IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 25, 2020, accepted December 11, 2020, date of publication December 16, 2020,
date of current version December 31, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3045175*

# Development and Evaluation of a Big Data Framework for Performance Management in Mobile Networks

**DIANA MARTINEZ-MOSQUERA[1], ROSA NAVARRETE[1], AND SERGIO LUJÁN-MORA[2]**
[1]Department of Informatics and Computer Science, Escuela Politécnica Nacional, Quito 170525, Ecuador
[2]Department of Software and Computing Systems, University of Alicante, 03690 Alicante, Spain

Corresponding author: Diana Martinez-Mosquera (diana.martinez@epn.edu.ec)

This work was supported by the Unidad de Gestión de Investigación y Proyección Social from the Escuela Politécnica Nacional.

**ABSTRACT** In telecommunications, Performance Management (PM) data are collected from network elements to a centralized system, the Network Management System (NMS), which acts as a business intelligence tool specialized in monitoring and reporting network performance. Performance Management files contain the metrics and named counters used to quantify the performance of the network. Current NMS implementations have limitations in scalability and support for volume, variety, and velocity of the collected PM data, especially for 5G and 6G mobile network technologies. To overcome these limitations, we proposed a Big Data framework based on an analysis of the following components: software architecture, ingestion, data lake, processing, reporting, and deployment. Our work analyzed the PM files' format on a real data set from four different vendors and 2G, 3G, 4G, and 5G technologies. Then, we experimentally assessed our proposed framework's feasibility through a case study involving 5G PM files. Test results of the ingestion and reporting components are presented, identifying the hardware and software required to support up to one billion counters per hour. This proposal can help telecommunications operators to have a reference Big Data framework to face the current and future challenges in the NMS, for instance, the support of data analytics in addition to the well-known services.

**INDEX TERMS** Big data, framework, mobile networks, network management system, performance management.

## I. INTRODUCTION

The mobile network industry is one of the most extensive and heterogeneous worldwide [1]. In the last two decades, it has evolved from 2G, 3G, and 4G technologies to the rollout of 5G and 6G technologies in the near future [2]. With the advent of 5G and 6G, mobile networks will be challenged to respond to the dramatic rise of connected devices.

Since the first generations were developed, mobile networks count using a centralized system that acts as a business intelligence tool specialized in monitoring and reporting the network's performance. This system is known as the Network Management System (NMS). Essentially, an NMS allows operators to manage the network and visualize its behavior by collecting Performance Management (PM) data from Element Management Systems (EMS) [3]. The purpose of PM is to continuously monitor and measure the performance of various subsystems [4].

As shown in [5], during 2020, there were around eight billion mobile connections in the world, and for 2025 it is estimated that there will be 8.8 billion mobile connections. The IDC Corporation forecasts that the global data will grow to 175 zettabytes (ZB) by 2025 [6] (1 ZB is equivalent to a trillion gigabytes). Therefore, due to the projected increase in the number of users, more extensive networks will need to analyze larger amounts of data in shorter periods of time in order to prevent network outages as soon as possible. With PM data, it is possible to plan and optimize the network to avoid outages and provide a better user experience. With this premise, we have focused our research on PM data.

The procedure to gather these PM data is shown in Fig. 1, where NMS collects PM data from each network element (NE). Mediation components are responsible for data

The associate editor coordinating the review of this manuscript and approving it for publication was Dominik Strzalka.

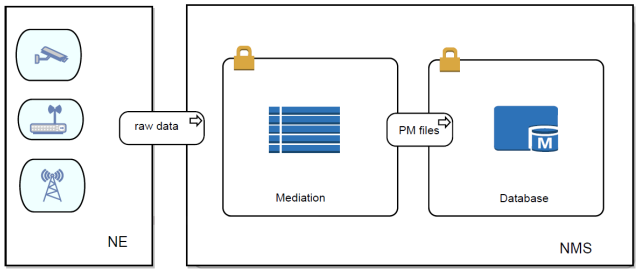


**FIGURE 1.** Collection of PM data from network elements.

ingestion and are developed per data format and transport protocol. Database components allow for time-level aggregations (hourly, daily, weekly, monthly), object-level aggregations, and busy hours calculations, based on default formulas and algorithms. In this component, it is also possible to generate SQL queries.

Performance Management data are collected at a particular frequency known as the granularity period. At the end of the granularity period, a file is generated with the PM data collected from each NE. Generally, PM data contains cumulative incremental counters triggered by the occurrence of the measured event. A counter is reset to zero at the beginning of each granularity period [4].

The minimum granularity period is five minutes and the maximum is one hour [4]. Therefore, an NMS is continuously collecting PM data from every NE. For these reasons, the scalability and performance requirements are particularly challenging and typically require hardware and software solutions with large capacities or load balancing between multiple NMSs [7], [8].

An NMS also allows the management of other types of data for reporting enrichment purposes, for instance, Fault Management (FM) data and Configuration Management (CM) data. Fault Management data helps to locate faults in the network devices and CM data allows for devices to be configured from a centralized system. Both activities are also considered important; however, in this article we focused our research on PM data.

Nowadays, an NMS provides, via traditional database paradigms, a solution for collecting, transforming, aggregating and storing PM files. Moreover, the NMS data warehouse is currently based on custom relational database solutions, and the architecture is based on shared disks that operate with scalability limitations when it comes to large-scale analytics. Therefore, vendors have already started to replace their NMS backend with a data lake infrastructure, but traditional solutions continue to be used [9]–[14].

A data lake has the ability to store raw data from different types of devices, such as the Internet of Things (IoT) devices, sensors, NEs, etc., in a fast way and with low storage costs. Several techniques, for instance, machine learning and predictive analysis, among others, can be applied to this data. Moreover, a data lake is highly scalable, very accessible, and quick to update; it does not require planning or prior knowledge about the data, since it assumes that analysis will happen later, on-demand [15].

The 5G and 6G technologies must support applications with low delay requirements [16], for instance, telemedicine, augmented and virtual reality, smart cities, smart grids, vehicular automation, or wearables. Hence, service availability for the user is currently crucial. In order to give an example of the requirements that must be achieved, in [17], the authors present use cases in 6G, such as telepresence, eHealth, industry 4.0, and unmanned mobility, that require latencies below one ms. To verify the compliance of these latencies, the PM data counters are required to calculate the main Key Performance Indicators (KPI) of 6G use cases. According to the 3GPP Technical Specification 32.401 V5.5.0 [4], counters from PM files are used to calculate KPI for traffic measurements, such as pages per location area per hour, busy hour call attempts per NE, or handovers; usage and availability of resources; Quality of Service (QoS) to measure the network performance expected to be experienced by the user; and Quality of Experience (QoE), among others.

Consequently, the volume, variety, and velocity of collected PM data from the NEs will inevitably increase within the next few years. Thus, the NMS capability to store, process and present the data in an efficient form must be assured. To overcome these new challenges, it is necessary for the current NMS to evolve by taking advantage of newer trends in Big Data storage, processing, and warehousing [18]. The data variation, allied with advanced analytical functions, also presents an opportunity to explore new insights and improve end-user experience and customer satisfaction.

In this context, we have considered the following main research questions: What is the format of PM files most used by the vendors? What Big Data architecture is most appropriate for the format of PM files most used by the vendors? How many counters can be ingested in the proposed architecture and what are the computation resources required? How many counters can be returned from querying a raw table in the proposed architecture and what are the computation resources required?

The intention of our proposal was to develop an NMS solution and evaluate the performance and scalability of the approach when running on a Big Data stack. This solution supports a Hadoop ecosystem based on an analysis of the following components: software architecture, ingestion, data lake, processing, reporting, and deployment. In order to answer the research questions raised, we used real data sets from four different vendors and 2G, 3G, 4G, and 5G technologies. Finally, we developed and assessed our proposed framework's feasibility through a case study involving 5G PM files.

The contribution of this article is the design of a Big Data framework for processing PM data of mobile networks; the implementation of the tools and procedures for analyzing the PM data in a test environment; and the evaluation of the results with real data sets for ingestion and batch reporting processes.

The challenge of this proposal was to match or reduce the delays in the ingestion and reporting processes with fewer computation resources. Furthermore, our proposal is intended to be a multi-vendor solution since Big Data stacks enable flexible deployment scenarios. All these features are aligned with industry trends.

The further structure of the paper is as follows. Section II refers to related works. Section III presents a format and size analysis of real PM files from four different vendors and 2G, 3G, 4G, and 5G technologies. Section IV offers an analysis of the PM files structure and the data modeling at the conceptual and logical levels. Section V discusses the components of the proposed Big Data framework: software architecture, ingestion, data lake, processing, reporting, and deployment. Section VI discusses the aspects considered in the test environments of the architecture, and the results of the ingestion and reporting processes for two test systems, one with 24 GB of RAM and the other with 30 GB. Finally, Section VII contains the conclusion and directions for future research.

## II. RELATED WORKS

In this section, we summarize some related works that have been published at academic and industry levels in the NMS context. The academic works include studies published in journals, conferences, and books, proposed reference models and some of the approaches evaluated. For the industry level works, we have considered information presented in white papers and product documentation, in order to understand the new trends.

Among the academic studies, Baik *et al.* [19] propose a multi-tier architecture for NMS in wired and wireless networks. They present a successful implementation of a three-tier architecture that consumes much fewer CPU resources than a two-tier architecture. This article is focused on the Internet Control Message Protocol. This is in contrast to our work, as we focused on mobile networks.

Samba [20] presents a model for an architecture for telecommunications NMS. The approach outlines a real-time and multi-vendor and describes three autonomous systems: a Network Management Service Creation System, Integrated NMS, and Cluster/Autonomous Sensor Entities. In contrast to our study, there is no assessment of the architecture or further explanation of the PM data processing.

Yi *et al.* [21] propose a reference model for NMS focused on 2G. The proposal is widely explained for 2G technologies. In contrast to our study, there is no assessment of their proposal and newer mobile networks generations technologies, such as 3G, 4G, and 5G are considered.

Pratama *et al.* [22] present a dashboard architecture for telecommunication billing systems of companies. Their proposal uses Apache Spark and Apache Zeppelin, and is focused on short message service as a case study. The goal of the project is to reduce the raw data processing time and they successfully achieve their statement. This work is very interesting in the use of Big Data tools. However, data from billing systems comprise a smaller set than the existing PM data, as their purposes are significantly different. One monitors customer equipment interactions with the network while the other tracks how all network components operate, so they can be configured to work better and more efficiently amongst them.

Maździarz [23] analyzes real data sets with FM data from a mobile telecommunication network. He uses an alarm correlation algorithm that employs the k-means method. As a result, the assessments successfully provide processing of 1200 and 2000 alarms in around 10 to 15 seconds. This study is interesting in relation to the network monitor process. However, it focuses solely on the realms of FM, whereas, by the use of the PM data, NE may also generate threshold alarms based on system-internal supervision of counters and their threshold values.

Rayens and Sage [24] describe in great detail a management framework for IP-based networks. They discuss an NMS example for PM, FM and others' plan, provision, operate, and bill different IP services. However, the authors do not consider an experimental validation of the proposed framework.

Skračić and Bodrušić [25] present a solution to use a Big Data framework for PM data. Their study provides a very useful benchmark of various scenarios for collecting and decoding binary log files from real Ericsson mobile networks in a Hadoop cluster. Their results are up to six times faster than existing solutions. A particularity of their proposed approach is that it evaluates binary files while our work evaluates Extensible Markup Language (XML) files. As we present in Section III.B, binary files are used in 2G technology for one vendor; on the contrary, XML is the format used for PM files in 2G, 3G, 4G, and 5G technologies for most vendors.

Suleykin and Panfilov [26] propose a Distributed Big Data Driven Framework (DBDF) for Cellular Network Monitoring Data. They implemented Apache Spark and Yarn and performed tests with data from Core NEs for roaming user detection. According to the authors, their results achieve around a one second delay on average compared to a batch solution that requires more than one minute and 15 seconds. In contrast to our study, their work is only focused on the final reports of the performance of the roaming users. Moreover, there is no evaluation of ingestion and reporting layers to understand the requirements of an NMS implementation based on Big Data.

Lin *et al.* [27] describe the importance of building an unified information model for an Operations Support System (OSS). They highlight that NMS is currently entering the era of Big Data management, and telecommunication operators are now facing great challenges. They underline three main ideas: (1) focusing on the existing information models, (2) identifying the key steps in the modeling process based on mainstream modeling methods, and (3) designing patterns to improve the quality of the information models. All their statements are important; however, this work does not present any proposal or implementation of NMS based on Big Data architecture.

Zhou *et al.* [28] describe a Big Data framework for processing network operation data in an Intelligent NMS (INMS) for China Telecom. They propose a generic Hadoop-based architecture for their INMS. For the collection phase, they propose merging the raw data to 64 MB. For the data storing phase, they propose the use of the Hadoop Distributed File System (HDFS) due to the fault-tolerant and low-cost commodity hardware features. And, for data reporting, HBase is the proposed Hadoop database due to the query speed and linear expansion features. The solution could be considered relevant; however, the paper does not present the implementation of the proposal or its advantages with experimental results. Our approach does include those aspects.

Among the industry studies, companies such as Ciena, Blueplanet, TMForum, Nokia, Huawei, Oracle, Ossline, amongst others, have published several white papers with trends about their Operation Support System (OSS). An OSS encompasses many highly technical network management processes, that is, it covers the NMS area. Next, we mention some of the industry studies.

Oracle [9] explains that even after two decades with independent PM solutions from vendors, operational efficiency improvements over network performance are demanded. With the 5G advent that traditional OSS is not equipped to handle. The results of the article present the desirable attributes for an OSS in the future, among them, scalability, open ecosystem driven, support for artificial intelligence, cloud-native, and the ability to run on a virtual architecture. This study shows that 95 % of customer operations are yet to implement these features in production.

In the white paper [10], the 5G PPP Architecture Working Group and Huawei company present a full explanation of every component in a 5G network. They show in the 5G architecture must have Big Data/ Management and Orchestration Data Analytics functionality in the operation and maintenance (O&M) and network layers. Raw data are collected from the network layer and contain the counters to monitor the performance and calculate KPIs, such as radio failures. In the O&M layer, data analytics must also be supported. It is vital to highlight, in the 5G context, the network layer is capable of supporting Network Function Virtualization (NFV) infrastructure; therefore, raw data can be collected from virtual machines instead of bare metal devices.

Vendors, such as Huawei, Nokia, and Ericsson and their NMS solutions U2000, NetAct, and Telcordia, respectively, do not freely their NMS solutions documentation, which is generally considered proprietary and confidential information. However, in the document [11] a vendor solution still uses relational databases and traditional processing paradigms for PM data.

Nokia [12] mentions network optimization jobs that are performed with the help of PM files as critical. Thus, they count on a Big Data team that researches Big Data solutions for PM processing. The company currently offers an NMS solution for operators with relational databases and traditional paradigms. Still, this study shows they are now in an early stage of the Big Data era. Moreover, they consider Big Data tools as the next logical evolution that opens massive data analysis opportunities. They mention components like acquisition, ingestion, storage, database, and visualization. The tools presented in the white paper are Hadoop, MapReduce, Cassandra, and MongoDB, among others. These types of documents do not present detailed information on the design and implementation of the PM processing with Big Data tools as ours.

According to the study [13], nowadays, the question of computational scalability is crucial because with the size of the 5G and IoT networks, the telecommunication networks may generate excessive control traffic overhead and the performance management entities will have to process a great amount of information. These challenges demand enrollment into the Big Data world. This study also mentions the implementation of the NFV, where an NFV manager realizes the adaptation of the configuration and event reporting between NFV interfaces and an NMS. In our study, we did not use raw data from an NFV infrastructure as a source as real implementations are still being deployed. However, our proposed NMS architecture complies with the support required for data analytics.

Ossline [14] mentioned OSS covers many highly technical network management processes. Furthermore, it presents that platforms like Hadoop begin to be implemented within the OSS. Moreover, telecommunication operators explain the need for Big Data analytics to comply with the challenges. These premises allowed us to address our research so that it is aligned with the industry trends, such as scalability, support of Big Data analytics, etc.

In the literature review, we could not find any similar research that presents a framework evaluation based on a Big Data stack for PM data. Although there are works that present reference models for an NMS [20], [21], [24], [28], none of them implement their proposals and assess the ingestion and reporting layers experimentally at the level of detail that is carried out in this research. For this reason, this article presents a novel proposal that overcomes the limitations of the previous academic studies that were analyzed and it is aligned with industry trends.

## III. FORMAT AND SIZE ANALYSIS OF PERFORMANCE MANAGEMENT FILES

In this section, we briefly analyze the format and the size of real data sets of PM files used by four vendors to transfer the PM data from NE to an NMS. Our goal is to examine the formatting trend of PM files during the different generations. This analysis also allows us to identify whether PM data comply with the characterization of Big Data. For instance, volume, because of the amount of PM data produced by NE, and variety, since the PM data formats vary for each vendor. Moreover, the PM file format is used to implement the database in the Big Data stack.

In order to undertake our analysis, we used the PM files produced by four different vendors A, B, C, and D, from real

**TABLE 1. Average Size of PM Files sent by a NE every 15 minutes.**

| Vendor | 2G [MB] | 3G [MB] | 4G [MB] | 5G [MB] |
|---|---|---|---|---|
| A | 11 | 0.2 | 0.1 | |
| B | 47 | 0.05 | 1 | |
| C | 6 | 25 | 2 | 0.2 |
| D | 1 | 0.05 | 0.02 | |

2G, 3G, 4G, and 5G mobile network nodes. Names of vendors cannot be disclosed due to confidentiality and non-disclosure agreements. Table 1 presents the average size of PM files collected from A, B, C, and D vendors in megabytes by technology. The granularity period is 15 minutes. Generally, for 2G, the PM sizes are larger than the newer technologies reaching up to 47 MB. For 5G, we could only obtain a PM file from vendor C and, in it, we can see that the size is smaller than the previous technologies. This smaller size is explicable because the roll-out of 5G networks is just starting worldwide and there are only a few sites installed [29].

In order to provide an estimated idea for the storage space required for PM files only in the radio access network, we used the number of installed NEs. For instance, taking as reference a country with 15 million mobile subscriptions, there are approximately 2200 base stations for 2G, 4800 base stations for 3G, and 3200 base stations for 4G [30]. With this number of NEs, we estimated 3000 base stations, named next generation Node B (gNB), for 5G. Therefore, in total, around 13300 base stations. The required storage capacity, taking a 5G PM file of 0.02 MB as reference, is around 26 GB daily, 182 GB weekly, or 728 GB monthly. If FM and CM data are also considered, and either NEs from other network mobile subsystems, like Packet Core, or other services such as the IoT devices, the storage space requirement greatly increases.

**TABLE 2. Format of PM files.**

| Vendor | 2G | 3G | 4G | 5G |
|---|---|---|---|---|
| A | Binary | XML | XML | |
| B | XML | XML | XML | |
| C | XML | XML | XML | XML |
| D | CSV | CSV | CSV | |

Table 2 presents the format of PM files by vendor and technology. As we can see, the most used file format is XML. From this information, we can answer the first research question: What is the format of PM files most used by the vendors?

According to this result, our approach considered this format type to be a trend and the test environments implemented in this work processed XML files. We consider this proposal could be useful for the next generation of technologies, since it could be arguable that 6G PM files are very likely to follow XML format.

## IV. PERFORMANCE MANAGEMENT DATA MODELING

In this section, we present the data modeling of the PM XML files for the vendors that use this format in section III. Modeling is presented at the conceptual and logical levels. With the results of this section, we were able to define the database to be used in our proposal.

### A. CONCEPTUAL MODEL

This section defines the conceptual model for PM XML files in order to determine Big Data architecture requirements in terms of storing and reporting. Since our approach is intended to be multi-vendor, we modeled the XML PM files from the A, B, and C vendors using feature modeling. We used the feature model since it defines common and variant features of a domain [31]. A feature model is commonly used to depict the characteristics and constraints of the software product line in terms of features. Conceptual models for every vendor are presented as follows.

#### 1) MODEL FOR VENDOR A

Vendor A uses the XML format for 3G and 4G technologies. The XML file format definitions are defined in annex A of the 3GPP TS 32.401-550 technical specification [4]. According to the defined nomenclature in [4], Fig. 2 presents the conceptual PM data model for 3G and 4G.

The file follows a tree structure, with the measDataCollection tag being the root node. This node has three childs: measFileHeader, measData, and measFileFooter. While the measFileHeader supplies information about the file and the NE itself, measFileFooter gives details regarding metrics timestamp. PM counter information is provided via multiple measData nodes arranged per counter type.

#### 2) MODEL FOR VENDOR B

Vendor B uses the XML format for 2G, 3G, and 4G technologies. In accordance with the nomenclature defined by this vendor, Fig. 3 presents the conceptual PM data model.

The file follows a tree structure, with the root node being the measCollectFile tag. This, in turn, is divided into the fileHeader and measData tags. FileHeader tag contains the file metadata such as version, vendor name, among others. The MeasData tag, in turn, includes the information about the source network element, the time, and the metrics themselves.

#### 3) MODEL FOR VENDOR C

Vendor C uses the XML format for 2G, 3G, and 4G technologies. In accordance with the nomenclature defined by this vendor, Fig. 4 presents the conceptual PM data model.

The file contains the OMeS tag as the node root, followed by a single PMSetup child node containing information regarding period start time and its duration. Multiple PMMOResult child nodes provide information about the network elements and the metrics themselves.

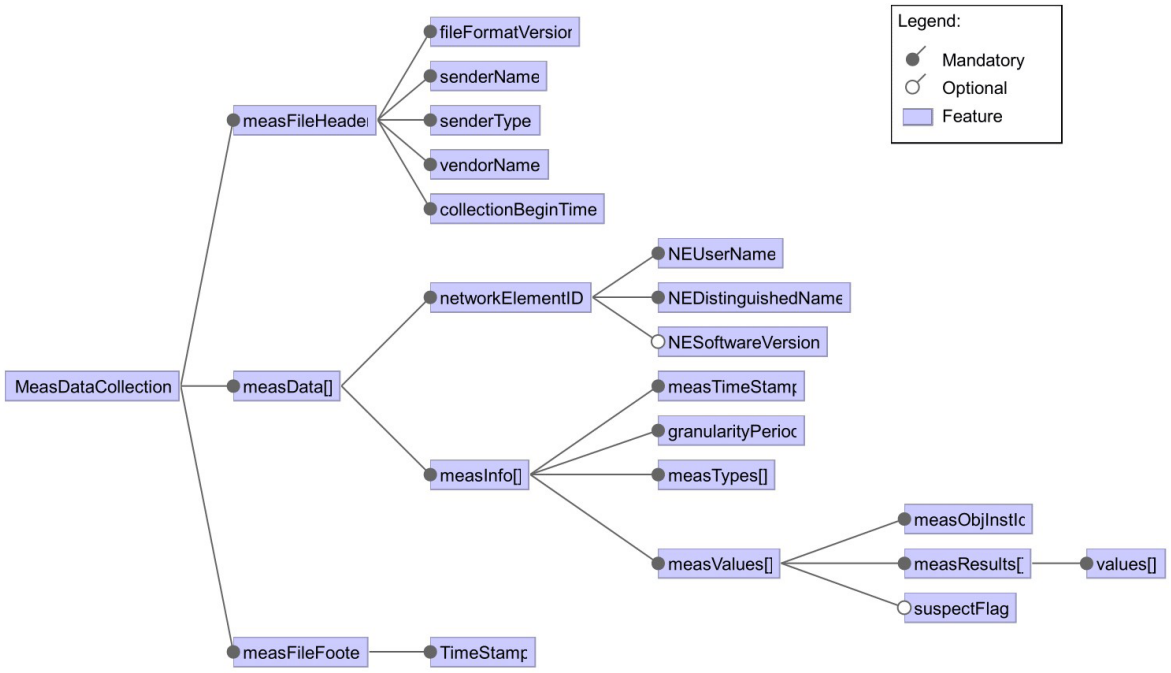


**FIGURE 2.** PM data model for Vendor A.

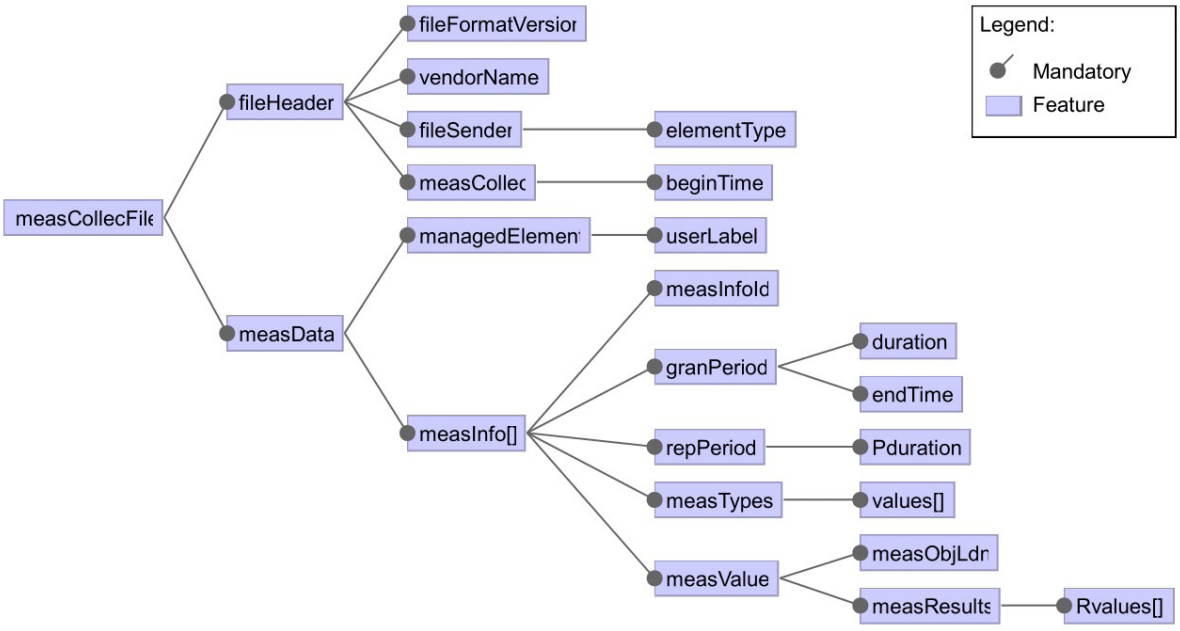


**FIGURE 3.** PM data model for Vendor B.

#### 4) METAMODEL

As a proposal to summarize the above-mentioned models, the conceptual metamodel for PM files is presented in Fig. 5. The root node MeasurementFile.xml is divided into two tags: FileConfiguration and MeasurementData. The FileConfiguration tag contains all the metadata for the PM file, for instance, timestamp, interval, data, etc. Meanwhile, the MeasurementData tag includes the metrics and related information, for example, managed object information, metric name and its value.

### *B. LOGICAL MODEL*

The logical model to be implemented mainly consists of a table, per vendor and technology, stored in the respective

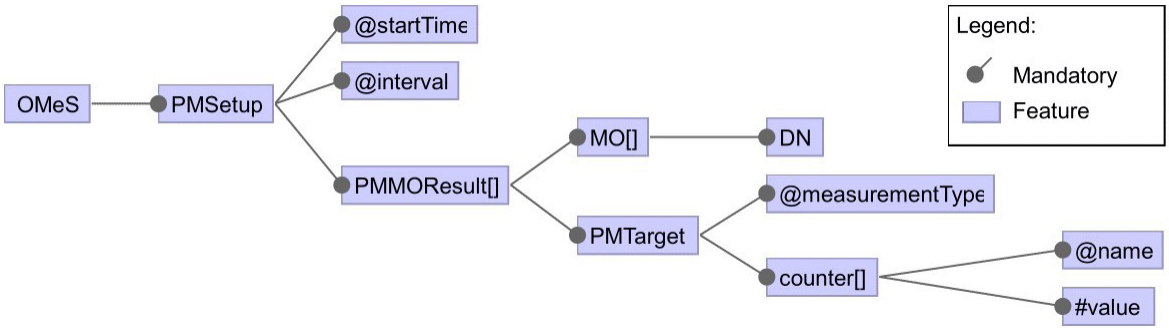


**FIGURE 4.** PM data model for Vendor C.

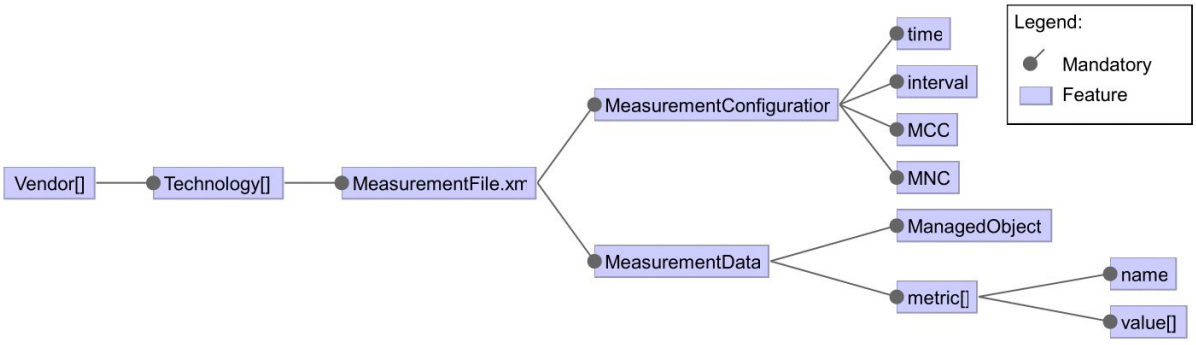


**FIGURE 5.** Conceptual PM data metamodel.

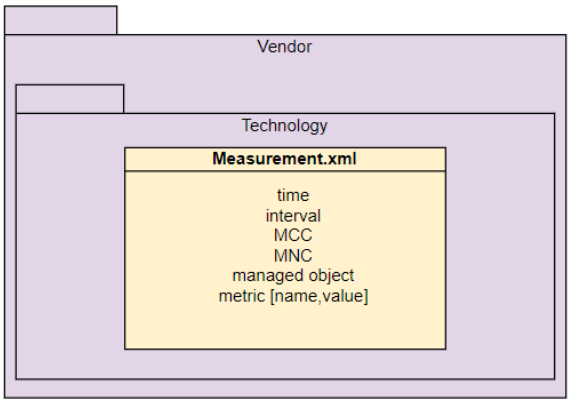


**FIGURE 6.** Logical PM data model.

file system. Common attributes of PM files are specified as configuration, for instance, time, interval, mobile country code (MCC), and mobile network code (MNC). Raw data are organized as a structure that contains the network element identifier, name, and metric value. Figure 6 presents a schema format for the PM logical model.

In conclusion, PM data from NEs will arrive in XML format to an NMS. Therefore, the Big Data framework must be able to collect the raw data and store it in XML format in a data lake where each vendor-technology set will have its own schema. For reporting, XML files must be queried according to the specific data structure of the vendor. After analyzing the type of data to be processed, the framework for PM data based on a Big Data stack is presented in the following section.

## V. BIG DATA FRAMEWORK

An NMS is an umbrella tool that handles all the performance data available in a mobile network. Current NMSs' data warehouses are traditionally based on a Relational Database Management System (DBMS). This work presents a proposal based on a Big Data paradigm, which has been designed specifically to handle large volumes of data. In this section, we answer the second research question: What is the Big Data architecture most appropriate for the format of PM files most used by the vendors?

### A. DESIGN

Table 3 presents the main features that any implementation must at least provide [32]. It follows the reference architecture [33] based on the use of an integrated stack of Big Data components such as software architecture, ingestion, data lake, processing, reporting, and deployment. In order to help the reader with the best understanding, the Table 3 briefly describes the components of the Big Data framework and the options available to implement them. In the next sections, we summarize the tool chosen for every component and the reasons according to the requirements.

**TABLE 3. Features and tools for a big data framework.**

| Components | Description | Options |
|---|---|---|
| Software Architecture | "The fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution" [34] | Lambda, Kappa |
| Ingestion | Mediator system for a massive collection of raw data from NEs to the data lake independently of format | Flume, Kafka, NiFi |
| Data Lake | Distributed file system that allows storing all kinds of data in an efficient way | Hadoop Distributed File System (HDFS) |
| Processing | Platform to provide a real-time processing of network data | Spark, Storm |
| Reporting | Framework to provide the use of SQL language to query data stored on the data lake | Hive, Impala, Pig, Spark SQL |
| Deployment | Platform to install and configure Hadoop cluster nodes and enabling agile application deployment | Cloudera, Hortonworks, IBM InfoSphere Big Insights, MapR, Pivotal HD |

#### 1) SOFTWARE ARCHITECTURE

In this section, we select the fundamental structure of a software system for a Big Data framework for PM data in mobile networks. According to the results obtained in the previous sections III and IV, the main requirements for the architecture are:

- Handling large volumes of data using batch processing to provide historical data management with reliable and accurate KPI calculations.
- Providing scalability for data processing.
- Ensuring low error possibility if the system crashes.

Here, two options are available: lambda or kappa architectures [35]. Lambda is composed of batch (BL), speed (SL), and serving layers (SRL). The operational sequencing of the lambda architecture is described in equation (1).

According to the equation (1), report $\lambda$ can combine the results from historical data from the BL and live streaming data from the SRL with the SL, where x relates to the data of each layer.

$$\lambda = \{x | x \in BL \vee x \in SRL\}. \tag{1}$$

Kappa is composed of SL and SRL. The operational sequencing of the kappa architecture is described in (2). According to (2), report $\kappa$ is gathered from live streaming data from the SRL with the SL, where x relates to the data of SRL layer.

$$\kappa = \{x | x \in SRL\}. \tag{2}$$

Starting from the requirements, we elected to follow a lambda architecture in order to support both batch and streaming data. In Section VA.3, reliability, scalability, and fault-tolerance are analyzed in depth.

#### 2) INGESTION

This component is responsible for collecting data from the source and transferring them to the target. In this proposal, the sources are the NEs and the target is the data lake. According to the information presented in Section III, the format of the raw data collected from the NEs is XML for most vendors and technologies. Therefore, the selected tool must be able to support this file format.

Currently, the PM files are sent from the NEs with a 15 minute of period granularity and are mainly composed of a set of elements described in Fig. 5. We depict each PM file as a vector $P = \{C \cup M_i\}$; $i = 1, \ldots, N$, which consists of the configuration of a PM file in the form of the vector $C$, the information related to the measurement data assigned to the vector $M_i$, and where N relates to the number objects of NEs. It is necessary to select an ingestion tool that allows storage of the approximated set PM on a data lake as a vector $P$.

Appendix A presents an example of an XML PM file for a generic network. There, the vector $C$ is composed by fileFormatVersion, vendorName, dnPrefix, among others' configuration data; the vector $M_i$ is composed of attTCHSeizures, succTCHSeizures, attImmediateAssignProcs, and succImmediateAssignProcs metrics and their apparent numeric values, and N is equal to three because the objects Gbg-997, Gbg-998, and Gbg-999 from the NE RNC-Gbg-1 are measured.

In this article, we consider the following tools that support collecting and transferring XML files: (1) Flume, (2) Kafka, and (3) NiFi [36].

Flume [37] is a service for collecting, aggregating, and moving large amounts of logging information. It works with three tiers: Tier1 is composed of agents that send data to Tier2. Tier2 is composed of the collector that moves the data to final storage. Finally, Tier3 is the target of the data.

Kafka [38] is a distributed streaming platform often used for operational monitoring data. In order to collect XML events, it requires the Zookeeper service to be installed and a mediating tool to transfer the data to the data lake, such as Flume or NiFi. However, that is not the only available working combination. In-house directory pooling connectors can also be deployed as possible transformations, cleanup, and enrichment, before storing the data in Kafka topics.

NiFi [39] is a system designed to process and distribute data. Through a graphic user interface, it allows the connection of sources to targets via any number of workflows. The Cloudera deployment framework supports this tool in the Enterprise version.

Kafka and Nifi have constraints, such as the need for an enterprise version of a deployment framework and other services installation. Therefore, we have selected Flume with the challenges of installing SFTP support, PM files serialization, and additional features.

#### 3) DATA LAKE

This component is the target of the ingestion data; therefore, it must support the XML files storage. Moreover, this must comply with reliability, scalability, and fault-tolerance features, as software architecture requirements mentioned. For reliability, it must be able to maintain data integrity. For scalability, we refer to scale by simply adding new nodes to the cluster to increase its storage capacity. For fault-tolerance, the data lake must provide redundancy, to avoid loss of data one node fails.

All these features are fulfilled by HDFS [40]. Hence, in this work, we selected HDFS as the data lake tool in the proposed Big Data framework. Furthermore, HDFS is the only file system supported by all the deployment frameworks, and it supports the batch-processing system.

As far as we know, nowadays, solutions with more advantages already exist, including solutions such as the MapR filesystem [41]. However, the filesystem is only available in the MapR framework [42].

#### 4) PROCESSING

This component refers to the capability to process the PM data near to real-time. In this study, we focused on batch processing; however, we consider it important to mention the available tools that could be compatible with our proposal.

Spark [43] and Storm [44] are stream processing frameworks that support the near real-time use cases. Spark allows a fast data analysis. Meanwhile, Storm is referred to as a complement to HDFS rather than a total replacement [45]. Therefore, Storm is required for KPI calculation and Spark for real-time monitoring. Both tools are selected for different proposals in the architecture.

#### 5) REPORTING

In this section, the XML files stored in the data lake must be available to be queried. Figures 3, 4, and 5 presented the file structure used for three different vendors. As we verified, every vendor has a different XML structure for PM files; therefore, customized tables must be created for each one. The Appendix A presents an example of the structure of an XML file according to the 3GPP TS 32.401.

Since we selected HDFS as a distributed file system in the data lake component, the reporting tool chosen in this section must be compatible with it. HDFS will store a set $P$ on a data lake as a vector $P = \{C \cup M_i\}$. $P$ regards to the PM file in XML format. The selected reporting tool must be compatible with SQL on the HDFS framework, aiming to provide low latency and high concurrency for reporting queries. This tool must provide direct access to the HDFS data for more efficient processing.

As candidates, we have Hive, Impala, Pig, and Spark SQL. Hive and Pig perform batch processing. Impala and Spark SQL can support near real-time processing.

Hive [46] is based on the HDFS and offers a query language based on the SQL, called HiveQL. Pig [47] allows the analysis of large data sets via a high-level language called Pig Latin. Impala [48] can query data from the HDFS directly and provides lower latency and higher concurrency than Hive. Like Hive, Impala also supports SQL. Spark SQL [49] is a distributed memory query tool, supports batch and streaming queries, and runs native SQL or HiveQL on existing data warehouses.

At this stage, we focused on batch processing. We selected Hive based on the native support to XML files and the benefits mentioned in a study [50] which compared the tools. It is important to highlight that we consider Impala as a good candidate, but, unfortunately, Impala does not support XML natively and additional development must be done. Instead, Hive can use the XML parsing serializer deserializer of IBM to create the tables. Appendix B presents an example of how to create a table in Hive for the XML PM file of Appendix A.

#### 6) DEPLOYMENT

The tools selected for every component analyzed in this section can be implemented individually. Nevertheless, to facilitate the implementation, we selected a deployment platform that provides all the software tools for Big Data frameworks ready to install, configure, and work. A relevant aspect is that these solutions offer support for the end-user and a great community for troubleshooting.

Several studies present a detailed comparison between different deployment frameworks, such as Cloudera, Hortonworks, MapR, IBM InfoS-phere BigInsights, and Pivotal HD as the most used [51], [52]. According to the results of the studies, the majority of the suppliers offer distributions based on Apache Hadoop and open-source projects. The studies do not mention any recommended framework since it will depend on the requirements of the project.

As far as we know, Cloudera announced the completion of its merger with Hortonworks in January 2019 [53] and advertised a 100 % open-source data platform. Since these two suppliers comply with our requirements and are the most used in the deployments [52], we selected them for testing scenarios. However, for production implementations, features such as technical support and training, which are not available in the community version, can be considered. It is important to highlight that Cloudera supports all the software tools previously selected, and the deployment can be performed on any server or cloud solutions.

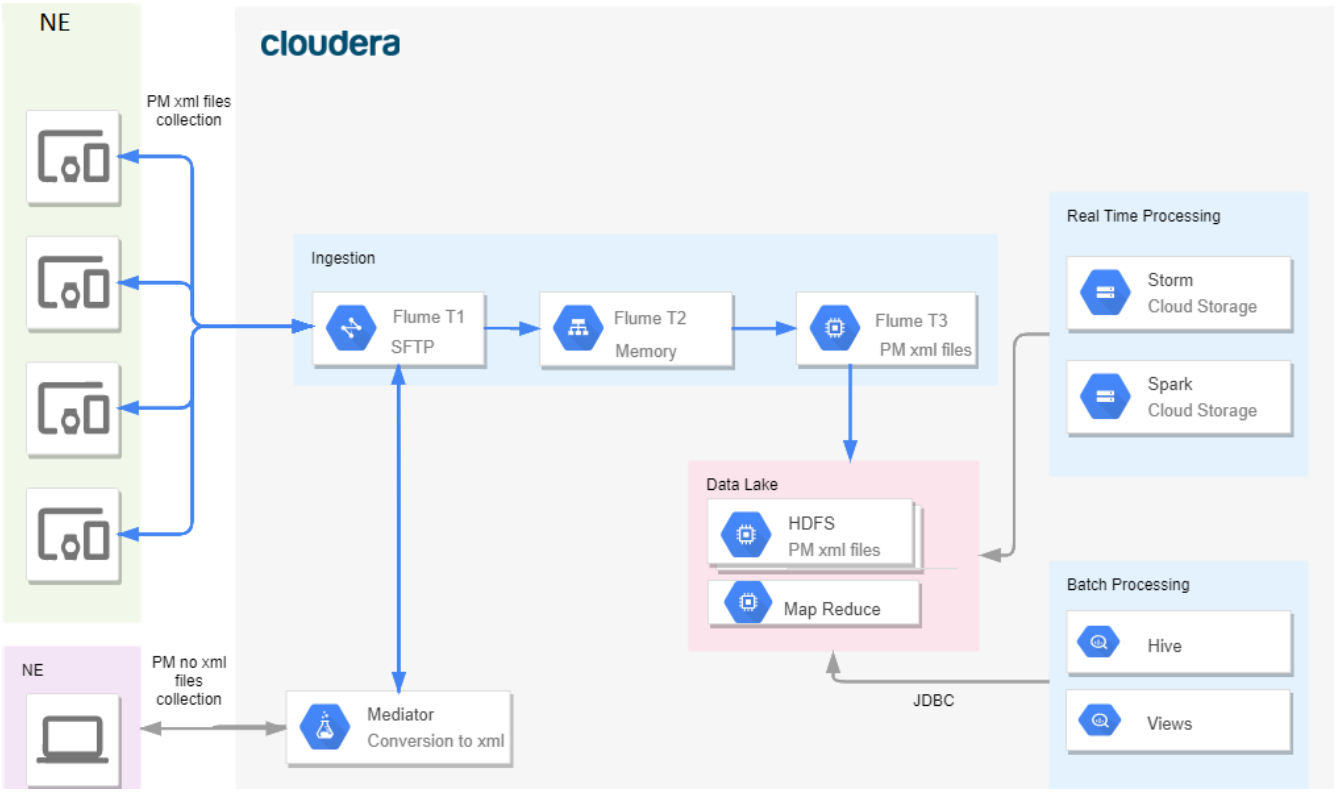


**FIGURE 7.** NMS architecture for PM in mobile networks using a big data framework.

### *B. ARCHITECTURE*

Following the analysis and selection of all the components of the Big Data framework for PM according to the project requirements, in this section we present the proposed architecture to be implemented to answer the second research question: What Big Data architecture is most appropriate for the format of PM files most used by the vendors?

To assist the reader, Figure 7 presents our proposed NMS architecture for PM in mobile networks using a Big Data framework. In order to explain the overall architecture and processing flow, we will explain three main layers: (1) deployment, (2) ingestion, and (3) reporting.

Starting from (1), the deployment involves the latest version of Cloudera Distribution Hadoop (CDH). The Cloudera open-source environment and some necessary services must be enabled, for instance, Flume, HDFS, Hive, Spark, and Storm. There is also the possibility of installing the tools individually.

For (2), the Big Data framework must collect PM data from NEs from different vendors using the SFTP protocol. For this, Flume acts as a loader, taking care of connecting to each NEs, locating the PM files stored as data at rest, transforming them to XML if required, and storing them in the HDFS. Flume works via agents that can be configured at the NMS side, since it may not be possible or recommended to install additional software into the NEs. These agents are arranged in three tiers, Tier1 is the source and connects and collects PM files from each NE, Tier2 as the sink is a memory buffer, and Tier3 as the target stores the transferred data into the HDFS. It is neccesary to have one set of $N$ Tier1 agents, one Tier2 agent and one Tier3 agent for each technology, where $N$ is the number of the NEs of that technology.

For (3), SQL queries are used as required. After storing the PM data in the HDFS, they can be used directly by Spark, Storm, or Hive for queries. In the batching process, each PM file is mapped to Hive tables, where each table structure depends on each vendor's technology for the differences previously mentioned in Section V.A5. As Fig. 6 presents, the schema consists of physical dimension files separated by vendor and technology. A single table stores the configuration and measurement information.

We consider that, in this layer, aggregation rules can be recreated, for instance, to calculate the busy hour, hourly, daily, weekly, and monthly reports. Graphical user interfaces are not considered in the approach since it is expected that the existing ones are reused. In the next section, we present the test environments used to evaluate the framework experimentally.

## VI. EVALUATION OF THE RESULTS

This section allows us to answer the two last research questions of this study: How many counters can be ingested in the proposed architecture and what are the computation resources required? How many counters can be returned from querying

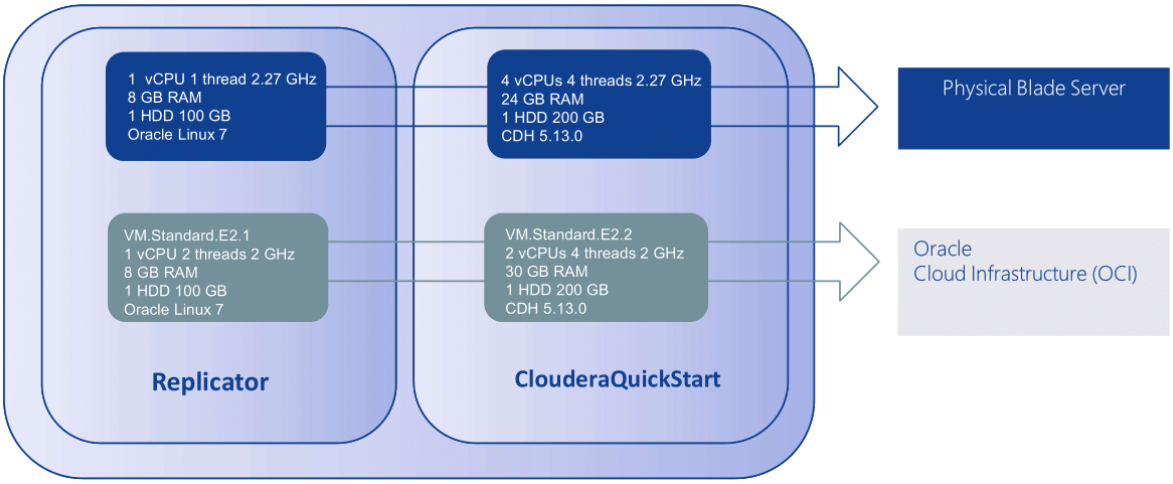


**FIGURE 8.** Test environments dimensioning.

a raw table in the proposed architecture and what are the computation resources required?

As a test strategy, PM files related to the 5G technology, consisting of 4722 counters each, were used to test the ingestion and reporting layers of the proposed architecture. These files were selected by the number of counters and for being the latest technology generation. The complete results of this study are available in our dataset in the IEEE DataPort.[1]

For the tests of the ingestion layer, we evaluated the time to transfer counters or metrics from the simulated NE to the HDFS. A counter refers to the values sensed in the network that are used to calculate the KPI. For the tests of the reporting layer, we measured the total time taken to query the counters from a raw table and a view with all the counters of a PM file. The main considerations in the test environments are described below.

### A. TEST ENVIRONMENTS

In this research, the architecture proposed in Fig. 7 has been implemented in two different test environments in order to compare results. In both environments, two Virtual Machines (VM) were deployed. The first, named Replicator, simulated a NE. Replicator VM replicated PM files using a data set from a real 5G network as reference. Another VM, named ClouderaQuickStart, hosted the Cloudera deployment framework. Figure 8 presents the dimensioning of the test environments.

The first test environment consisted of a physical blade server with 48 GB of RAM and eight cores capable of running 16 threads at 2.27 GHz.

- For the Replicator VM, hosting an Oracle Linux 7.7 system, one vCPU, 8 GB of RAM, and 100 GB of storage were allocated. In this VM, PM files were generated with a granularity period of 15 minutes, and a maximum consumption of 7.84 GB of memory was detected.
- For the Cloudera VM, four vCPUs, 24 GB of RAM, and 200 GB of storage were allocated, and it hosted Cloudera CDH 5.13.0, which is based on CentOS 6.7.

The second test environment was hosted on Oracle Cloud Infrastructure (OCI), and the same software as in the first environment was deployed.

- For the Replicator VM, a VM.Standard.E2.1 shape with 100 GB of storage was used. This shape provided one vCPU capable of running two threads at a base clock of 2 GHz and 8 GB of RAM.
- For the Cloudera VM, a VM.Standard2.2 shape with 200 GB of storage was used. This shape provided two vCPUs capable of running four threads at a base clock of 2 GHz and 30 GB of RAM.

We highlight that Cloudera cannot be installed with less than 16 GB of RAM and, according to the tests, 24 GB was the minimum requirement for enabling Cloudera services and data processing.

### B. DATA REPLICATION

Here, we developed a software tool to replicate the PM files. Our goal was to simulate the production of PM files in the NE every 15 minutes as in a real network. For this, the following packages were required:

- Python 2.7.5 - An interpreted, interactive, object-oriented programming language.
- JRE 1.8 - Java Platform Standard Edition Runtime Environment.
- P7zip - Very high compression ratio file archive.

The PM files produced were regularly placed, every 15 minutes, under different folders as per vendor and technology. This allowed the use of a single system to replicate the real behavior of the NEs.

[1] https://ieee-dataport.org/documents/ingestion-and-reporting-times-processing-5g-performance-management-files-big-data#files

It is important to highlight that PM files transfer between NEs and NMS is usually performed through SFTP in either a push or pull model:

- In the push model, the NEs periodically connect to an NMS and send newly created data. In the case of success, transferred data is moved to a backup folder. In the case of failure, data are kept in a ''pending'' folder to attempt re-transmission at the next cycle.
- In the pull model, one or more NMS periodically connect to the NEs and collect new files. It is highly recommended that each NMS implements an accounting mechanism to ascertain what files have already been transferred. If configured to do so, the NMS can also delete files from the NEs after transfer and forego the accounting mechanism.

As this work does not propose changes at the NE side, it is necessary that the NMS implements the pull model.

### C. DATA INGESTION

As described in Section IV.B, for data ingestion, we used the Flume tool with a plugin to support SFTP. After being collected, PM files were parsed and loaded into the HDFS. We highlight that PM files are collected and transferred in XML format without any conversion.

The relevant configurations in the test environment were:

- Replicator VM as the source, where the PM files were collected via SFTP.
- The delay to discover new PM files equal to 60000 ms.
- Recursive search for XML files.
- Data was temporarily stored in memory in Flume.
- HDFS as a target with data stream as the file type.
- The HDFS path must correspond to the vendor and technology. The HDFS is the long-term data lake used by reporting and aggregation processes.
- Flume logs were used as a notification service to keep track of what data has been stored in the HDFS.

As far as we know, HDFS was built to work with large files, and according to Table 1, PM files are small, between 0.02 MB and 47 MB. Therefore, we first experimented with the behavior, from the data ingestion perspective, to ascertain whether merging several generated files was better than loading them individually in delay terms. We tested a different number of PM files in order to assess ingestion times.

Figure 9 presents the results obtained. For merged files, the fitted regression line complies with the equation (3), where $y_1$ corresponds to the time variable and $x_1$ the number of PM merged files. For the equation (3), the determination coefficient, denoted $R^2$, is equal to 0.9621 and the Pearson correlation coefficient $R$ is equal to 0.9809. Therefore, we can observe the direct relationship between the time and the number of merged files variables, and as $R$ is very close to the value 1, there is a strong linear dependence between them.

$$y_1 = 0.2313x_1 - 109.5 \tag{3}$$

For individual files, the fitted regression line complies with the equation (4), where $y_2$ corresponds to the time variable and $x_2$ the number of PM individual files. The determination coefficient $R^2$ and the Pearson correlation coefficient $R$ are equal to 0.9999. Therefore, there is a strong linear dependence between them, even more than for merged files.

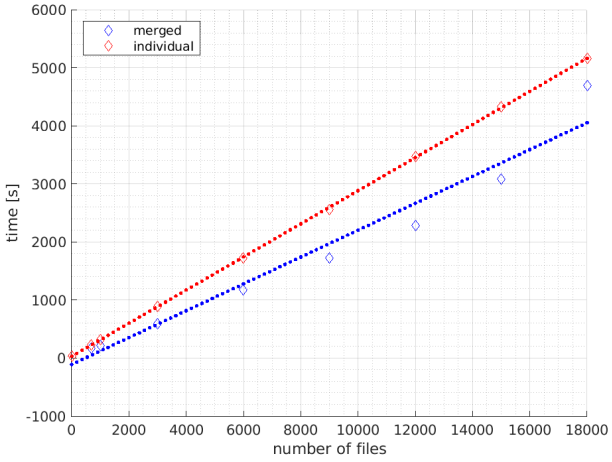


**FIGURE 9.** Ingestion times comparison.

$$y_2 = 0.2851x_2 + 30.83 \tag{4}$$

Minor delays were observed when the files were merged into a single file, according to the equation (5). We depict the ingested PM file as $PM_{total}$, which is equal to a set of elements $PM_i$, and where N relates to the number of NEs.

$$PM_{total} = \{PM_1 \cup PM_2 \cup \ldots.. \cup PM_N\} \tag{5}$$

For 1000 files, we can reduce around 100 s, while for 18000 files, we can reduce around 500 s, the time to merge the files considered. Therefore, sending individual files from the NEs to the HDFS causes longer delays.

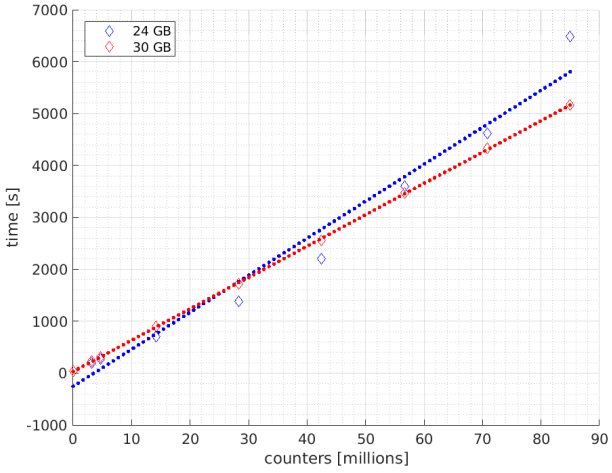


**FIGURE 10.** Ingestion time per counter.

We also evaluated the number of ingested counters that the solution can reach. Figure 10 presents the number of

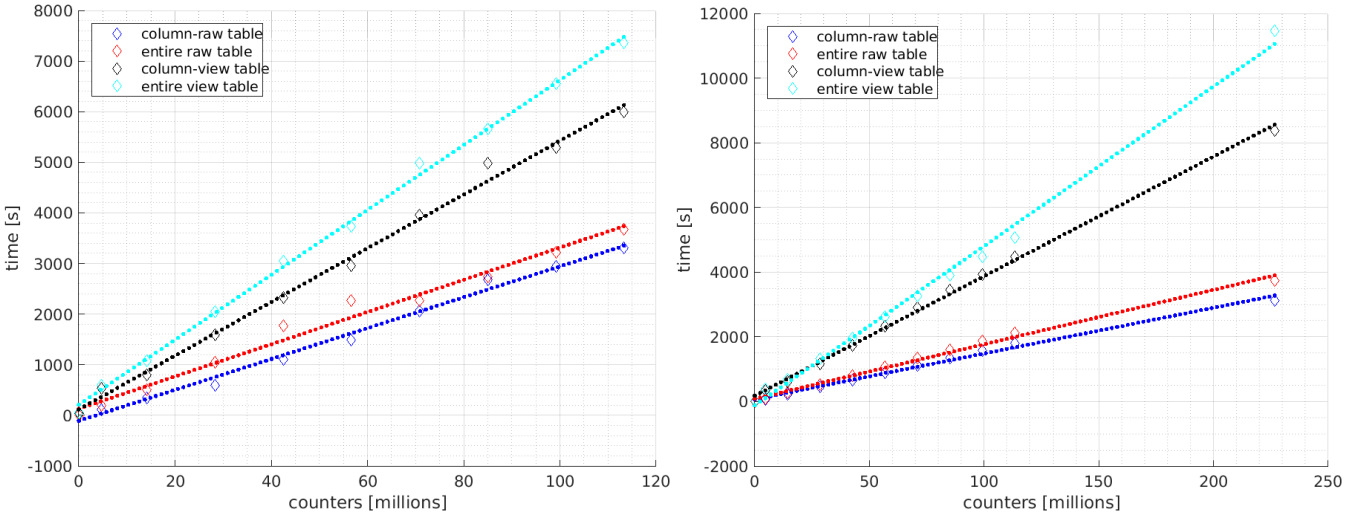


**FIGURE 11.** Reporting time per counter.

counters and the ingestion times attained in both test environments. We tested individual 5G PM files, with 4722 counters each.

For the test environment with 24 GB of RAM, the fitted regression line in Fig. 10 complies with the equation (6), where, $y_3$ corresponds to the time variable and $x_3$ the number of counters. The determination coefficient $R^2$ is equal to 0.9718 and the Pearson correlation coefficient $R$ is equal to 0.9858. As previously explained, since it is close to 1, there is a strong linear dependence between the variables.

$$y_3 = 71.29x_3 - 254.3 \tag{6}$$

For the test environment with 30 GB of RAM, the fitted regression line in Fig. 10 complies with the equation (7), where, $y_4$ corresponds to the time variable and $x_4$ the number of counters. The determination coefficient $R^2$ and the Pearson correlation coefficient $R$ are equal to 0.9999. Therefore, the variables are more linearly dependent than in the test environment with 24 GB of RAM.

$$y_4 = 60.38x_4 + 30.83 \tag{7}$$

As a result, around 57 million counters per hour can be loaded into HDFS with 24 GB of RAM and 59 million counters with 30 GB of RAM. Therefore, to reach one billion counters per hour, we estimate that at least 512 GB of RAM is needed if sending individual files and neither other data compression nor analysis techniques are used.

While sending merged files, it is possible to reach around 85 million counters per hour with a node with 30 GB of RAM. Therefore, we estimate the need for 11 nodes with 30 GB or a single node with 320 GB of RAM to reach one billion counters per hour.

### D. DATA REPORTING

Since the PM files format was XML, the table structure created in Hive for reporting was based on the XML structure. In this work, we did not perform tests related to aggregations, only over the raw data and a view for evaluating complex queries. Appendix C presents an example of the table's creation in Hive. As mentioned in Section V.A5, it was important to use the IBM XML parsing serializer deserializer to avoid extra processing in the PM files.

Figure 11 presents the results obtained by report generations based on different numbers of PM files. The tests compared execution times between selections for a single column and all columns of a raw table, and a view.

For the environment with 24 GB of RAM, and according to Fig. 11 a), around 113 million counters per hour were returned from querying the raw table and 57 million from the view. When the queries were focused on a single column, the maximum number of counters increased to around an additional 14 million. For the 30 GB environment, and according to Fig. 11 b), around 226 million counters per hour were queried from the raw table and 85 million from the view. When the queries were focused on a single column, the maximum number of counters also increased by around an additional 14 million.

It is clear that an increase in RAM directly impacts on report execution time and, consequently, on the number of counters that can be queried. Through these results, we noted that the additional 6 GB of RAM returned an additional 113 million counters, that is, 100 % more. Furthermore, we estimated that with 72 GB of RAM, we can query one billion counters per hour from a raw table. Report execution time degrades linearly as load increases and, as expected, there is a clear relationship between the number of counters and the report execution time.

### E. SYSTEM RESOURCES

During the Ingestion tests, CPU utilization was under 10 % in the Cloudera server, while the consumed memory was near 100 %. For the Reporting tests, CPU utilization maxed at around 74 %, and memory consumption was nearly 100 %. During these tests, no issues were encountered in the HDFS, network, or disk input/output.

Finally, in order to respond to the last two research questions, in Table 4 we summarize the results obtained for the proposed architecture. The computation resources required are presented to support one billion counters per hour for the ingestion and reporting components. The presented values were calculated from Fig. 10 and Fig. 11.

**TABLE 4. Computation resources to support 1 billion counters per hour for ingestion and reporting.**

| Component | RAM [GB] | vCPU 2.27 GHz | Hard Disk [TB] |
|---|---|---|---|
| Ingestion individual files | 512 | 68 | 10 |
| Ingestion merged files | 320 | 44 | 10 |
| Reporting | 72 | 12 | 10 |

## VII. CONCLUSION

Based on this research, several important aspects were uncovered. First, we could not find related works presenting results regarding the evaluation of frameworks based on the Big Data stack for PM data in mobile networks. Therefore, this article presents a novel approach to the problem of supporting scalability, volume, variety, and velocity in NMS, by analyzing the software architecture, ingestion, data lake, processing, reporting, and deployment components.

Furthermore, after analyzing PM files from four different vendors and four telecommunication technologies, we determined that some vendors adopt the XML format as defined by the 3GPP TS 32.401. The conceptual and logical data models were presented for the PM files of each vendor. In summary, each PM file consisted of configuration information and measurement data. The PM files can be loaded in different file systems for each vendor and technology, and the database engine allowed for SQL queries to run directly over the XML file. Another aspect analyzed was that PM file sizes ranged from 0.02 MB to 47 MB depending on the number of NEs reported in the file.

Once the data type to be stored and queried were analyzed, the research determined the appropriate framework implementation tools. Based on the Big Data components, the suitable tools were as follows. For the software architecture component, Lambda was selected for batch processing support. For the ingestion component, Flume was selected to transfer XML PM files from the NEs to the HDFS, chosen as the data lake. Spark was selected for the processing component. For the reporting component, Hive was selected for its XML and SQL native support. All tools can be implemented in a Cloudera community deployment.

The selected components for the Big Data framework were implemented in virtual machines with 24 GB and 30 GB of RAM in order to compare our solution. We evaluated the number of counters that could be ingested per hour from the NEs to the HDFS and the amount of resources needed to support up to one billion counters per hour. The number of counters that could be queried in a report per hour was also evaluated by creating raw tables and views in Hive that directly consulted the XML files in the HDFS. The resources required to support up to one billion counters per hour were also presented.

We can conclude that a Big Data framework allows the processing of a larger number of PM files with fewer consumed resources, for example, up to one billion counters per hour with 320GB of RAM in the ingestion component, and 72 GB of RAM in the reporting component. We have tested for only one node of every tool; however, more Flume, HDFS name, HDFS data, and Hive nodes can be added to the framework in order to achieve better performance results and comply with the scalability requirement. We can also prove our solution maintains the integrity of the PM data, from the beginning to the end of the process, according to the reliability requirement.

As future work, we plan to evaluate data aggregations, real-time reports by implementing a kappa architecture, and 6G PM files based on the 3GPP Technical Specification version 32.401.

## APPENDIX A
## EXAMPLE OF XML PM FILE

The following file presents an example of the structure of an XML file for PM in mobile networks according to the 3GPP TS 32.401.

```
<measCollecFile
<fileHeader fileFormatVersion
   ="32.401 V5.0"
vendorName="Company Generic"
dnPrefix="DC=a1.companygeneric.com,
   SubNetwork=1,IRPAgent=1">
<fileSender
localDn="SubNetwork=Country,MeContext
   =MEC-Gbg-1,ManagedElement=RNC-Gbg-1"
elementType="RNC"/>
<measCollec beginTime="2020-10-01T14:
     00:00+02:00"/>
</fileHeader>
<measData>
<managedElement
localDn="SubNetwork=Country,MeContext
   =MEC-Gbg-1,ManagedElement=RNC-Gbg-1"
userLabel="RNC generic"/>
<measInfo>
<granPeriod duration="PT900S"
   endTime="2020-10-01T14:14:30+02:00"/>
<measType p="1">attTCHSeizures
```

```
      </measType>
<measType p="2">succTCHSeizures
      </measType>
<measType p="3">attImmediateAssignProcs
      </measType>
<measType p="4">succImmediateAssignProcs
      </measType>
<measValue measObjLdn="RncFunction=RF-1,
      UtranCell=Gbg-997">
     <r p="1">234</r>
     <r p="2">345</r>
     <r p="3">567</r>
     <r p="4">789</r>
</measValue>
<measValue measObjLdn="RncFunction=RF-1,
      UtranCell=Gbg-998">
     <r p="1">890</r>
     <r p="2">901</r>
     <r p="3">123</r>
     <r p="4">234</r>
</measValue>
<measValue measObjLdn="RncFunction=RF-1,
      UtranCell=Gbg-999">
     <r p="1">456</r>
     <r p="2">567</r>
     <r p="3">678</r>
     <r p="4">789</r>
     <suspect>true</suspect>
</measValue>
</measInfo>
</measData>
<fileFooter>
<measCollec endTime="2020-10-01T14:15:00
      +02:00"/>
</fileFooter>
</measCollecFile>
```

## APPENDIX B
## CREATION OF HIVE TABLE FOR XML PM FILE

The following SQL sentences present an example of how to create a table in Hive for the XML PM file of Appendix A. Option INPUTFORMAT sets the use of the parsing serializer deserializer of IBM.

```
CREATE EXTERNAL TABLE XML_PM_TABLE_HIVE
  ( xml STRING )
ROW FORMAT SERDE "com.ibm.spss.hive.
  serde2.xml.XmlSerDe"
WITH SERDEPROPERTIES
  ( "column.xpath.xml"="/" )
STORED AS
    INPUTFORMAT "com.ibm.spss.hive.
      serde2.xml.XmlInputFormat"
    OUTPUTFORMAT "org.apache.hadoop.
      hive.ql.io.
      IgnoreKeyTextOutputFormat"
    LOCATION "/HDFS_DIRECTORY"
TBLPROPERTIES ( "xmlinput.start"="
  <measCollecFile", "xmlinput.end"="
  </measCollecFile>" );
```

## APPENDIX C
## CREATION OF HIVE TABLE FOR XML 5G PM FILE

The following SQL sentences present an example of how to create a table in Hive for a 5G XML PM file.

```
CREATE EXTERNAL TABLE 5G_RAW_TABLE (
    time STRING,
    interval INT,
    rawData ARRAY<STRING>,
    orgData ARRAY<STRUCT<
        PMMOResult:STRUCT<
            MO:ARRAY<STRUCT<DN:STRING>>,
            PMTarget:MAP<STRING,STRING>
        >
    >>
)
ROW FORMAT SERDE 'com.ibm.spss.
  hive.serde2.xml.XmlSerDe'
WITH SERDEPROPERTIES (
    "column.xpath.time
      "="/PMSetup/@startTime",
    "column.xpath.interval
      "="/PMSetup/@interval",
    "column.xpath.rawData
      "="/PMSetup/PMMOResult",
    "column.xpath.orgData
      "="/PMSetup/PMMOResult"
)
STORED AS
    INPUTFORMAT 'com.ibm.spss.hive.
      serde2.xml.XmlInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.
      hive.ql.io.
        IgnoreKeyTextOutputFormat'
    LOCATION "/HDFS_DIRECTORY"
TBLPROPERTIES (
    "xmlinput.start"="<PMSetup ",
    "xmlinput.end"="</PMSetup>"
);
```

## ACKNOWLEDGMENT

The authors would like to thank telecommunication operators' support engineers for their valuable contribution.

## REFERENCES

[1] D. Kovačević, A. Krajnović, and D. Čičin-Šain, "Market analysis of the telecommunications market—The case of croatia," in *Proc. Dubrovnik Int. Econ. Meeting*, 2017, vol. 3, no. 1, pp. 161–175.
[2] K. David and H. Berndt, "6G vision and requirements: Is there any need for beyond 5G?" *IEEE Veh. Technol. Mag.*, vol. 13, no. 3, pp. 72–80, Sep. 2018.
[3] J. Sathyan, *Fundamentals of EMS, NMS and OSS/BSS*, 1st ed. Boca Raton, FL, USA: Auerbatch Publications, 2010.

[4] *Telecommunication Management;Performance Management (PM);Concept and Requirements*, document TS 32401.550, 3GPP. Accessed: Aug. 18, 2020. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/32_series/32.401/

[5] GSM. *The Mobile Economy 2020*. Accessed: Aug. 28, 2020. [Online]. Available: https://www.gsma.com/mobileeconomy/wp-content/uploads/2020/03/GSMA_MobileEconomy2020_Global.pdf

[6] *The Digitization of the World From Edge to Core*. Accessed: Aug. 28, 2018. [Online]. Available: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf/

[7] S. Yantao, Y. Fangnan, and S. Zhi'qiang, "Distributed network management system with load balancing," *J. Commun.*, vol. 30, no. 3, pp. 34–41, 2009.

[8] G. Krzysztof and N. Leszek, "Testing and scalability analysis of network management systems using device emulation," in *Proc. Int. Conf. Comput. Netw.*, Berlin, Germany, 2012, pp. 91–100.

[9] *Future OSS: Towards an Open Digital Architecture*. Accessed: Oct. 14, 2020. [Online]. Available: https://inform.tmforum.org/research-reports/future-oss-towards-an-open-digital-architecture/

[10] *View on 5G Architecture*. Accessed: Oct. 14, 2020. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf

[11] *iManager U2000 Unified Network Management System V100R002C00 Product Description*, Huawei Industrial Base, Huawei Technol., Shenzhen, China, 2010, pp. 3.1–3.10, vol. 3.

[12] *Shaping the Future of Telecommunication*. Accessed: Oct. 14, 2020. [Online]. Available: https://nokiawroclaw.pl/wp-content/uploads/2019/03/NOKIA_Book_2nd.pdf

[13] G. Saadon, Y. Haddad, and N. Simoni, "A survey of application orchestration and OSS in next-generation network management," *Comput. Standards Interfaces*, vol. 62, pp. 17–31, Feb. 2019.

[14] *The 2019 Guide to Modern OSS*. Accessed: Oct. 14, 2020. [Online]. Available: https://media.ciena.com/documents/The-Guide-to-Modern-OSS-2019-Edition.pdf/

[15] M. Wibowo, S. M. Shamsuddin, and N. Simoni, "2nd machine learning in data lake for combining data silos," in *Proc. Int. Conf. Data Mining Big Data*, Fukuoka, Japan, 2017, pp. 294–306.

[16] R. Magalhães, "5G y IoT—Tendencias y Aplicaciones," Fundação Instituto Nacional de Telecomunicações–Finatel, Santa Rita do Sapucaí, Brazil, Tech. Rep. 1, Accessed: Jul. 14, 2020.

[17] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, Mar. 2020.

[18] D. Martinez-Mosquera, R. Navarrete, and S. Lujan-Mora, "Modeling and management big data in databases—A systematic literature review," *Sustainability*, vol. 12, no. 2, p. 634, Jan. 2020.

[19] S. Baik, Y. Jeon, C. Hwang, and Y. Lee, "A tiering architecture for integrated network management system," in *Proc. 15th Asia–Pacific Netw. Oper. Manage. Symp.*, Hiroshima, Japan, 2013, pp. 1–3.

[20] A. Samba, "A network management framework for emerging telecommunications networks," in *Modeling and Simulation Tools for Emerging Telecommunication Networks*. Boston, MA, USA: Springer, 2006, pp. 179–200.

[21] M. Yi, S. Jing, S. Junde, and S. Mei, "Considerations for the development of large scale mobile network management system," in *Proc. Can. Conf. Electr. Comput. Eng.*, Niagara Falls, ON, Canada, 2004, pp. 1139–1142.

[22] M. A. L. Pratama, T. F. Kusumasari, and R. Andreswari, "Data processing architecture using opensource bigdata technology to increase transaction speed," in *Proc. 3rd Int. Conf. Informat. Comput. (ICIC)*, Palembang, Indonesia, Oct. 2018, pp. 1–6.

[23] A. Mazdziarz, "Alarm correlation in mobile telecommunications networks based on k-means cluster analysis method," *J. Telecommun. Inf. Technol.*, vol. 2, pp. 95–102, Jul. 2018.

[24] A. Rayes and K. Sage, "Integrated management architecture for IP-based networks," *IEEE Commun. Mag.*, vol. 38, no. 4, pp. 48–53, Apr. 2000.

[25] K. Skračić and I. Bodrušić, "A big data solution for troubleshooting mobile network performance problems," in *Proc. 40th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, Opatija, Croatia, May 2017, pp. 472–477.

[26] A. Suleykin and P. Panfilov, "Distributed big data driven framework for cellular network monitoring data," in *Proc. 24th Conf. Open Innov. Assoc. (FRUCT)*, Moscow, Russia, Apr. 2019, pp. 430–436.

[27] Y. Lin, Y. B. Wang, Y. B. Wang, H. Zhang, and J. H. Yang, "Research on information models and modelling methods of the new generation of OSS for data management," in *Proc. Int. Conf. Comput. Inf. Syst. Ind. Appl.*, Bangkok, Thailand, 2015, pp. 410–413.

[28] P. Zhou, Z. Yang, L. Li, and S. Qiu, "Application of big data processing technology in the intelligent network management system," in *Proc. Web Technol. Appl.*, Guangzhou, China, 2015, pp. 26–34.

[29] *OOKLA 5G MAP*. Accessed: Aug. 18, 2020. [Online]. Available: https://www.speedtest.net/ookla-5g-map/

[30] Agencia de Regulación y Control de las Telecomunicaciones. *Boletín Estadístico Julio 2020*. Accessed: Jul. 17, 2020. [Online]. Available: https://www.arcotel.gob.ec/wp-content/uploads/2018/11/1.2-Radiobases-por-operador-y-tecnologia-nivel-provincial_May-2020-1.xlsx

[31] C. A. Salma, B. Tekinerdogan, and I. N. Athanasiadis, "Domain-driven design of big data systems based on a reference architecture," in *Software Architecture for Big Data and the Cloud*. Burlington, MA, USA: Morgan Kaufmann, 2017, pp. 49–68.

[32] Microsoft. *Big Data Architecture Style*. Accessed: Aug. 28, 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/big-data

[33] J. Tomcy and M. Pankaj, *Data Lake for Enterprises*. Birmingham, U.K.: Packt, 2017.

[34] *IEEE Recommended Practice for Architectural Description for Software-Intensive Systems*, Standard IEEE 1471-2000. Accessed: Jul. 21, 2020. [Online]. Available: https://standards.ieee.org/standard/1471-2000.html

[35] S. Ounacer, M. Amine, S. Ardchir, A. Daif, and M. Azouazi, "A new architecture for real time data stream processing," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 11, pp. 44–51, 2017.

[36] A. Matacuta and C. Popa, "Big data analytics: Analysis of features and performance of big data ingestion tools," *Inf. Economica*, vol. 22, no. 2, pp. 25–34, Jun. 2018.

[37] Flume. *Apache Flume*. Accessed: Jul. 24, 2020. [Online]. Available: https://flume.apache.org/

[38] Kafka. *Apache Kafka*. Accessed: Jul. 24, 2020. [Online]. Available: https://kafka.apache.org/

[39] NiFi. *Apache Kafka*. Accessed: Jul. 24, 2020. [Online]. Available: https://nifi.apache.org/

[40] D. Borthakur, "HDFS architecture guide," *Hadoop Apache Project*, vol. 53, no. 2, pp. 1–13, 2008.

[41] *MapR 6.1 Documentation Hewlett Packard Enterprise*. Accessed: Jul. 24, 2020. [Online]. Available: https://mapr.com/docs/61/MapROverview/c_maprfs.html

[42] *MapR 6.1 Documentation Hewlett Packard Enterprise*. Accessed: Jul. 24, 2020. [Online]. Available: https://mapr.com/

[43] Spark. *Apache Spark*. Accessed: Jul. 24, 2020. [Online]. Available: https://spark.apache.org/

[44] Storm. *Apache Storm*. Accessed: Jul. 24, 2020. [Online]. Available: https://storm.apache.org/

[45] T. S. Morais, "Survey on frameworks for distributed computing: Hadoop, spark and storm," in *Proc. 10th Doctoral Symp. Informat. Eng.*, vol. 1, 2015, pp. 95–105.

[46] Hive. *Apache Hive*. Accessed: Jul. 24, 2020. [Online]. Available: https://hive.apache.org/

[47] *Apache Pig*. Accessed: Jul. 24, 2020. [Online]. Available: https://pig.apache.org/

[48] Impala. *Apache Impala*. Accessed: Jul. 24, 2020. [Online]. Available: https://impala.apache.org/

[49] Spark SQL. *Apache Spark SQL*. Accessed: Jul. 24, 2020. [Online]. Available: https://spark.apache.org/sql/

[50] X. Li and W. Zhou, "Performance comparison of hive, impala and spark SQL," in *Proc. 7th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.*, Hangzhou, China, Aug. 2015, pp. 418–423.

[51] A. Erraissi, A. Belangour, and A. Tragha, "A comparative study of Hadoop-based big data architectures," *Int. J. Web Appl.*, vol. 9, no. 4, pp. 129–137, 2017.

[52] G. S. Bhathal and A. S. Dhiman, "Big data solution: Improvised distributions framework of Hadoop," in *Proc. 2nd Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Jun. 2018, pp. 35–38.

[53] Cloudera. *Cloudera and Hortonworks Complete Planned Merger*. Accessed: Jul. 21, 2020. [Online]. Available: https://www.cloudera.com/about/news-and-blogs/press-releases/2019-01-03-cloudera-and-hortonworks-complete-planned-merger.html

**DIANA MARTINEZ-MOSQUERA** received the B.Sc. degree in electronic and information networks engineering from the Escuela Politécnica Nacional, Ecuador, in 2008, and the M.Sc. degree in networks and telecommunications in 2014. She is currently pursuing the Ph.D. degree in computer science with the University of Alicante, Spain. She has around ten years of experience as a Support Engineer of network management systems. She currently works as a part-time Professor and a Researcher with the Department of Informatics and Computer Science, Escuela Politécnica Nacional. She has authored several publications at international conferences and journals. Her main research interests include big data engineering, semi-structured data modeling, and operation business platforms for telecommunication operators.

**ROSA NAVARRETE** received the Ph.D. degree in informatics from the University of Alicante. She currently works as a full-time Professor and a Senior Researcher with the National Polytechnic School. She has published several research papers at international conferences and in high-impact journals. Her main research interests include web accessibility and usability, UX, and open educational resources. She has chaired several international conferences and workshops. She serves as an Editor on multiple editorial boards.

**SERGIO LUJÁN-MORA** received the B.Sc. degree in computer science and engineering from the University of Alicante, Spain, in 1998, and the Ph.D. degree in computer engineering from the Department of Software and Computing Systems, University of Alicante, in 2005. He is currently a Senior Lecturer with the Department of Software and Computing Systems, University of Alicante. He has authored several books and numerous articles published in various conference proceedings (ER, UML, and DOLAP) and high-impact journals (DKE, JCIS, JDBM, JECR, JIS, JWE, IJEE, and UAIS). His main research interests include web applications, web development, and web accessibility and usability. His current research interests include e-learning, massive open online courses, open educational resources, and the accessibility of video games.