Benchmarking and Profiling 5G Verticals' Applications: an Industrial IoT Use Case

Anastasios Zafeiropoulos
Ubitech
Athens, Greece
azafeiropoulos@ubitech.eu

Eleni Fotopoulou
Ubitech
Athens, Greece
efotopoulou@ubitech.eu

Manuel Peuster
Paderborn University
Paderborn, Germany
manuel.peuster@upb.de

Stefan Schneider
Paderborn University
Paderborn, Germany
stefan.schneider@upb.de

Panagiotis Gouvas
Ubitech
Athens, Greece
pgouvas@ubitech.eu

Daniel Behnke
Weidmüller Group
Detmold, Germany
daniel.behnke@weidmueller.com

Marcel Müller
Weidmüller Group
Detmold, Germany
marcel.mueller@weidmueller.com

Patrick-Benjamin Bök
Weidmüller Group
Detmold, Germany
patrick-benjamin.boek@weidmueller.com

Panagiotis Trakadas
Synelixis S.A.
Athens, Greece
ptrak@synelixis.com

Panagiotis Karkazis
Synelixis S.A.
Athens, Greece
pkarkazis@synelixis.com

Holger Karl
Paderborn University
Paderborn, Germany
holger.karl@upb.de

***Abstract*—The Industry 4.0 sector is evolving in a tremendous pace by introducing a set of industrial automation mechanisms tightly coupled with the exploitation of Internet of Things (IoT), 5G and Artificial Intelligence (AI) technologies. By combining such emerging technologies, interconnected sensors, instruments, and other industrial devices are networked together with industrial applications, formulating the Industrial IoT (IIoT) and aiming to improve the efficiency and reliability of the deployed applications and provide Quality of Service (QoS) guarantees. However, in a 5G era, efficient, reliable and highly performant applications' provision has to be combined with exploitation of capabilities offered by 5G networks. Optimal usage of the available resources has to be realised, while guaranteeing strict QoS requirements such as high data rates, ultra-low latency and jitter. The first step towards this direction is based on the accurate profiling of vertical industries' applications in terms of resources usage, capacity limits and reliability characteristics. To achieve so, in this paper we provide an integrated methodology and approach for benchmarking and profiling 5G vertical industries' applications. This approach covers the realisation of benchmarking experiments and the extraction of insights based on the analysis of the collected data. Such insights are considered the cornerstones for the development of AI models that can lead to optimal infrastructure usage along with assurance of high QoS provision. The detailed approach is applied in a real IIoT use case, leading to profiling of a set of 5G network functions.**

***Index Terms*—Network Function Virtualization, NFV orchestration, benchmarking, profiling, big data analytics, 5G networks**

I. Introduction

5G networks design and implementation is considered as a key issue to support the introduction of digital technologies in economic and societal processes, leading to the fourth industrial revolution and impacting multiple sectors [1]. The rationale for the development of the 5th generation of mobile communications was not only to expand the broadband capabilities of mobile networks, but also to provide advanced network connectivity, Quality of Service (QoS), security and reliability guarantees for a wide variety of vertical industries. Under this perspective, the integration of verticals is considered as one of the key differentiators between 4G and 5G systems. One of the main distinguishing points in comparison with previous generations is the 5G technologies' strong focus on machine-type communication and the Internet of Things (IoT) [2]. This focus makes 5G technologies very well suited for serving networking requirements in the Industry 4.0, given that Industry 4.0 integrates the IoT technologies within the industrial manufacturing sector [3].

Moving one step further and aiming to cope with the increasing demands of next generation industrial automation, the adoption of Artificial Intelligence (AI) technologies is emerging, leading to a convergence of 5G, AI and IoT mechanisms. Such a convergence provides a set of opportunities for the transition to an Industry 4.0 era, with notable improvements in terms of efficiency, modularity, management, automation and usability of future smart factories across all layers of the production processes. Actionable insights can be produced based on real-time or offline processing of the collected data from the industrial IoT devices, leading to continuous injection of intelligence in 5G orchestration mechanisms. Machine learning mechanisms can be designed and applied, supporting various industrial processes, such as time-critical operations, mobile robotics and predictive maintenance functionalities. By adopting such novel technologies, operating expenses saving of 15-20 percent can be achieved in smart factories, helping production operations to become more flexible, efficient, safer,

978-1-7281-5684-2/20/$31.00 ©2020 IEEE

and cheaper to maintain [4].

Given the aforementioned trends, it can be claimed that various opportunities are arising for designing and developing novel, reliable and efficient Industry 4.0 services. However, as stated by the 5G Alliance for Connected Industries and Automation [3], in order to make the transition from research lab to commercial market, the adoption of 5G technologies in the manufacturing industry has to be combined with sufficient validation from testbed/trial activities. Profiling of vertical industries' applications in terms of consumption of resources, elasticity efficiency, ability to adapt to dynamic network conditions and failures as well as identification of patterns in the overall behaviour of the application, can lead to insights extremely helpful to system administrators and production managers for optimally managing their production processes and automating software development and information technology operations (DevOps) over a 5G enabled infrastructure.

In this paper, we provide an integrated methodology and approach for benchmarking and profiling 5G vertical industries' applications. Focus is given on the interconnection of the proposed benchmarking and profiling tools in a way that can facilitate end users to extract insights with regards to resources usage patterns. A workflow is provided that supports experimentation with each considered application, the aggregation of monitoring data for resource usage and virtual network function (VNF)-specific metrics and the analysis of the aggregated data.

The main novelty of the proposed approach regards the provision of an integrated framework -based on open-source tools- that offers flexibility in software developers and service providers to realise experiments and achieve profiling of their applications in terms of resource and elasticity efficiency. Based on the produced results, proper dimensioning of the applications in terms of resources' usage needs and design of efficient operational policies can be achieved. Furthermore, the provided framework enables data scientists to easily onboard analysis scripts and realise analysis over the collected time series data based on the consumption of open Application Programming Interfaces (APIs). A DevOps approach for applications lifecycle management is adopted, where profiling and feedback produced through experimentation and analysis leads to continuous improvements in the developed software. The proposed approach is applied in a real smart manufacturing use case in the Weidmüller Group that provides products, solutions and services for the Smart Industrial Connectivity and Industrial Internet of Things (IIoT).

II. State of the art and beyond

A. Benchmarking Methodologies and Frameworks

The cloud-computing community introduced several solutions for performance benchmarking of virtualized applications. Most of them focus on solutions to benchmark single-VM applications [5], [6], but some solutions also support complex applications [7]. However, none of them focuses on packet processing elements or specific 5G scenarios. To this end, the Network Function Virtualization (NFV) community started to define benchmarking and test methodologies that allow to collect data that describes how single VNFs or complex Service Function Chains (SFCs) behave under different configurations or in different environments [8], [9].

Based on these methodologies, a series of benchmarking frameworks has been proposed that either focus on benchmarking single VNFs or on evaluating NFV infrastructure deployments [10]–[12]. Others do consider benchmarking of complex SFCs [13] to characterize the performance behavior of end-to-end services, which cannot be derived from isolated VNF benchmarks [14]–[16]. Still, these tools and frameworks suffer from the fact that they provide only limited automation of the benchmarking process or the unavailability of their source code. Such a limitation is tackled in our previous work presented in [14] that provides a first prototype for end-to-end automated NFV benchmarking. This prototype has been turned into an open-source project, called *tng-bench*, which allows to automatically perform benchmarks of NFV components, such as single VNFs or complex SFCs, in a fully automated fashion and in a wide variety of scenarios, such as 5G verticals [17].

In this paper, we utilize this novel tool to apply benchmarking methods to an Industry 4.0 scenario. More specifically, we perform a series of benchmarking experiments in order to mine, to the best of our knowledge, one of the first NFV benchmarking datasets of a realistic 5G smart manufacturing scenario.

B. Profiling Network Applications and Services

Similarly to benchmarking frameworks, lot of work has been realised for the design and implementation of profiling approaches for cloud computing applications, focusing mainly on resource efficiency aspects and optimal infrastructure setup to serve the applications workload [18], [19]. In the 5G world and especially in the NFV community, specifications evolve with regards to standard mechanisms for realising analysis, e.g. in IETF [20] and in the 5G PPP Test, Measurement and KPIs Validation working group [21]. Some VNF profiling approaches like NFV Inspector [16] support profiling and classification of VNFs based on resource capacities and underlying system properties and extract the correlation among QoS metrics and resource utilization of VNFs. Reference [22] proposes a model-based analytics approach for profiling VNF workloads that captures traffic burstiness, while in our previous work [14], we have introduced a fully automated, flexible, and platform-agnostic profiling system that allows to profile entire SFCs at once. In [23], the use of reinforcement learning to promote resilience in Software Defined Networking (SDN) environments is considered. Finally, in [24], a set of NFV workload efficiency quality attributes are specified, including VNF resource efficiency, VNF elasticity efficiency and NFV orchestration efficiency aspects.

In the current work, the profiling approach is based on an analytics engine that we have developed and made available as an open-source project, called *tng-analytics-engine* [25]. The analytics engine permits the easy incorporation and execution of analysis processes, taking advantage of time series data

collected by the realised benchmarks [25]. An integrated framework is provided, combining benchmarking and profiling kits and significantly reducing the overhead of data preparation and configuration of analysis processes. The role of the data scientist for designing and applying novel machine learning mechanisms is highlighted, since their design is totally decoupled from the way that the benchmarking experiments are realised and the experiments results are collected. In this way, newly developed or existing analysis scripts can be easily integrated and supported, without any dependency on the System Under Test (SUT) setup and the workload characteristics.

C. Artificial Intelligence for Smart Manufacturing

Several recent works have highlighted the importance of including AI techniques for smart manufacturing [26]. In this paper, we detail a framework that supports onboarding of machine learning mechanisms in an analytics engine and apply a set of analysis processes in a 5G smart manufacturing use case, which concerns the interconnection and integration of machine data from various manufacturing machines. We use an automated benchmarking approach to generate vast amounts of data, which we then analyze using network service profiling.

To generate big data for analysis purposes in the manufacturing domain, Tao et al. [27] propose to use the so called digital twins. We apply this suggestion and consider digital twins in our use case. Moreover, we go beyond this existing work by applying our automated benchmarking and profiling approach to achieve in-depth analysis insights. It should be noted that the proposed work and architectural framework is generic enough and can be applied in various 5G scenarios, not limited in smart manufacturing verticals. For instance, similar work has taken place for verticals related to immersive media, video and VoIP communications [28].

D. Progress Beyond the State of the Art

The main innovation introduced in this paper is the provision of an integrated framework for benchmarking and profiling of 5G verticals' applications. This framework builds upon our already developed open-source tool for benchmarking, called *tng-bench* [17], and a newly developed open-source tool for profiling, called *tng-analytics-engine* [25]. A workflow is provided that supports the realisation of a set of benchmarking experiments, the collection of the experiments' time-series data, the realisation of various analysis processes over them and the extraction of performance insights. Emphasis is given on the easiness of realising benchmarking experiments and analysis processes, as well as the provision of a set of simple steps for integration of further analysis scripts by data scientists. As already stated, the proposed framework is applied in a real manufacturing use case, aiming at an initial evaluation of the framework's validity and usefuleness for the extraction of performance characteristics of 5G verticals' applications. Such an evaluation process is considered as a proof of concept, where the instantiation of the detailed workflow can guide more advanced experiments and analyses in upcoming works in the future.

III. Smart Manufacturing Use Case

To test and verify the concepts proposed in this paper, we consider a smart manufacturing scenario, which we have initially introduced in our previous work [29]. This use case uses NFV-based services, each consisting of multiple VNFs, to allow the flexible control and monitoring of manufacturing machines, such as injection molding machines. The use of NFV technology complements the envisioned 5G scenarios in which manufacturing machines are interconnected by 5G radio technology deployed within the machine park. By using software-based, e.g., NFV, solutions to deploy and operate the required control and monitoring functionalities as part of the 5G network, quick and automated reconfigurations of the machine parks become possible. For example, in today's manufacturing machine parks, manufacturing machines have to be connected and configured manually. This is time-consuming and error-prone when setting up new machine parks and makes on-demand reconfigurations of production lines, as envisioned in Industry 4.0 scenarios, infeasible.

Together with the Weidmüller group, we developed such a flexible NFV-based system, consisting of multiple network services and VNFs, that can automate large parts of the machine interconnection and configuration. Each of these VNFs can be deployed on top of edge infrastructure, e.g., Kubernetes clusters, directly at the the machine park. Once deployed, the involved network services automatically collect manufacturing data (machine parameters or sensor measurements) from the connected machines, aggregate them, and send them to the company's cloud backend (e.g., Azure IoT Hub). While the analysis at the cloud allows long-term insights and correlation of manufacturing data across machine parks, the system also provides an edge analytics engine (EAE) for real-time analysis of the machine data at the edge within the machine park.

More specifically, we developed two network services (NSs): A *factory edge service (NS1)* and a *machine interconnection service (NS2)*, as shown in Figure 1. NS2 is instantiated once per manufacturing machine and provides a machine data connector VNF (MDC) that handles the interconnection with the machines and periodically collects their manufacturing data. It connects to NS1, which is typically started once per machine park but can be scaled according to the demand using a MQTT broker VNF inside NS1. NS1 further handles the temporary local storage of machine data in a database, their local visualization and analysis in the EAE, and the connection of the company's cloud backend. Figure 1 illustrates this data flow between the involved VNFs which is realised as an on-demand service overlay on top of 5G infrastructure. The used system also contains an intrusion detection system (IDS) as part of NS1 which is a key component for additional security measures preventing malicious traffic to spread within a factory network. Additional VNFs, like a software-based router (RTR), are used for flexible traffic control within the system.

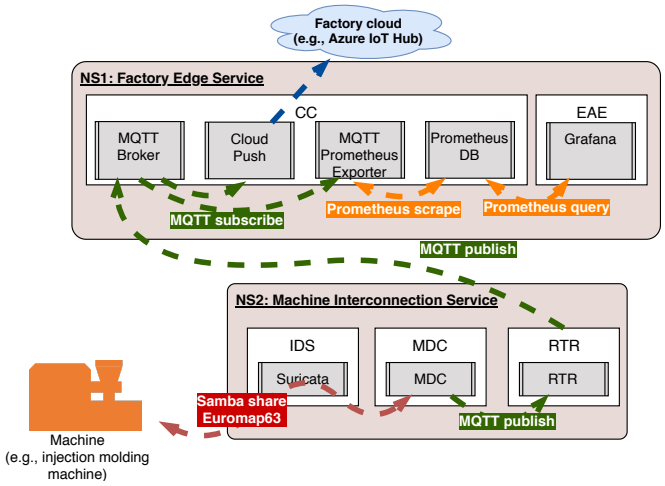

Fig. 1. Smart manufacturing network services and data flow.

Besides optimising the deployment and provisioning of the network services over the 5G infrastructure, guaranteeing certain QoS levels is one of the key challenges for such a software-based smart manufacturing system. Furthermore, the design of machine learning models for forecasting and undertaking of proactive actions (e.g. predictive maintenance) is considered crucial. We address exactly this by performing advanced performance analysis realised over results coming from benchmarking tests in the case study in Section V. Those analyses produce resources efficiency profiles for the involved VNFs, identifying correlation between resources consumption and QoS metrics and -where applicable- provide feedback for the design of effective scaling scenarios. Such results are the first steps towards the design and development of AI-oriented processes applicable to operational environments.

More specifically, we report detailed benchmarking results and the resulting performance models, e.g., efficiency profiles, for the two key VNFs in our presented system, the IDS VNF and the MQTT broker VNF, in the case study presented in Section V. We picked those two VNFs because they are most relevant for the overall system performance as well as for the correct functionality of the system, e.g., if the IDS VNF is not able to capture the complete traffic of the connected machine it might miss intrusions. A performance model, produced by our solution, that focuses on the number of missed IDS packets as main metric can help to avoid such situations. Similarly, performance models of the MQTT broker that give insights into the achieved performance of the broker under different MQTT QoS level configurations (message delivery at most once (value 0), at least once (value 1), exactly once (value 2)), can help to correctly configure and deploy the MQTT VNF to meet the required message rates.

IV. Benchmarking and Profiling Framework

In this section, we describe the proposed integrated benchmarking and profiling approach. Initially, we detail the overall methodology and workflow followed for data aggregation and analysis. Following, the main components of the benchmarking and the profiling tools are presented, focusing on the provided functionalities, the interconnection interfaces, openness, interoperability and usability aspects.

A. Methodology and Analysis Workflow

We consider an end-to-end analysis process for benchmarking and profiling of network services, covering the overall lifecycle of the design, setup and execution of benchmarking experiments, the collection of results in big data repositories and the realisation of big data analysis over them. The following steps are considered, as they are shown in Figure 2. The first step defines the analysis scope, for example, extraction of a performance profile, identification of a bottleneck or capacity limit, verification of achievement of a performance target or dimensioning of the required 5G infrastructure resources for the provision of the network service. As a second step and in accordance with the analysis scope, a set of experiments are defined and executed. Depending on the type of the considered software, different methods are followed to realise experiments, including white, black and grey-box approaches. The SUT definition is taking place and made available as metadata of each experiment. Within the SUT's definition, we consider the assigned resources (compute, storage, disk, network) and the type of the orchestrator to be used for network services deployment and operation. The overall experiment configuration (e.g. number of iterations, test method, automation techniques in terms of DevOps processes, deployment and operational policies) is also specified. This step also defines the stimuli used as input for an experiment, in particular, the input workload characteristics, e.g., parameters for traffic generators or traffic traces used to stimulate the SUT.

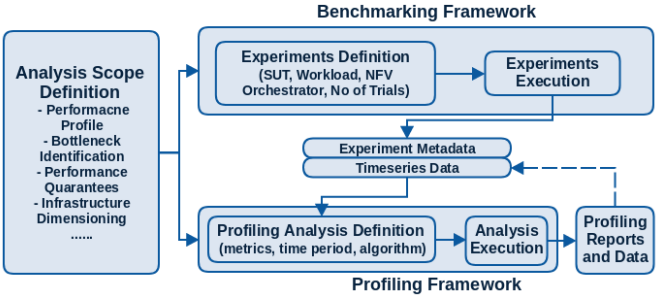

Fig. 2. Benchmarking and profiling workflow.

Following, experiments are executed to collect the intended data. This is called "benchmarking process". During each benchmarking process execution, monitoring data with the experiment results is collected and stored in a time-series database. Such data includes resource usage metrics (e.g. CPU/memory usage, incoming/outgoing traffic), QoS metrics (e.g. jitter, delay, packet loss, throughput), orchestration metrics (e.g. deployment/scaling/reconfiguration time) and metrics associated with the deployed VNFs (e.g. average http requests, average query response time, packet processing time). The

collected time-series data along with the outcome of each experiment constitute the benchmarking process results.

The next step analyses the provided results; it is called "profiling process". A profiling process can be automated and executed upon the end of the experiments, or realised on demand. In the latter case, software developers or service providers are interested on defining the type of the analysis to be executed. The analysis leads to a set of profiling reports related to resources usage efficiency (CPU, memory intensiveness), elasticity efficiency of the VNFs (time and capacity for supporting scaling operations), identification of capacity limits upon stressing the VNFs, identification of metrics whose high correlation is statistically significant, reliability reports taking into account time series data with identified failures, as well as reports regarding behavioural aspects of the VNFs in terms of capability for recovery from failures.

After the analysis, the results are made available to data scientists, application providers and telecom operators for interpretation. Moving one step further, the collected data and results can be used as input towards the training and evaluation of machine learning models, empowering the design of automation mechanisms, considering both automated orchestration mechanisms and vertical applications-oriented automation mechanisms (e.g. industrial automation).

B. Benchmarking

As mentioned in the last section, benchmarking describes the process of running experiments and measurements on a SUT with the goal to collect the required data to understand the behaviour of the SUT. Executing such experiments and measurements manually is not an option, since SUTs (e.g., complex network services) will have many different configurations that need to be tested [14], [30]. And it is especially not an option if an agile, DevOps-based environment is considered where each deployment step between a code commit and the deployment has to be automated.

To this end, the proposed benchmarking framework regards an extension of an open-source tool [14], as it has been implemented in our previous work, that provides a *NFV benchmarking automation framework*. This tool is called "tng-bench" and is shown in Figure 3. Each tng-bench setup consists of two main components: the benchmarker (tng-bench) and one or multiple NFV platforms to execute the actual experiments, i.e., deploy and run the SUTs (VNFs or services). Both components should run in separated environments, e.g., on two separated physical machines, and tng-bench must be able to connect to the execution platform to control and monitor them.

A typical workflow to benchmark a given VNF or NS looks as follows. First, a user (e.g. developer or analyst) specifies a performance experiment description (PED), which is a YAML document that describes the entire experiment, e.g., in terms of configurations to be tested. This PED also references the VNF or service package that contains the SUT. Once this document is created, it is, together with the SUT package, given to tng-bench which then reads it and starts the benchmarking process (s1 in Figure 3). In the next step, tng-bench explores

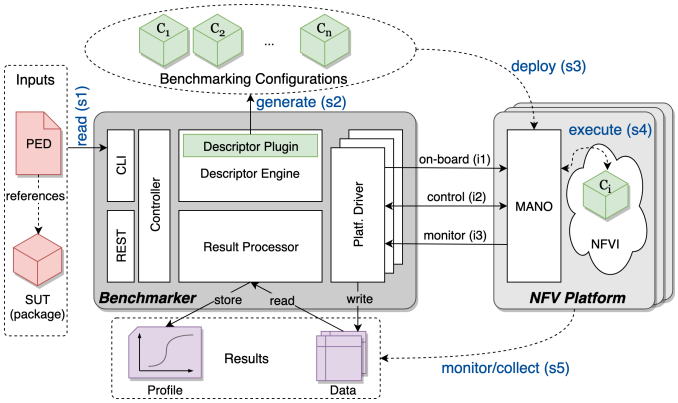

Fig. 3. High-level architecture, artifacts and workflows of the benchmarking framework.

the complete configuration space that should be tested, i.e., it computes the Cartesian product of all configuration options and number of experiment repetitions specified in the PED. Once this is done, the different configurations (which can be thousands) are applied to the descriptors of the PED. For example, new descriptors are generated in which 1 vCPU is assigned to a SUT VNF, another is generated with 2 vCPUs, and so on. In addition, probe VNFs are added to the SUTs, as it is specified in the PED. Those probes can contain, e.g., traffic generators used to stimulate the SUT during the experiments. All these new configurations and probes are then used to generate a series of new VNF or service packages, one for each configuration to be tested (s2).

Once those SUT packages are generated, tng-bench enters the next phase in which it starts to actually on-board those packages on the connected execution platforms and deploy them, one after each other (s3). After a new SUT package is deployed (SUT and probes are instantiated), tng-bench instructs the probes to start the experiment, e.g., to generate traffic. This execution phase runs for a fixed amount of time as defined by the PED, e.g., 10 minutes (s4). During this time, tng-bench collects monitoring data from the execution platforms, SUT, and probes and stores it (s5). Once the experiment reaches its runtime limit, the SUT is terminated and deleted before the next SUT package, containing another configuration, is deployed. This process continuous until every configuration has been deployed and tested and all results and monitoring data is collected. Finally, the resulting data is kept in a connected time-series database from where it can be picked up for the profiling process.

C. Profiling

The profiling process is initiated based on the data from a benchmarking process in the aforementioned time-series database. A set of analysis processes are supported to extract insights such as:

- Resource efficiency analysis for the identification of resource consumption trends and capacity limits, used for planning optimal reservation of resources. The considered

monitored metrics combine a resource usage metric (e.g. CPU usage, memory usage) with a service output metric (e.g. traffic served, HTTP requests served, active users). Such an analysis is realised through the production of (multiple) linear regression models.

- Elasticity efficiency analysis to assess the performance of scaling operations, along with the impact of scaling actions in the service output efficiency (e.g. traffic served by a VNF). Elasticity efficiency is expressed as a pair of discrete metrics, namely application capacity change (incremental capacity change related to a scaling action) and capacity change lead time (time required for a capacity change). Such an analysis is primarily based on monitoring and visualisation of elasticity actions. In a second stage, training and application of machine learning models for automated elasticity actions enforcement is considered by service providers, facilitating the undertaking of proactive elasticity actions for guaranteeing QoS.
- Correlation analysis for the identification of strong and statistically significant correlations among infrastructure and VNF-specific metrics, leading to various insights (e.g. which parameters are highly dependent, which parameters can create bottlenecks in the overall performance). Such an analysis is realised through correlograms.
- Forecasting based on time-series decomposition mechanisms. Such mechanisms are applied over resource usage or workload metrics and provide feedback to elasticity efficiency mechanisms. Various forecasting models are supported based on the type of the time series data.
- Graph analysis for identification of bottlenecks in software functions' calls and the consideration of software updates for optimal service provision. Such an analysis is valuable for software consisted of microservices, where performance issues and bottlenecks due to software functions' calls can be identified and provided as feedback to software developers.

The high level architectural approach of the "Profiling" component is depicted at Figure 4. A primary design principle is the decoupling of the execution of an analysis process from the overall analysis configuration and data management (interfaces i1 and i2). We considered crucial to make it easy to include extra data analysis services, based on analysis scripts provided by data scientists that may cover a diverse set of needs and objectives, as well as be developed in different programming languages (e.g. R, Python). To do so, we introduced the notion of a Proxy that can support – through open APIs – the registration and execution of analysis processes. In the current implementation, two kinds of proxies are supported. Namely, the OpenCPU framework [31] for embedded scientific computing that acts as a middle layer interface to analysis scripts in R and the Flask microframework [32] for analysis scripts in Python. The overall implementation of the "Profiling" component and the set of APIs is based on Java. The developed APIs support the registration of an analysis service, the fetching of the required time series analysis data from the time series data repository, the execution of an analysis service and the provision of the analysis results.

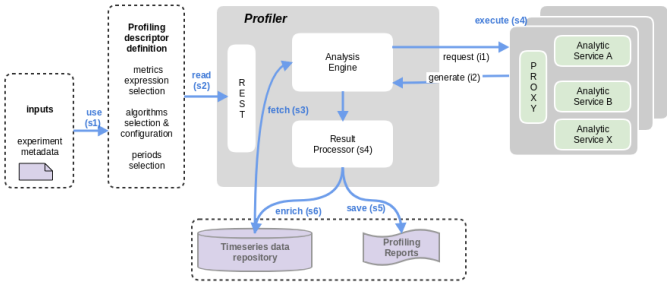

Fig. 4. High-level architecture, artifacts and workflows of the profiling framework.

As already stated, a profiling process is realised over the results made available from a benchmarking process (s1). Given the support of a set of analysis services, the end user is able to define a profiling descriptor (s2) that includes information for the algorithm (analysis script) to be executed, metadata related to configuration of the algorithm execution part (e.g. input dataset, iterations) and the set of monitoring metrics to be considered for the input dataset. It should be noted that at the current implementation status, the supported analysis services include (multiple) linear regression, clustering, time-series decomposition, forecasting and correlation analysis mechanisms. Upon the definition of the profiling descriptor, an analysis may take place (s3, s4), leading to a set of results that are usually made available through a set of URLs (s5). In some cases, the results are also made available (s6) for further processing (e.g. for supporting the increase in the accuracy and reliability of machine learning models). Concurrency in terms of parallel and isolated realization of the analysis processes is ensured by design by the supported proxies, while horizontal scalability is also tackled in cases where big data analysis services can be executed (e.g. big data parallel processing via Spark clusters).

V. Case Study

A. Benchmarking Experiments and Systems under Test

We performed two benchmarking experiments to collect the data needed to evaluate the presented methodology, approach and tools. All experiments have been executed on a testbed based on two machines with Intel(R) Xeon(R) W-2145 CPU at 3.70 GHz CPU, 32 GB of memory, running Linux 4.4.0-142-generic. We used vim-emu [33] as NFV platform connected and controlled by tng-bench. Vim-emu allows to deploy and control NFV scenarios on a single physical machine using Docker containers. The tested VNFs as well as the probes used to stimulate them are deployed as Docker containers, each of them always pinned to its own set of physical CPU cores to achieve isolation between VNFs and probes [14].

In the first experiment, we benchmark the intrusion detection system (IDS) VNF of NS2, as presented in Section III. The VNF is based on Suricata 4.0 [34] and is deployed

in a Docker container. This VNF acts as an example for a typical security appliance used in vertical 5G scenarios. During the experiment, the IDS is stimulated by two different traffic traces, containing normal and malicious traces, taken from [35]. We configure the VNF with a small and a large IDS ruleset taken from [36]. Throughout the experiment, we modify the resource configurations of the VNF container, namely the number of available vCPU cores (1 to 12 cores), to simulate an elastic deployment in which the VNF is vertically scaled. Each configuration is executed for 10 minutes and the data is collected with a frequency of 0.5 Hz. We collect a total of 157 different time-series metrics, ranging from generic metrics, like CPU usage, to application-specific metrics, like the number of matched packets in the IDS resulting in a total of 2.26 million data points in the resulting data set.

The second set of experiments focuses on benchmarking the MQTT broker VNF that is a central part of the use case presented in Section III and commonly used in smart manufacturing scenarios. This VNF is realised by a container running the Mosquitto 1.6.2 [37] broker. To stimulate the broker, we use the MQTT load generator Malaria [38] configured to test the VNF with two different message sizes (10 bytes and 10,000 bytes) as well as with three different MQTT QoS levels (0, 1, 2; 2 corresponds to the highest level). During the experiment, the CPU time available to the VNF container is changed (10 % to 100 % step size 10 %) and each configuration is again tested for 10 minutes. We record 93 different time-series metrics resulting in a total of 1.67 million data points.

B. Profiling Analysis Results

Based on the benchmarking results, we have executed a set of analysis processes, aiming mainly at realising resource efficiency analysis of both VNFs. The overall analysis has been realised by fetching the required data from the collected monitoring metrics in the time-series database and triggering the relevant analysis services on behalf of the Profiler.

In the case of the intrusion detection system (IDS), we have tried to identify the main metrics that affect the performance of the Suricata 4.0 VNF. Initially, we analysed correlation in a subset of the available monitoring metrics for all the experiments with malware traffic and small ruleset, producing the correlogram shown in Figure 5. The subset of the examined metrics include resource usage metrics (e.g. cpu usage, memory usage, incoming/outgoing traffic) and VNF-specific metrics (e.g. detected alerts, examined flows, decoded packets). The main objective is to get insights regarding correlations among metrics that may not be easily observable. High correlation values can lead to further examination of the relationship among the considered metrics and the associated effect on performance aspects.

A set of positive and negative statistically significant correlations are identified between resource usage and VNF-specific metrics. For instance, high positive correlation is identified between the CPU usage and the traffic served by the VNF, while small negative correlation is identified between the memory usage and the number of produced alerts. Such an

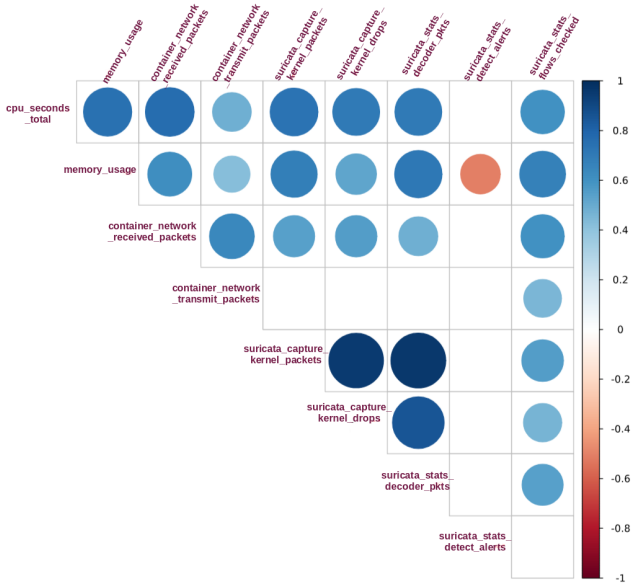

Fig. 5. Correlogram with selected Suricata monitoring metrics.

insight led us to examine the relevant relationships based on linear regression models. A series of linear regression analysis processes has shown that Suricata 4.0 is a CPU-intensive VNF as a function of the served network traffic (e.g. see Figure 6 for the experiments with 1 CPU core and big ruleset). The type of the traffic – whether it is malware or normal traffic – does not seem to significantly affect CPU usage. However, it seems to affect memory usage, since in case of malware traffic, higher memory usage (600MB on average) is noticed, even with much smaller amount of network traffic compared to the normal traffic trace (e.g. see Figure 7 for the experiments with 1 CPU core and big ruleset). Higher memory usage is also noticed in case of enforcement of the big ruleset (200MB on average). Malware traffic refers to values from 0 to 40 Mbytes per second, while normal traffic refers to values from 60 to 120 Mbytes per second in both Figure 6 and Figure 7. With regards to the allocation of multiple CPU cores, it seems that there is no need for allocation of more than 2 CPU cores for serving the considered traffic. The total CPU usage, considering the availability of 12 CPU cores, is depicted at Figure 8. Such insights are useful for planning computational resources allocation for the specific VNF in the real deployment of the considered use case in the Weidmüller Group.

A similar process is followed for the MQTT broker, where we have tried to identify the main metrics that affect the performance of the Mosquitto 1.6.2 VNF. Mosquitto 1.6.2 turned to be a memory intensive VNF with regards to the messages served, while no significant effect is noticed for the CPU usage. In Figure 9, the relationship between the memory usage and the messages sent by the Mosquitto 1.6.2 VNF is depicted, for the experiments with small size and QoS equal to 2 that corresponds to the highest QoS level. A strong and statistical significant linear relationship between the two

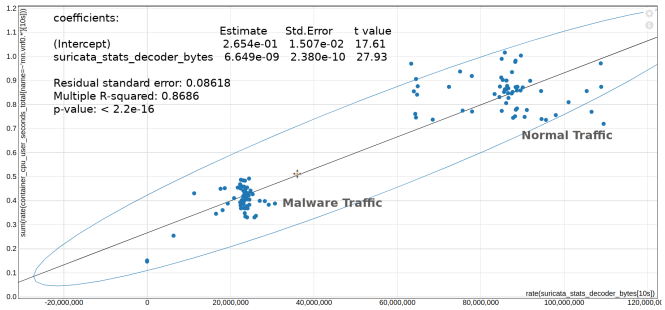

Fig. 6. Linear Regression: CPU Usage vs Decoder Bytes (Suricata VNF).

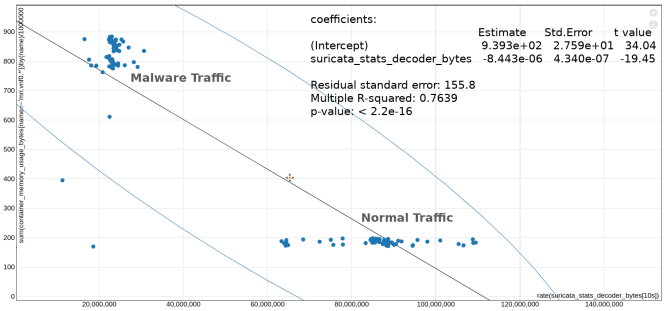

Fig. 7. Linear Regression: Memory Usage vs Decoder Bytes (Suricata VNF).

Fig. 8. CPU Usage of the Suricata VNF.

metrics is identified. Such a relationship is stronger as the QoS level increases. For instance, higher trend and correlation between memory usage and packets served is noticed in case of QoS value 2 and small packet size (adjusted Rsquared value 0.9 for QoS 2, 0.55 for QoS 1 and 0.69 for QoS 0). Dropped packets are also noticed only in case of QoS 2, notifying us that more resources have to be assigned to the deployed VNF. On the contrary, even if the the CPU time available to the VNF container is changed (10% to 100% with step size 10%), it seems that the maximum usage is close to 40%, providing an insight that the provided resources can easily accommodate the provided workloads (e.g. see Figure 10).

This smart manufacturing case study clearly demonstrates how our proposed integrated benchmarking and profiling methodology can be used to get detailed insights with regards to the resource consumption trends of VNFs and NSs. These insights lead to appropriate dimensioning of the reserved 5G infrastructure as well as to the preparation of deployment and runtime elasticity policies that can assure the provision of the required network services performance.

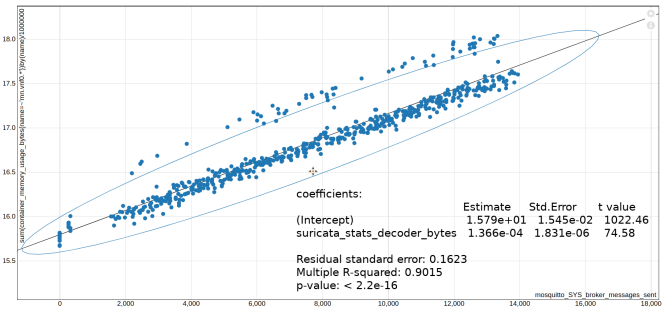

Fig. 9. Linear Regression: Memory Usage vs Broker Messages Sent (Mosquitto VNF).

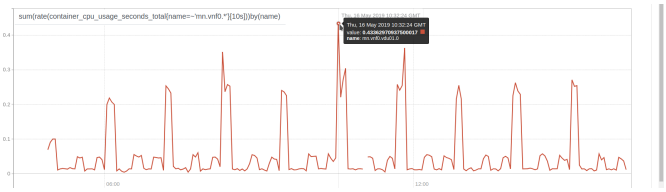

Fig. 10. CPU Usage of the Mosquitto VNF.

C. Open data sets

We make the raw measurement data that has been produced by our experiments and used for the presented evaluations available as open data sets. This allows the community to utilize our results for their own research, e.g., to test novel big data analysis approaches for the 5G domain. All data sets are published as part of our SNDZoo project [39] under the CC-BY-SA 4.0 license.

VI. Conclusions

In this paper, we detailed an integrated solution for benchmarking and profiling 5G-oriented vertical applications, along with a use case applied in a smart manufacturing industry. The proposed workflow combines the design and execution of benchmarking experiments, along with the analysis of the collected data for extracting insights. In the presented case study, the proposed approach is applied, leading to a set of resource efficiency outcomes for two VNFs that are part of the deployed network services. These outcomes helped us to prepare optimal deployment plans in the smart factory setting, as well as to provide feedback for the design of runtime policies to automatically and proactively allocate resources in case of high workloads, considering the identified capacity limits.

In our next steps, we plan to work on the development and evaluation of reinforcement learning models that can support automated management of scaling actions of VNFs as well as further orchestration actions, taking as input results from resource efficiency analysis. To achieve so, a definition of a Markov Decision Process will take place representing the performance goals of the orchestration environment, the available actions for horizontal and vertical scaling and the rewards that will guide a set of agents to predict on real

time the optimal action to support. Furthermore, in the smart manufacturing use case, we are going to apply time-series decomposition and forecasting mechanisms over operational data, aiming to get alerts for upcoming events that can be used for proactive decision making.

ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation programmes under grant agreement No. H2020-ICT-2016-2 761493 (5GTANGO) and grant agreement No. H2020-ICT-2016-2 761898 (MATILDA), and the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901).

REFERENCES

[1] Vision Papers and Roadmaps, 5G empowering vertical industries - 5G-PPP. https://5g-ppp.eu/roadmaps/ (accessed Apr. 04, 2020).

[2] Making Industry 4.0 Happen, 5G ACIA Report for Connected Industries and Automation. https://www.5g-acia.org/index.php?id=5050 (accessed Apr. 04, 2020).

[3] 5G for connected industries and automation, 5G ACIA Report for Connected Industries and Automation. https://www.5g-acia.org/index.php?id=5125 (accessed Apr. 04, 2020).

[4] 5G business potential report, Ericsson, Oct. 04, 2019. https://www.ericsson.com/en/5g/forms/5gforbusiness-2019-report (accessed Apr. 04, 2020).

[5] T. Wood, L. Cherkasova, K. Ozonat and P. Shenoy, "Profiling and Modeling Resource Usage of Virtualized Applications," in Middleware 2008, Berlin, Heidelberg, 2008, pp. 366–387.

[6] J. Taheri, A. Y. Zomaya and A. Kassler, "vmBBThrPred: A Black-Box Throughput Predictor for Virtual Machines in Cloud Environments," in Service-Oriented and Cloud Computing, Cham, 2016, pp. 18–33.

[7] B. C. Tak, C. Tang, H. Huang and L. Wang, "PseudoApp: Performance prediction for application migration to cloud," in 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), May 2013, pp. 303–310.

[8] Network Functions Virtualization (NFV); Pre-deployment Testing; Report on Validation of NFV Environments and Services, ETSI GS NFV-TST 001, 2016.

[9] R. V. Rosa, C. E. Rothenberg, M. Peuster and H. Karl, "Methodology for VNF Benchmarking Automation," IETF, Internet-Draft, Jul. 2018, work in Progress. https://datatracker.ietf.org/doc/draft-rosa-bmwg-vnfbench/ (accessed Apr. 04, 2020).

[10] R. V. Rosa, C. E. Rothenberg and R. Szabo, "VBaaS: VNF benchmark-as-a-service," in 2015 Fourth European Workshop on Software Defined Networks. IEEE, 2015, pp. 79–84.

[11] M. Baldi and A. Sapio, "A network function modeling approach for performance estimation," in Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2015 IEEE 1st International Forum on. IEEE, 2015, pp. 527–533.

[12] R. V. Rosa, C. Bertoldo and C. E. Rothenberg, "Take Your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking," IEEE Communications Magazine, vol. 55, no. 9, pp. 110–117, 2017.

[13] L. Cao, P. Sharma, S. Fahmy and V. Saxena, "NFV-VITAL: A Framework for Characterizing the Performance of Virtual Network Functions," in Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on. IEEE, 2015, pp. 93–99.

[14] M. Peuster and H. Karl, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments," in 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017.

[15] J. Nam, J. Seo and S. Shin, "Probius: Automated approach for vnf and service chain analysis in software-defined nfv," in Proceedings

[16] M. G. Khan, S. Bastani, J. Taheri, A. Kassler and S. Deng, "Nfv-inspector: A systematic approach to profile and analyze virtual network functions," in 2018 IEEE 7th International Conference on Cloud Networking (CloudNet). IEEE, 2018, pp. 1–7.

[17] M. Peuster. tng-bench: Automated Benchmarking of NFV Scenarios. https://github.com/sonata-nfv/tng-sdk-benchmark (accessed Apr. 04, 2020).

[18] W. J. Lloyd, S. Pallickara, O. David, M. Arabi, T. Wible, J. Ditty and K. Rojas, "Demystifying the clouds: Harnessing resource utilization models for cost effective infrastructure alternatives," IEEE Transactions on Cloud Computing, vol. 5, no. 4, pp. 667–680, Oct 2017.

[19] C. B. Hauser and S. Wesner, "Reviewing cloud monitoring: Towards cloud resource profiling," in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), July 2018, pp. 678–685.

[20] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure," IETF Internet-Draft, AT&T Labs, Internet-Draft, 2016. https://tools.ietf.org/html/draft-ietf-bmwg-virtual-net-03 (accessed Apr. 04, 2020).

[21] 5G PPP, "5g ppp working groups," https://5g-ppp.eu/5g-ppp-work-groups/, 2019.

[22] R. Bruschi, F. Davoli, P. Lago and J. F. Pajo, "Model-based analytics for profiling workloads in virtual network functions," in 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), May 2017, pp. 916–921.

[23] L. S. R. Sampaio, P. H. A. Faustini, A. S. Silva, L. Z. Granville and A. Schaeffer-Filho, "Using nfv and reinforcement learning for anomalies detection and mitigation in sdn," in 2018 IEEE Symposium on Computers and Communications (ISCC), June 2018, pp. 00 432–00 437.

[24] Q. Forum, "Nfv workload efficiency whitepaper," https://tl9000.org/ (accessed Apr. 04, 2020).

[25] E. Fotopoulou and A. Zafeiropoulos. tng-analytics-engine: Profiler for 5G Network Services and Applications. https://github.com/sonata-nfv/tng-analytics-engine (accessed Apr. 04, 2020)..

[26] N. Papakostas, J. O'Connor and G. Byrne, "Internet of things technologies in manufacturing: Application areas, challenges and outlook," in 2016 International Conference on Information Society (i-Society). IEEE, 2016, pp. 126–131.

[27] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," The International Journal of Advanced Manufacturing Technology, vol. 94, no. 9-12, pp. 3563–3576, 2018.

[28] 5GTANGO Consortium. D7.3 Final demonstrators and evaluation report. https://www.5gtango.eu/project-outcomes/deliverables/78-d7-3-final-demonstrators-and-evaluation-report.html (accessed Apr. 04, 2020).

[29] S. Schneider, M. Peuster, D. Behnke, M. Müller, P. Bök and H. Karl, Putting 5G into Production: Realizing a Smart Manufacturing Vertical Scenario. European Conference on Networks and Communications (EuCNC'19).

[30] M. Peuster and H. Karl, "Understand Your Chains and Keep Your Deadlines: Introducing Time-constrained Profiling for NFV," in 2018 14th International Conference on Network and Service Management (CNSM), Nov 2018, pp. 240–246.

[31] OpenCPU. OpenCPU system for embedded scientific computing. https://www.opencpu.org/ (accessed Apr. 04, 2020).

[32] Armin Ronacher. Flask is a microframework for Python. http://flask.pocoo.org/ (accessed Apr. 04, 2020).

[33] M. Peuster, H. Karl and S. van Rossem, "MeDICINE: Rapid Prototyping of Production-ready Network Services in Multi-PoP Environments," in 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016, pp. 148–153.

[34] OISFoundation. Suricata: Open Source IDS / IPS / NSM engine. https://suricata-ids.org/ (accessed Apr. 04, 2020).

[35] S. Garcia, M. Grill, J. Stiborek and A. Zunino, "An empirical comparison of botnet detection methods," Computers & Security, vol. 45, pp. 100 – 123, 2014.

[36] ET Labs. Emerging Threats Open Ruleset. https://rules.emergingthreats.net/ (accessed Apr. 04, 2020).

[37] Eclipse Foundation. Eclipse Mosquitto: An open source MQTT broker. https://mosquitto.org/ (accessed Apr. 04, 2020).

[38] Malaria Project. Malaria: Attacking MQTT systems with Mosquittos. https://github.com/etactica/mqtt-malaria (accessed Apr. 04, 2020).

[39] M. Peuster, S. Schneider and H. Karl, The Softwarised Network Data Zoo. arXiv preprint arXiv:1905.04962, 2019.