Pérez, Ramón; García-Reinoso, Jaime; Zabala, Aitor; Serrano, Pablo; Banchs, Albert. (2020). A Monitoring Framework for Multi-Site 5G Platforms. *2020 European Conference on Networks and Communications (EuCNC2020)*, Piscataway, NJ: IEEE. Pp.: 52-56.

DOI: https://doi.org/10.1109/EuCNC48522.2020.9200914

A Monitoring Framework for Multi-Site 5G Platforms

Ramon Perez*†, Jaime Garcia-Reinoso†, Aitor Zabala*, Pablo Serrano†, Albert Banchs†‡
*Telcaria Ideas, Spain †Universidad Carlos III de Madrid, Spain ‡IMDEA Networks Institute, Spain

***Abstract*—The fifth generation (5G) of mobile networks will have to accommodate different types of use cases, each of them with different and stringent requirements and key performance indicators (KPIs). To support this, apart from novel technologies such as network slicing or artificial intelligence, 5G will require a flexible and efficient monitoring system. The collected metrics serve to optimize the performance of the network, and to confirm the achievement of the KPIs. Furthermore, in the envisioned multi-site, multi-stakeholder scenarios, having a common monitoring system is even more critical for an efficient optimization and service provisioning. In this paper, we present a Monitoring architecture for the distribution and consumption of metrics and KPIs for 5G multi-site platforms, where different verticals from different stakeholders are implemented over a shared infrastructure. We also assess the performance of the implemented publish-subscribe paradigm, to confirm that it suits the requirements of these scenarios, and discuss how the architecture could be mapped to other 5G scenarios.**

Keywords*— *Monitoring, data collection, 5G experimental validation, 5G trials, 5G multi-site platform, publish-subscribe paradigm

I. Introduction

The main idea behind Network Slicing [1] is the ability to define multiple isolated logical networks from a single physical one. Moreover, each logical network may support a particular type of service: low latency, high bit rate, massive number of terminals, etc. 5G telecommunication operators have to design their networks to support all these services and, what is even more important, to guarantee that the Key Performance Indicators (KPIs) demanded by their verticals are satisfied. In this aspect, monitoring of network metrics is crucial to guarantee all Service Level Agreements (SLAs) between operators and users.

Triggered by the complexity and novelty of 5G, several research initiatives have started to gather an understanding of the envisioned features of these types of networks. 5G EVE[1] is a European project that is deploying a validation 5G multi-site platform, involving four main facilities located in Spain, Italy, France, and Greece, where verticals and other projects can execute extensive trials. After an initial phase where verticals have provided their requirements (reported in [2]), the project presented in [3] the proposed architecture and the main innovation areas addressed, including the KPI Framework for performance diagnosis.

One of the main components of this architecture related to the aforementioned innovative topics is the Monitoring service, which is intended to collect all the metrics generated by the different elements involved in an experiment to show their evolution over time to the experimenter, and to feed such data to the KPI Validation Framework.

This paper presents the Monitoring framework defined by the 5G EVE project, which has been designed to be flexible enough to be implemented in other projects as well as by telecommunication operators within the scope of advanced 5G networks. One example is the 5GROWTH project[2], integrating some ideas and concepts present in the 5G EVE Monitoring platform in the so-called Vertical-oriented Monitoring System (5Gr-VoMS), an extension of the monitoring solution already proposed in the 5G-TRANSFORMER project[3].

The rest of the paper is organized as follows:

- Section II describes the Monitoring architecture, which has been designed as a scalable, reliable, low-latency, distributed, multi-source data aggregation and re-configurable architecture.
- Section III justifies and details the implementation selected by 5G EVE to instantiate the proposed architecture, based on the publish-subscribe paradigm.
- Section IV validates such implementation against the requirements imposed to the Monitoring architecture from the 5G EVE project specifications.
- Finally, Section V concludes the paper and presents our future work.

II. Monitoring Architecture Overview

A thorough analysis of the 5G EVE infrastructure and service requirements [2] results in the following characteristics to be offered by the Monitoring service: (1) the Monitoring distribution architecture must support multi-site experiments involving distant sites; (2) the platform must deal with experiments that may generate monitoring data in the order of gigabytes; (3) that data has to be available to experimenters after the experiment has concluded (estimating a retention time of at least 2 weeks); (4) redundancy is needed to offer a fault-tolerant system; (5) the architecture must be flexible enough to accommodate a wide variety of elements to be monitored; (6) supporting some pre-processing techniques (*e.g.*, translation across formats) may be needed for an efficient subsequent processing; and (7) collected metrics may be used and post-processed by a KPI Validation Framework, which can also distribute the calculated KPIs' and results' values from a specific set of metrics using this platform.

Figure 1 presents, in light blue, the proposed architecture for the collection, distribution and pre-processing of monitoring data that satisfies all the requirements described above. The figure also includes in dark blue some elements of the experiment infrastructure to be monitored, included here for the sake of completeness, and which may be User Equipment devices (UEs), monitoring tools, (4G or 5G) radio antennas, Physical Network Functions (PNFs) or Virtual Network Functions (VNF). Next, all elements of the architecture will be presented following a bottom-up/west-east approach.

The first component of the architecture is the **Metrics Management** entity, whose main role is to properly configure the other components of the architecture, providing the configuration of the necessary data service function chains in order to enable metrics to be gathered, filtered, normalized and relayed to upper layers in the architecture to be further processed. The exact configuration of each component is out of the scope of this paper.

The component of the architecture directly connected to each experiment's infrastructure is the **Metrics Extractor Function (MEF)**,

[1]https://www.5g-eve.eu/

[2]http://5growth.eu/

[3]http://5g-transformer.eu/

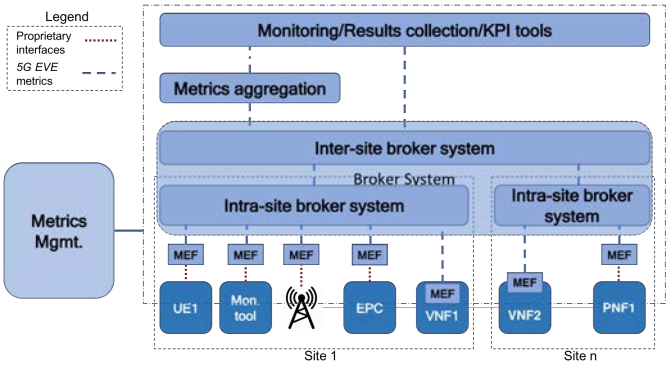

Fig. 1. Monitoring metrics architecture.

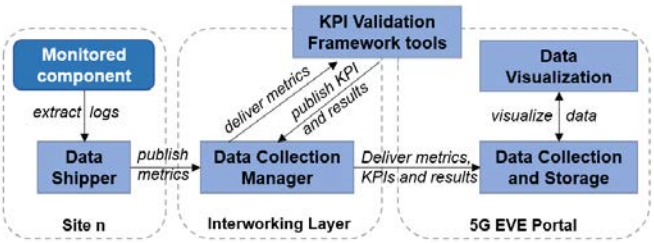

Fig. 2. Component chain that implements the general Monitoring metrics architecture in the 5G EVE platform.

which takes care of extracting and translating (if required) the metrics generated by a heterogeneous set of infrastructure components. In the proposed architecture, it is assumed that there is a one-to-one logical relationship between a particular *MEF* and its monitored infrastructure component, although this may be implemented in different ways, mainly depending on if it is fully, partially or not integrated in the monitored components, as presented in Fig. 1. How to implement the *MEF* for each case is out of the scope of this paper.

This modular design allows to have dedicated *MEFs* per infrastructure device, which satisfies the requirement (5) explained above. This way, it would be possible to implement dedicated *MEFs* to handle any kind of proprietary interfaces (dotted red lines in Fig. 1). Then, the *Metrics Management* entity instructs each *MEF* to extract metrics from its monitored component and to make them available to the upper layer. It is important to remark that all these metrics have to follow the 5G EVE format to satisfy constraint (6) presented before.

The **Broker system** is in charge of storing and distributing not only the metrics obtained from different sites, but also the KPIs' and results' values generated in upper layers. For accomplishing requirement (1), two brokering levels have been defined: the **Intra-site broker**, deployed per site, whose role is to eventually harmonize the metrics' format to provide data in an unified way, preserving the data privacy of each site, and the **Inter-site broker**, which interconnects all sites together to both aggregate metrics through the **Metrics aggregation** component, generating new metrics based on those provided by the *MEFs*, and directly provide them to the different tools grouped in the **Monitoring/Results collection/KPI tools** entity, which lays the ground for a set of value-added additional components that range from the KPI Framework for performance diagnosis already presented, which allows to fulfill requirement (7), to more complex modules such as data analytics platforms, SLA enforcement mechanisms or data visualisation services, which can be fed from the monitoring data provided by the system.

Finally, in order to satisfy requirements (2), (3) and (4), the *Metrics Management* entity is the responsible for properly configuring all levels of the broker system in a per-experiment basis, also enabling the necessary security mechanisms to ensure that only the actors belonging to a given experiment can manage the monitored data of their experiment and not others.

III. Implementation: The Publish-Subscribe Paradigm

The instantiation of the Monitoring architecture presented in Section II over the 5G EVE architecture [4] [5] results in the composition of a specific component chain, depicted in Figure 2. The cornerstone of this chain is the publish-subscribe messaging pattern, providing a distributed system with parallel data processing capabilities which allows to meet the requirements imposed to the Monitoring platform. This paradigm suits the multipoint-to-multipoint monitoring data flow of the 5G EVE project, closer to a big data pipeline rather than to a classic relational database model, as a massive volume of data is pushed from site facilities without a specific format, which is not suitable to be stored in a relational model [6].

Following the above, the *Broker system* is mapped into a set of publish-subscribe queues, starting from local queues deployed in each site facility (*Intra-site broker*) that aggregate metrics to the Interworking publish-subscribe queue (*Inter-site broker*), which provides a transparent and seamless access to metrics', KPIs' and results' values from all sites to components from upper layers. These queues, together with the *Metrics Management* service, are implemented by the **Data Collection Manager** component in 5G EVE architecture, based on *Apache Kafka*[4], an open-source, industry-proven publish-subscribe tool that manages data pipes and forwards the published data to the different components subscribed, providing a higher maximum sustainable throughput than other broker-based message-oriented middleware technologies [7]. This makes *Kafka* an optimal solution for data-movement, frequently adopted as pipe to different processing systems [8].

The information model which defines the different data that is handled by the *Data Collection Manager* in a concurrent way is described in the so-called Topic framework proposal [9]. In that way, each topic is designed to manage a specific set of data (mainly related to a single metric, KPI or result to be monitored) that will be different to the data consumed by the other topics, enabling dataset isolation.

The components that interact with the *Data Collection Manager* can be classified as publishers and subscribers. The main component which performs the metrics' data publishing operation is the **Data shipper**, playing the role of the *MEF* component from the general architecture, transforming the heterogeneous, raw logs obtained from components and collection tools into metrics with a common, homogeneous format. These data shippers can be placed within each component as a lightweight software (ranging from general-purpose solutions already developed and packaged like *Beats*[5] to more complex solutions programmed for specific-purpose cases) or can be in a separated server, but in both cases, they must be connected to the *Data Collection Manager* with a logical connection.

Moreover, the **KPI Validation Framework tools** also contain publishers providing KPIs and results related to a given set of metrics received from the *Data Collection Manager* after being published by specific *Data shippers*, which means that these *KPI tools* also implement a subscriber for each metric to be consumed.

Finally, the *Experiment Monitoring and Maintenance and Results Collection tool* performs the expected functionalities provided by the *Monitoring/Results collection* entities with a solution based on the *Elastic (ELK) Stack*[6] [10]. It is separated in two main blocks [11]: (i) the **Data Collection and Storage** component, which collects the

[4]https://kafka.apache.org/
[5]https://www.elastic.co/beats
[6]https://www.elastic.co/products/

metrics, KPIs and results to which this component is subscribed and provides data persistence, searching and filtering capabilities (related to the *Metrics aggregation* functionality from the general architecture), and (ii) the **Data Visualization** component, in charge of enabling the monitoring of the progress of the experiment in terms of that monitoring data displayed.

IV. Performance Evaluation

To assess and validate the proposed Monitoring framework implementation, we have followed the testing process described below, based on the application of a top-down approach.

A. System assumptions

As a first approach to the evaluation, the following assumptions were made. First, although the Monitoring platform should deal with the simultaneous execution of a considerable amount of experiments, the initial set of use cases defined in 5G EVE is six [2], this being the number of simultaneous experiments to be handled. Each experiment can define a different number of metrics, KPIs and results to be monitored during the experiment execution, depending on vertical's needs. For this evaluation process, we assume that each experiment will require the monitoring of 20 parameters, so each experiment requires the creation of 20 topics in the Monitoring platform.

The size of the message containing the values of metric, KPI or result managed by the Monitoring platform depends on the nature of the data transported. As a result, four different alternatives have been defined for the tests: 100 B and 1 KB messages for data traffic (*i.e.*, numeric or string values), representing the 80% of all the monitoring traffic (40% for each case), and 100 KB and 1 MB messages for multimedia traffic (*i.e.*, images or video frames), which would be the remaining 20% (10% for each case). The percentages have been selected assuming that most of the data will be small-side messages, which fit best in this kind of platforms, but also considering that there may be larger messages, mainly related to multimedia data. The *Data shipper's* publication rate is set to 1000 messages/s for data traffic, but reduced for each case of multimedia traffic, as the received throughput almost never reached that value due to the message size, with 10 messages/s for 100 KB messages and 1 message/s for 1 MB messages, which results in a concurrent publication rate of approximately 102,4 Mbps per experiment.

The rates for each message size are aligned with the disk size estimation for each *broker node*, which is computed as $D = s * r * t * f/b$, where s is the message size, r is the publication rate, t is the retention time (at least 2 weeks, as discussed in Section II), and f and b are both the replication factor and the number of brokers in the system, typically $f = b - 1$, this leading to a value slightly below 100TB, which is an estimation of the expected amount of data handled in the project.

B. Testbed setup

The testbed used for the evaluation of the architecture consists on two Virtual Machines (VMs) deployed in a host located in the 5G EVE Spanish site facility, 5TONIC[7], using *Proxmox*[8] as virtualization environment. This host is equipped with 40 Intel(R) Xeon(R) CPU E5-2630 v4 at 2.20GHz and 128 GB RAM. The proposed scenario intends to simulate the monitoring and data collection process of the metrics, KPIs and results related to a set of experiments where only one VNF (VM#1) is publishing monitoring data in the Monitoring architecture (VM#2). The characteristics and software deployed in each VM, both based on Ubuntu Server 16.04 and executed with 16 virtual CPU cores and 32 GB of RAM, are the following:

- VM#1: data publishers are emulated with *Sangrenel*[9], a *Kafka* cluster load testing tool that allows to configure parameters such as the message/batch sizing and other settings, writing messages to a specific topic and obtaining, as output, the input message rate or the batch write latency, which are the performance parameters under study, being dumped every second.
- VM#2: for emulating the 5G EVE Monitoring architecture, a *Dockerized*[10] environment for testing the 5G EVE Monitoring and Data Collection tools[11] has been used, implementing the *Data Collection Manager*, *Data Collection and Storage* and *Data Visualization* components from Fig. 2 with a solution based on *Apache Kafka* and the *Elastic Stack*. For monitoring the resource consumption of each container, *Docker* native tools (e.g. $docker\ stats$) have been used.

C. Experiments with one topic

To start with the performance analysis of the Monitoring platform, experiments with only one topic created were performed, checking that the system operates correctly and consistently for each message size and publication rate proposed in Section IV-A without limit of resources, and also with the objective of defining the minimum set of computing resources (RAM and vCPU) for the most critical components of the architecture.

It was observed that the resource consumption in the components of the Monitoring architecture is CPU-intensive for the most critical components of the platform, which are *Kafka*, *Logstash* and *Elasticsearch*, leaving the RAM for working as buffer and cache before saving data to disk. As a consequence, this fact facilitates the sizing of these components, as the RAM value can be fixed with a specific value (in this case, with 2 GB of RAM is enough for working properly during the testing process), whereas the CPU value is the only variable term.

In terms of CPU, for a single-topic experiment, *Logstash* is the most critical component, with a consumption that ranges from 100 to 200%, needing 4 vCPU in order not to lose performance. However, the CPU consumption in *Kafka* and *Elasticsearch* stays below 100% for all types of traffic, so 1 vCPU for both of them should be enough to cover single-topic experiments. However, in multi-topic experiments, which will be studied next, *Kafka* becomes the most critical component with a noticeable increase in its CPU consumption, whereas *Logstash* and *Elasticsearch* approximately maintain the same consumption profile.

D. Multi-topic experiments

In multi-topic experiments, the distribution of performance parameter values between topics of the same type (*i.e.*, that handle the same type of data, message size and publication rate) in a given experiment is expected to be uniform in general conditions, where there are no more priority topics than others. This assumption is confirmed in Figure 3 for the batch write latency analysis in one experiment with multiple topics, according to the per-experiment topic distribution described in Section IV-A. As a result, this confirmed assumption is used in subsequent tests for accumulating and averaging the values obtained from performance parameters in topics of the same type, as if they were a single topic, which allows to simplify the performance analysis.

The different tests related to the performance impact assessment for simultaneous multi-topic experiments aim at evaluating two design parameters that causes variations in the Monitoring platform's workload: the number of topics created and running in the system as concurrent processes and the total throughput received by the Monitoring system, calculated as the sum of all input message rates received for each topic. However, a variation in any of these design parameters may cause different effects in the system in terms of CPU consumption or performance that must be characterized, also checking if the superposition property can be applied when both

[7]https://www.5tonic.org/

[8]https://www.proxmox.com/en/

[9]https://github.com/jamiealquiza/sangrenel

[10]https://www.docker.com/

[11]https://github.com/5GEVE/5geve-wp4-monitoring-dockerized-env

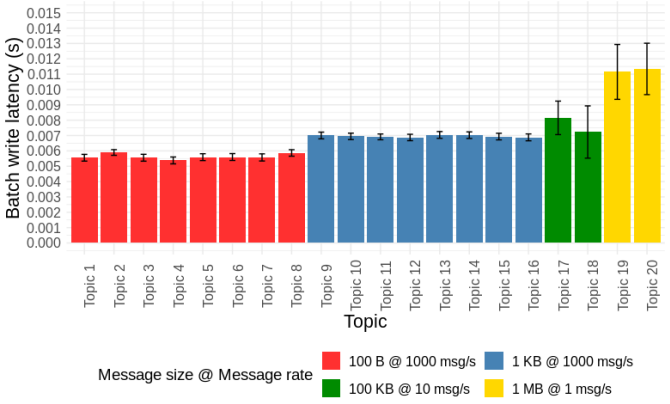

Fig. 3. Batch write latency distribution in one experiment with 20 topics.

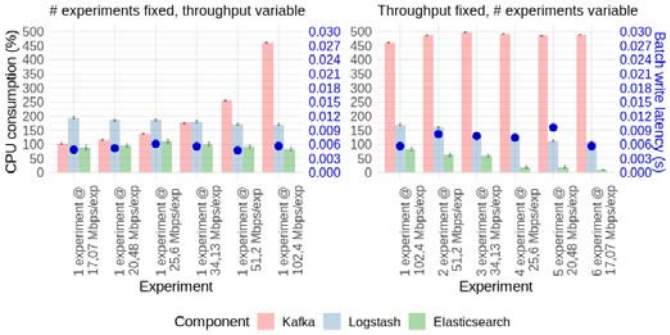

Fig. 4. CPU consumption and batch write latency evolution for 100 B data traffic in different experiments, modifying a different design parameter in each case whereas the other one remains fixed.

parameters are modified simultaneously. This study was divided in two parts: (i) a first analysis where one of the design parameters is modified while the other one stays fixed, and (ii) a final test including the modification of both parameters at the same time, checking if the superposition of individual effects is present.

Case (i) is presented in Figure 4, where the CPU consumption and the batch write latency related to 100 B aggregated data traffic[12] are evaluated for different examples of experiments. On the left side, the number of experiment is fixed in 1, whereas the total throughput is modified, using the theoretical input message rate as upper limit (*i.e.*, 102,4 $Mbps$) and dividing it by values between 1 and 6. However, on the right side, the number of experiments is variable, ranging from 1 to 6, but the total throughput for all experiments is conserved, which is achieved by dividing the message rate aforementioned by the number of experiments deployed.

In both cases, it is observed that the batch write latency does not vary when modifying one of the design parameters, and it is also true for the I/O message rate (*i.e.*, the received throughput divided by the publication rate), which tends to 1. However, in the first case, when the total throughput increases its value, the *Kafka* CPU consumption increases with a trend that seems exponential, but in the second case, the CPU consumption also remains constant in average.

As a result, while the total throughput has an effect in the *Kafka* CPU consumption with an exponential tendency, the number of experiments (*i.e.*, the number of topics in the system) does not seem to influence the system performance, as long as the total throughput

[12]This size is used in the rest of the analysis because it presents a lower value of latency with a tighter 95% confidence interval, according to Fig. 3

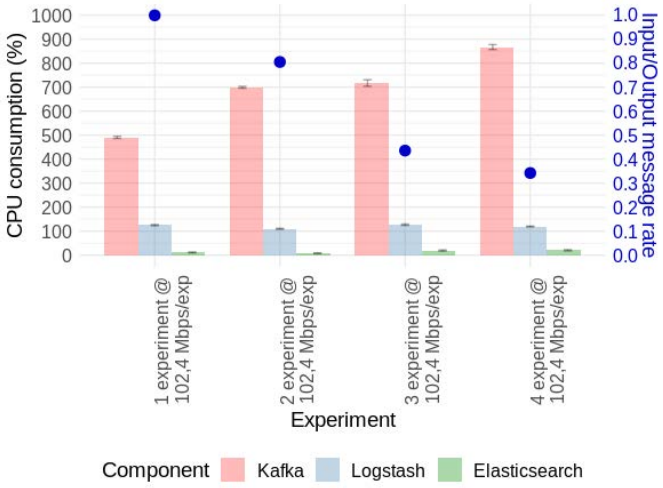

Fig. 5. CPU consumption and I/O message rate evolution for 100 B data traffic in different experiments, modifying both number of experiments and total throughput in all cases.

is conserved when there is an increase in the number of topics, taking care of specifying correctly the publication rate in order not to exceed the system limits. However, this is true while the system is not saturated. When this happens, the effect is similar to the one shown in Figure 5, related to the case (ii).

Here, when the number of experiments increases, the total throughput is also higher, and in fact, it can be noticed that message loss is present from two experiments deployed, as the I/O message rate is nearly 0,8 (so the 20% of messages are lost), and falling until less than 0,4 in the case of four experiments deployed simultaneously, value that remains constant even if more experiments are deployed (these experiments have not been included in Fig. 5 just to present the saturation process with more detail). The evolution of the CPU consumption in *Kafka* is also stopped due to this saturation state, as well as the latency starts to present variations as it is calculated based on the messages that are eventually received.

These results are quite aligned with the outcomes from [12], where it was reported that *Kafka* throughput depends linearly on the number of topics, reaching a hard limit at some specific point. According to this study, when there is only one *Kafka* replica, the limit is reached for around 15000-20000 packets per second, value which is close to our results, as one experiment in our testbed means around 16000 messages per second and the deployment of a second experiment causes a loss of performance, since that limit, which should be between 16000 and 32000 messages per second, is exceeded.

E. System scalability validation

The direct solution to avoid this saturation effect is to build mechanisms and processes that allow system scalability, mainly oriented to the application of horizontal and/or vertical scaling processes depending on the current status of the platform.

For this evaluation process, a preliminary vertical scaling system for this Monitoring platform is proposed, based on the results obtained in the previous tests as training data, used to refine the different cases that can occur in terms of resource consumption (mainly related to CPU) and performance evaluation (mainly based on the batch write latency and the I/O message rate), and the conditions related to each case that trigger the system scale process.

Figure 6 presents an example of vertical scaling for one experiment deployed in the platform. In this case, the *Kafka* container is scaled by increasing its vCPU assignment until the system is able to handle the workload received without saturating, decision that depends on

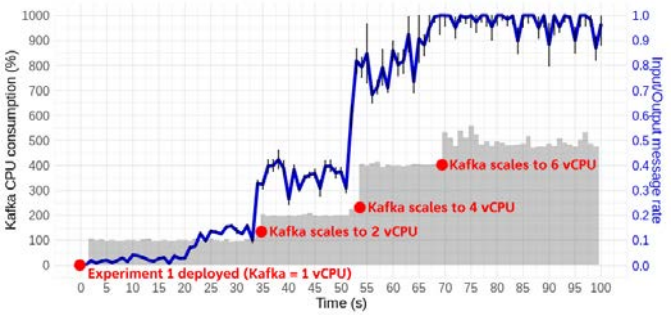

Fig. 6. Evolution of the I/O message rate related to 100 B data traffic in one experiment when vertical scaling mechanisms are enabled.

different parameters, such as, *e.g.*, the current CPU consumption, the delay to compute a KPI or some other performance variable. In our case, for illustrative purposes we trigger an upscale only when a CPU is fully occupied for relatively long periods of times, but more "agile" schemes could be easily implemented.

V. Conclusions and Future Research

To conclude, the results of section IV have revealed some interesting insights related to the Monitoring architecture. The first one is that the distribution of the performance parameter values in topics of the same type is uniform, allowing the aggregation of the performance values obtained for each topic of the same type.

It has been also detected that the total throughput is the parameter that can cause the greatest impact on system performance, with two different possibilities: while the system has enough free resources to work, the CPU consumption tends to increase exponentially, keeping batch write latency and I/O message rate constant. However, when the system is saturated, which seems to happen for a total throughput between 16000 and 32000 packets per second, this exponential growth is stopped and the I/O message rate fails below 0,4% in the worst case. To solve this, a preliminary vertical scaling mechanism has been also proposed, which calculates how many resources are needed for a given workload.

As a consequence of these results obtained during the evaluation process, many topics for future research can be defined. First of all, knowing that the evaluation process presented in Section IV is based on synthetic data, it is expected to repeat the experiments with real values in order to confirm the results obtained, also checking what kind of monitoring traffic is more important for each experiment for fully characterizing each possible experiment. These real monitoring data may come from the different vertical industries involved in the 5G EVE project or from other European projects that are intended to use the 5G EVE platform for validating 5G pilots.

Other future research topics are related to the architecture itself. In fact, the current implemented architecture is based on a very basic *Dockerized* environment, where no component has been built in a clustering or redundant configuration for improving the overall performance of the system. In that way, it is intended to include these kind of techniques to make the Monitoring platform evolve, orienting it towards real production environments. Moreover, this may allow the introduction of horizontal scaling processes, complementary to the current implementation of the vertical scaling system proposed, with the objective of composing a better integrated scaling system of the Monitoring platform that may also include AI and ML techniques for improving the decision-making process.

Furthermore, this framework can also be useful for filling specific gaps declared in 3GPP standards for certain 5G value-added functionalities, so that it can be easily integrated as a complementary module in those cases, with the goal of evolving the Monitoring platform towards a multipurpose framework. This is motivated by the way that Network Functions' (NFs) Service-based interfaces (SBIs) expose their services in the 5G Core Control Plane, based on publish-subscribe mechanisms [13]. Some examples detected and under tracking are the following: (i) data collection framework for data retrieval from Application Functions (AFs) to be processed by the Network Data Analytics Functionality (NWDAF) for enabling Network Automation processes [14], or (ii) inclusion of the publish-subscribe messaging pattern in the communication between management service producers and consumers in a Management and Orchestration (MANO) architecture, being an approach similar to the use of the "subscribe-notify" paradigm proposed in [15].

Acknowledgements

This work was partly funded by the European Commission under the European Union's Horizon 2020 program - grant agreement number 815074 (5G EVE project). The paper solely reflects the views of the authors. The Commission is not responsible for the contents of this paper or any use made thereof.

References

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[2] 5G-EVE, "Requirements definition and analysis from participant vertical industries," Deliverable D1.1, Oct. 2018. [Online]. Available: https://zenodo.org/record/3530391#.XjqwxDJKhIY

[3] ——, "5G EVE end to end reference architecture for vertical industries and core applications," Deliverable D1.3, Dec. 2019. [Online]. Available: https://zenodo.org/record/3628333#.XjqwxDJKhIY

[4] M. Gupta, R. Legouable, M. M. Rosello, M. Cecchi, J. R. Alonso, M. Lorenzo, E. Kosmatos, M. R. Boldi, and G. Carrozzo, "The 5G EVE End-to-End 5G Facility for Extensive Trials," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2019, pp. 1–5.

[5] J. Garcia-Reinoso, M. M. Roselló, E. Kosmatos, G. Landi, G. Bernini, R. Legouable, L. M. Contreras, M. Lorenzo, K. Trichias, and M. Gupta, "The 5G EVE Multi-site Experimental Architecture and Experimentation Workflow," in *2019 IEEE 2nd 5G World Forum (5GWF)*. IEEE, 2019, pp. 335–340.

[6] 5G-EVE, "Interworking Reference Model," Deliverable D3.2, Jun. 2019. [Online]. Available: https://zenodo.org/record/3625689#.XjqwpjJKhIY

[7] P. Sommer, F. Schellroth, M. Fischer, and J. Schlechtendahl, "Message-oriented Middleware for Industrial Production Systems," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Munich, 2018, pp. 1217–1223.

[8] L. Magnoni, "Modern Messaging for Distributed Sytems," *Journal of Physics: Conference Series*, vol. 608, p. 012038, May 2015.

[9] 5G-EVE, "First implementation of the interworking reference model," Deliverable D3.3, Oct. 2019. [Online]. Available: https://zenodo.org/record/3628179#.Xjta48tKg5k

[10] ——, "Experimentation tools and VNF repository," Deliverable D4.1, Oct. 2019. [Online]. Available: https://zenodo.org/record/3628201#.XjqxUDJKhIY

[11] ——, "First version of the experimental portal and service handbook," Deliverable D4.2, Dec. 2019. [Online]. Available: https://zenodo.org/record/3628316#.XjqxVzJKhIY

[12] P. Dobbelaere and K. S. Esmaili, "Kafka versus RabbitMQ: A Comparative Study of Two Industry Reference Publish/Subscribe Implementations: Industry Paper," in *Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems*, ser. DEBS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 227–238.

[13] 3GPP, "System architecture for the 5G System (5GS)," TS 23.501, v16.3.0, Sep. 2019. [Online]. Available: https://www.3gpp.org/DynaReport/23501.htm

[14] ——, "Study of Enablers for Network Automation for 5G," TR 23.791, v16.2.0, Jun 2019. [Online]. Available: https://www.3gpp.org/DynaReport/23791.htm

[15] ——, "Management and orchestration; Architecture framework," TS 28.533, v16.2.0, Dec. 2019. [Online]. Available: https://www.3gpp.org/DynaReport/28533.htm