Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 217 (2023) 806–815

Procedia Computer Science

www.elsevier.com/locate/procedia

4th International Conference on Industry 4.0 and Smart Manufacturing

Fleet management systems in Logistics 4.0 era: a real time distributed and scalable architectural proposal

Ricardo Dintén[a,*], Sebastián García[a], Marta Zorrilla[a]

[a] *ISTR Research Group, Universidad de Cantabria, Av. Los Castros s/n, Santander 39005, Spain*

Abstract

In an era marked by the big data paradigm and ubiquitous computing systems, it is increasingly common to see devices embedded in any type of object with the aim of collecting and sharing owned or read data from its environment. This type of interconnection between devices is known as the Internet of Things (IoT). In particular in the logistics sector, vehicles are equipped with control units that are capable of monitoring a large number of parameters to ensure the correct operation of the vehicle. In addition, they are now able to share this data in near real time so that this information can be accessed and analysed at any time. However, due to the large amount of shared data, the frequency of data generation and delivery, and the high potential for growth in the number of devices, traditional technologies and architectures are not able to meet the performance demands of these real time decision making systems. In this article we describe and evaluate the benefits and potential trade-offs of implementing services based on a distributed and scalable architecture, called RAI4.0, in a truck fleet management company, which currently has 20,000 on-board devices and expects to grow to 80,000 devices in the next 2 years. With the change of architecture, the company expects to be able to implement near real-time services to monitor and notify its drivers of driving tips and diagnose possible vehicle failures in advance, among others.

Keywords: Big data; Cloud/Fog/Edge Computing; Real time analytics; Stream processing architecture; Logistics

1. Introduction

The Internet of Things (IoT) is becoming prevalent within many different industries, and an area of growth is being seen, especially in transportation. One of the key areas where this investment has been used is fleet management. IoT-enabled fleet management systems allow companies to receive real-time information of their vehicles, allowing them to make more informed decisions. For instance, time and costs may be saved by choosing the best possible route, or by warning drivers in near real time to avoid congestions or giving them advice on their driving in order to reduce

* Corresponding author. Tel.: +34 942 20 67 64.
E-mail address: ricardo.dinten@unican.es

10.1016/j.procs.2022.12.277

fuel consumption or to limit engine wear-and-tear. Furthermore, many of these actions would help to reduce CO2 emissions, which consequently will make transportation sector to be more sustainable [1].

Modern vehicles are progressively equipped with an outsized quantity of sensors, actuators, and communication devices (mobile devices, GPS devices and embedded computers). The information they generate is a source of great value that, when properly processed, allows organizations to get real-time visibility into business KPIs; to send actionable alerts to drivers about road status or inefficient driving; to know the status and location of the vehicle fleet: which vehicles are stopped, running, parked, the consumption per vehicle or even the pollutant emissions, among others.

The amount and variety of data being generated is so high and at such a fast pace that big data technology-based platforms are required to ingest, pre-process and deliver actionable knowledge in real time. Data centered, distributed and scalable architectures are currently the software solution that better resolves these needs/challenges of the transport sector.

The main contribution of this paper is to present a reference architecture which meets these premises and allows logistics companies to evolve their digital platforms in a gradual and demand-driven manner, while building capacity to offer new services aimed at real-time decision making (data analytics, predictive tools...). This cutting edge architecture enables processing and analysing IoT data from heterogeneous sources with different data structures and combines the real-time stream processing paradigm for information processing and transforming together with the complex event processing for information analysis.

For the sake of a better understanding, we show an instantiation of the architecture for a real use case. A company that provides a system for fleet management currently based on centralized architecture, process-oriented, that supports the ingestion of data from 26000 board computers (CLVs) with an average load of 92 MBytes/CLV which is stored in 900 relational databases, one per logistics company. Its prevision is to triple the number of CLVs and consequently the volume of data in two years and its current solution is no longer maintainable. Likewise, we discuss the advantages and difficulties found for its adoption in a medium-sized company.

The rest of the paper is organised as follows. Section 2 describes the related work. Section 3 describes our case study, the solution proposed and its implementation. Section 4 discusses the advantages and difficulties found for its adoption according to IT responsible of the company. Finally, section 5 draws the conclusions of this paper and mentions the future work lines.

2. Literature review

This section is organized in two folds. First, we describe proposals extracted from the literature in the arena of Smart Transportation Systems. Next, we relate works that deal with IoT data processing in near real time through the use of big data technologies using architectural frameworks as a software solution to implement the digital platform.

2.1. Smart Transportation Systems

Intelligent Transportation Systems (ITS) aim to provide innovative services relating to different modes of transport and traffic management and enable users to be better informed and make safer and ‘smarter’ use of transport networks. The emerging technologies of the Internet of Things (IoT) and cloud computing provide unprecedented opportunities for the development and realization of innovative ITS where sensors and mobile devices can gather information and cloud computing, allowing knowledge discovery, information sharing, and supported decision making [2]. Among others, ITS can provide based on this data, business intelligence solutions and dashboards addressed to improve the performance and efficiency of companies in the transportation and logistics space.

For instance, we find in the literature works about green drivers such as [3], where authors analyzed the impact of driving context on driving behavior and consequently on energy consumption or [4] where researchers built personal driving style of various vehicle drivers based on several driving parameters. Others focused on the application of AI for designing optimized routes to collect waste in smart cities [5]. The predictive maintenance is another issue under research. At this respect, [6] describes a predictive maintenance system for a fleet of public transport buses, which attempts to diagnose faulty buses. On the other hand, this survey [7] offers various dimensions of applying deep learning methodologies to various types of ITS problems, ranging from supervised learning for traffic flow prediction, to deep reinforcement learning for traffic signal control and unsupervised learning for trajectory clustering.

2.2. Software data-centered architecture for BigData/IoT

The number of Internet of Things (IoT) and smart devices capable of producing, consuming and exchanging information is constantly increasing. The centralized architectures presents limitations to meet performance, scalability and latencies for decision-making applications in real time due to the massive generation of data at high speed. This along with the constrained computation and storage resources that IoT devices present have led to the use of the powerful cloud computing to process their data.

The digital platform that supports IoT applications must be versatile, easily extensible and scalable to respond to a complex, heterogeneous, geographically distributed environment in which there will be applications with different levels of criticality, leading to the need to combine the use of edge, fog and cloud computing nodes. [8]

Kappa-type architectures [9] are best suited for the design and implementation of near real-time data-intensive applications using distributed technologies from the big data domain [10] [11], such as those provided by Apache ecosystem. Kappa architectures are comprised of two layers: a stream processing layer and a serving layer. The former runs the stream processing jobs, generally to enable real-time data processing. The latter is used to query the results. An architecture for processing and analysing IoT data under this scope is described in [12]. A similar architecture follows the work described in [13] addressed to predictive maintenance tasks or the model proposed in [14] for the deployment of data flow and distributed deep learning based IoT-Edge applications.

The design, development and deployment of these applications are complex mainly due to the great number of different technologies that must be orchestrated, configured and deployed on a mixed cloud/fog/edge environment in order to fulfil their functionality. Therefore, it is highly recommended to work with conceptual tools that facilitate its conception, setting-up as well as its deployment. At this regard, we proposed RAI4.0[15] a metamodel that collects the description of all elements involved in an RAI4.0-compliant digital platform (data, resources, workloads and metrics) as well as the necessary information to configure, deploy and execute workloads (applications) on it. We believe that working with models not only facilitates the definition and comprehension of the architecture but also facilitates its extension and reconfiguration to host new IoT applications.

3. Real use case

A medium size Spanish company that offers monitoring and data analysis services for truck fleet management is facing serious problems in scaling up its application as the number of clients increases. Its application gathers data from the CLV installed in the trucks and stores it to show statistics about the last trips through a web application. Currently it provides a tracking system that allows companies to monitor location of vehicles at all times. Every minute, the CLV sends the truck's positioning and diagnostic data (consumption, kilometres travelled, number of accelerations, etc.) to a central system via Internet. However, the data preprocessing and analytics is carried out in backend and its response is not in real time. Thus, it cannot offer services of re-routing or vehicle diagnosis for instant decision-making by the fleet owner. During the last months the number of clients has raised considerably, and the application is experiencing unexpected delays when trying to process and present data due to the volume and velocity at which data is generated. Moreover, the developers now find that the backend code is getting more complex and harder to maintain.

IT responsible wants to improve the software architecture to allow an easier maintenance and scaling, so they can keep up with the increase of clients and the new hardware releases which measure more parameters and with higher accuracy. In addition, they want to offer near real-time analytics to allow a faster decision making and deploy new services addresed to monitoring and predictive maintenance.

As case study, this paper describes the migration of the truck trip calculation process from a centralized architecture to a real time distributed and scalable platform using RAI4.0 reference architecture.

3.1. Objective: Migrating the current truck trip statistics generation

We start describing the process of trip statistics generation with the existing plaform (see Figure 1): first, trucks send their data from the CLV devices and CANBUS to a set of web services. Secondly, services read data with the CLV identifier and send them to an intermediate database which contains the information needed to map each CLV

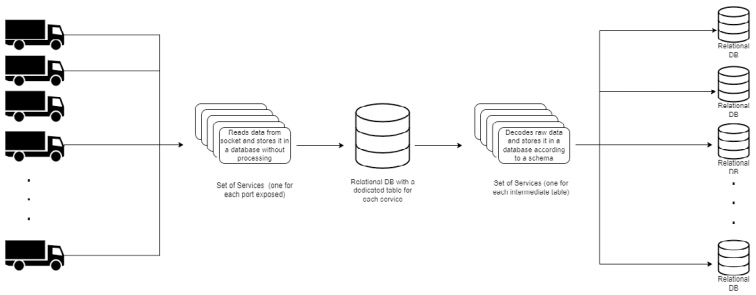

Fig. 1. Description of the current centralised architecture of this case study

to the corresponding client. Next, another set of services read and decode the data from the intermediate database. Lastly, there is a relational database for each client where the data is stored in a structured manner.

Next, the trip calculation is performed by a stored procedure that follows Algorithm 1 implemented in a SQL Server database management system. The fact of retrieving from disk and processing such a huge volume of data requires a lot of computational power that has direct impact on the performance of the application and the overall user experience. Because of that, the processing must be carried out in batch in off-peak hours. Even more, the system can only scale vertically, i.e. changing the server for another more powerful. This strategy is neither efficient nor adaptable to the demand.

3.2. Challenge: Data volume growth and performance

This centralized solution supports the ingestion of data from 26000 CLVs with an average load of 92 MB/CLV stored in 900 relational databases, one per logistics company. Its prevision is to triple the number of CLVs and consequently the volume of data in two years. This will cause that its system cannot keep up with the high demand due, primarily, to the following problems:

- A relational database is being used as if it was a queue management system, which writes data to disk limiting the rate of ingestion of the data. This fact can cause delays and data losses.
- There is a fixed number of services responsible for receiving and decoding data associated with the ports exposed by the system, so once the capacity of these services is exceeded, it would be necessary to modify the core of the application to incorporate new services.
- The database acting as a queue manager has a table for each service, therefore, would be also necessary to modify the database if you want to increase the capacity of the application.
- There is a decoding service for each intermediate table, therefore in case of scaling the application it would be necessary to create another new service.
- About 900 databases are being managed and if the number of clients increases, this number will also grow. Therefore, having the logic of the application and the calculations centralized in the database server makes the development and maintenance of the application more complex, the processing will be slower and the generation of inconsistencies between the databases of the different clients will raise.

3.3. Solution: Big data platform based on RAI4.0

To deal with these challenges we have adopted the principles of the kappa architecture proposed by Wingerath et al. [9] and implemented a digital platform based on RAI4.0 architecture described in [15]. Likewise this paper gathers two case studies that validate the proposed architecture. This reference architecture is designed to meet the following Industry 4.0 requirements: i) the architecture must be operated in a reactive and decentralized mode in order to handle data streams that are generated in the environment; ii) the distribution, heterogeneity and scalability of the computational resources required to meet the functional and non-functional requirements of the applications deployed in the environment are conceived as a set of services with the aim of keeping a unique strategy for its

Algorithm 1: Trip statistics calculation algorithm

```
for each vehicle in the fleet do
    for each vehicle event do
        Check the status (e.g. Start, Stop, Driver Change, Country Change, etc.)
        if It is the first event then
            It is marked as the beginning of a trip
        else
            It is compared with the previous one
            if It belongs to the same trip then
                Metrics are accumulated (e.g. distance, duration...)
                Move to the next event
                if It is marked as the end of the trip then
                    The end date is saved
                    Overall metrics are calculated
                    if distance <100m or speed <3Km/h then
                        It is considered a stop
                    end
                    The trip is saved in a temporary table
                end
            end
        end
    end
    for each trip of the temporary table do
        if It is the first then
            The data is saved in temporary variables
        else
            if It is consecutive to the previous trip and maintains driver, location, vehicle ... then
                The trip is considered a section and it is accumulated and grouped together with the previous one
            else
                The final trip is stored
            end
        end
    end
    for each final trip do
        Technical metrics from CANBUS are collected and processed (fuel consumption, acceleration, braking time...)
        Driving ratings are calculated
        Data about incidents that occurred during the trip are incorporated
        The idle and driving times are incorporated
        The values obtained are checked
        if there are values out of range then
            It is labeled as a trip with errors
        end
    end
    Final trip with its metrics is stored in the database
end
```

management; iii) the monitoring of the environment is an essential task since its maintenance and management rely on adaptive and dynamic strategies defined from the levels of use of the resources and the state of operation. The detailed description of RAI4.0 and the justification of its suitability for I4.0 is written in [15]. This digital platform comprises a communication service, a distribution service, a persistance service and a monitoring service (future work). The communication service implements the data bus where every component of the service can share data, it is the core of the architecture. The distribution service implements the stream processing engine where the workflows are deployed. It retrives data from the data bus and distributes the processing among the available worker nodes. The persistance service persists the data after the cleaning and processing is done. Finally, the monitoring service,

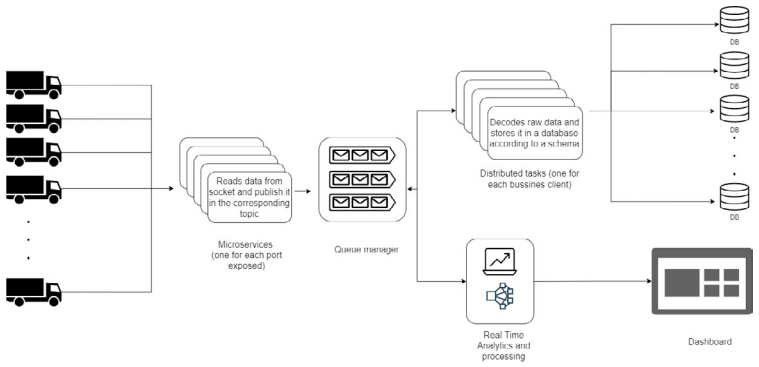

Fig. 2. Proposal of RT distributed architecture for this case study

measures and displays the resource usage to scale up/down the components of the platform when needed with the aim of avoiding excesive costs, bottlenecks or delays.

Figure 2 depicts the proposed solution for our case study. The main changes with respect to the current architecture are:

- The services responsible for reading data have been replaced by stateless workflows that merge and clean the different frames reported by the CLV and CANBUS and next publish them to the data bus.
- Data is now published in a data bus implemented by means of a message queue manager that uses memory to increase the performance.
- This data is consumed and processed by a set of distributed tasks deployed on a data stream processing engine, which allows computing each trip in near real time and storing it in a NoSQL database. These are prepared to be analised in a dashboard. Additionaly, they can be sent to the relational database for further processing.
- Another set of tasks deployed in the data stream processing engine is responsible for performing the calculation of the final trips (merge of events which comprise a trip) in a distributed way, which allows the system to speed up the calculations, to relieve the load of the database and to make the business logic independent of the database manager.

3.4. Prototype description

The solution proposed in the previous section has been developed and locally deployed using the following technologies:

- RabbitMQ has been chosen for the queue manager. This implements the MQTT protocol widely used in IoT system.There are several reasons for its adoption, it is very light, adds a very small overhead in the devices and allows declaring diferent queues to organize the data sources in diferent data streams.
- Apache Flink has been selected for the data flow processing engine. This allows exactly once processing at tuple level. This prevents duplicate rows from reaching the database and allows developers to add timestamps that serve to detect data collected out of order. This services works in a reactive way, executing the tasks every time a new record is published in the data bus. It also allows operating over time windows (e.g. tumbling windows or sliding windows) or user-defined event based windows. In the implementation of Algorithm 2 and 3, available on [16], we have employed tumbling windows to merge the data sent from different devices of the same truck, and event-based windows to determine when trips start and finish using the state reported in the frames sent by CLV devices.
- As a persistence layer, SQL Server has been maintained for transactional data management and Cassandra has been added to perform a quick data ingestion and query, in order to enable certain complex views to be loaded from the interface without overloading the relational database server.

The Algorithm 1 has been refactored into two Flink jobs. The first one, described in Algorithm 2, joins the data streams coming from the truck (CANBUS and CLV). This algorithm receives and joins all the frames from each vehicle within a determined time window and returns a new frame with all the data together. The second one, described in Algorithm 3, receives the joined frames for each vehicle and checks the status reported to determine the begining and the end of each trip using an event-based window. At the end of each trip, it retrieves all the data received within the window and computes all the statistics (e.g. distance, fuel consumption...)

Algorithm 2: Data stream joining

```
canbus ← subscribe to CANBUS RABBITMQ datastream
events ← subscribe to EVENTS RABBITMQ datastream
joinedStream ←
  events.join(canbus).where(idVehicle).equalsTo(IdVehicle).window(TumblingWindow(30s).joinFunction(joinFn)
joinedStream → RABBITMQ
def joinFn(event, canbus):
    joined ← new JoinedEvent(event) ;  /* Initialized event parameter and sets canbus related values
     to null */
    if canbus ! = null then
        for each parameter p in canbus do
            joined.p ← p
        end
    end
    return joined
```

Algorithm 3: Window based trip calculation

```
events ← subscribe to joined RABBITMQ datastream
events.keyBy(idVehicle).window(GlobalWindow).trigger(customTriggerFn).function(computeTrip)
def customTriggerFn(event):
    if event.status! = "endTrip" then
        return CONTINUE
    end
    return FIRE_AND_PURGE
def computeTrip(events[]):
    trip ← newTrip
    trip.startDate ← events[0].date
    trip.endDate ← events[events.length − 1].date
    /* Statistic calculation details omitted for the sake of simplicity                    */
    trip.computeStatistics(events)
    return trip
```

The model depicted in Figure 3 describes graphically the architecture designed following the RAI4.0 architecture metamodel [15]. This model gathers the information about how each of the components is configured and related to the others. In Figure 3 we can see that there is an instance for each service hosted in a *NodeCluster* that points to the nodes where it will be deployed. In this case, as it is a local deployment, every *NodeCluster* points to the same node. Each service has an instance that contains its deployment configuration (Rabbit_conf, Cassandra_conf, Flink_conf), the data streams (Events, Cantrama and JoinedStream) point to the *ComunicationService* (*RabbitService*) that acts as holder, the tasks (decode_and_join and trip_calculation) are linked to the root workflow (Trip_workflow) to which they belong and the workflow is connected to the *SchedulingService* (*FlinkService*) that host it.

Moreover, this model allows the developers to switch between different deployments and to scale vertically and horizontally with some minor changes. An example of a distributed and replicated version of this model that represent the deployment of the system on a set of GCE nodes with the specifications detailed on Table 1 is available on github [16]. To build the distributed and replicated version, the only action needed was to add the new *ProcessingNodes* deployed on GCE and update the links between the *NodeCluster* instances and the new *ProcessingNodes*.

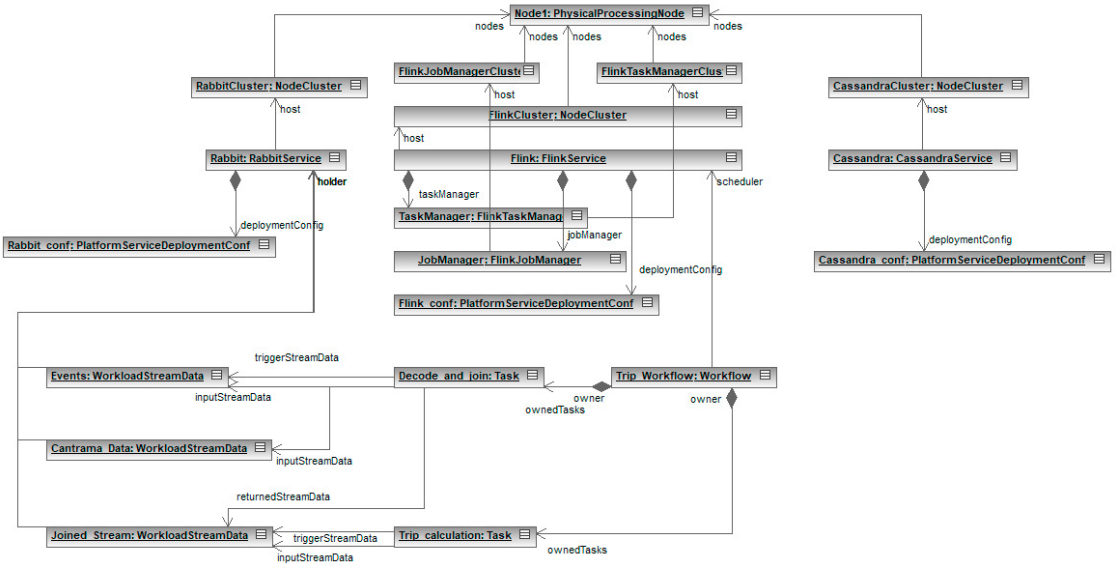

Fig. 3. Deployment model conforming RAI4.0 metamodel

Type	CPU Family	CPU Specs	RAM	Storage	OS
E2-medium	Intel(R) Xeon(R)	2vCPU @ 2.3Ghz Cache:46.1 MB	4GB	10GB HDD	Ubuntu 18.04.4 LTS

Table 1. GCE nodes specifications

4. Discussion

In this section, we discuss some of the advantages and disadvantages of the proposed platform focusing on different aspects that are limiting factors in the current implementation of the application.

1 **Extensibility and latency reduction for decision making**
The inclusion of a data stream processing engine in the architecture allows the company to know statistics in near real time with different levels of granularity. Some of them could be sent to the drivers during their trip as warning messages or driving tips. With the current architecture, it is necessary to wait until the next day, after finishing the trip, for the computation to be accesible.
In addition, the fact of being an extensible and scalable architecture allows the company to easily adapt the deployment to the demand, something that in a centralized architecture like the one described would not be possible.

2 **Coding and maintenance complexity. Learning Curve.**
Real time architectures requiere the use of a different programming paradigm which implies a period of training and adaptation. This process of adapting and migrating the system can be costly as there are a large number of technologies and components that have to be configured correctly in order to communicate one with each other. In addition, this system requires a high degree of control over the versions used for each of the components and their dependencies, as the compatibility of the elements with previous versions is not guaranteed. This complexity can be mitigated by modeling every component of the applications following the RAI4.0 metamodel, so there is a single source of truth where every item is represented and described once. Furthermore, RAI4.0 provides a deployment tool, see [15], that automates this tedious task.
On the other hand, the fact of dividing the system into microservices and separating the business logic from the database simplifies the tasks of maintenance and production of new versions.

3 **High availability vs single point of failure**
The use of a distributed and replicable architecture provides redundancy of all the data and processes so that, if there was a failure, the system would recover in matter of seconds without human interaction and the information would be preserved. The centralized solution requires to have an alternative fallover system which is most of the time underutilized.

4 **Evaluation of the improvement for the calculation of the statistics**
This project is still in progress and, at this moment, we have developed a local version and a 7-nodes distributed one comprised of: 1 node for RabbitMQ, 4 nodes for Flink (1 jobmanager and 3 taskmanagers), 1 node for Cassandra and 1 node for Prometheus and Grafana deployed in GCP nodes (see Table 1) in order to measure and compare their perfomance. The dataset used contain real data from 3 months for only 30 trucks, near 3 million events. This dataset was augmented to represent a total of 2000 vehicles. A 3 Flink taskmanagers + 1 RabbitMQ nodes set-up vs 1 Flink taskmanager + 1 RabbitMQ node configuration has been tested using this augmented dataset. As a result, the system distributes the workload in a balanced way, one third on each cpu, which means that the system capacity can be increased by simply adding more nodes. No node was saturated. The CPU load is inferior to 15% on each node. The throughput is 4000 events/second. We grew the load to 10.000 vehicles but the nodes hired in GCE (free tier) did not work well, they started to lose events and the delivery rate dropped drastically. However, the system still had plenty of resources available so we are currently analysing whether this is due to the usage of GCE free tier. That is why the benchmark carried out is not conclusive but promising.

5 **Costs**
In terms of costs, on-premise solutions implies both Capital expenditure (Capex) and Operational expenditure (Opex) costs while cloud computing only originates Opex. Cloud computing allows the project to start working with at a smaller cost. However, Opex costs are usually much higher for cloud computing so after a few years the on-premise solution can become cheaper. In [17], the authors carried out a comparative analysis between the cost of an on-premise system and a similar system but hosted on the cloud and determined that, depending on the costs and rate of maintenance services of the on-premise solution, the cumulative costs becomes higher for the cloud-based solution after three to five years. This means that if you are planning to maintain the same infrastructure, cloud based solutions may not be the best alternative. In our case study the business owner expects a fast growth of the number of clients, so keeping the same infrastructure over a long period of time may not be possible, making the cloud-based alternative a better suited option.

5 **Security**
Ensuring the security of cloud-based applications is harder and more complex than it is in a centralized on-premise version due mainly to the number of different components to be taken into account, the comunication among them and the location of data that is being stored and processed somewhere on the cloud. RAI4.0 architecture security service is still under implementation.

5. Conclusions and future work

This paper proposes a reference architecture to migrate IoT applications designed under a centralised architecture to a scalable, distributed, near-real-time one to provide actionable services, i.e. sending real-time alerts or notifications in order to reduce costs or improving the performance of certain processes.

A real use case is described to show the opportunity, suitability, advantages and challenges that a distributed and scalable data bus architecture provides to solve the scalability problems that arise when the pace and volume of data grows and near real-time requirements need to be met.

At the moment, we are working with different deployment configurations in the Google Cloud to carry out a cost vs scalability and performance study. The results are promising in terms of performance. No costs have yet been incurred as the cloud has been used for a free period.

Acknowledgements

This work was supported in part by MCIN/ AEI /10.13039/501100011033/ FEDER "Una manera de hacer Europa" under grant PID2021-124502OB-C42 (PRESECREL), RUT-IA project and the predoctoral program "Concepción Arenal" funded by Universidad de Cantabria and Cantabria's Government (BOC 18-10-2021).

References

[1] Inmarsat, 2018. Iot is a leading force in helping the transport and logistics sector to drive down its carbon footprint, finds inmarsat. URL: https://www.inmarsat.com/en/news/latest-news/enterprise/2018/iot-is-a-leading-force-in-helping-the-transport-and-logistics-sector-to-drive-down-its-carbon-footprint-finds-inmarsat.html.

[2] Mohammed, S., Arabnia, H.R., Qu, X., Zhang, D., Kim, T.H., Zhao, J., 2020. Ieee access special section editorial: Big data technology and applications in intelligent transportation. IEEE Access 8, 201331–201344. doi:10.1109/ACCESS.2020.3035440

[3] Faria, M.V., Baptista, P.C., Farias, T.L., 2017. Identifying driving behavior patterns and their impacts on fuel use. Transportation Research Procedia 27, 953–960. URL: https://www.sciencedirect.com/science/article/pii/S2352146517309353, doi:https: //doi.org/10.1016/j.trpro.2017.12.038. 20th EURO Working Group on Transportation Meeting, EWGT 2017, 4-6 September 2017, Budapest, Hungary.

[4] Constantinescu, Z., Marinoiu, C., Vladoiu, M., 2010. Driving style analysis using data mining techniques. INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL 5, 654–663. URL: http://www.univagora.ro/jour/index.php/ijccc/article/view/2221.

[5] Ghahramani, M., Zhou, M., Molter, A., Pilla, F., 2022. IoT-based Route Recommendation for an Intelligent Waste Management System. ArXiv:2201.00180v1.

[6] Killeen, P., Ding, B., Kiringa, I., Yeap, T., 2019. Iot-based predictive maintenance for fleet management. Procedia Computer Science 151, 607–613.URL: https://www.sciencedirect.com/science/article/pii/S1877050919306519, doi:https://doi.org/10.1016/j. procs.2019.04.184. the 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd Interna-tional Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops.

[7] Veres, M., Moussa, M., 2020. Deep learning for intelligent transportation systems: A survey of emerging trends. IEEE Transactions on Intelligent Transportation Systems 21, 3152–3168. doi:10.1109/TITS.2019.2929020.

[8] Kumar, M., Dubey, K., Pandey, R., 2021. Evolution of emerging computing paradigm cloud to fog: Applications, limitations and research challenges, in: 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence), pp. 257–261. doi:10. 1109/Confluence51648.2021.9377050.

[9] Wingerath, W., Gessert, F., Friedrich, S., Ritter, N., 2016. Real-time stream processing for big data. it - Information Technology 58, 186–194. URL: https://doi.org/10.1515/itit-2016-0002, doi:doi:10.1515/itit-2016-0002.

[10] Philip Chen, C., Zhang, C.Y., 2014. Data-intensive applications, challenges, techniques and technologies: A survey on big data. Information Sciences 275, 314–347. URL: https://www.sciencedirect.com/science/article/pii/S0020025514000346, doi:https://doi.org/10.1016/j.ins.2014.01.015.

[11] Arooj, A., Farooq, M.S., Akram, A., Iqbal, R., Sharma, A., Dhiman, G., 2022. Big data processing and analysis in internet of vehicles: Architecture, taxonomy, and open research challenges. Archives of Computational Methods in Engineering 29, 793–829. URL: https: //doi.org/10.1007/s11831-021-09590-x, doi:10.1007/s11831-021-09590-x.

[12] Corral-Plaza, D., Medina-Bulo, I., Ortiz, G., Boubeta-Puig, J., 2020. A stream processing architecture for heterogeneous data sources in the internet of things. Computer Standards & Interfaces 70, 103426. URL: https://www.sciencedirect.com/science/article/pii/S092054891930008X, doi:https://doi.org/10.1016/j.csi.2020.103426.

[13] Dintén Herrero, R., Zorrilla, M., 2022. An I4.0 data intensive platform suitable for the deployment of machine learning models: a predictive maintenance service case study. Procedia Computer Science 200, 1014–1023. URL: https://www.sciencedirect.com/science/article/pii/S187705092200309X, doi:https://doi.org/10.1016/j.procs.2022.01.300. 3rd International Conference on Industry 4.0 and Smart Manufacturing.

[14] Veeramanikandan, Sankaranarayanan, S., Rodrigues, J.J., Sugumaran, V., Kozlov, S., 2020. Data flow and distributed deep neural network based low latency iot-edge computation model for big data environment. Engineering Applications of Artificial Intelligence 94, 103785. URL: https://www.sciencedirect.com/science/article/pii/S0952197620301780, doi:https://doi.org/10.1016/j.engappai.2020.103785.

[15] López Martínez, P., Dintén, R., Drake, J.M., Zorrilla, M., 2021. A big data-centric architecture metamodel for industry 4.0. Future Generation Computer Systems URL: https://www.sciencedirect.com/science/article/pii/S0167739X21002156, doi:https://doi.org/ 10.1016/j.future.2021.06.020.

[16] Sebastian, G.B., 2022. Trip statistics calculations with flink. https://github.com/Sgb597/Rabbit.

[17] Fisher, C., 2018. Cloud versus on-premise computing. American Journal of Industrial and Business Management 08, 1991–2006. URL: https://doi.org/10.4236/ajibm.2018.89133, doi:10.4236/ajibm.2018.89133.