the experiments is related to the study of the scalability of IoT applications based on Cloud databases. The goal is to determine if scaling can produce changes with respect to the reference scenario evaluated in the first (long term) experiment. In details, in the reference experiment only one device is using the Cloud application and only one database access per minute is considered. When the scenario scales, it is expected to have more and more devices publishing their data to the Cloud. In this last experiment, the number of devices that publish data has been increased to 10, 100 and 500. The devices are not synchronized, and they publish data randomly with the same rate, that is one new data message every minute. In other words, now, the database must be accessed 10, 100, and 500 times per minute. Since, the deployment of hundreds of devices is not feasible with the budget of this project, the publishing of data is simulated with a software that also takes care of randomizing transmission instants.

The details of this particular experiments are shown in Figure 6. The path between the user application and the database is interrupted and the Device Simulator software is now generating the messages as they come from many devices. Another important block is introduced in these experiments, the Rate Limiter. It is always needed to respect the maximum allowed write rate, r_w, of the selected database. This block is not used in the long-term experiments, since there was no possibility that a single device sending data every minute could violate database constraints. Conversely, as the number of devices increases, the use of the rate limiter becomes mandatory, since usually databases rise errors and drop data if the maximum access rate is exceeded. In other words, scalable systems must use such blocks, so it is inserted in this experiment. The behavior of the Rate Limiter is simple: it works dividing the time in a slot of one second; it postpones the sending of write request to the database if the maximum number of write per seconds has been reached in the current second.

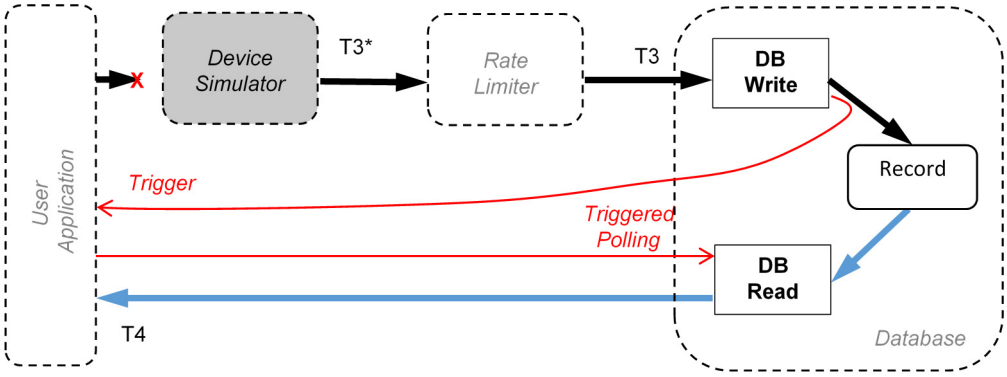

Figure 6. Adding the Device Simulator software for testing the scalability of the system under test. The Rate Limiter block is required not to violate the Database write rate limit.

Even using a Rate Limiter, there is an upper limit to the scalability of the system that is due to the database write rate limit and to the average generation period, t_g, of the devices

$$N_{max} = r_w \cdot t_g. \quad (8)$$

Referring to the Figure 6, an additional metric can be defined:

- The delay, called D_{RL}, that the Rate Limiter introduces in order to enforce the write rate limit of the database (this delay varies for each message since it depends on previous message), formally

$$D_{RL} = T3 - T3^*. \quad (9)$$

For the application of the previous discussion to the current use case, the following parameters have been used: r_w = 10 write/s (standard rate for IBM Cloudant Lite), t_g = 60 s, N = [10,100,500]. Note that, with the given values, the N_{max} = 600 for the considered use case. The experiment simulates an operation interval of 30 min during two moments (day and night) in order to highlight possible problems of congestions of the cloud infrastructure.

The results regarding the D_{RL} are shown in Table 7 and Figure 7. It is clear that the impact of the Rate Limiter is negligible if the number of devices is far from the N_{max}, but it becomes dominant when the devices are 500. In that case, delays up to 3 s are possible. As expected, there are no noticeable differences between experiments carried out during the day or during the night, since the Rate Limiter requires a very limited computation.

Table 7. Statistics of the D_{RL} (delay introduced by the rate limiter block) collected in the scalability experiments. All values are in milliseconds.

Nodes	Conditions	Min	Average	P95%	P99%	Max	Std. Dev.
10	Day	0	5	12	54	68	126
10	Night	0	4	7	60	113	61
100	Day	0	16	97	143	281	67
100	Night	1	17	100	169	301	57
500	Day	2	731	1970	2434	3063	72
500	Night	3	778	1989	2457	3273	76

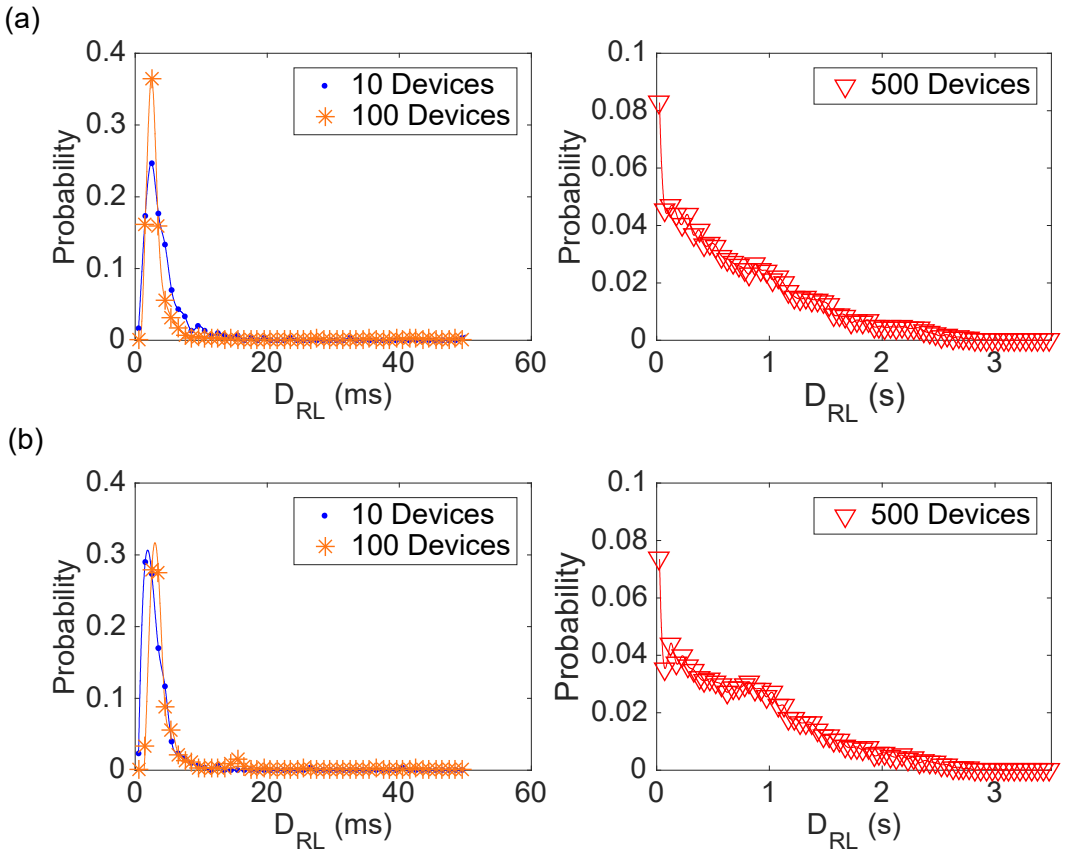

Figure 7. Distribution of the D_{RL} metric varying the number of devices and the moment of the day: (**a**) during the day; (**b**) during the night.

The results regarding the L_{CDC} metric in the scalability experiment are shown in Table 8 and Figure 8. The database itself introduces an overall latency that seems depending on the effective write rate. In detail, the higher the number of devices, the higher the write rate and the lower the database

latency. From the L_{CDC} distribution it appears that the support of the distribution shrinks as the number of devices grows. Although, this behavior seems non-intuitive, it may be in accordance with the previous results about the sporadic long delay introduced by IBM Cloudant. Further discussion is carried out in the next section. Last, the moment of the day when the experiment is done has a very limited effect, it appears to have some influence only when the number of devices is low.

Table 8. Statistics of the L_{CDC} (latency introduced by the database) collected in the scalability experiments. All values are in milliseconds.

Nodes	Conditions	Min	Average	P95%	P99%	Max	Std. Dev.
10	Day	172	308	487	716	1622	126
10	Night	164	252	360	456	682	61
100	Day	107	218	313	409	1432	67
100	Night	109	219	309	399	1238	57
500	Day	101	186	259	327	2033	72
500	Night	104	196	273	377	2657	76

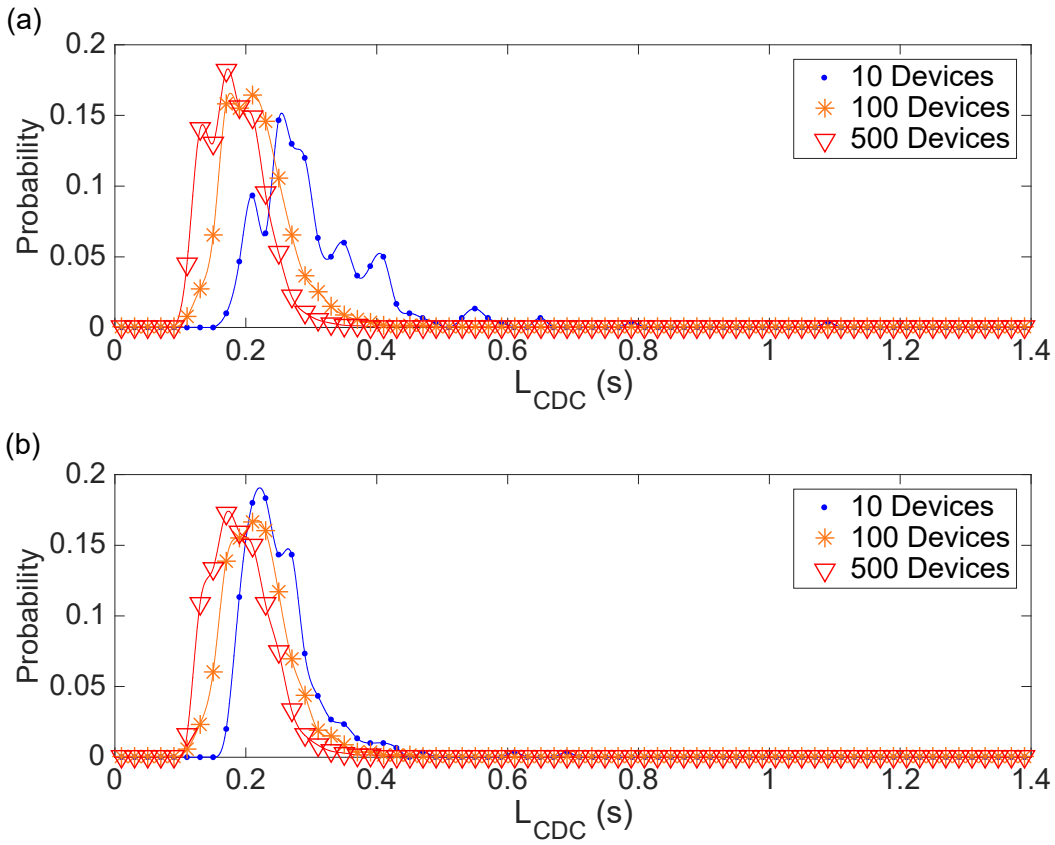

Figure 8. Distribution of the L_{CDC} (latency introduced by the database) collected in the scalability experiments varying the number of devices and the moment of the day: (**a**) during the day; (**b**) during the night.

5.4. Final Discussion about Results and Effectiveness of the Proposed Methodology

The proposed experimental methodology for the characterization of scalable industrial applications based on Cloud services and databases has been applied to an example architecture producing significant results with important insights. It is worth to say that no optimization nor configuration has been done, so the results are just a picture of the situation and they might be improved. However, the important thing "per se" is that the proposed procedure is able to highlight such inner behavior of architecture blocks, which are usually considered as black boxes.

The primary result is a full characterization of all the components of the architecture under test, with the possibility, for the end user, to determine its weak and strong points. Modifications or improvements can be quantitatively evaluated using the proposed procedures.

The first secondary result is the highlighting of the database behavior.

Going more specifically, the IBM Cloudant database has a variable delay that depends on the write rate. Contrasting with usual expectations, the IBM Cloudant reduces latency as the write rate increases. With one write per minute, the delay can be very long, up to 30 s in 5% of the cases. When the number of writing increases, the delay decreases, reaching a minimum for a number of writing close to the maximum allowed write rate. There could be several explanations for this behavior; the most probable is the presence inside the database interfaces of a timeout task that waits some time (for additional incoming requests) before executing the write. When the access is sporadic the entire waiting period must expire (i.e., this is the long delay), while if more requests are received the write operations are performed (as a batch) when the number of requests reach a threshold.

Another secondary result is the underlining of the indirect effect of database write limitation on the scalability. While it is clear that the write rate limit is the upper bound of the scaling range of the considered application, the effect that such limit has on delays is clearly visible only with the proposed methodology. In particular, every single device experiences a variable delay that is a function of the total number of devices. Higher number of devices means higher delays.

The last secondary result is about the evaluation of the real-time capability of architecture under tests. The extensive statistics obtainable with the proposed method can quantify the reaction time of the entire loop (i.e., L_{PDS}, from machine to machine passing in the cloud and in the database). For instance, the use case architecture can be adopted for slowly changing industrial applications only, given that the longest reaction time can be greater than 30 s.

6. Demo of the Use Case and Last Remarks

The experimental setup used for the experiments has been transformed in a demo show bench that Metalwork uses in the trade fair. In particular, the demo system (shown in Figure 9), has been operating during all the 2019 at Hannover Fair and SPS Fair in Europe. The visitors can interact with the EB80 pneumatic valve systems (e.g., causing malfunctions like "air leakage") and see the feedback of the Cloud based predictive maintenances in the dashboard.

Figure 9. Real demo of the predictive maintenance systems based on Cloud services following the architecture of the use case in Figure 3.

During the show it was possible to carry out some additional experiments about the quantity of data exchanged by the field devices and the Cloud. Based on a 10-day long experiment, the publisher EB80 in "demo mode" (which, in order to easy visual interaction with visitors, means a faster publishing rate of t_g = 10 s and a full parameter upload,) and the subscriber (i.e., the dashboard panel) consume about 16 Mbytes of aggregate data per day. Such a last result is given just to underline that also the amount of published and subscribed data should be taken into account for scalability analysis.

7. Conclusions

The Industrial IoT or IIoT is one of the most important parts of the Industry 4.0. Thanks to it, the so-called cyber physical space can be realized, and the overall efficiency and quality of industrial processes can be improved by means of new services.

The large amount of data generated in these industrial applications is usually stored in Cloud databases, leveraging on the DBaaS approach. Therefore, timeliness of Cloud databases is a main concern in any IIoT solution. Unfortunately, neither real-time is easily ensured on large distributed system nor accurate and effective methodologies have been established to evaluate time-related performance.

In this work, authors propose: a measurement methodology based on an abstraction view of IIoT control applications: and a set of metrics able to objectively evaluate time performance of such systems. As a result, the Cloud is treated as a gray box and details about the Cloud database can be estimated. The suggested approach can be easily used for "stress tests" in order to verify the scalability of the system under evaluation.

The on-the-field validation of the propose method has been carried out by means of an extensive measurement campaign based on a real system dealing with predictive maintenance. More in detail, this real-world use case includes: a Siemens S7 PLC, an electro-pneumatic actuators EB80 from Metalworks; and the IBM Bluemix Cloud platform, offering the IBM Cloudant database. Data are transferred using MQTT message-oriented protocol. Thanks to the proposed measurement strategy, some interesting peculiarities of the real system under evaluation are highlighted; for instance, the way the Cloud database stores and retrieves records has a large impact on the overall time performance; and, since different optimization techniques are probably applied depending on the estimated activity, this delay may hugely vary when the number of nodes changes.

Author Contributions: Conceptualization, P.F.; Data curation, S.R. and E.S.; Formal analysis, E.S.; Funding acquisition, A.F.; Investigation, M.P. and A.D.; Methodology, P.F.; Project administration, P.F. and A.F.; Software, S.R., and P.B.; Supervision, A.F.; Validation, P.F.; Writing—original draft, P.F., S.R. and E.S. All authors have read and agreed to the published version of the manuscript.

Funding: The research has been partially funded by University of Brescia and MoSoRe Project.

Acknowledgments: The authors would like to thank eLux laboratory and the University of Brescia for hosting the experimental setup.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xu, L.D.; Xu, E.L.; Li, L. Industry 4.0: State of the art and future trends. *Int. J. Prod. Res.* **2018**, *56*, 2941–2962. [CrossRef]
2. Xu, H.; Yu, W.; Griffith, D.; Golmie, N. A Survey on industrial internet of things: A cyber-physical systems perspective. *IEEE Access* **2018**, *6*, 78238–78259. [CrossRef]
3. Li, J.; Yu, F.R.; Deng, G.; Luo, C.; Ming, Z.; Yan, Q. Industrial internet: A survey on the enabling technologies, applications, and challenges. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1504–1526. [CrossRef]
4. Tao, F.; Zuo, Y.; Xu, L.D.; Zhang, L. IoT-Based intelligent perception and access of manufacturing resource toward cloud manufacturing. *IEEE Trans. Ind. Inform.* **2014**, *10*, 1547–1557.
5. Du, S.; Liu, B.; Ma, H.; Wu, G.; Wu, P. IIOT-based intelligent control and management system for motorcycle endurance test. *IEEE Access* **2018**, *6*, 30567–30576. [CrossRef]

6. Kurte, R.; Salcic, Z.; Wang, K.I. A distributed service framework for the internet of things. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4166–4176. [CrossRef]
7. Nguyen, T.; Gosine, R.G.; Warrianl, P. A systematic review of big data analytics for oil and gas industry 4.0. *IEEE Access* **2020**, *8*, 61183–61201. [CrossRef]
8. Tang, H.; Li, D.; Wan, J.; Imran, M.; Shoaib, M. A reconfigurable method for intelligent manufacturing based on industrial cloud and edge intelligence. *IEEE Internet Things J.* **2020**, *7*, 4248–4259. [CrossRef]
9. Villalonga, A.; Beruvides, G.; Castaño, F.; Haber, R.E. Cloud-based industrial cyber–physical system for data-driven reasoning: A review and use case on an industry 4.0 pilot line. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5975–5984. [CrossRef]
10. Muhuri, P.K.; Shukla, A.K.; Abraham, A. Industry 4.0: A bibliometric analysis and detailed overview. *Eng. Appl. Artif. Intell.* **2019**, *78*, 218–235. [CrossRef]
11. Wollschlaeger, M.; Sauter, T.; Jasperneite, J. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Ind. Electron. Mag.* **2017**, *11*, 17–27. [CrossRef]
12. Bellagente, P.; Ferrari, P.; Flammini, A.; Rinaldi, S.; Sisinni, E. Enabling PROFINET devices to work in IoT: Characterization and requirements. In Proceedings of the IEEE Instrumentation and Measurement Technology Conference, I2MTC, Taipei, Taiwan, 23–26 May 2016.
13. Szymanski, T.H. Supporting consumer services in a deterministic industrial internet core network. *IEEE Commun. Mag.* **2016**, *54*, 110–117. [CrossRef]
14. Rocha, M.S.; Sestito, G.S.; Dias, A.L.; Turcato, A.C.; Brandão, D.; Ferrari, P. On the performance of OPC UA and MQTT for data exchange between industrial plants and cloud servers. *Acta IMEKO* **2019**, *8*, 80–87. [CrossRef]
15. Ferrari, P.; Rinaldi, S.; Sisinni, E.; Colombo, F.; Ghelfi, F.; Maffei, D.; Malara, M. Performance evaluation of full-cloud and edge-cloud architectures for Industrial IoT anomaly detection based on deep learning. In Proceedings of the 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT), Naples, Italy, 4–6 June 2019; pp. 420–425.
16. Tan, D.P.; Li, L.; Zhu, Y.L.; Zheng, S.; Ruan, H.J.; Jiang, X.Y. An embedded cloud database service method for distributed industry monitoring. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2881–2893. [CrossRef]
17. Ferrari, P.; Sisinni, E.; Depari, A.; Flammini, A.; Rinaldi, S.; Bellagente, P.; Pasetti, M. Evaluation of the impact of cloud database services on industrial IoT applications. In Proceedings of the 2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Dubrovnik, Croatia, 25–28 May 2020; pp. 1–6.
18. Stojmenovic, I. Machine-to-machine communications with in-network data aggregation, processing, and actuation for large-scale cyber-physical systems. *IEEE Internet Things J.* **2014**, *1*, 122–128. [CrossRef]
19. Yifeng, L.; Junshi, G.; Jiaye, Z.; Jihong, G.; Shuigeng, Z. Towards efficiently supporting database as a service with QoS guarantees. *J. Syst. Softw.* **2018**, *139*, 51–63.
20. Wang, P.; Chen, X.; Sun, Z. Performance modeling and suitability assessment of data center based on fog computing in smart systems. *IEEE Access* **2018**, *6*, 29587–29593. [CrossRef]
21. Grau, A.; Indri, M.; Lo Bello, L.; Sauter, T. Industrial robotics in factory automation: From the early stage to the internet of things. In Proceedings of the IECON 43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, China, 29 October–1 November 2017; pp. 6159–6164.
22. Yun, U.; Lee, G.; Yoon, E. Efficient high utility pattern mining for establishing manufacturing plans with sliding window control. *IEEE Trans. Ind. Electron.* **2017**, *64*, 7239–7249. [CrossRef]
23. Yun, U.; Lee, G.; Yoon, E. Advanced approach of sliding window based erasable pattern mining with list structure of industrial fields. *Inform. Sci.* **2019**, *494*, 37–59. [CrossRef]
24. Semančik, L. Recording of data monitoring access to databases using triggers. In Proceedings of the 2019 Communication and Information Technologies (KIT), Vysoke Tatry, Slovakia, 9–11 October 2019; pp. 1–5.
25. Benhamida, F.Z.; Casado-Mansilla, D.; Bennani, C.; Lopez-de-lpina, D. Toward a delay tolerant internet of things. In Proceedings of the 9th International Conference on the Internet of Things, art. 34, Bilbao, Spain, 22–25 October 2019; pp. 1–4.
26. Guha Roy, D.; Mahato, B.; De, D.; Buyya, R. Application-aware end-to-end delay and message loss estimation in internet of things (IoT)—MQTT-SN protocols. *Future Gener. Comp. Syst.* **2018**, *89*, 300–316. [CrossRef]

27. Daponte, P.; Lamonaca, F.; Picariello, F.; de Vito, L.; Mazzilli, G.; Tudosa, I. A survey of measurement applications based on IoT. In Proceedings of the IEEE Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2018, Brescia, Italy, 16–18 April 2018; pp. 157–162.
28. Ferrari, P.; Flammini, A.; Rinaldi, S.; Sisinni, E.; Maffei, D.; Malara, M. Impact of quality of service on cloud based industrial IoT applications with OPC UA. *Electronics* **2019**, *7*, 109. [CrossRef]
29. Silva, D.R.C.; Oliveira, G.M.B.; Silva, I.; Ferrari, P.; Sisinni, E. Latency evaluation for MQTT and WebSocket protocols: An Industry 4.0 perspective. In Proceedings of the IEEE Symposium on Computers and Communications, Natal, Brasil, 25–28 June 2018; pp. 1233–1238.
30. Ferrari, P.; Flammini, A.; Sisinni, E.; Rinaldi, S.; Brandao, D.; Rocha, M.S. Delay estimation of industrial IoT applications based on messaging protocols. *IEEE Trans. Instr. Meas.* **2018**, *67*, 2188–2199. [CrossRef]
31. Depari, A.; Fernandes Carvalho, D.; Bellagente, P.; Ferrari, P.; Sisinni, E.; Flammini, A.; Padovani, A. An IoT based architecture for enhancing the effectiveness of prototype medical instruments applied to neurodegenerative disease diagnosis. *Sensors* **2019**, *19*, 1564. [CrossRef]
32. Carvalho, F.; Ferrari, P.; Sisinni, E.; Depari, A.; Rinaldi, S.; Pasetti, M.; Silva, D. A test methodology for evaluating architectural delays of LoRaWAN implementations. *Pervasive Mobile Comput.* **2019**, *56*, 1–17. [CrossRef]
33. Ferrari, P.; Bellagente, P.; Depari, A.; Flammini, A.; Pasetti, M.; Rinaldi, S.; Sisinni, E. Evaluation of the impact on industrial applications of NTP Used by IoT devices. In Proceedings of the 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 3–5 June 2020; pp. 223–228.
34. Compare, M.; Baraldi, P.; Zio, E. Challenges to IoT-enabled predictive maintenance for industry 4.0. *IEEE Internet Things J.* **2020**, *7*, 4585–4597. [CrossRef]
35. Yu, W.; Dillon, T.; Mostafa, F.; Rahayu, W.; Liu, Y. A global manufacturing big data ecosystem for fault detection in predictive maintenance. *IEEE Trans. Ind. Inform.* **2020**, *16*, 183–192. [CrossRef]
36. Meraji, S.; Lavoy, C.; Hall, B.; Wang, A.; Rothenstein, G.; Davis, P. Towards performance evaluation of cloudant for customer representative workloads. In Proceedings of the 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW), Berlin, Germany, 4–6 April 2016; pp. 144–147.