Impact of Data Quality in Real-Time Big Data Systems

Jorge Merino[1]*, Xiang Xie[2], Ajith Kumar Parlikad[3], Ian Lewis[4], Duncan McFarlane[5]

[1235]Institute for Manufacturing, University of Cambridge, Cambridge, United Kingdom
[1]*jm2210@cam.ac.uk
[4]Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom

Abstract. Data Quality is one of the main challenges in any type of Big Data System. Timeliness is one of the main factors in real-time Big Data. Limiting data quality evaluations to data sources may be insufficient in Big Data Systems with high Velocity and Variability. On the other hand, real-time Data Quality evaluations throughout the Big Data Pipeline can be costly (i.e., latency introduced by Data Quality Evaluations). This paper identifies four categories – embedded, parallel, in-line, and independent– of approaches for Big Data Quality Evaluation available in the literature. A real-time Big Data System based on the SmartCambridge Real-Time Data Platform is deployed and used as basis to implement a representative case for each one of the four categories identified. An application for bus catching dynamic prediction is used as case study to quantify the impact of these Data Quality Evaluations in the Real-Time Data Platform in terms of latency introduced in the system. Results suggests that the impact of Data Quality Evaluations differ depending on the type of method used, and that the main factors are the data transfers between Data Quality modules and the data processing algorithms, the synchronisation of messages, and the complexity of the Data Quality algorithms.

Keywords: Big Data · Data Quality · Real-Time · Timeliness · Internet of Things · Smart Cities · Velocity · Latency · Streaming · SmartCambridge

1 Introduction

Smart cities provide intelligent services over six axes (i.e., economy, mobility, environment, people, living, and governance) to improve the quality of life of citizens [1]. Social Network, meteorology, and IoT are often used as sources of real-time data for smart commuting (e.g., traffic, public transport), pollution analysis (e.g., quality of air and water, noise levels) among others. Most smart cities applications have to deal with high-volume, high-velocity, and/or high-variety data [2].

Real-Time Big Data systems collect and process data from different streaming sources like sensors, smart tags, networks, and other systems (e.g., meteorological, social networks, etc.) while minimising the latency. Data is often integrated with non-real-time data of spaces, assets, products, and processes to make timely and informed decisions.

Each one of these sources have different data quality levels depending on factors like errors in the readings, granularity in time and space of the data, quality of communication networks, and data processing capability [2]. All these factors affect Timeliness, which can be defined as the minimum acceptable latency in a decision-making process [3]. Timeliness is one of the main factors that affect the decision quality in Real-Time Big Data systems [4], [5]. Timeliness is also one of the dimensions that define Data Quality, commonly defined as "fit-for-use" [6]. If decision is made on outdated data, it is considered as poor-quality and will likely be suboptimal [1], [2], [7]–[10].

A myriad of Big Data Quality Evaluation alternatives coexists in the literature. Many conduct the evaluation of Data Quality on a subset or a sample of the data [11]–[13] which can help optimising the time to evaluate Data Quality but may reduce the significance and trust of the evaluation. A number of approaches suggest to evaluate data quality separately from the main Big Data pipeline in order to avoid impacting the latency [14]–[16]. However, it has been proven that data quality errors can be introduced at different points in the analytics pipeline [17]. A Big Data analysis may perceive data sources as trusted if these data sources scored good quality levels in the past, but this trust may be outdated. Introducing data quality evaluation steps (i.e., real-time evaluation) may enhance timeliness, and thus, decision-making, but may also incur cost every time the pipeline is executed. Extra cost is introduced in terms of a) additional latency inserted in every new evaluation step added to the system, b) additional memory load to temporarily store data for its quality evaluation, c) concurrent computation demand to evaluate data quality at the same data is being processed by Big Data analyses, or d) higher complexity of Big Data analyses hindering maintenance.

This paper introduces a classification of methods for Big Data Quality Evaluation approaches: Embedded, Parallel, In-line, and Independent. An implementation of each type of evaluation approaches is implemented into a Real-Time Big Data System (Intelligent City Platform [18]). The implementations are tested using a smart city application for Bus catching prediction as a case study. The latency introduced by each implementation is quantified to measure the impact of the different Big Data Quality evaluation categories. The results of each category are compared, and the benefits and drawbacks of each alternative are analysed.

Section 2 contextualizes the concept of Data Quality for Big Data and extracts four categories of methods for Big Data Quality evaluation from the literature. A research method based on latency quantification and benchmarking is presented in section 3. Section 4 describes the framework and the case study used to benchmark the different categories of Big Data Quality evaluation methods. The results of the benchmark are shown in section 5. Section 6 provides conclusions.

2 Data Quality Evaluation in Big Data

Gartner defines Big Data as "high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for

enhanced insight and decision making" [19]. Some authors have introduced Value, and Veracity to the set [20], [21]. Big Data is often managed through Big Data Pipelines (Collection, Preparation/Curation, Analysis, Visualisation, and Access) [22]. [23] differentiates between data quality (i.e., assessed at the time data is collected) and information quality (i.e., assessed at the time data is being used).

[17] proved that even that Data Quality is reduced in the same rate as the Volume increases, and Variety exponentially reduces the Consistency of data. [17] also identified that many errors are introduced during the traditional Big Data Pipeline and not all data defects may be filtered. Thus, it highlights the compromise between filtering data defects versus the use of a quality-based trust factor for the Big Data Analysis in a very much needed multiphase data quality evaluation. Most approaches conduct the evaluation of Big Data Quality during the Data Collection phase [11]–[13], [24] or during Data Preparation/Curation phase [5], [7], [16], [25]–[27]. For instance, data miners focus on accuracy and outliers removal as the most important factors that define data quality [28], [29]. [30] measures the quality of integration of different data schemas which affect the quality of the data used in the Big Data Analysis phase. [14] proposes a framework to evaluate quality of Big Data after each phase of the traditional Big Data Pipeline and acknowledges that different Data Quality dimensions apply to some/all phases of the Big Data Pipeline.

[31] proposes an ontology-based data quality measurement and monitoring framework for data streams, including content (i.e., semantics of the data flowing in the stream), queries (i.e., aggregation and integration), and application (i.e., context-dependent quality requirements). [3] analyses the quality measures for image-based crowd-sourced big data, emphasising the time related dimensions over the rest in real-time Big Data systems. [15] centres its attention in the timeliness of the Big Data Analysis providing a method to measure the freshness of data. Some of these authors emphasize the trade-off between Data Quality and performance for real-time data in their approaches. [31] affirms that while performance must ensure real-time processing, sufficient quality of computed results must be achieved. [11], [12] suggests boosting performance by sampling the available data for quality evaluation, and then use regression to extrapolate results. Sampling may help to reduce the overall latency of the Big Data Pipeline and keep performance, but control mechanisms like sampling must be tuned to preserve the balance between performance and sufficient data quality evaluation [31].

2.1 Classification of methods for Big Data Quality evaluations

This section introduces one of the contributions of this paper: a classification of the methods available in the literature for Big Data Quality evaluation. Considering the above, four categories of methods have been identified:

1. *Embedded:* These methods evaluate quality during the Data Processing step by incorporating quality constraints to the Big Data Analysis. It allows tighter connections to the business applications (e.g., most Machine Learning regression algorithms include outlier removal) at the cost of higher complexity in maintenance of

the system and lower flexibility. These methods are used when adequate levels of Data Quality are essential to either filter out defective data or annotate it for further analysis.

2. *Parallel:* In this type of methods the evaluation is conducted concurrently in a separate data flow from the Big Data Analysis. Performance is often the main driver of the Big Data Analysis (e.g., high-Velocity or high-Volume data that must be rapidly processed), but quality also plays a key role. Hence, quality evaluation must not introduce any latency in the main data processing flow. Quality evaluation results are used as risk indicators (i.e., analysis metadata) that can be checked after the Big Data Analysis is finished (e.g., when the Data Quality levels are not sufficient, the risk of using the results of a Big Data Analysis on the same data to make a decision is high). These methods are commonly used to monitor Data Quality in in Data Stream applications.
3. *In-line:* These methods evaluate data quality before the Big Data Analysis, but in the same data flow. These scenarios are more commonly known as data curation, and they prioritise the identification and correction or filtering of data defects. Used in Big Data systems supporting vital decision-making and performance being secondary. These methods include data profiling, assurance, lineage, data tagging, and filtering.
4. *Independent:* These methods conduct the evaluation in an unconnected data flow and usually before the main Big Data Pipeline is even executed. It the least disruptive approach as the evaluation is conducted independently from the Big Data Analysis. The main drawback is that these approaches are not designed to evaluate the data on-the-go as it flows throughout the Data Pipeline. Thence, the Data Quality levels do not normally consider the message being processed in real-time by the Big Data Analysis, but rather historical data. The results of the data quality evaluations are often stored as metadata of the data sources, or in a metadata repository when evaluations are more granular and/or it comes from different sources. This category includes methods for data linage, data source evaluation, data provenance, data cataloguing.

Table 1 classifies the Big Data Evaluation methods in the literature according to these 4 categories (some methods can be classified into more than one category).

Table 1. Classification of methods from the literature

Embedded	[9], [14], [25], [26], [28]
Parallel	[13]–[15], [24], [31], [32]
In-Line	[3], [7], [13]–[16], [21], [25]–[27], [29], [30], [32]
Independent	[5], [11], [12], [16], [17], [24], [25], [29]–[31], [33]

3 Research method

This paper uses the term “real-time” to refer to an unobstructed flow of incoming data through a Big Data system. We use a publish-subscribe Real-Time system that manage fast-paced data (also referred as Data Streams) and prioritise the minimization of

end-to-end latency (i.e., Velocity). It is important to highlight that in this system, the messages passing are handled asynchronously. This means that the messages are not directly available in the memory of the subscriber when a module publishes them, but when the subscriber has utterly received them.

The impact of Data Quality in this type of Big Data Systems will be analysed by measuring and comparing the latency introduced by Embedded, Parallel, In-Line, and Independent methods for Big Data Quality Evaluation. Not all the alternatives from the literature are benchmarked, but rather, a custom implementation applicable in the selected framework (see section 4) representing each one of the identified categories. The impact is always quantified in terms of latency introduced in the normal flow of data, but it varies depending on the addressed category. **Fig. 1** visually shows the formulas to calculate the latency.

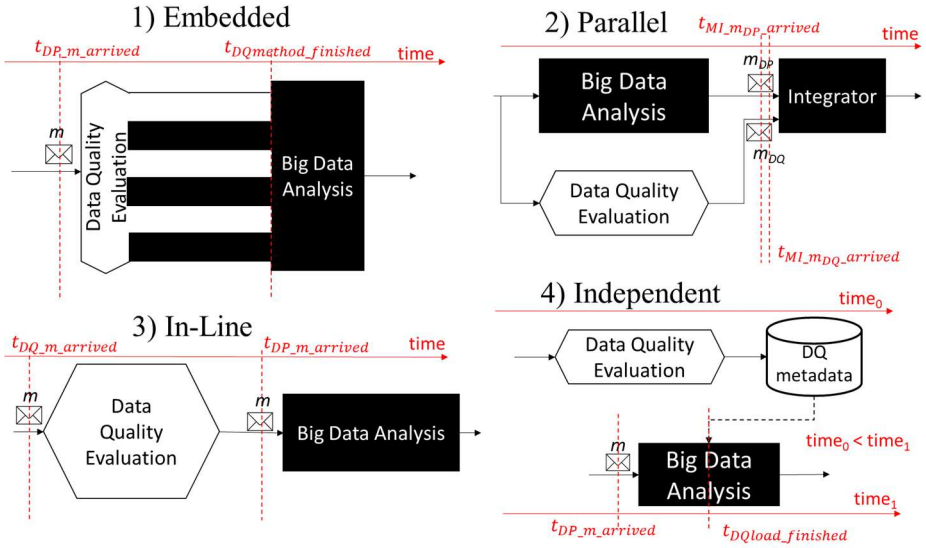

Fig. 1. Latency introduced by the four categories of methods

Given a data quality method *DQmethod* that evaluates the quality of a message *m*, and a data processing task *DP* that processes the message *m*:

- *Embedded*: $latency = t_{DQmethod\_finished} - t_{DP\_m\_arrived}$, with $t_{DQmethod\_finished}$ being the time when the algorithm of the *DQmethod* finishes, and $t_{DP\_m\_arrived}$ being the time when the message *m* arrives to the data processor DP. As the *DQmethod* is embedded into the DP, the data transfer does not introduce latency. Hence, the latency is always a positive number, $latency \in (0, \infty)$.
- *Parallel*: In this approach, the data processing tasks are not directly impacted by the data quality evaluation algorithms as they run concurrently. Both may impact each other in the cases when they share infrastructure. Notwithstanding, it is necessary to introduce a MsgIntegrator MI to integrate the processed data and the data quality evaluation results. Consequently, $latency = t_{MI\_m_{DQ}\_arrived} -$

$t_{MI\_m_{DP}\_arrived}$, with $t_{MI\_m_{DQ}\_arrived}$ being the time when the message m_{DQ} from *DQmethod* arrives to the MsgIntegrator, and $t_{MI\_m_{DP}\_arrived}$ being the time when the message m_{DP} from the data processor DP arrives to the MsgIntegrator MI. Thereby, the latency could be negative if m_{DQ} arrives before m_{DP}, latency $\in (-\infty, \infty)$.

- *In-line*: $latency = t_{DP\_m\_arrived} - t_{DQ\_m\_arrived}$, with $t_{DP\_m\_arrived}$ being the time when the message *m* arrives to the data processor DP and $t_{DQ\_m\_arrived}$ being the time when the message *m* arrives to the *DQmethod*. The *DQmethod* is executed before the *DataProcessor*. As a result, the latency includes the duration of the *DQmethod* itself plus data transfers. The latency is a positive number, $latency \in (0, \infty)$.
- *Independent*: These alternatives run the *DQmethod* independently in a different timeline ($time_0$) and store the result in a DQ metadata repository. DQ levels are used during the normal flow of data ($time_1$). Thus, the latency introduced depends only on the duration of loading the data quality levels from the data quality metadata repository. Then, the latency is defined as $latency = t_{DQload\_finished} - t_{DP\_m\_arrived}$, with $t_{DQload\_finished}$ being the time when data quality levels have been loaded from the data quality metadata repository, and $t_{DP\_m\_arrived}$ being the time when the message *m* arrives to the data processor DP. Thence, the latency is a positive number, $latency \in (0, \infty)$.

The formulas have been defined with the objective of quantifying the entire latency introduced by each DQ approach. This means that the formulas include not only, the time to run the *DQmethod* algorithm itself, but also the time to transfer the messages, loading the data quality metadata from repositories, or synchronising messages for integration. In this manner, the latencies can be compared between different categories. The implementation of these formulas is described in the following section alongside the depiction of the Big Data System that serves as a framework for the implementation, and the case study used for the benchmark.

4 Framework and Case Study – Smart Cambridge iCP

The Smart Cambridge Intelligent City Platform (iCP) gathers data from sensors installed around the Cambridge region. A Data Hub is used to ingest, process, visualise, and provide access to collected data in real-time. The main purpose of the iCP is to explore the future of transport around Cambridge, including how traffic congestion can be reduced and air quality improved. Implemented applications include car parks surveillance and status (Cameras and barriers counters), traffic status (traffic lights, traffic monitoring cameras), public transport (busses GPS, train schedules), and air quality (Air quality sensors, including CO_2). A new LoRa (Low Power Long Range) network has also been established in collaboration with the University of Cambridge to transfer the data flowing in from multi-purpose sensors to the Data Hub. One of the main drivers of the iCP is data availability and visualisation to enable the creation of a

city level digital twin. This information is visualised in smart panels in some buildings at the University of Cambridge and is publicly available at [34].

The architecture of the Data Hub proposed by [18] is implemented as a framework to benchmark the latency introduced by the identified types of Big Data Quality Evaluation approaches in section 2.1. The core architecture [35] of the Data Hub is depicted in **Fig. 2**. The design of the architecture is based on the real-time publish-subscribe model and it is implemented using Java and Vertx [36]. Latency between messages input to output is in the range of milliseconds. The main modules in this architecture are described below:

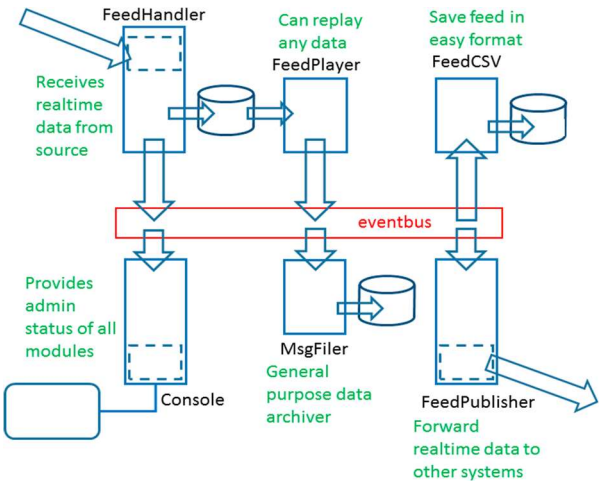

Fig. 2. Architecture of the Real-Time Data Platform for the Data Hub in iCP [35]

- *EventBus*: serves as a route of push communication between the different modules. It has different topics that the modules can subscribe and publish to. A topic is an address in the *EventBus* (e.g., buses/sensors/, is a topic for all the sensors in the buses).
- *FeedHandlers*: subscribe to different data sources (e.g., vehicle position data) and publishes it on the *EventBus*. In some cases, the format of incoming data may be in a different format. Thus, the *FeedHandlers* can parse incoming data into a format that the platform can easily manage and understand (i.e., JSON). *FeedHandler* also archives every post of binary data as a timestamped file in the server filesystem.
- *FeedMakers*: Often, it is not possible to subscribe to data sources from legacy systems. Most cases these systems store data in relational data bases, XML, Excel files, etc. In those cases, a *FeedMaker* will query data from those sources periodically (typically minutes) store it raw in the file system, and parse and publish it as a message on the *EventBus*.
- *MsgFilers*: are general-purpose modules that subscribe to messages on the *EventBus* and persist them in the filesystem. *FeedCSV* is a specific type of *MsgFiler* that stores the bus position data in CSV format.

4.1 Data Quality Evaluation in the Real-Time Data Platform

Two new modules called *DQAnalyser* and *MsgIntegrator* were developed to enable Data Quality Evaluation in the Real-Time Data Platform, and an additional module named *DataProcessor* responsible for the data processing tasks (i.e., Big Data Analysis). These modules are used to implement the formulas to calculate the latency introduced by data quality. **Fig. 3** shows how these modules implement those formulas for each one of the four categories.

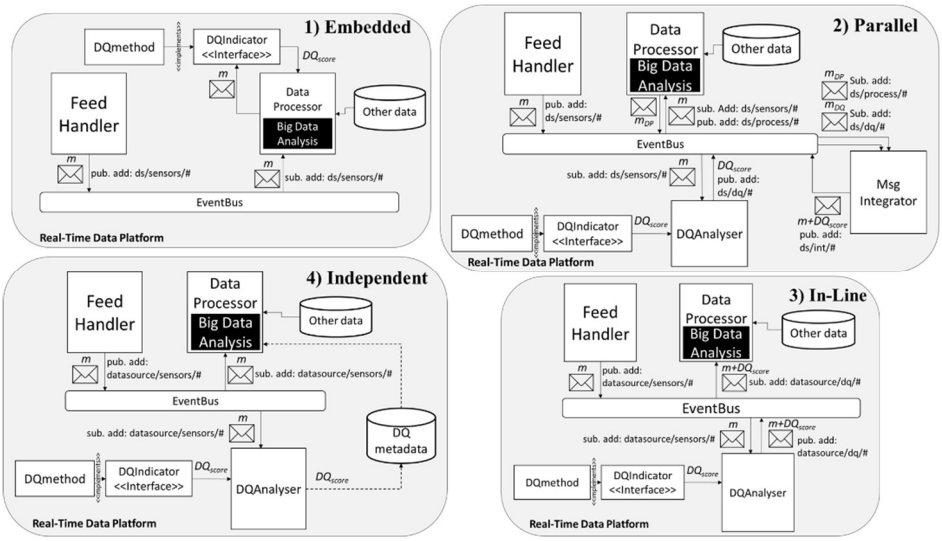

Fig. 3. Implementation of the four categories for Big Data Quality Evaluation in the Real-Time Data Platform from iCP

- *DQAnalyser*: analyses data quality of messages flowing in a particular topic in the *EventBus*. *DQAnalyser* can be configured to either attach the data quality score to the message or store it as metadata in a data-quality-metadata repository. *DQAnalyser* can persist the messages with the Data Quality score in the file system and publish it back to the *EventBus*. *DQAnalyser* creates *DQIndicators* to configure the algorithm for the *DQmethod* that calculates the data quality score of the messages. Thus, new algorithms can be easily incorporated.
- *MsgIntegrator*: acts like a funnel, reading messages from multiple topics in the EventBus and publishing them back to a common topic. *MsgIntegrator* can also synchronise the messages from the subscribed topics and integrate them.
- *DataProcessor*: dedicated to reading messages from one or multiple addresses in the *EventBus* and from the filesystem and conduct processing tasks on them. The *DataProcessor* can publish the processed messages back in the *EventBus* or store them in the filesystem. *DataProcessors* can be configured to be data quality aware, in which case, they read from a data-quality-metadata repository. Data Processors can also call *DQIndicators* that implement *DQmethod* algorithms

4.2 Case study: Bus catching prediction

A bus catching prediction application is used as a case study to generate data on the latency introduced by each one of the four types of alternatives for Big Data Quality (Embedded, Parallel, In-line, Independent). The application reads the user position and predicts whether the user can catch the next bus to arrive to the desired bus stop. It uses the data from buses geolocation, routes, and schedules from the iCP platform, plus the geolocation from the user. Then calculates the likelihood of the user arriving to the bus stop before the next bus and suggests the user to either take the next bus or wait for the following one. **Fig. 4** shows the general data flow and decision-making process of the application. Given that the application itself is not the main goal of the case study, the decision has been simplified to facilitate understanding.

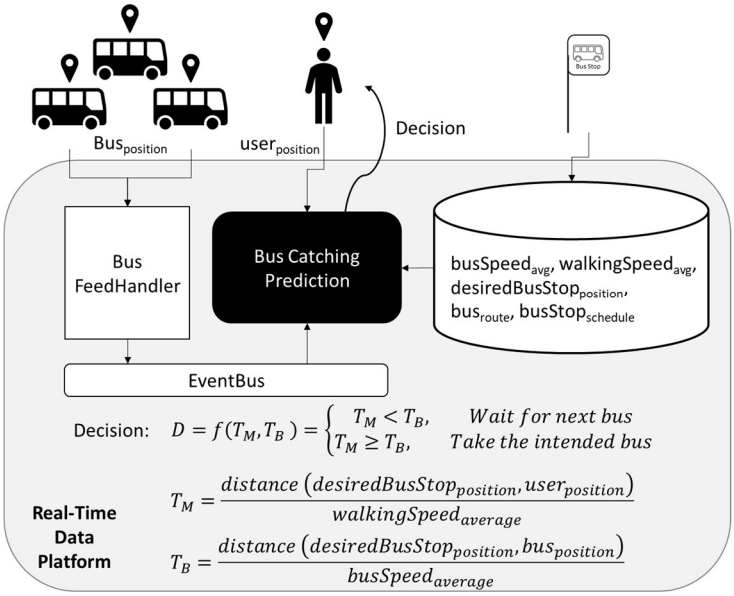

Fig. 4. Bus catching prediction application data flow and decision-making

Some data quality concerns arise from the decision-making process of this application. As it uses asynchronously collected real-time data, there is always a delay between the last geolocation position submitted by a given bus and its real geolocation position. This is an example of Timeliness of data, as it creates a time window in which it is safe to make the decision using the last bus position reading. Given that the purpose of this case study is to quantify latency, the DQ evaluation has been simplified.

5 Analysis of the Impact of Data Quality in Real-Time Big Data

1000 messages from buses were ingested to analyse the impact (i.e., in terms of latency) of the different categories for Big Data Quality Evaluation: Embedded, Parallel,

In-Line, and Independent. **Table 2** compares the latency introduced by these four categories in the system. **Fig. 5** visualises the latency introduced while processing the 1000 messages in chronological order and its distribution. These experimental results were obtained from taking timestamps for each method as defined in section 3.

Table 2. Aggregated Indicators on the latency introduced by the four categories for Big Data Quality Evaluation

Latency Aggregated Indicators (in ms)	Embedded	Parallel	In-Line	Independent
Average	0,980	5,838	1,221	0,447
St. Dev.	2,903	28,695	1,299	1,330
Mode	0,000	0,000	1,000	0,000
Min	0,000	-23,000	0,000	0,000
Max	79,000	175,000	10,000	22,000

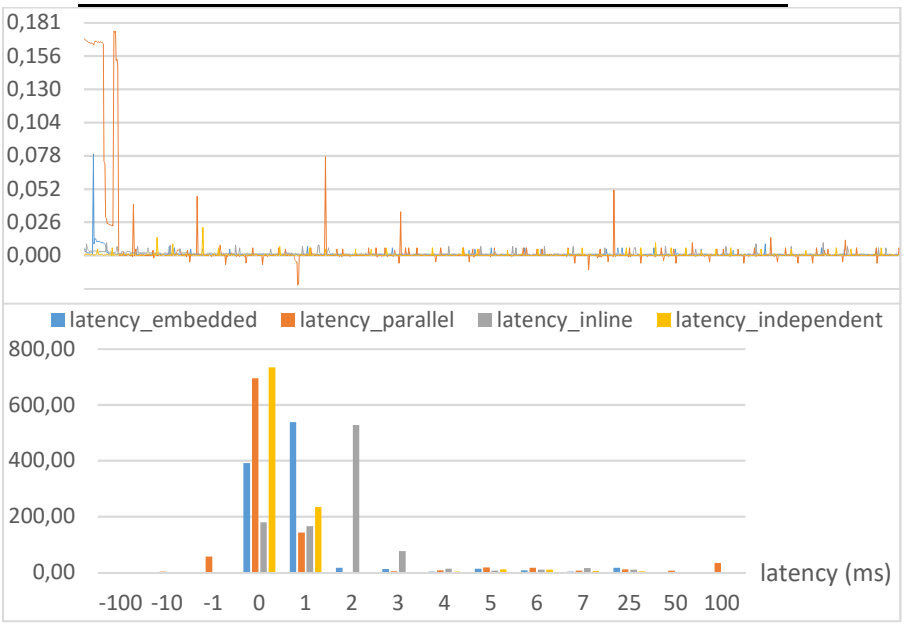

Fig. 5. Latency introduced by the four categories of approaches for Big Data Quality. a) Chronological order; b) Distribution of latency introduced in milliseconds

Fig. 5 shows that counter-intuitively, the impact of Parallel evaluation is higher on some occasions. It is also possible to see that the latency frequently stays below 1ms. **Table 2** confirms these facts. Even that there are some occasions when the latency of the *DQmethod* can be nullified (i.e., negatives data points in the graph), the indicators show that the impact of synchronising the messages from the *DataProcessor* and the *DQAnalyser* can be huge in an asynchronous systems when compared to the other alternatives. Latency introduced by each type of method depends on different factors:

- *Embedded*: No *DQAnalyser* module is necessary in this case, *DQIndicator* modules are called from the *DataProcessor*. *DQIndicator* modules are responsible for calling the right *DQmethod* (for this case study, the Timeliness evaluator; see sections 4.1, and 4.2). In this implementation, the calls from the *DataProcessor* and the *DQAnalyser* also follow the publish-subscribe paradigm as all the modules work asynchronously, avoiding waits and locks. Hence, the latency can be defined as a function of the complexity of the *DQmethod* used for the evaluation plus the data transfers depending on the framework, $latency = O(DQmethod)[+data_{transfer}]$.
- *Parallel*: The implementation of this approach in this paper includes a *MsgIntegrator* responsible for collecting, synchronising, and integrating the messages m_{DP} and m_{DQ} from the *DataProcessor* and the *DQAnalyser* respectively (see section 4.1). Therefore, the latency can be defined as a function of the syncronisation phase, as the *MsgIntegrator* needs to wait for the last message to arrive: $latency = sync(m_{DP}, m_{DQ}) = \left| t_{MI_{m_{DP}arrived}} - t_{MI_{m_{DQ}arrived}} \right|$.
- *In-Line*: Similarly to embedded alternatives, the *DQAnalyser* needs to evaluate the quality of the message *m* before it enters the *DataProcessor* (see sections 3, and 4.1). Thereby, the latency can be defined as a function of the data transfers plus the complexity of the *DQmethod* used for the evaluation, $latency = O(DQmethod) + data_{transfer}$.
- *Independent*: In the implementation of this approach for the case study, the *DataProcessor* queries a metadata repository with the data quality information on the topics in the *EventBus* (see sections 3, and 4.1). Consequently, the latency can be defined as a function of the data transfer (i.e., data querying from the metadata repository), $latency = data_{transfer}$.

Table 3. Impact of the factors introducing latency for each approach

Embedded		**Parallel**	
DQmethod < data transfer	546	Slower DQmethod	247
DQmethod > data transfer	241	Slower DataProcessor	71
DQmethod = data transfer	222	Equal	683
Inline		**Independent**	
DQmethod < data transfer	573	DQ metadata import slow	269
DQmethod > data transfer	233	No latency introduced by import	740
DQmethod = data transfer	196		

Table 3 shows which of the aforementioned factors supposed the larger portion of latency the 1000 messages analysed. None of the approaches for Big Data Quality Evaluation seems to be best in all cases. Generally, for *Embedded* and *In-line* approaches, the latency introduced in the system by the Data Quality algorithms – represented by the *DQmethods*– can be calculated as a function of the complexity of the algorithms themselves. Regardless of the complexity, the algorithms will likely process individual messages, ergo the latency will remain low. Even for the *Independent* evaluation approach, where the algorithm typically processes entire Big Data sources, the evaluation happens in an independently from the main data flow, causing

no impact in it. As a result, the latency due to the Data Quality algorithms alone can remain low.

In the cases where multiple messages are necessary to evaluate Data Quality, the complexity of the algorithms can cause a larger impact. In such cases, it is likely that the *DataProcessors* need to analyse multiple messages as well, making *Parallel* and *Independent* (historical data) evaluations may be more appropriate. Even that the Parallel evaluation approach suffers from synchronisation delays, the latency due to synchronization of Big Data Analysis and Data Quality results would be much lower.

Latency of data transfers between modules can differ from system to system. Generally, it can be represented as a function directly proportional to the number and duration of data transfers between modules. In the case study, the latency introduced by data transfers remained relatively comparable to the latency of the *DQmethods*, albeit it drove the main cost of the latency functions. This can be exacerbated in the cases when the modules of the platform are distributed over a network rather than in the same server –like in the case study. Considering that one of the main advantages of asynchronous systems is the ability of distributing the modules over a network, it is possible to assert that the latency of the data transfer will drive the main impact of the Data Quality in real-time Big Data Systems since more transfers are added to enable it. In such cases, the Embedded methods may be more appropriate since the data transfer factor for those methods will likely be local or inexistent.

6 Conclusions

Multiple new technologies are being used to ingest data in scales never seen in the past. From IoT to Social Networks, these technologies are generating data in volumes, shapes, and at paces that set special requirements. Big Data Systems need to meet those requirements in order to ingest, process, and providing insights on that data.

Data quality is one of the main challenges when creating these insights. Literature has addressed data quality dimensions like accuracy, completeness, or Consistency more frequently in Big Data. Nonetheless, current DQ measures are simplistic for real-time spatiotemporal data where recognising a DQ issue requires multiple messages with an implied latency. Timelines is being identified as one of the most important dimensions of data quality when it comes to real-time Big Data systems. Thus, making decisions based on outdated data will utterly drive to incorrect insights.

One of the contributions of this paper is a classification of the methods available in the literature for Big Data Quality evaluation, from the performance of the real-time systems angle. The identified categories include Embedded, Parallel, In-line, and Independent methods. A second contribution is a benchmark of the different types of methods from the classification in terms of the latency introduced in a real-time Big Data system. The following actions have been conducted to enable this comparison. First, four formulas have been designed to quantify the latency of the four categories. Secondly, A real-time Big Data System has been deployed to serve as a framework. Third, one method representing each category has been implemented in the aforementioned framework. Fourth, the methods have been executed using data from a bus

catching prediction application in order to measure the latency. Finally, the results have been analysed and the categories have been compared.

Results suggests that the impact of Data Quality Evaluations differ depending on the category of the method used. The main factors driving the impact in terms of latency are the data transfers between Data Quality and the Data Analytics algorithms, the synchronisation of messages, and the complexity of the Data Quality algorithms. The nature of the Big Data system will strongly influence the design of the Big Data Quality Evaluation [3], [31], [32]. Each Big Data system has different characteristics (i.e., V's) as well as different priorities or needs (e.g., analysis performance in terms of latency or amount of processed data). For instance, Big Data Analysis using fast-paced data (i.e., Velocity is high), the Data Quality evaluation will get outdated as new data is analysed in the system. These aspects constraint how the data quality measures can be executed.

7 References

[1] A. Caragliu, C. D. Bo, and P. Nijkamp, 'Smart Cities in Europe', *Journal of Urban Technology*, vol. 18, no. 2, pp. 65–82, Apr. 2011.

[2] P. Barnaghi, M. Bermudez-Edo, and R. Tönjes, 'Challenges for Quality of Data in Smart Cities', *J. Data and Information Quality*, vol. 6, no. 2–3, p. 6:1–6:4, Jun. 2015.

[3] Z. Kugler *et al.*, 'Time related quality dimensions of urban remotely sensed Big Data', in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Delft, The Netherlands, Sep. 2018, vol. XLII–4, pp. 315–320.

[4] M. Ghasemaghaei and G. Calic, 'Can big data improve firm decision quality? The role of data quality and data diagnosticity', *DSS*, vol. 120, pp. 38–49, May 2019.

[5] M. Janssen, H. van der Voort, and A. Wahyudi, 'Factors influencing big data decision-making quality', *J. of Business Research*, vol. 70, pp. 338–345, Jan. 2017.

[6] R. Y. Wang and D. M. Strong, 'Beyond Accuracy: What Data Quality Means to Data Consumers', *J. Manage. Inf. Syst.*, vol. 12, no. 4, pp. 5–33, Mar. 1996.

[7] M. Mezzanzanica, R. Boselli, M. Cesarini, and F. Mercorio, 'A model-based evaluation of data quality activities in KDD', *Information Processing & Management*, vol. 51, no. 2, pp. 144–166, Mar. 2015.

[8] S. El Kadiri *et al.*, 'Current trends on ICT technologies for enterprise information systems', *Computers in Industry*, vol. 79, pp. 14–33, Jun. 2016.

[9] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, 'Machine learning for internet of things data analysis: a survey', *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, Aug. 2018.

[10] P. Sotres, J. R. Santana, L. Sánchez, J. Lanza, and L. Muñoz, 'Practical Lessons From the Deployment and Management of a Smart City Internet-of-Things Infrastructure: The SmartSantander Testbed Case', *IEEE Access*, vol. 5, pp. 14309–14322, 2017.

[11] H. Baqa, N. B. Truong, N. Crespi, G. M. Lee, and F. L. Gall, 'Quality of Information as an indicator of Trust in the Internet of Things', in *17th IEE TrustCom/12th IEE BigDataSE*, Aug. 2018, pp. 204–211.

[12] I. Taleb, M. A. Serhani, and R. Dssouli, 'Big Data Quality: A Data Quality Profiling Model', in *Services 2019*, 2019, pp. 61–77.

[13] P. Ceravolo and E. Bellini, 'Towards Configurable Composite Data Quality Assessment', in *21st IEEE Conf. on Business Informatics*, Jul. 2019, vol. 01, pp. 249–257.

[14] A. Immonen, P. Pääkkönen, and E. Ovaska, 'Evaluating the Quality of Social Media Data in Big Data Architecture', *IEEE Access*, vol. 3, pp. 2028–2043, 2015.

[15] D. Firmani, M. Mecella, M. Scannapieco, and C. Batini, 'On the Meaningfulness of "Big Data Quality" (Invited Paper)', *Data Sci. Eng.*, vol. 1, no. 1, pp. 6–20, Mar. 2016.
[16] M. T. Baldassarre, I. Caballero, D. Caivano, B. Rivas Garcia, and M. Piattini, 'From Big Data to Smart Data: A Data Quality Perspective', in *1st ACM SIGSOFT Int. Workshop on Ensemble-Based Software Engineering*, New York, NY, USA, 2018, pp. 19–24.
[17] D. Becker, T. D. King, and B. McMullen, 'Big data, big data quality problem', in *IEEE Int. Conf. on Big Data*, Santa Clara, CA, USA, Oct. 2015, pp. 2644–2653.
[18] 'Intelligent City Platform. Smart Cambridge', *Intelligent City Platform. Smart Cambridge*. https://www.connectingcambridgeshire.co.uk/smart-places/smart-cambridge/data-intelligent-city-platform-icp/ (accessed Oct. 01, 2020).
[19] 'Big Data', *Gartner*, 2020. https://www.gartner.com/en/information-technology/glossary/big-data (accessed Feb. 06, 2020).
[20] S. Soares, *Big Data Governance: An emerging imperative*, 342 vols. MC Press, 2012.
[21] J. Debattista, C. Lange, S. Scerri, and S. Auer, 'Linked "Big" Data: Towards a Manifold Increase in Big Data Value and Veracity', in *IEEE/ACM 2nd Int. Symp. on Big Data Computing*, Dec. 2015, pp. 92–98.
[22] W. L. Chang, A. Roy, and M. Underwood, 'NIST Big Data Interoperability Framework: volume 6, Reference Architecture version 3', National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 1500-6r2, Oct. 2019.
[23] R. Clarke, 'Big data, big risks', *ISJ*, vol. 26, no. 1, pp. 77–90, Jan. 2016.
[24] L. Ramaswamy, V. Lawson, and S. V. Gogineni, 'Towards a Quality-centric Big Data Architecture for Federated Sensor Services', in *IEEE Int. Cong. on Big Data*, Jun. 2013, pp. 86–93.
[25] M. Mahdavi, F. Neutatz, L. Visengeriyeva, and Z. Abedjan, 'Towards Automated Data Cleaning Workflows', in *Conf. on 'Lernen, Wissen, Daten, Analysen'*, Berlin, Germany, 2019, pp. 10–19.
[26] W. Shi *et al.*, 'An Integrated Data Preprocessing Framework Based on Apache Spark for Fault Diagnosis of Power Grid Equipment', *J. Signal Processing Systems*, vol. 86, no. 2–3, pp. 221–236, Mar. 2017.
[27] M. Farooqi, H. A. Khattak, and M. Imran, 'Data Quality Techniques in the Internet of Things: Random Forest Regression', in *14th Int. Conf. on Emerging Technologies*, Nov. 2018, pp. 1–4.
[28] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, 'Time-series clustering – A decade review', *Information Systems*, vol. 53, pp. 16–38, Oct. 2015.
[29] R. Andrews, C. G. J. van Dun, M. T. Wynn, W. Kratsch, M. K. E. Röglinger, and A. H. M. ter Hofstede, 'Quality-informed semi-automated event log generation for process mining', *DSS*, p. 113265, Feb. 2020.
[30] L. Ehrlinger and W. Wöß, 'Automated Schema Quality Measurement in Large-Scale Information Systems', in *Data Quality and Trust in Big Data*, 2019, pp. 16–31.
[31] S. Geisler, C. Quix, S. Weber, and M. Jarke, 'Ontology-Based Data Quality Management for Data Streams', *JDIQ*, vol. 7, no. 4, p. 18:1–18:34, Oct. 2016.
[32] M. Zhang, T. Wo, T. Xie, X. Lin, and Y. Liu, 'CarStream: an industrial system of big data processing for internet-of-vehicles', *Proc. VLDB Endowment*, vol. 10, pp. 1766–1777, Aug. 2017.
[33] J. P. McCrae, C. Wiljes, and P. Cimiano, 'Towards assured data quality and validation by data certification', in *1st Workshop on Linked Data Quality in 10th International Conference on Semantic Systems*, Leipzig, Germany, Sep. 2014, vol. 1215.
[34] 'SmartCambridge'. https://smartcambridge.org/ (accessed Jun. 03, 2020).
[35] *SmartCambridge/tfc_server*. Smart Cambridge, 2020.
[36] 'Eclipse Vert.x'. https://vertx.io/ (accessed Jun. 03, 2020).