UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Claudio Roberto Bertoldo Junior

# Avaliação Comparativa de Virtualização de Função de Rede: Análise de Desempenho de um Subsistema Multimídia IP com o Framework Gym

***Network Function Virtualization Benchmarking: Performance Evaluation of an IP Multimedia Subsystem with the Gym Framework***

Campinas

2017

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Claudio Roberto Bertoldo Junior

# Avaliação Comparativa de Virtualização de Função de Rede: Análise de Desempenho de um Subsistema Multimídia IP com o Framework Gym

***Network Function Virtualization Benchmarking: Performance Evaluation of an IP Multimedia Subsystem with the Gym Framework***

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia de Computação.

Supervisor: Prof. Dr. Christian Rodolfo Esteve Rothenberg

Campinas
2017

**Agência(s) de fomento e nº(s) de processo(s):** Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

Bertoldo Junior, Claudio Roberto, 1987-
B462n Network Function Virtualization benchmarking : performance evaluation of an IP Multimedia Subsystem with the Gym Framework / Claudio Roberto Bertoldo Junior. – Campinas, SP : [s.n.], 2017.

Orientador: Christian Rodolfo Esteve Rothenberg.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Virtualização. 2. Redes. 3. Desempenho. 4. Multimídia (Computação). I. Esteve Rothenberg, Christian Rodolfo,1982-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

**Título em outro idioma:** Avaliação comparativa de Virtualização de Função de Rede : análise de desempenho de um Subsistema Multimídia IP com o Framework Gym
**Palavras-chave em inglês:**
Virtualization
Networking
Performance
Multimedia (Computing)
**Área de concentração:** Engenharia de Computação
**Titulação:** Mestre em Engenharia Elétrica
**Banca examinadora:**
Christian Rodolfo Esteve Rothenberg [Orientador]
Mateus Augusto Silva Santos
José Raimundo de Oliveira
**Data de defesa:** 16-05-2017
**Programa de Pós-Graduação:** Engenharia Elétrica

## COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

**Candidato:** Claudio Roberto Bertoldo Junior RA: 140999
**Data da Defesa:** 16 de maio de 2017

**Título da Tese:** "Avaliação Comparativa de Virtualização de Função de Rede: Análise de Desempenho de um Subsistema Multimídia IP com o Framework Gym".

Prof. Dr. Christian Rodolfo Esteve Rothenberg (Presidente, FEEC/UNICAMP)
Dr. Mateus Augusto Silva Santos (ERICSSON)
Prof. Dr. José Raimundo de Oliveira (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

*Dedico esta dissertação aos meus pais Claudio (in memoriam) e Maria José, a minha irmã Caroline e a minha esposa Thaynara.*

# Acknowledgements

The completion of this work could not have been possible without the unconditional support of several important people in my personal and academic lives.

I would like to thank my advisor Prof. Dr. Christian Esteve Rothenberg from the Faculty of Electrical and Computer Engineering (FEEC) at University of Campinas (UNICAMP). Even during weekends and holidays, always solicitous and ready to help and give sincerely feedback about my activities and doubts.

I would like to thank my research group fellow at INTRIG (Information & Networking Technologies Research & Innovation Group) at University of Campinas, Doctorate Raphael Vicente Rosa, for the development of the framework widely exploited in this work and for the several feedbacks and concerns about this work development as a whole.

I would like to thank all INTRIG members (INTRIGERS) for their support during moments of doubts of a fresh researcher, for their reception, for the support for frequent troubleshooting helps regarding our cloud and network infrastructures and also for the moments of fun and friendship. This work was partially supported by FAPESP grant #14/18482-4 and by the Innovation Center, Ericsson S.A., Brazil, grant UNI.58.

I would like to thank my wife Thaynara Felicori for accepting and understanding my absence with her during most part of this work development, including nights, weekends and holidays. My goal has become her goal as well and without her support with all other daily tasks I could not accomplish this work.

Last but not the least, I would like to thank my mother Maria José Lopes and my father Claudio Bertoldo (in memoriam) and to my sister Caroline Bertoldo for supporting me throughout this work and for always cheering for my professional and personal success. I lost my father during the development of this master's program and its conclusion is then dedicated to him.

# Abstract

The trend towards Network Functions Virtualization (NFV) calls for new frameworks allowing rich performance evaluation of the Network Functions Virtualization Infrastructures (NFVIs) during the pre-deployment and run-time phases. A common concern among users, such as Virtualized Network Function (VNF) developers/integrators and NFVI/Virtualized Network Function-as-a-Service (VNFaaS) providers is the effective correspondence of given VNF and its target NFVI regarding resource allocations, i.e., required/negotiated Service Level Agreement (SLA) versus available resources/infrastructure. A quote by Irish physicist William Thomson summarizes the proposition in this scenario: "If you can not measure it, you can not improve it." –nor choose the best for some specific measurable objectives. This work aims at contributing to the development of a so-called Gym Framework integrated into a larger NFV architecture. The focus of this work is on developing components suitable to evaluate physical and virtual machine performance in heterogeneous software/hardware platforms. The use case scenario for the proof of concept evaluation is a mobile telecom network virtual IP Multimedia Subsystem (IMS), a widely deployed functions chain that enables voice call service over Internet Protocol (IP), among other multimedia services.

**Keywords**: Network Functions Virtualization; Benchmarking; Performance.

# Resumo

A tendência em direção à Virtualização de Funções de Rede (NFV) exige novas ferramentas que permitem uma profunda avaliação de desempenho das Infraestruturas de Virtualização de Funções de Rede (NFVIs) durante as fases de pré-implantação e de execução. Uma preocupação comum entre os usuários, como desenvolvedores/integradores de Função Virtualizada de Rede (VNF) e provedores de NFVI/Função Virtualizada de Rede-como-um-Serviço (VNFaaS), é a correspondência eficaz de determinada VNF e sua NFVI-alvo, no que diz respeito à alocação de recursos, i.e., entre o Acordo de Nível de Serviço (SLA) exigido/negociado versus os recursos/infra-estrutura disponíveis. Uma citação do físico irlandês William Thomson resume a proposição neste cenário: "Se você não puder medir algo, não pode melhorá-lo", nem escolher o melhor para alguns objetivos específicos mensuráveis. Este trabalho visa contribuir com o desenvolvimento do chamado Framework Gym, integrada numa arquitetura NFV mais ampla. O foco deste trabalho é o desenvolvimento de componentes adequados para avaliar o desempenho da máquina física e virtual em plataformas heterogêneos de software/hardware. O cenário do caso de uso, tendo em vista a prova de conceito, é um Subsistema Multimídia IP (IMS) virtualizado para redes móveis de telecom: uma cadeia de funções amplamente implantada que permite o serviço de chamadas de voz através do Protocolo de Internet (IP), entre outros serviços multimídia.

**Palavras-chave**: Virtualização de Funções de Rede; Avaliação Comparativa; Desempenho.

*“If you can not measure it, you can not improve it.”*
*(William Thomson)*

# List of Figures

# List of Tables

# Contents

# 1 Introduction

Network Functions Virtualization (NFV) (ROSA *et al.*, 2014) is happening due to the service providers' needs in creating an open to innovation and cost-effective environment for implementing their services upon their network functions. One of the main concern about implementing VNF is its performance. In public clouds, for instance, is possible to set various entities, as virtual CPU, memory, disk space, network logical topology, firewall rules, but is difficult to have a thorough control of network performance, although the cloud providers make available options for the instances network quality, between pessimistic and optimistic scenarios. Thus, in business models based on Infrastructure-as-a-Service (IaaS), where there is a "locked down" environment (SOBEL *et al.*, 2005), a detailed study/performance assessment is needed to forecast how a particular function will behave in this environment, often considered dynamic and closed. The question is to assess whether public clouds, or even private ones, are carrier-grade-ready or at least allow customization/optimization for that. Through Rimal *et al.* (2009), for instance, is possible to better understand this "locked down" environment of the most popular public cloud infrastructures.

Existing related works are limited to performance evaluation and improvement of generic clouds, many times focused only on web applications (see Sec. 2.2) and having no integration with upper components, such as the Orchestrator in NFV-context. The intention is the same: evaluate infrastructure and/or application performance in order to optimize it or support decision.

This work, in addition to others in NFV field and specifically synergistic to the Next Generation Mobile Networks (NGMNs), moves on together with the conceptual evolution of the technology. Within this context, some challenges are characterized concerning the reliability of the model, due to its prematurity. The proposal aims to address the problems of the "low-developed" and open-to-innovation environment faced by Mobile Network Operators (MNOs), particularly with regard to development agility and functions implementation (deploy and test). If successful, it will contribute to the demystification of the proposed model and to the evolution of the model acceptance and adoption by MNOs worldwide. In this field, the efforts and initiatives of the Industry Specification Group (ISG) for NFV of the European Telecommunication Standard Institute (ETSI)[1] can also be highlighted. Initiatives related to the Gym framework, former VBaaS, introduced by Rosa *et al.* (2015), and the Use Case (UC)/Proof of Concept (PoC) itself

[1] http://www.etsi.org/technologies-clusters/technologies/nfv

deserve attention and will be treated in the summarized literature review. More specifically, this work addresses an issue that is currently being discussed in the academy and industry: the performance of general-purpose applications and network functions running over public and private cloud, with a view to Quality of Experience (QoE), compliance of required/negotiated SLAs and cost reduction with computing and networking resources, i.e., Capital Expenditure (CapEx) and Operational Expenditure (OpEx).

## 1.1 Problem Statement

In this context, there is a need for specific frameworks for evaluating virtualized environments for network functions. This is understandable due to the maturity of the technology and the key metrics definitions moment for assessment and improvement/optimization, still ongoing by the relevant authorities, e.g. ETSI . Generally, this assessment seeks to correlate the results obtained in a full or paravirtualized, i.e., a comparison between different levels of virtualization overhead, environment with those known from traditional physical domain (baseline). The benefits of knowing that, concerning optimization initiatives and decision support, are numerous. Starting from the premise that the requested workload is estimated during the network design phase, Solution Architects have the task of designing the infrastructure to support such traffic.

The implementation of a VNF to support legacy traffic of a function previously implemented into a black box, i.e., the “physical-virtualized migration”, where the values of all its Key Performance Indicators (KPIs) at a given time are known, such as throughput effectively supported, number of packets processed by unit of time and maximum concurrent sessions, is an example of activity that requires such study, i.e., prior knowledge of the target infrastructure properties. In short, it is possible to determine through measurements and comparisons whether a given public or private infrastructure is able to meet the demands imposed by service providers and VNF Independent Software Vendors (ISVs).

## 1.2 Value Proposition

The carriers’ dilemma in selecting VNF deployment over private or public cloud involves many economic and geographic factors, as well those related to performance, availability and security. In the case of performance, considering that carriers will deploy their own cloud (private), so what is the purpose of performance assessment, whether, by strategic decision, there will not be at least one peer for comparison? The search for optimization in software-level is the answer.

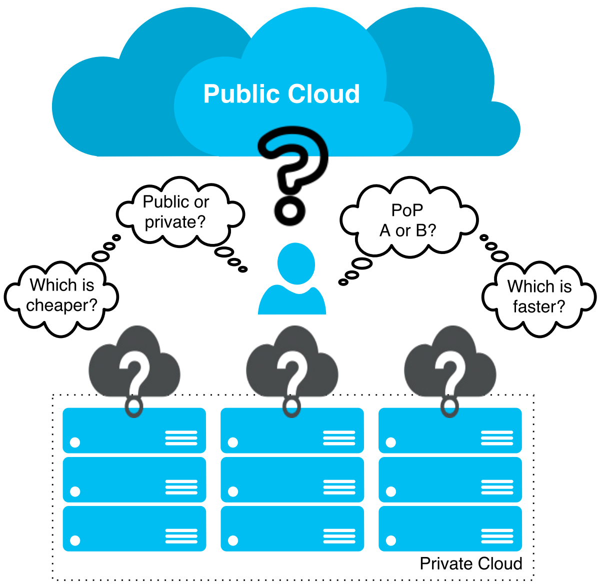


Figure 1.1 – Developer/VNF-Provider and NFVI-Provider's Challenge.

Thus, the value proposition of this work is to contribute to an under-development benchmarking framework for NFVI (public and private clouds) and VNF assessment in order to bring:

**VNF-NFVI Correspondence**

Knowledge of the correspondence between a given VNF and its target NFVI, concerning resources allocation and the required/negotiated SLA during the pre-VNF deployment phase, i.e., the verification during the benchmarking whether an infrastructure meets the VNF performance requirement;

**Identification of Optimization Triggers**

Identification of available platform optimization features in virtual processing and networking level, in private clouds mainly, based on obtained values and compared with baselines, enabling the conclusion whether there is or not such optimization(s) implemented, e.g. shorter Virtual Machine (VM) CPU time slices in the context of vCPU scheduling in Xen[2]/physical CPU sharing (SHEA *et al.*, 2014); and Transmission Control Protocol (TCP) performance improvement with Xen through TCP acknowledgments offloading to the driver domain (KANGARLOU *et al.*, 2010);

**Baselining**

In order to support decision about choosing between the private cloud implementation and the public cloud rental or even the best Point of Presence (PoP), rack or node, complementing benchmarkings based exclusively on economic parameters,

[2] http://www.xenproject.org/

as shown on Figure 1.1. Hereupon it is important to highlight the inflexibility of public infrastructures for optimization, that can be configured as an entry barrier for cloud provider in the NFV market;

**Benchmarking**

In order to support decision about choosing the best NFVI provider, inside the "test before deploy" context, also complementing benchmarkings based exclusively on economic parameters. It is expected to justify values discrepancies obtained through the underpinning environment characteristics. An example of it is the employment of Xen by Amazon, where its default settings, theoretically immutable, of vCPU scheduling and TCP acknowledgments not being offloaded to the driver domain (KANGARLOU *et al.*, 2010), do not favor its choice as NFVI for "demanding" functions;

**Fault Tolerance**

Support for NFVI fault tolerance mechanisms, generally implemented in the NFV-Management and Orchestration (MANO) component, that correlates performance degradation, such as low throughput and high latency, with failures. It is related to a probe role and not related to an "pre-deployment" benchmarking, but the proposed application can be easily extended to it.

## 1.3 Expected Results

The expected results of this work are:

1. Development of a suitable methodology for NFVI performance assessment: generic metrics related to computation, network, memory and storage and also evaluation of services offered by a particular VNF, i.e, the specific metrics;

2. Design of an architecture, based on the methodology of the previous topic, and implementation of a framework for NFVI and VNF assessment, contributing the Gym framework project. The main idea is to automate the tasks of provisioning and testing/benchmarking different infrastructures and create a knowledge plan that supports decision-making;

3. Execution of performance benchmarkings: data collection and comparison through the developed methodology and framework (PoC), over private cloud (deployed testbed), correlating the obtained outcomes for NFVI and VNF metrics and then comparing between different virtualized infrastructures/instances, thus, identifying

gaps related to performance and determining relevant trade-offs that can trigger optimization, scalability actions and searches for infrastructures with greater capacity and/or better performance;

4. Last but not least, results analysis and sharing with the academic community and interest groups in NFV performance benchmarking and NFV for NGMN, thus contributing to the model demystification for early adopters service providers. Therefore, the aim of this study is to compile this information with the final goal of contributing incrementally with the evolution of technology as a whole, regardless of the frameworks and enabler technologies employed. Eventually, new features can be raised and proposed, while focusing on performance evaluation and enhancements.

## 1.4 Monograph Outline

This monograph is organized as follows. Chapter 2, after a premier on NFV, reviews the literature related to the methodology itself, the proposed benchmarking framework, the virtualized infrastructure performance and finally the target Use Case (UC). In Chapter 3 is presented the methodology employed in the proposed framework design and implementation, including the analysis of the candidate metrics and tools to achieve it. It also covers the methodology to prove its concept through a real-world UC. Chapter 4 presents the experimental results divided in “before-Gym”, i.e., a non-automated data collection and results output, and “after-Gym”, a completely automated process. It also presents an analysis of these results and the lessons learned from them. Chapter 5 concludes the work, discussing the challenges faced, the lessons learned as a whole and the open research challenges that can become future work.

# 2 Background and Literature Review

## 2.1 Network Function Virtualization

### 2.1.1 NFV for Mobile Networks

NFV is a trend that aims to leverage an increasingly innovation-friendly environment, compared to the current scenario, being able to allow new functions to be quickly tested and integrated to those already existing and allowing customization in software-level of several network elements. From an economic standpoint, this paradigm introduces a new approach compared to existing middle-boxes that integrate hardware and software into a single solution (appliance). Service Providers will have the flexibility to separate hardware and software, share Information Technology (IT) infrastructures through virtualization, use open source software and COTS hardware solutions. The CapEx and OpEx reduction is one of the goals of this new approach. In general, we can highlight several network functions that can be virtualized, such as routers, firewalls, Deep Packet Inspections (DPIs), Intrusion Detection Systems (IDSs) and Network Address Translations (NATs). The technology is at a place where Telecommunications finally meets IT. In short, it is an initiative that aims to outdistance the Black Boxes and start the evangelization of the so-called White Boxes.

Considering the current scenario, where service providers' profit margin is more and more decreasing and the demand for data traffic is increasing, mainly due to the rise of Over-The-Top (OTT) services, it is necessary to develop creative solutions that contribute to the investments optimization, that exceed the users' expectations and that has a median learning curve. It is expected that in a few years the network infrastructure of MNOs be fully outsourced, for Access Network through radio base stations and controllers infrastructure leasing and Operation & Maintenance (O&M) concession, for Backhaul with the practice of Leased Line over metro optical networks and for Core Network, where IaaS providers are going to be protagonists. In this context, carriers are trying to focus on offering more value-added services, such as those already cited OTTs, and on brand management, as the emerging Mobile Virtual Network Operators (MVNOs) have been doing. The consulting company "Mind Commerce" estimates that NFV global investments will grow up at a Compound Annual Growth Rate (CAGR) of 83.1% between 2015 and 2020. The revenues will reach $ 8.7 Billion by the end of 2020 (MINDCOMMERCE, 2015). "Dell'Oro Group" believes that the market can represent US$ 2 billions in equipment sales by 2018 (MATSUMOTO, 2014). "Research Doyle" estimates a more optimistic scenario, in which the technology will reach a market of US$ 5 billions by 2018, including software, servers and storages (DOYLE, 2013).

Currently, we notice the evolution of the fourth generation of mobile networks (Long Term Evolution (LTE)/System Architecture Evolution (SAE)) on global scale and the first steps of fifth one. Like other technologies of mobile networks standardized by 3rd Generation Partnership Project (3GPP), the 4G's topology is basically segmented into wireless access network (Evolved Universal Mobile Telecommunications System Radio Access Network (eUTRAN)) and core network (EPC), entirely Internet Protocol (IP)-based. Complementary, still inside mobile networks context, the IMS aims to enable Voice over LTE (VoLTE). The high complexity of both system's O&M, considered "closed", and the low capacity for customization and innovation turn them the most biased in order to implement its functions in a virtualized environment. "Infonectics" consulting consulted several carriers about the beginning of NFV applications. 59% answered that they intend to deploy virtualized EPC (vEPC) by 2016 or later, and several also reported about plans for deploying virtualized IMS (vIMS) to support VoLTE. (LYNCH *et al.*, 2014). Finally, it is concluded that, as well as EPC, the implementation of IMS functions in virtualized environments brings several benefits to MNOs, such as flexibility, scalability and cost savings.

### 2.1.2 NFV Performace Assessment

In the NFV context, the computing and networking performance of the NFVIs is considered the biggest technical barrier in the migrating process from physical functions to virtualized ones, i.e., the search for carrier-grade performance is one of the biggest challenges of the technology. Cloud and Communication Service Providers aims at optimizing hardware resources in view of high capacity systems that are cost-effective, consume low energy and are physically compact. Over the past few years, Telecommunications Equipment Manufacturers (TEMs) has been supplying this demand through the use of "common" components in their appliances (a.k.a. Black Boxes), such as Application-Specific Integrated Circuits (ASICs), Network Processing Units (NPUs), Digital Signal Processors (DSPs) and even Field-Programmable Gate Arrays (FPGAs), which are more flexible compared to previous, but less specific. Several initiatives are emerging in computing and networking resources optimization field, provided by COTS hardware, in order to support carrier-grade NFVIs.

Introducing an overview about NFV technical details, ETSI presents a reference architectural framework (Figure 2.1), that shows didactically the actual NFV panorama. It naturally takes advantage of the modern virtualization techniques and technologies widely explored by the cloud computing era. And, in order to also avail the convenience of resource sharing, cost reduction and scalability, NFV aims to bring all telecommunications and data communication technologies, in a so-called VNF or VNF-FG format. These VNFs are, as their predecessor "Black-boxes" eventually managed by their correspondent Element Manager Systems (EMSs), in the context of their high-level function, i.e., the application itself. Above EMS, there is an Operation Support System (OSS) and Business Support System (BSS) layer, interfacing the mandatory Management and Orchestration (MANO)[1,2], that in turn, is composed by an Orchestrator, that instantiate, monitors and maintain NFVs, a VNF Manager(s), that manages the VNF's life cycle, and a Virtualized Infrastructure Manager (VIM), that is being represented by OpenStack since it is becoming the de-facto component for this role. VIM manages the NFVI that supports and underlies the NFVs. "Service, VNF and Infrastructure Description", according to ETSI, is "data-set that provides information regarding the VNF deployment template, VNF-FG, service related-information, and NFV infrastructure information models".

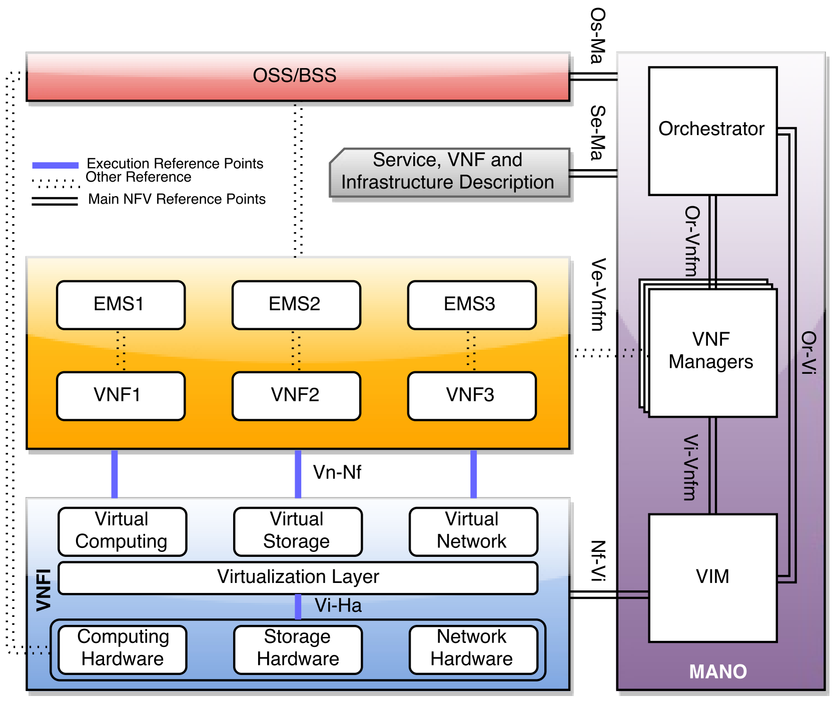


Figure 2.1 – NFV Reference Architectural Framework.
Adapted from ETSI Group Specification NFV 002 V.1.1.1 (2013-10).

Throughout this work, will be known many frameworks, libraries, settings etc. with the main objective of evaluating/optimizing such environments in order to make them as similar as possible to traditional solutions (non-virtualized) with respect to the performance with the assistance of software and hardware-based virtualization technology

1 https://osm.etsi.org
2 https://openbaton.github.io

enhancements. Optimizations of this nature is intrinsically related to scalability concepts. It is possible to scale vertically (down/up) and horizontally (in/out) and/or optimize the available resources, doing “more with less”. Service Providers should look for ways to realize the greatest number of transactions with the allocation of a few resources but, most importantly, that are capable to handle the whole given workload. For example, considering a hypothetical node that has the capacity to process $n$ packets of size $l$ per unit time per core without any optimization in software-level and then, after applying some acceleration technique which support the processing of $2 * n$ packets of size $l$ per unit time per core, so there is a hardware saving when there is a need to scale-out and/or scale-up when there is a function that consumes a processing load greater than $n$ packets of size $l$ per unit time per core in busy hours.

In order to optimize or just to deploy virtualized network functions, such as those that constitute an IMS architecture, it is essential to know the characteristics of the virtualized environment (NFVI). Achieving benchmarkings for choosing the best provider/platform or just to support the decision to deploy a private cloud is considered one of the most important stages of the technology proposed road-map. Sub-sizing means bottleneck insertion and, on the other hand, super-sizing means waste of resources. Financial ones, mainly. These benchmarkings are based on workload testing (stressing) over the network/interfaces, e.g. the generation of varying packets sizes and the their transmission via different protocols compared to the consumption of computational resources.

## 2.2 Related Work

There are several related works, from industry and academia, that aims to analyze metrics, methods and performance benchmarking frameworks for testing cloud infrastructures, such as Cloud WorkBench (SCHEUNER *et al.*, 2014), CloudBench (SILVA *et al.*, 2013), Expertus (JAYASINGHE *et al.*, 2012), Cloud Crawler (CUNHA *et al.*, 2013), CloudCmp (LI *et al.*, 2010), C-Meter (YIGITBASI *et al.*, 2009), Cloud-Gauge (EL-REFAEY; RIZKAA, 2010), CloudSuite (FERDMAN *et al.*, 2012) and Yahoo YSCB (COOPER *et al.*, 2010), besides of the recent Google Perfkit Benchmarker[3]. These frameworks are not specific for NFV performance benchmarking. They are generally test-oriented for web-based applications. In fact, the panorama of Cloud Testing as a whole is still very oriented to web application performance analysis, as in Gao *et al.* (2011). However Riungu *et al.* (2010) and Spirent (2010) highlight infrastructures testings, independently of the application nature: web, network function etc.

From an industry perspective, the TEM Alcatel-Lucent introduced a white

[3] https://github.com/GoogleCloudPlatform/PerfKitBenchmarker

paper in 2014 (ALCATEL-LUCENT, 2014), which advocates a point of view focused on marketing, listing some benefits of virtualizing the LTE Packet Core, such as cost reducing, ROI maximizing, speed up in delivering new services to the end user, operational efficiency, scalability, etc. Due to having a less technical approach compared with other related works, is configured as a great reference for introducing the NFV concept and motivation for this study. It also highlights the EPC along with the IMS technologies as catalysts for the dissemination of solutions based on NFV, which is in line with the strategies of operators in deploying IMS in order to support Voice over LTE (VoLTE), as mentioned in the objectives and justification of this research project.

Existing open source projects sprint common abstractions for benchmarking VNF and their underlying infrastructure. In OPNFV[4], highlights go for three of them. Yardstick[5] targets infrastructure compliance when running VNF applications. QTIP[6] approaches provides definitions towards platform performance benchmarking. And Bottlenecks[7] proposes a framework to execute automatic methods of benchmarks to validate VNF deployment during staging. An independent but related effort is ToDD[8], which walks in the direction of an on-demand distributed and highly extensible framework for distributed capacity and connectivity testing.

However, the most related work as a whole is introduced by Hiray (2014), which adopts the open-source Clearwater vIMS[9] VNF for evaluating. A so-called "monitoring engine" was developed in order to monitor the system's performance in concern of application and infrastructure. Differently from the work proposed here, that has the same VNF, but just as an UC, the author has proposed a framework fully based on Clearwater Project. Rather than employing the Clearwater's "All-in-One" approach, used here at the pre-implementation phase of the fully-automated framework and detailed throughout this document, the "A-Node-per-VM" approach was adopted and used here after the implementation of the fully-automated framework and also detailed throughout this document. Concerning the UC, the methodology and tools are very similar.

In the following, we discuss relevant related work from different perspectives, namely, (*i*) methodology, (*ii*) benchmarking frameworks, (*iii*) performance evaluation of virtualized infrastructures, and (*iv*) vIMS use case. In a nutshell, the main comparison of the scope and approach of this dissertation with regard to related work could be summarized as follows:

[4] ttps://www.opnfv.org
[5] https://wiki.opnfv.org/yardstick
[6] https://wiki.opnfv.org/display/qtip/Platform+Performance+Benchmarking
[7] https://wiki.opnfv.org/display/bottlenecks
[8] https://github.com/toddproject/todd
[9] http://www.projectclearwater.org

- Relates to others cited due to the similar objective of evaluating "as-a-Service" applications and the computational infrastructure that supports them, through the design and implementation of monitoring and benchmarking frameworks based on cutting-edge tools;
- Differentiates from the most part of the cited ones because they are explicitly web-oriented, i.e., instead of targeting network applications (or generic ones), such as data forwarding functions, they aim to evaluate web applications' performance, correlating with cloud computing infrastructure's performance, e.g. processing and database.

### 2.2.1 Methodology

***Cloud Testing Tools. (BAI* et al.*, 2011)***

Purely theoretical, this work is configured as a compilation of methods, modern architectures and recent "state-of-the-art" implementations from industry and academia that enable cloud testing and that will be exploited for insights concerning the architecture design here proposed. Probably due to the chronological mismatch, this work does not introduce some new benchmarking frameworks, such as the Google PerfKit Benchmarker and Cloud WorkBench framework, which will be also exploited in this work as a basis for the proposed framework development. As well as other works, its focus is not the evaluation of NFVI, then it can be seen, for example, the citation of implementations focused in cloud storage performance analysis. It is being exploited for insights concerning the architecture design here proposed.

***On a Catalogue of Metrics for Evaluating Commercial Cloud Services. (LI* et al.*, 2012)***

This work introduces a set of metrics that supports decision-making about choosing the best cloud provider, based on cost-benefit analysis, specifically involving performance, economic and security-related variables. It also highlights the inflexibility of optimizing underpinning layers. This paper will be used as reference for the evaluation metrics compilation proposed here for NFVI and also, consequently, to define tools for workload generation and data collection/analysis that will be part of the proposed framework.

***Cloud-Testing: Issues, Challenges, Needs and Practice. (GAO* et al.*, 2011)***

This work introduces a clear approach about the concept of Testing-as-a-Service (TaaS) that, oftentimes, is approached only as a software testing methodology over cloud only and not the cloud itself or in a complementary way. Also purely theoretical, it presents methods and implementations for cloud testing. Its focus is clearly on web-based applications, i.e., on cloud and cloud-based applications testing. With regard to test methods, they are presented in different domains: public, private and hybrid clouds. It also introduces an interesting comparative table on the different types of tests and their respective focuses, relating them to these domains. In the "Issues and Challenges in Cloud Testing - Scalability and performance testing" section, it is highlighted an important point related to the different approach about static infrastructure and cloud testing, with regard to the consideration of this latter on special features related to metrics and benchmarking frameworks.

### 2.2.2 Benchmarking framework

***Cloud WorkBench – Infrastructure-as-Code Based Cloud Benchmarking. (SCHEUNER* et al.*, 2014)***

Also focused on web-based applications, this work introduces the concept of Infrastructure-as-Code and the Cloud WorkBench (CWB) framework, based on this concept (IaC) and on the modern concept of DevOps. This work presents an implementation itself and a case study oriented to storage performance analysis in public cloud (Amazon EC2[10]). Anyway its challenges and research questions are compatible with those proposed here, such as the delay in provisioning non-automated benchmarkings, such as agents instantiation. Finally, the introduced CWB framework architecture will be reference for the framework architecture proposed here. For interaction with the cloud providers, the CWB uses their APIs, differently og Google Perfkit Benchmarker, that uses their CLIs commands. CWB uses some modern tools in its composition, such as Cron[11] as scheduler, Vagrant[12] as VM environment management tool, through a Ruby-based Domain-Specific Language (DSL), and Opscode Chef[13] as provisioning tool, exploiting the integration between the last two tools. Implemented in Ruby on Rails under the Apache 2.0 license, this framework is available as an open source project at Github. This paper also presents a great review of related frameworks, such as CloudBench (SILVA *et al.*, 2013), Expertus (JAYASINGHE *et al.*, 2012) and Cloud Crawler (CUNHA *et al.*, 2013).

***CloudBench: Experiment Automation for Cloud Environments. (SILVA* et al.*, 2013)***

The motivation of this work is similar to this proposed here, which there is a concern with the matching between application and infrastructure and optimization goals.

[10] https://aws.amazon.com/ec2
[11] https://cloud.google.com/solutions/reliable-task-scheduling-compute-engine
[12] https://github.com/mitchellh/vagrant
[13] https://www.chef.io

It brings a high-level approach, with the adoption of metrics such as VM provisioning throughput and latency, which is more related to the managers (VIM) performance than the VM or the container itself, but also includes the "runtime performance" metrics, as they are called in the paper: latency, throughput and bandwidth. It also introduces the VirtualApplication (vApp), which is an abstraction for the definition/description of the scenario for executing the benchmarking, which includes some popular benchmarking tools ("vApp types", according to the paper). It is implemented in Python and is also available in public repository. CloudBench supports Amazon EC2, IBM Smart Cloud Provisioning (SCP)[14], OpenStack[15] and direct libvirt[16]. A Use Case was performed over OpenStack for DayTrader[17] and Hadoop[18] for the manager evaluation itself, which addressed the "VM provisioning latency" metric, related to the deployment performance.

***Expertus: A Generator Approach to Automate Performance Testing in IaaS Clouds. (JAYASINGHE* et al.*, 2012)***

This work also highlights the benefits of cloud testing automation. It addresses clearly the separations of application versus platform and deployment versus runtime. It introduces the use of multi-step XML handling validated by XSLT templates for code generation, e.g. a script for SSH connection with public cloud, in order to automate complex test scenarios. After the specification phase and code generation, its life-cycle is similar to that presented by related frameworks and that proposed here: platform configuration, application deployment, application configuration, test execution and data collection. In the topic "Extensibility and Flexibility" is highlighted the ease of increasing the code to support new applications and clouds. It was not possible to reach out the developers/authors from Georgia Institute of Technology in order to know about the framework availability for public use.

***A Declarative Environment for Automatic Performance Evaluation in IaaS Clouds. (CUNHA* et al.*, 2013)***

Work focused on an architecture for describing and automatically executing application performance tests in IaaS. It introduce a DSL called "Crawl" (Cloud resource application and workload language), an engine called "Crawler", implemented in Java and based on RESTful web service, responsible for the described benchmarking life cycle by the Crawl, and an Use Case: the evaluation of Olio, an open-source social network web application, over Amazon EC2 and Rackspace[19]. It presents, through the table "Olio Per-

[14] https://www.ibm.com/developerworks/downloads/tiv/smartcloud
[15] https://www.openstack.org
[16] https://libvirt.org
[17] http://geronimo.apache.org/GMOxDOC22/daytrader-a-more-complex-application.html
[18] http://hadoop.apache.org
[19] https://www.rackspace.com

formance Results in the Amazon EC2 and Rackspace Clouds" the results of the performed benchmarking, comparing the amount of executions in which the application meets the required SLA for different VM settings, with their respective costs, in U.S. Dollar per hour, and with different workloads: low and moderate demands related to the concurrent users amount, over different cloud providers (Amazon and Rackspace). It has more overhead compared to the other frameworks because it, according to the paper: "requires from the developer/operator to deploy virtual machine images and application components in the cloud and to implement the Crawler's Java communication interface". The paper does not give details about the authors' plans to make available the framework source code at public repository.

### 2.2.3 Performance evaluation of virtualized infrastructure

***Performance Evaluation of Virtualization Solutions for Telecommunication Applications. (ALOMARI, 2015)***

This work highlights the importance of high computational, memory/cache, storage and networking performance in virtualized environments with a view to networking application hosting. It mostly consists of a benchmarking between three different scenarios: physical machine vs. physical machine, physical machine versus virtual machine and virtual machine vs. virtual machine. For this latter, a machine being the SUT and the other one being the the load generator. In regard to the second case, "physical machine versus virtual machine", the virtual machine holds the SUT role. It also highlights the benefits of binary translation and direct execution techniques in virtualization, also network performance enhancement techniques, carried out by virtualization solution from VMware[20], used in the testbed. Interestingly in this study is the root cause analysis that justifies the poor performance of a given scenario or only justify why a value is larger or smaller than others for different scenarios. For network performance in particular, it performs a comparison between TCP and UDP for different scenarios justifying its results and suggests the study of SCTP as future work.

***A Deep Investigation Into Network Performance in Virtual Machine Based Cloud Environments. (SHEA* et al.*, 2014)***

This work proves the relationship between network performance, such as throughput, delay and packet drop rate, and CPU utilization. It presents a gap in the virtual network performance due the non-optimized CPU scheduling policy in Xen Hypervisor concerning the high throughput and low latency and packet drop rate demand imposed by NFV. The SUT in this work was the public cloud provided by Amazon EC2 and con-

[20] http://www.vmware.com

sequently Xen Hypervisor system, once EC2 overlies it. The most important conclusions here are the relationship between virtualized network performance and VMs' processing load of a non-NFV-optimized environment and the raise of the parameter that switches from a conventional VM workload processing to a NFV workload paradigm, optimizing the CPU scheduling mechanism for this purpose. It is important to highlight that the researches working on this project were just able to figure it out, after performing tests over a local private cloud running Xen Hypervisor as well.

### 2.2.4 Target Use Case: vIMS

***NFV-VITAL: A framework for characterizing the performance of virtual network functions. (CAO*** **et al.*****, 2016)***

This work has a very similar test scenario, using Clearwater vIMS, SIPp, OpenStack and similar testbed in terms of hardware. It employs CPU pinning to avoid potential effects of a shared infrastructure, instead of not putting concurrent background workload as done here. SIPp runs on a dedicated physical server, unlike this work that runs purposely in a "concurrent scenario". It stress vIMS with a granularity of 50 calls/second from 200 calls/second to 1100 calls/second. Here, a granularity of 100 calls/second from 100 calls/second to 1000 calls/second (10 steps). It also repeats the test 10 times to average the results. Their work also considers vIMS efficiency, here presented as the ratio between the there so-called ICR (Input Call Rate) and SCR (Successful Call Rate), with no waiting time. The approach is the same: the more powerful the VNFs are, the higher the efficiency of the whole system is, but instead of analyzing the VNFs' CPU load against input SIP traffic, as it is done here through Gym, it focuses on the saturation of SCR for each VNF (or VNF-FG) size (OpenStack's flavor: homo or heterogeneous platform), besides of also focusing on their CPU utilization. Important: in this work, a call has one step more: subscriber's deregistration, i.e, it is a more complex three-way transaction. When inputting this call scenario at 400 calls/second, for instance, Bono's vCPU reaches a load of 30%, while Sprout 80% and Homestead 60% in a homogeneous m1.medium configuration. Here at 400 calls/second, according to the Figure 4.b: Bono reaches around 20%, Sprout 50% and Homestead 40%, what means a compatible result between both works.

***Dependability Evaluation and Benchmarking of Network Function Virtualization Infrastructures. (COTRONEO*** **et al.*****, 2015)***

With little relation to the works presented above, which have focused on applications performance, usually web ones, and generic clouds, this work has a very similar proposal with this proposed here, focused on NFV. It seeks to contribute to the popularization of NFV in the carrier-grade domain, but in the field of NFVI reliability (availability/resilience) evaluation and benchmarking, based on fault injection (detection/localization/recovery). It is complementary to the pursuit of high performance solutions (goal here and that are also involved by fault detection, since performance degradation occurs when faults occur. Its Use Case exploits the open-source VNF vIMS Clearwater [47], over COTS server and the VMware ESXi hypervisor, the same to be exploited here. Its methodology also exploits the definition of metrics and workloads. Finally, it is concluded that the central idea is a performance study (SIP sessions success rate and latency) in the presence of failures in both NFVI and VNF itself.

***ETSI NFV ISG PoC#1 - CloudNFV Open NFV Framework Project. (CIMI, 2014)***

The official documentation of this PoC is not made publicly available by ETSI NFV ISG, but through CIMI (2014) is possible to see its relationship with the work proposed here. This is the first PoC of ISG. Its Use Case is also based on MetaSwitch's Project Clearwater IMS. This work is one of the biggest drivers to perform the Use Case proposed here. Even without many details about its implementation, it is possible to be seen the exploitation of state-of-the-art implementations, such as some data plane optimizations/accelerations, powered by 6WINDGate™, the deployment over OpenStack and the integration with SDN. The project is a consortium of the following technology providers: 6WIND, CIMI Corporation, Dell Inc., EnterpriseWeb, Overture Networks and Qosmos.

***Evaluating the Performance of an IMS/NGN Deployment. (THISSEN* et al.*, 2009)***

This work is based on the "IMS/NGN Performance Benchmark" specification (TS 186008-1, October 2007) developed by the ETSI (European Telecommunications Standards Institute Telecommunications and TISPAN (Internet converged Services and Protocols for Advanced Networking). It introduces an interesting overview on the IMS technology besides of showing in detail the dynamics of the ETSI specification, the SUT (IMS implementation) and Test System, used to start several scenarios (related to the various types of messages) and then to stress the SUT. The chosen IMS implementation is the Open Source IMS Core[21], unlike that here chosen (Clearwater). The greatest contribution is the introduction of the IMS Bench SIPp tool [22], which will be detailed in Example Tools section (Specific Metrics). Both the test system and the SUT are evaluated with regard to computing performance and memory consumption.

[21] http://www.openimscore.org
[22] http://sipp.sourceforge.net/ims_bench

# 3 Methodology

The methodology to achieve the expected results of the project as a whole is divided into three parts: (i) definition of metrics and tools that are core of the proposed framework, (ii) design and implementation of the framework itself, and (iii) execution of an UC/PoC to test and prove its effectiveness. The design and implementation phase is summarized in the Gym framework development itself. Implementation involves Operation System (OS) compilation (images), API development, their integration and testing etc. The evaluation phase (PoC) is summarized in ensuring the efficiency and effectiveness of this framework through a UC, which additionally aims to correlate performance results in VNF and NFVI and eventually triggering optimization. The specific assessment, through specific metrics and tools, is hardly reused because each function has its standard features, but the proposal is to highlight the concept of VNF benchmarking by the methodology independently of the services provided by it.

## 3.1 Gym: Design and Implementation

The proposed framework architecture is based on performance metrics to be collected from VNF Under Test (VUT) and NFVI Under Test (NUT). The framework is a Virtualized Network Function Forwarding Graph (VNF-FG), as well as VUT itself. The VUT has specific metrics (Layer 5-7), plus generic metrics that comprise computation, network and storage resources. The metrics and KPIs definition for performing benchmarkings should take into consideration three relevant domains:

- System as a whole, considering for instance metrics related to the provisioning delay, scheduling, VM migration, etc. (XILOURIS *et al.*, 2014). Therefore, this domain is not taken into account in the context of Gym since it is Virtualized Infrastructure Manager (VIM) role;
- Instance itself (generic metrics), being it bare metal or full or lightweight virtualized and it is related to machine in-loco, e.g. CPU, memory and storage, and network resources (Layer 2-4);
- Application, i.e., VNF that covers a wider range of specific metrics (Layer 5-7) dependent of the variety of network functions candidates to the virtualization, e.g. SIP Proxy and SAE-GW.

Although in Xilouris *et al.* (2014) the first two domains mentioned above are grouped into “System Level Metrics”, it is necessary to split them concerning the propositional abstraction of the NFV overlying functions. A full benchmarking should include the generic metrics, i.e., application/VNF-independent, besides of having the ability of crossing them with a view at forming KPIs that will support decisions. With the architecture, metrics and default tools definition, the framework implementation covers the development of the three main building blocks (Manager, Monitor and Agent, detailed hereafter and shown on Figure 3.1 and 3.2) with the same design patterns.

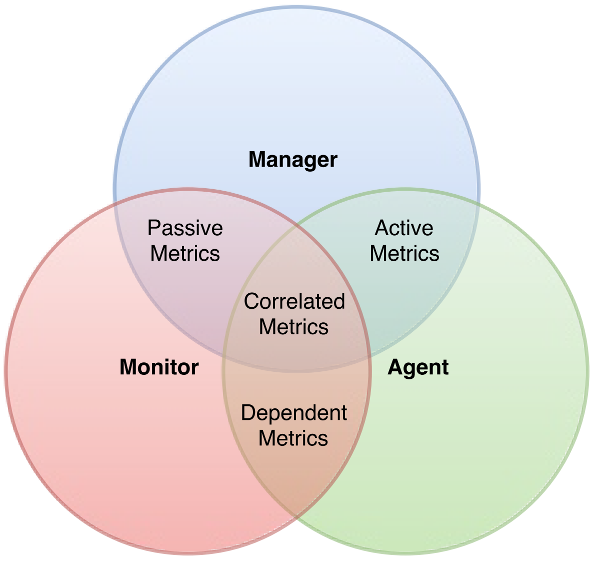


Figure 3.1 – Gym, former VBaaS, Building Blocks’ Relationship.
Credits: R. V. Rosa, co-author of Rosa *et al.* (2014) and Rosa *et al.* (2015).

Manager and Agent will lightweight Linux instances, being VMs or eventually containers, running over an abstracted NFVI, managed and orchestrated by an abstracted MANO. Monitor will be a plug-in attached to the NUT due to its goal of collecting in-loco metrics. The most important features of the whole system are its openness and integrability, through the proposed APIs. In order to reuse code from other FOSS benchmarking frameworks (see Sec. 2.2 on work related to the benchmarking frameworks), Python language was defined.

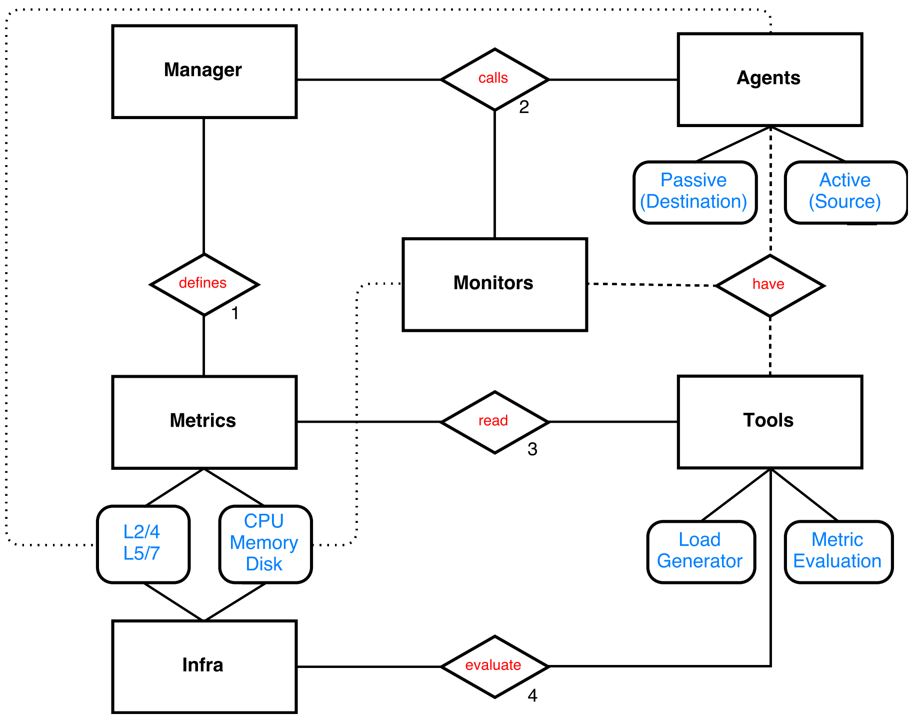


Figure 3.2 – Gym Building Blocks' Actions and Properties.

The three building blocks plus the Infra (VUT and NUT) are detailed below:

- Agents (a.k.a. Gym-AG) are active or passive building blocks that generate workloads and evaluate incoming features, e.g. TCP throughput and Session Request Delay, when active (source), or, when passive, it is just as destination for the traffic generated by the active one and processed/switched by the VNF/Infra. It is important to highlight that Agents are related both to networking (Layer 2-4/generic) and application (Layer 5-7/specific) evaluation. Agents can be distributed, acting in multi-point emulating users and terminal systems' behavior geographically . It contains an API that will be explained below;

- Monitors (a.k.a. Gym-MO) are always passive building blocks attached to the VNF VM or container, as a plug-in, that evaluate intrinsic features of Infra, e.g. CPU and memory utilization under incoming workload from agent(s). Like Agents, it also contains third-party benchmarking tools that are triggered according to the Manager's demand. It also contains an API that will be explained below;

- Manager (a.k.a. Gym-MN) has the role of synchronizing Agents and Monitors' activities, during both in pre-benchmarking and post-benchmarking phases. After sending its requirements ("parameters"), received previously through its northbound interface, connected to the Orchestrator, Manager synchronizes both Agent and Monitor's workload generation (when applicable) and performs data collection/processing, correlating received reports and then providing feedback to the Orchestrator and/or directly to the user.

- Infra is the target NFVI supporting the VNF itself, both under assessment according to the benchmarking objectives raised by the user (NUT+VUT). From a infrastructure point-of-view, it can be considered the candidate Point of Presence (PoP) to host the VNF or the VNF-FG. From an application point of the view, it is defined as a VNF or a VNF-FG itself.

Agent and Monitor's API are open interface, RESTful, for receiving Manager's demand (benchmarking parameters, such as metrics, duration and granularity). This process is transparent from NFVO point-of-view once there it is just a matter of VNF-FG provisioning. Agents can send traffic for Infra (VUT) as its final destination, e.g. vMME, or to another target, a new agent, having Infra as intermediate node(s), e.g. vS-GW/P-GW, evaluating its throughput, latency etc. It is related to Control and Data Plane transactions. Through the proposed API, Agents must know that from the REST Manager, including the metrics to be evaluated, in order to provision benchmarking tools and load generators, the tests duration etc. Additionally, when Infra is an intermediate node, the Agents must know whose are source or destination, besides of the IP address provisioned for the other Agents.

The flowcharts below (Figures 3.3 and 3.4) show the workflow followed after both building blocks receiving parameters in its northbound RESTful API:

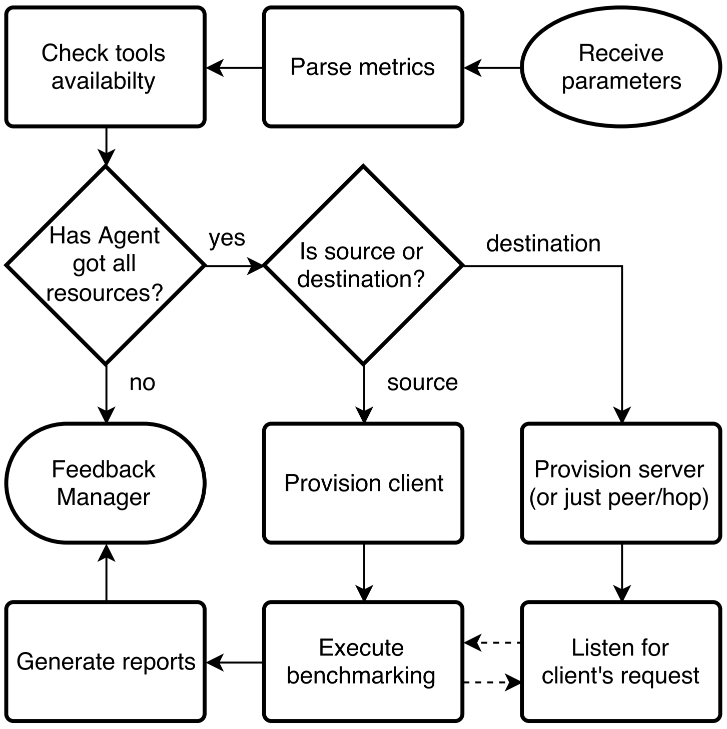


Figure 3.3 – Agent's API Workflow.

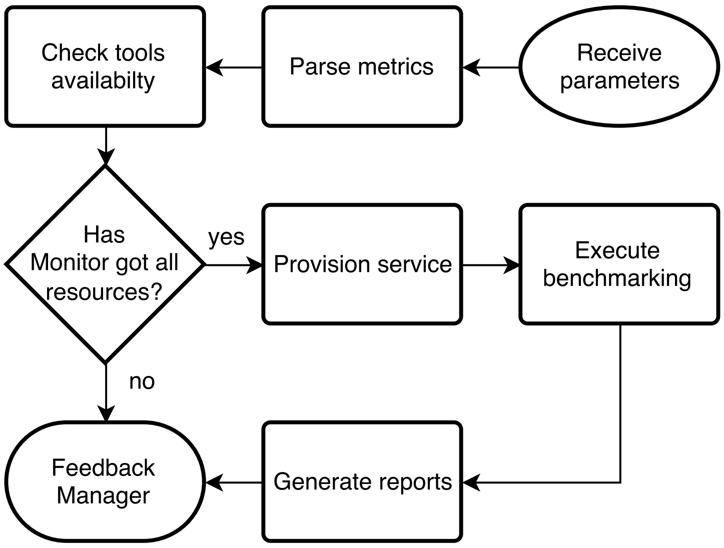


Figure 3.4 – Monitor's API Workflow.

## 3.2 Metrics and Tools

### 3.2.1 Generic Metrics and Tools

The definition of metrics and tools related to the infrastructure, a generic scenario, i.e., without considering high-level application/VNF performance assessment, was based on (LI *et al.*, 2012) which in turn gave rise to "Metrics for Cloud Services Evaluation"[1], a public web-based cloud metrics lookup system.

Table 3.1 – Generic Metrics.

| **Network** | **Computation** | **Memory** | **Storage** |
|---|---|---|---|
| Bandwidth for Different Transport Protocols | Number of Physical Cores | Size | Size |
| Latency (Packet Transfer Delay) | Clock Frequency | Load/Usage | Load/Usage |
| Jitter (Std. Dev. of Latency) | CPU Load/Usage | Transaction Speed (RAM Update Rate) | Data I/O Latency |
| Small-length Packet Processing Rate | Transaction speed (Benchmark Flop Rate) | Latency (Mean Hit Time) | Data I/O Throughput |
| Availability (Dropped Frames Rate) | Latency (Benchmark Run Time) | Throughput (Memory Bit Speed) | Availability |

For the eventual load generation and results collection/analysis of these metrics, FOSS tools will be used. Through an arbitrary way with tendency to choice the "popular" ones, with active community and detailed documentation, aiming at the benchmarking of each infrastructure section, i.e., network, computation and memory. Some

[1] http://cloudservicesevaluation.appspot.com

candidate tools for each section were listed in the Table 3.2. Only one will be employed for each metric. The choice of ideal tools for this project will be done throughout its development.

Table 3.2 – Candidate Tools for Generic Metrics.

| Domain | Metric | Unit | Candidate tools |
|---|---|---|---|
| Network | Availabilty | Packet Loss Rate | Badabing / ifstat |
| | Throughput | TCP/UDP/SCTP Bit Rate | Iperf / Netperf |
| | Latency | Packet Round-Trip Time | CARE / Ping / Hping3 |
| Computation | Processing | CPU Load | Mpstat / Unixbench / dstat |
| | Throughput | Floating Point Execution Rate | HPCC / Unixbench |
| | Latency | Benchmark Run Time | Sysbench |
| Memory | Usage | Memory Load | vmstat |
| | Throughput | I/O Bit Rate | CacheBench |
| | Latency | Mean Hit Time | Land Elevation Change App |
| Storage | Usage | Disk Usage | df |
| | Throughput | I/O Bit Rate | Bonnie/Bonnie++ / iostat |
| | Latency | I/O Delay | NPB: BT |

Despite of the available tools above, the Gym implementation has just taken in account Psutil[2]. It is Python library widely deployed in system utilization data collection that enables Gym-MO to collect generic metric values in-loco from VUTs that, in turn, will be delivered to Gym-MN through Gym's API.

## 3.2.2 VNF-Specific Metrics and Tools

The metrics definition is related to the services that wishes to assess (JAIN, 1991). In this case it is related to the network function and its protocols. Starting from the premise that all generic metrics, and their collection/analysis tools, are known and that specific metrics and also their tools are unknown due to the aforementioned diversity of candidates functions for virtualization and, going beyond, to those that still do not exist, are defined some high-level specific metrics, based on Xilouris *et al.* (2014):

- Concurrent users/sessions/flows;
- Transaction success rate;
- Maximum session rate;
- Application response times;
- Transaction delay.

[2] https://github.com/giampaolo/psutil

These metrics, while seemingly related to network metrics, depend on all generic metrics mentioned above, besides of the design principles, implementation programming languages and settings of the application/VNF itself. Hereupon it is important to notice that the Transport Layer (a.k.a. Layer 4), in network evaluation, is the boundary between the specific and generic metrics, since few protocols are actually implemented in commercial applications, e.g. TCP, UDP and SCTP. On the other hand, these protocols are strongly linked to the application/VNF requirements, which basically involves guaranteed packet delivery and/or speed.

Example Tools, mainly based on the compilation by Xilouris *et al.* (2014):

- D-ITG[3]: acronym for Distributed Internet Traffic Generator, this FOSS tool is, according to its own documentation, "a platform capable to produce traffic at packet level accurately replicating appropriate stochastic processes for both Inter Departure Time and Packet Size random variables." Its a Layer 4-7 generic traffic generator that supports both IPv4 and IPv6 and several transport layer protocols, including SCTP and Datagram Congestion Control Protocol (DCCP). This tool is detailed in Botta *et al.* (2012).
- Ostinato[4]: a "full stack" FOSS packet crafting, generic traffic generation and analyzing tool that support several transport protocol and application-level benchmarkings, e.g. TCP and Session Initiation Protocol (SIP). It has an enhancement powered by the Intel® DPDK, called "DPDK Accelerated Ostinanto"[5], that meets this work goals concerning performance evaluation. According to its own documentation, the intention of the tool is to be a "Wireshark[6] in Reverse", complementing it. Since it also generates and measure Layer 2-7 traffic, it was quoted as a candidate tools in generic metrics and tools section, especially for providing a Python API, called python-ostinato[7];
- Seagull[8]: also a FOSS, under GPL, multi-protocol traffic generator. According to its own documentation, it was "primarily aimed at IMS protocols and thus being the perfect complement to SIPp for IMS testing". SIPp, the next cited tool below, is a default Gym tool concerning the proposed UC, so this synergy between both tools, can be leveraged for the proposed project as a whole. HP® Development Company introduces a White Paper (HEWLETT-PACKARD, 2006) detailing the tool;

[3] http://traffic.comics.unina.it/software/ITG/
[4] http://ostinato.org/
[5] http://www.slideshare.net/pstavirs/dpdk-accelerated-ostinato
[6] https://www.wireshark.org
[7] https://pypi.python.org/pypi/python-ostinato/
[8] http://gull.sourceforge.net/

- SIPp[9]: developed by HP® under GNU GPLv2 license, this SIP Traffic Generation/Measurement Tool will be employed in the proposed UC/PoC due its synergy and native support for the chosen application/VNF. In summary, it stress systems such as the IMS, establishing and releasing calls with the INVITE and BYE methods. It can also generate audio and/or video (Real Time Transfer Protocol (RTP)) traffic through RTP echo and RTP pcap replay. A variant tool, called IMS Bench SIPp was developed by Intel® Corporation (GNU GPL license) that, besides of having got as goal to collect IMS performance data, also collects infrastructure performance data such as memory and CPU usage thorough cpum and generates reports in accordance to the ETSI Tecnical Specification (TS) 186 008 (IMS Network Testing (INT); IMS/Next Generation Networks (NGN) Performance Benchmark) (THISSEN *et al.*, 2009).

Since VNF KPIs are specific for each service function or service function chain, e.g. Clearwater vIMS and nwEPC, and the recommendation here proposed is that, such as the exploited UC (vIMS) and detailed hereafter, be correlated with KPIs values raised in the underlying system, i.e., into NFVI, aiming its implementation over an environment as optimal as possible. Taking the UC as example, it is interesting to follow standards, such as Thißen *et al.* (2009) did:

- ETSI TS 186 008-1: "IMS Network Testing (INT); IMS/NGN Performance Benchmark; Part 1: Core Concepts"[10];
- ETSI TS 186 008-2: "IMS Network Testing (INT); IMS/NGN Performance Benchmark; Part 2: Subsystem Configurations and Benchmarks"[11];
- ETSI TS 186 008-3: "IMS Network Testing (INT); IMS/NGN Performance Benchmark; Part 3: Traffic Sets and Traffic Profiles"[12];
- ETSI TS 186 008-4: "IMS Network Testing (INT); IMS/NGN Performance Benchmark; Part 4: Reference Load network quality parameters"[13].

## 3.3 Gym Main Extension: SIPp Prober as the Stressor Agent

The main extension introduced to Gym is the SIPp Prober (a.k.a. sipp-gym) as the "Stressor Agent" – Gym-AG – based on the already cited open-source SIPp tool.

[9] http://sipp.sourceforge.net/
[10] http://www.etsi.org/deliver/etsi_ts/186000_186099/18600801/01.02.01_60/ts_18600801v010201p.pdf
[11] http://www.etsi.org/deliver/etsi_ts/186000_186099/18600802/02.01.01_60/ts_18600802v020101p.pdf
[12] http://www.etsi.org/deliver/etsi_ts/186000_186099/18600803/02.01.01_60/ts_18600803v020101p.pdf
[13] http://www.etsi.org/deliver/etsi_ts/186000_186099/18600804/02.01.01_60/ts_18600804v020101p.pdf

The generated workload is based on a “Stressful Ladder”, inspired in Din (2008), with a simple two-way SIP registration default procedure, as it is going to be shown on Chapter 4, Figure 4.2, against a(n) (v)IMS deployment, targeting naturally its Edge Proxy (Bono). However, other call scenarios can also be employed[14].

Sipp-gym takes the following parameters as pre-test data:

- SIP Proxy IP address;
- Call scenario file path (SIPp default *xml* file);
- Dummy subscribers database path (*csv* file);
- Maximum simultaneous calls;
- Call rate increase step (calls per second);
- Interval of call rate increase step (seconds);
- Maximum call rate (calls per second);
- Transport (TCP or UDP);
- Test output file (txt or log file).

It also collects and delivers the following metrics by default:

- Amount of Average Transaction Rate per Step;
- Sent Calls;
- Failed Calls;
- Stressor Efficiency [%] (Sent Calls divided by $TransactionsAmountLimit$ - Equation 3.1)
- Completed Calls;
- vIMS Efficiency [%] (Completed Calls divided by Sent Calls).

Where:

$$TransactionsAmountLimit = \sum_{n=1}^{p} Ri * Fd * n \tag{3.1}$$

[14] http://sipp.sourceforge.net/doc/reference.html

Where $p$ (rate_max $Rm$ divided by rate_increase $Ri$) is the number of steps of the "Stressful Ladder" and $Fd$ is the increase_interval.

The amount of time spent by the stress testing is represented by:

$$RunTime = \frac{Rm}{Ri} * Fd = p * Fd \tag{3.2}$$

Efficiency of the stressing system and the VUT itself:

$$StressorEfficiency[\%] = \frac{TrulyGeneratedTransactions}{TransactionsAmountLimit} * 100 \tag{3.3}$$

$$VUTEfficiency[\%] = \frac{SuccessfulAnsweredTransactions}{TrulyGeneratedTransactions} * 100 \tag{3.4}$$

The amount of calls that are not failed neither completed are those not received during stressing/collection $RunTime$ (Equation 3.2), i.e., those delayed. Original SIPp tool does stop receiving calls responses after a pre-configured period, but sipp-gym does. This feature aims to avoid:

- Unreal vIMS flow rate capability, since delayed calls are not taken into account;
- Tasks of Gym's Agents taking more time than planned and then finishing after Monitor ones, mismatching collection time periods.

The input parameters, raised by the user/developer and send from Player to Manager through its API, in JSON format, that will then sent to the Agent, are:

The current Gym version provides 81 metrics. For the current Use Case, only ı$cpu_percent$ȷ is being exploited.

Each metric is seen together with the SIPp generated Stressful Ladder in order to allow their visual correlation. This graphical output puts together the all VNFs' results, as it going to be seen in Chapter 4 (Results).

## 3.4 Proof of Concept Use Case

Before telling what this PoC addresses, is better to understand what the expected outputs are with the benchmarking results raised by the proposed framework:

- VNF (specific) and NFVI (generic) metrics values correlation in order to better understand the Infra and then to perform human or machine-triggered optimization and then to reevaluate (decision support);
- Compare different virtualized infrastructures (VM flavors) on top of the proposed testbed (private cloud) in concern flexibility and performance itself;
- Select or prepare the environment where the incoming VNF fits better based on its performance history, required/negotiated SLA etc.

The assessment phase (proposed UC), besides of demonstrating the Gym framework effectiveness as a whole, aims to evaluate/correlate the performance results between VNF and NFVI. The NFVI, with the exception to its management and orchestration entities and interfaces with other systems, e.g. Operation Support System (OSS) and Business Support System (BSS), that can be abstracted hereupon, reaffirming what has been outlined previously about disregarding NFV overlying functions, such as MANO, has consolidated metrics that are already employed since the emerging of cloud computing. The application/VNF, in turn, suggests specific metrics, which in the majority of cases are variations of throughput and latency of transactions (sessions) that it performs. I.e., the crucial point in the methodology is the clarity of the purpose to evaluate the testbed, composed by two main domains:

- NUT - generic assessment, where methodology/framework can be reused for independent assessments, independently of the function(s) over it;
- VUT - specific assessment. As an UC, this work will adopt the integrated vIMS function thus evaluating specific metrics of the technology.

### 3.4.1 vIMS as VNF under Test

For choosing the VNF, considering the aforementioned synergy of this work with Next Generation Mobile Networks (NGMNs) and the performance evaluation for different NFVI for determined NFV and the subsequent correlation results, were listed, qualitatively, two open-source network functions as potential VUTs in the context of NGMNs: the first representing the EPC and the second representing the IMS:

- nwEPC[15]: A. Chawre (2011) introduces an open source SAE/EPC Serving Gateway framework available in public repository, called "nwEPC". This implementation also includes emulation of the EPC control plane through a built-in MME emulator. The system was developed "out-of-NFV-world", so it suggest to be deployed over Bare Metal, but there is no restriction in deploying it over virtualized infrastructure. It was also developed with the objective of enabling small-scale projects, prototypes and simulations. The scant documentation and technical support absence (no active community) are seen as a major challenge when adopting this framework for the UC;
- Metaswitch's Clearwater IMS project (a.k.a. Clearwater vIMS): an open source project developed by Metaswitch Networks. As quoted in related work section, this framework is protagonist in the ETSI NFV ISG PoC#1 (CIMI, 2014). It has been widely employed in PoCs of several NFV projects, such as Cotroneo *et al.* (2015), Carella *et al.* (2014) and Hiray (2014). It is licensed under GNU GPLv3. Additionally to the "a VM per component" approach (that support horizontal scalability and is a more professional way), the frameworks' community also offers a complete VM (Open Virtualization Format) called "All-in-One" that gathers all core functions of IMS and turns it simpler to deploy and evaluate. Its architecture, according to the Figure 3.5, is composed by the following modules: Edge Proxy (Bono), SIP Router (Sprout), HSS Cache (Homestead), XDMS (Homer), CTF (Ralf) and a provisioning WebUI (Ellis).

[15] http://github.com/thomasbhatia/nwEPC—EPC-SAE-Gateway

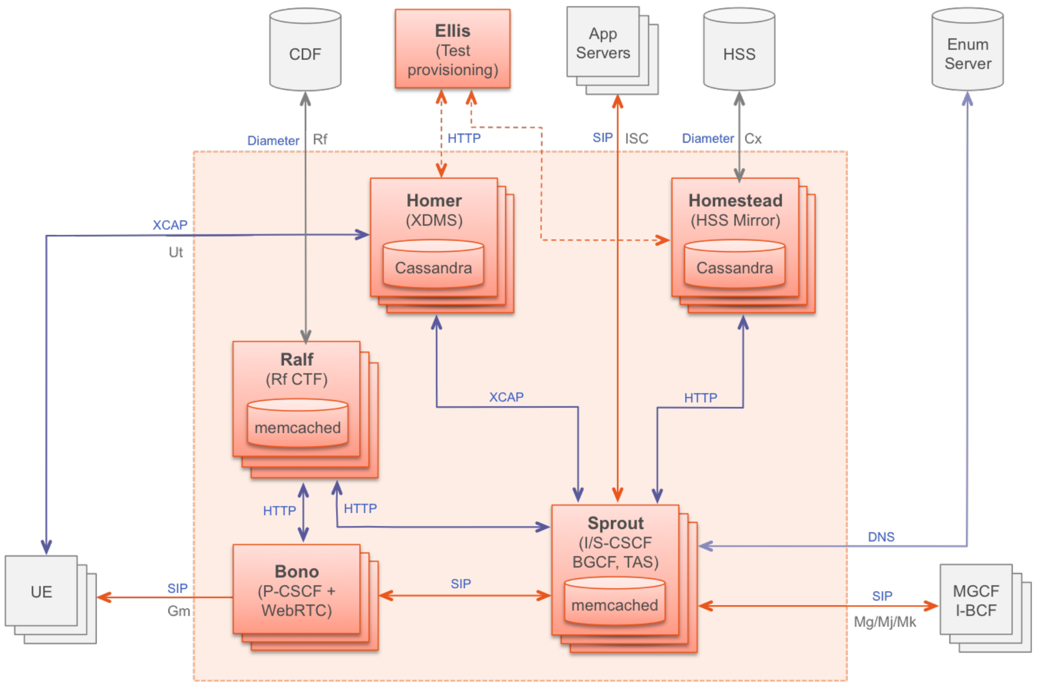


Figure 3.5 – Clearwater vIMS Architecture.
Source: http://www.projectclearwater.org/technical/clearwater-architecture

The choice of vIMS as VNF for the UC was broadly motivated by:

- The ETSI NFV ISG UC #5 ("Mobile Core Network and IMS Virtualization"), that even not referring any performance improvement/evaluation, meets the aim of demonstrating the receptivity of mobile network technologies by NFV technology;
- The influence of three related work: Cotroneo *et al.* (2015), Carella *et al.* (2014)) and Hiray (2014);
- The extensive documentation and active community of Metaswitch's Project Clearwater (IMS) and its integration support with SIPp tool, cited previously as default tool.

### 3.4.2 Testbed

The UC will explore a Private Cloud as NFVI in order to evaluate the Gym framework and the VNF-FG (Clearwater vIMS) behavior. Once Public Cloud is inflexible and close concerning its underpinning environment, through Private Cloud is possible to better interpret underlying issues (RIMAL *et al.*, 2009). Besides of the attempt of the Public Cloud architecture reproduction of Amazon Web Services (AWS) (with Xen), the document (INTEL, 2014) will be exploited as Reference Architecture with a view to implementing an environment (hardware/software) optimized for NFV. This architecture, in turn, suggests state-of-the-art implementations, such as the Open vSwitch accelerated by

Intel® DPDK[16], the deployment over OpenStack [17] and the integration with OpenDayLight[18]. The greatest advantage of having a high performance platform is the flexibility to run different virtualization and cloud management solutions and to modify kernel, hypervisor, and virtual switches settings, in view of the environment optimization for NFV. Since OpenStack is turning the de facto VNF VIM and due to its compatibility to AWS, it is going to be widely explored. The release deployed on Ubuntu Server 14.04.03 is the codenamed "Kilo" ("2015.1.1", the last stable version when OpenStack was chosen), with KVM[19] as hypervisor. It offers, by default, five VM flavors, according to the Table 3.3.

Table 3.3 – Configuration of OpenStack's Default Flavors

| **Configuration** | **OpenStack's Flavors** | | | | |
|---|---|---|---|---|---|
| | ***Tiny*** | ***Small*** | ***Medium*** | ***Large*** | ***XLarge*** |
| vCPU (Num.) | 1 | 1 | 2 | 4 | 8 |
| RAM (GB) | 0.5 | 2 | 4 | 8 | 16 |

The hardware infrastructure to support it consists of i) a high performance COTS server, equipped with two Intel® Xeon 8-Core CPUs, E5 line (2450v2), clocked at 2.5 GHz and cache memory of 20 MB / RAM DDR3-1333 64GB / 1x SSD 120 GB and 2x 300 GB HDD / 2x Intel® NIC X540-T2 Dual 10 GbE Port; ii) a high performance COTS server, equipped wirh one Intel® Xeon 6-Core CPUs, E5 line (2620v2), clocked at 2.1 GHz and cache memory of 15 MB / RAM DDR3-1333 24GB / 2x 1TB HDD / 3x GbE on-board NICs.

Based on the VNF Architecture proposed by ETSI, independent of the NFVI-provider (and placement), the NUT and the Gym framework is placed as shown in Figure 3.6, with OpenStack as VIM, Clearwater vIMS as NUT and OSS/BSS, Orchestrator and VNF Managers abstracted.

[16] https://github.com/01org/dpdk-ovs
[17] https://www.openstack.org/
[18] https://www.opendaylight.org/
[19] http://http://www.linux-kvm.org

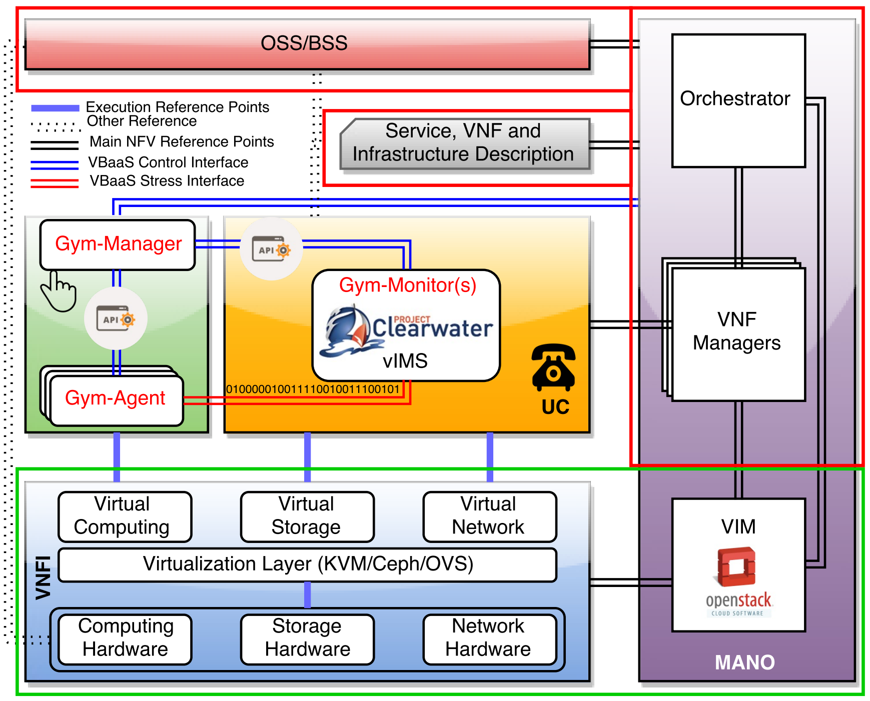


Figure 3.6 – UC Based on the NFV Architecture by ETSI.
Components circled in red are abstracted; In green are part of the testbed.

Due to the Tiny flavor's RAM size incompatibility with Clearwater vIMS minimum requirements, it is not going to be considered for the tests. The candidate NUTs (physical and virtualized hardware) for the UC are shown on Table 3.4:

Table 3.4 – Candidate NUTs for the UC.

| **Hardware**[20] | | | **OpenStack Flavor** | | | | |
|---|---|---|---|---|---|---|---|
| **CPUs**[21] | **Clock (GHz)** | **RAM (GB)** | ***Tiny*** | ***Small*** | ***Medium*** | ***Large*** | ***XLarge*** |
| 12 | 2.1 | 24 | X | ✓ | ✓ | ✓ | ✓ |
| 32 | 2.5 | 64 | X | ✓ | ✓ | ✓ | ✓ |

In order to make things easier, from now on the server with 12 CPUs will be called "Smaller" and the server with 32 CPUs will be called "Larger".

OpenStack, the layer between physical infrastructure and virtual one, is turning the de facto VNF VIM. Its open APIs facilitate integration with other VNF components, such as VNF Managers and Orchestrators. It is powered by KVM/QEMU hypervisor and runs over a single high performance COTS server, equipped with two Intel Xeon 8-Core CPUs, E5 line (2450v2), clocked at 2.5 GHz and cache memory of 20 MB, with 64GB RAM DDR3-1333.

This server, due to the Intel® HT technology[22], provides 32 pCPUs. The deployed infrastructure must fill the available physical resources, as an initial setup, ensuring that operators are not exploiting/stressing less resource than they really have. To enable it, an homogeneous VNF-FG with the largest VIM's available flavor: $m1.xlarge$. Therefore, the present VNF-FG can fill up to 24 pCPUs, i.e., three VNFs (Bono, Sproud and Homestead) with 8 vCPUs each ($m1.xlarge$ flavor). SIPp Prober (Gym's Agent) fills also up to 8 pCPUs with the same OpenStack's flavor. Support instances (DNS/NTP Server and Gym's Manager) fills, by definition, 2 vCPUs each, i.e., $m1.medium$ instances, what results in a total of 36 vCPUs, against 32 pCPUs (an overcommit ratio[23] of 1.125:1). By default, OpenStack is configured with a shared CPU policy, supporting an overcommit ratio up to 16:1. Dedicated CPU policy is also possible (pinning[24]), but its not being considered here. Memory is also not being taken in account in this Use Case, but it is highly recommend as well as network and storage resources, in order to complement the analysis together with CPU.

Ensuring that the physical platform can be fully filled, the desired threshold, measured in calls per second, can also be truly proved, then after reaching it with maximum the capacity, VNFs can be downsized in order to release computing resources to other applications ensuring this threshold. Due to the real-world unknown concurrent background workload, implementing a dedicated CPU and memory policy through VIM could avoid the nondeterminism issue, but the advantages of a shared policy mostly overlap the dedicate ones and will not be discussed here since it is a operator's choice. Anyway no concurrent workload is present during the tests here presented.

Concerning the overlaying platform, the mandatory Clearwater vIMS components (Bono, Sprout and Homestead) were running over OpenStack's compute nodes with Linux Ubuntu Server 14.04.3 upon different OpenStack's default flavors: $m1.small$ (1-vCPU/2GB-RAM), $m1.medium$ (2-vCPUs/4GB- RAM) and $m1.large$ (4-vCPUs/8GB-RAM). Just for information, the other OpenStack's default flavors ($m1.tiny$ and $m1.xlarge$) offers respectively 1-vCPU/0.5GB-RAM and 8-vCPUs/16GB-RAM. Besides of Gym's Agent and Manager, as already mentioned, a third extra instance serving DNS, in order to allow Clearwater vIMS components communicating to each other, and NTP, in order to allow fine-grained time synchronism, also runs over the deployed platform. Gym's Monitor run as a plug-in for each VNF.

[22] http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html

[23] http://docs.openstack.org/ops-guide/arch-compute-nodes.html#overcommitting

[24] https://specs.openstack.org/openstack/nova-specs/specs/juno/approved/virt-driver-cpu-pinning.html

# 4 Experimental Evaluation

After presenting the workflow for the results collections, this chapter presents the results, i.e., the target output, in a graphical performance data correlation format, after collecting and processing performance data in a non-automated process, i.e., "manually" and later in a completely automated process, i.e., after Gym's first release. The main goal of this method is to allow the implementation through a top-down approach, getting things done firstly manually in order to know it and then automating this process.

## 4.1 Result Collection Workflow

The flowchart below (Figure 4.1) shows the methodology employed to collect the partial results through a non-fully-automated way, i.e., without taking advantage of the high-level automation introduced by the Gym Framework yet.

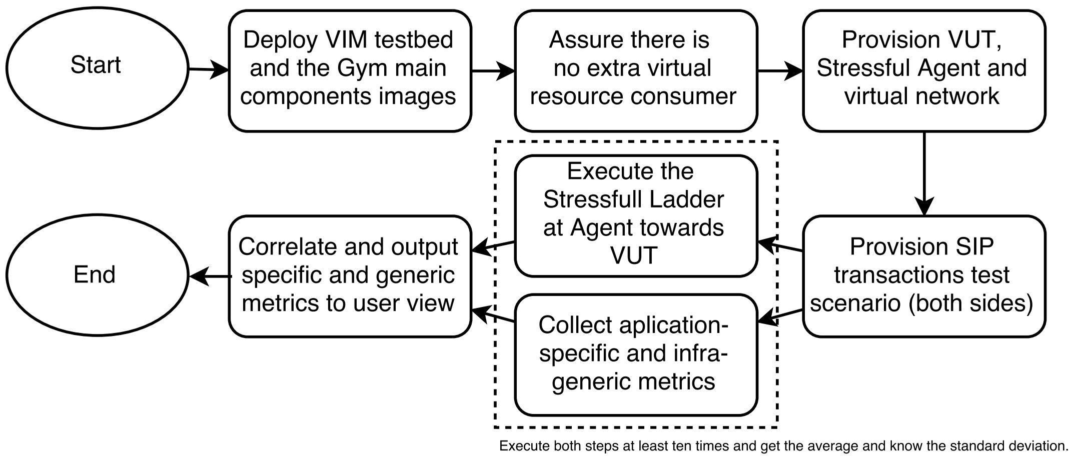

Start
Deploy VIM testbed and the Gym main components images
Assure there is no extra virtual resource consumer
Provision VUT, Stressful Agent and virtual network
Provision SIP transactions test scenario (both sides)
Execute the Stressfull Ladder at Agent towards VUT
Collect aplication-specific and infra-generic metrics
Correlate and output specific and generic metrics to user view
End
Execute both steps at least ten times and get the average and know the standard deviation.

Figure 4.1 – Results Collection Methodology.

The first step is independent of the next ones and once the testbed and the Gym's main components images are deployed, they will be useful until the end. The next steps are executed with minimal automation support, what includes manual execution, absence of fine-grained synchronization and so on. Assuring there is no extra virtual instance consuming resources, such as an "out-of-the-circle" concurrent VM, is important to better understand the results in a shared resources environment. The provisioning phase (VUT, "Stressful Agent" and virtual network) is performed manually through VIM's Graphical User Interface (GUI) or CLI. Provisioning the test scenario includes bulk provisioning of "ghost" subscribers in vIMS side and installing and configuring the application load generator in the "Stressful Agent" side. Then, through a "Stressful Ladder", generating SIP

Registration Transactions, as simple as shown on Figure 4.2, by SIPp tool, and created by a different instance (XLarge) at same physical node, the "All-in-One" Clearwater vIMS (VUT) process this load while collecting its vCPU usage and the vIMS Edge Proxy process' (Bono) RAM demand for after-correlation. Then VMs and virtual network's capacity are evaluated.

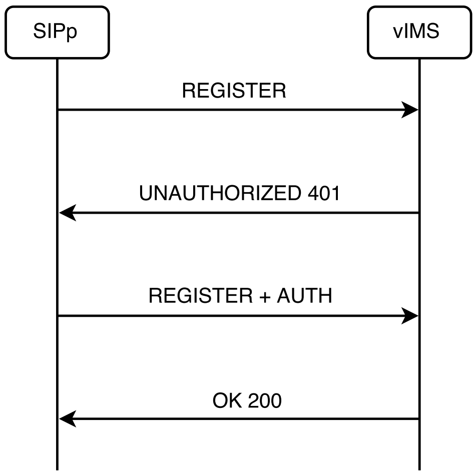


Figure 4.2 – SIP Registration Transaction.

## 4.2 Non-automated Data Collection

As shown on Figure 4.3, 4.4, 4.5 and 4.6, these collection were done before any implementation in order to ensure that the desired output would be valid and conclusive besides of guiding the definition of the given framework requirements and features. Follow the results of generating the theoretical "Stressful Ladder" towards the vIMS instance and then crossing with the collected CPU and RAM by Bono usage at VUT on both nodes: Larger and Smaller.

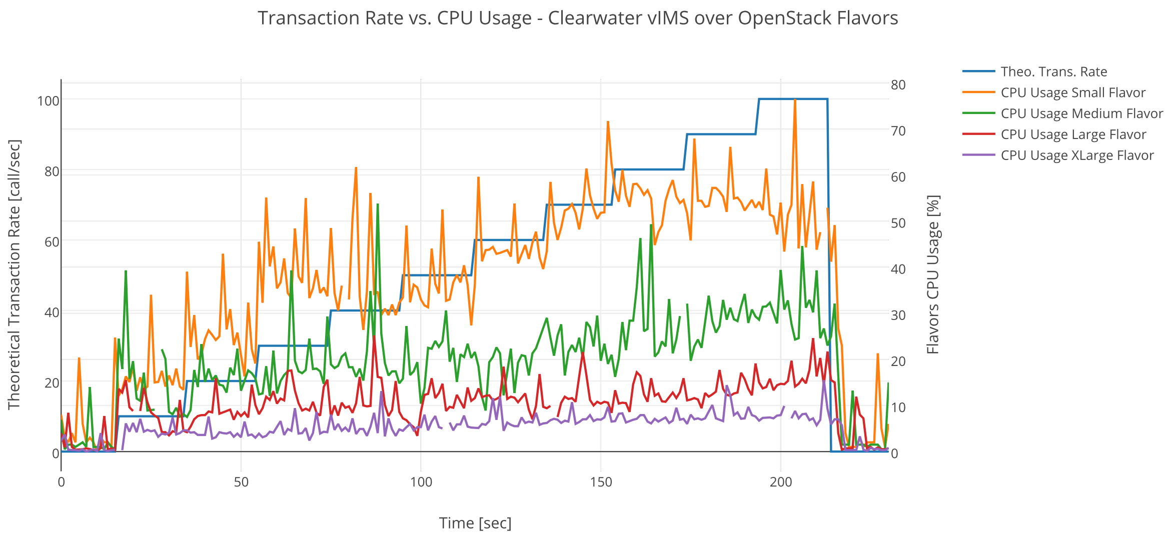


Figure 4.3 – CPU Usage - vIMS over OpenStack Flavors over Larger Node.

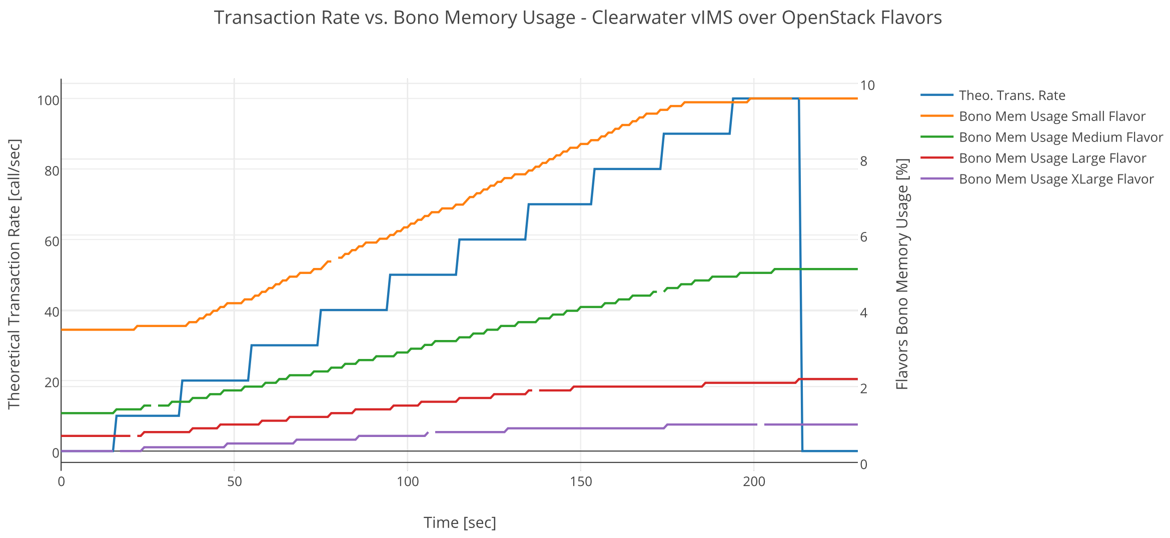


Figure 4.4 – Bono Memory Usage - vIMS over OpenStack Flavors over Larger Node.

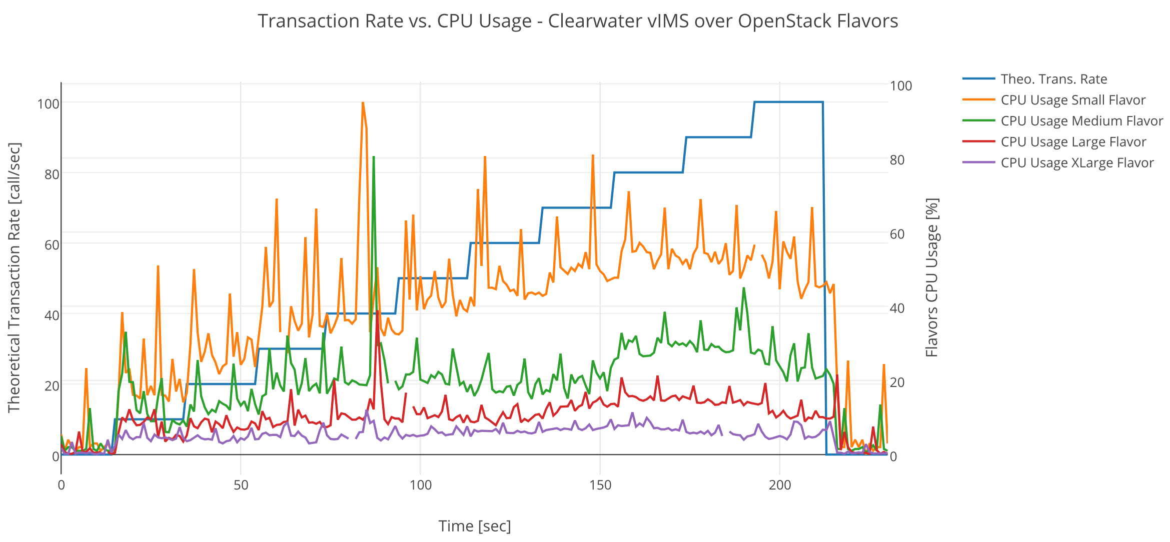


Figure 4.5 – CPU Usage - vIMS over OpenStack Flavors over Smaller Node.

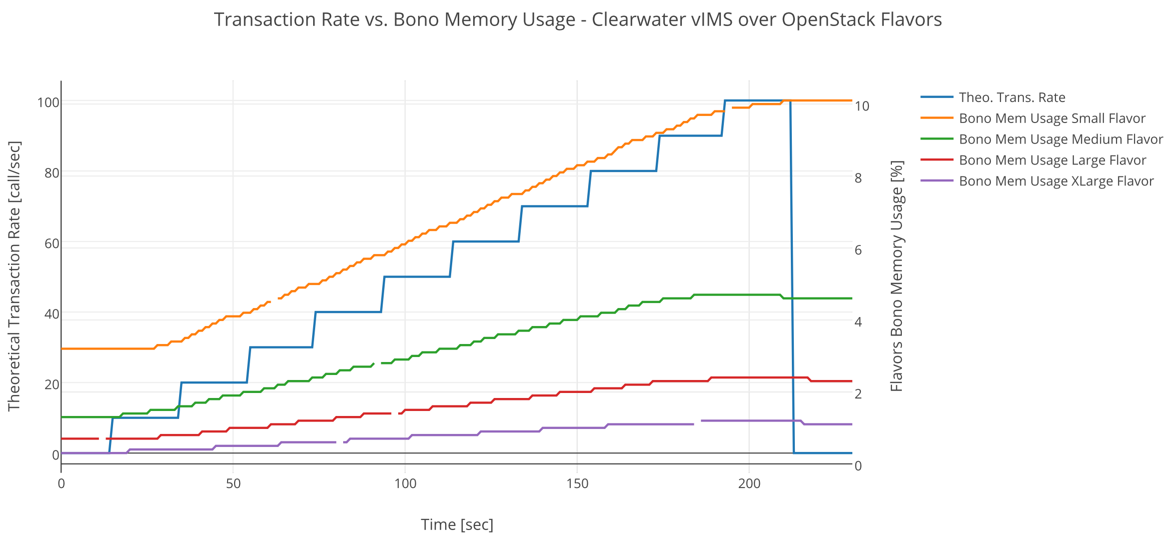


Figure 4.6 – Bono Memory - vIMS over OpenStack Flavors over Smaller Node.

From this testing scenario definition, it is possible to determine some theoretical limits, such as the maximum number of generated transactions and run time and the test efficiency, as follows:

- Transactions Amount Limit (Eq. 3.1): 11,000 calls (10 valid steps of 20 seconds each with a call rate increase of 10 calls per seconds per step).
- Run Time* (Eq. 3.2): approximately 250 seconds .

*Approximately because the test is manually stopped. Also a "tail" of zero call per second is intentionally awaited.

The $StressorEfficiency$ (Eq. 3.3) values of each collection performed and shown above are shown on Table 4.1. For all tests so far, the scenario configuration purposely does not allow simultaneous transactions and this KPI only makes sense for this scenario.

Table 4.1 – SIP Registration Stress Test Efficiency.

| **Hardware** | | | **OpenStack Flavor** | | | |
|---|---|---|---|---|---|---|
| **CPUs** | **Clock (GHz)** | **RAM (GB)** | ***Small*** | ***Medium*** | ***Large*** | ***XLarge*** |
| 12 | 2.1 | 24 | 92% | 89% | 88% | 87% |
| 32 | 2.5 | 64 | 88% | 95% | 81% | 78% |

This efficiency can be affected by several factors, such as network performance, processing saturation and application processing delays. As for the data shown on table above the "Stressful Agent" and the VUT are both guests of the same host, then the virtual network between them is not an issue. In order to validate it, a "Stressful Agent" was installed into a "XLarge" VUT and then the same SIP workload was generated towards "localhost". The results were not almost 100%, as expected. For instance, with a "10/5/200 Stressful Ladder", i.e., with an increase rate of ten transactions per second; step duration of five seconds; and maximum transaction rate of two hundred transactions per second; resulting in a ladder of duration of hundred seconds, according to the Equation 3.3, the result was as low as 59.4%. The efficiency decreases according to the transaction rate increasing. It can be a good KPI to perform benchmarking between different network scenarios and correlate with network delays, for example, but from now on it is going to be ignored and the simultaneous transactions at "Stressful Agent"-side will unlimited, resulting in tests efficiency of almost 100%.

### *Generic Metrics Collection for Pre-Benchmarking*

In order to find a correlation between generic and specific metrics, avoiding specific metrics collection (pre-benchmarking) and enabling conclusions about the VNF considering only NUT's performance, Sysbench[1], a generic benchmarking tool, was ran into a OpenStack's instance configured with a vCPU quantity greater than the quantity of pCPUs of its bare metal host. An expected result, that could not be found in literature, is the saturation when the amount of VM's threads are equal to threads capacity of bare metal host.

[1] https://github.com/akopytov/sysbench

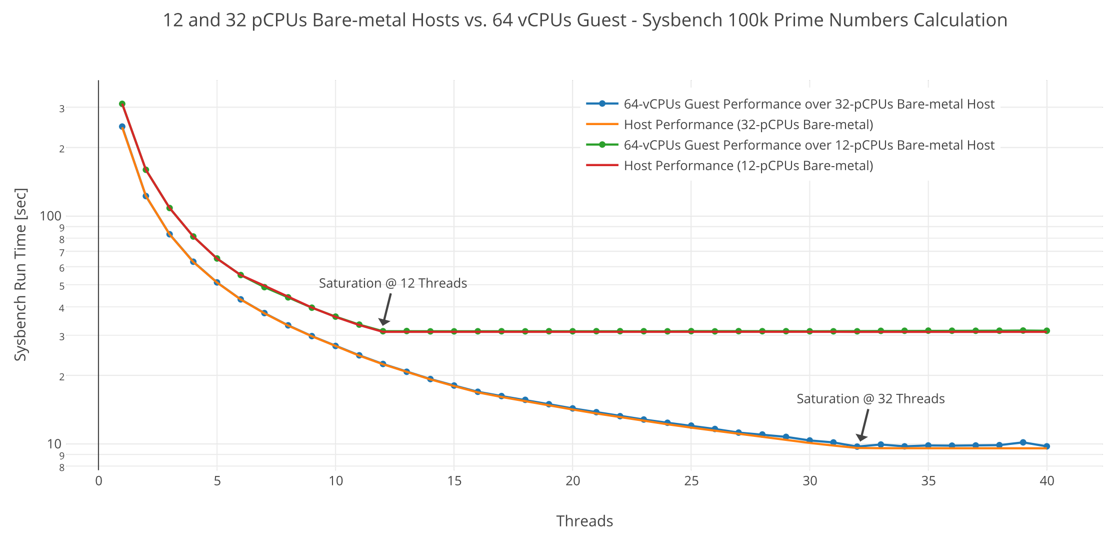


Figure 4.7 – Processing Performance of Smaller and Larger Hosts vs. 64 vCPUs Guest.

From the chart above (Figure 4.7), it can be concluded:

- In the testbed deployed, the results concerning the equality of both guests and bare metal hosts' curves were unexpected and it proves the OpenStack/KVM's efficiency in terms of vCPU-pCPU resources allocation, i.e., the virtualization overheads minimization for the workload generated;
- Unless the specification is to reserve a large number of pCPU time slices (related to the OpenStack's Overcommitting[2]) for a given VNF in order to ensure a considerable performance when the concurrent load is relatively high, instantiating VNFs with custom flavors with a number of vCPUs greater than the number of pCPUs makes no sense due to the saturation;
- Once guest and bare metal host's curves for each node are overlapped, there is no concurrent workload, i,e., there is no other VNF on the same host consuming processing. May even be instantiated, but without significant consumption. This analysis triggers a new assumption about the determinism of NFVI's capacity concern concurrent workload on the same physical host, that affects all collection so far concerning VNF-specific metrics and computation ones, collected with no concurrent workload.

Another interesting result was the difference between Sysbench run time when calculating a given amount of prime numbers (50k for instance) for both Small and Large servers and comparing the processing performance between a VM and its bare metal

[2] http://docs.openstack.org/openstack-ops/content/compute_nodes.html#overcommitting

host (Smaller and Larger nodes), using the same amount of threads, equal to the vCPU quantity of the he VM under test. In this case, the test was configured with eight threads in order to compare with the XLarge Flavor's performance. Both results for both servers in red on table 4.2 reinforce the affirmation of VIM and the efficiency of the hypervisor.

Table 4.2 – OpenStack Flavors over Smaller and Larger Nodes - Sysbench 50k Prime Numbers Calculation Run Time [sec].

| **Hardware** | | | **OpenStack Flavor** | | | | *Bare Metal* |
|---|---|---|---|---|---|---|---|
| CPUs | Clock (GHz) | RAM (GB) | *Small* | *Medium* | *Large* | *XLarge* | *(8 Threads)* |
| 12 | 2.1 | 24 | 122.8 | 62.34 | 31.5 | 17.02 | 16.98 |
| 32 | 2.5 | 64 | 95 | 48.06 | 24.66 | 12.86 | 12.72 |

The results, comparing different flavors at the same node, after collecting a first dataset is easily predicted due to its proportionality. It is important from the operator's point of view, to know the limitations, i.e., the maximum thresholds, in terms of application metrics, of its available virtual resources.

## 4.3 Automated Data Collection

Some conclusions were already raised in the previous section and are going to be used and cited here. Smaller Node is not being employed anymore and all tests are running over the so-called Larger Node, since it was concluded that VM's performance is almost the same of its underlaying platform (Bare Metal), according to Figure 4.7 and Table 4.2, and generic metrics, e.g. CPU and memory, are naturally and obviously proportional to hardware power, there is no need to continue comparing both servers.

From now on, besides of fully employing Gym (results were collected automatically through the developed Gym framework and its extensions (SIPp Prober, a.k.a. $sipp-gym$ and *collector* a module that will be part of the Gym's Player)), a new approach for the VFG-FG under test is taken: a defragmented vIMS architecture, i.e., Clearwater's VNFs are no longer part of an All-in-One instance. Each of its mandatory components (Bono, Sprout and Homestead) is an independent instance.

New approaches are also taken into account from now on:

- The theoretical "Stressful Ladder" was replaced by a real one. The "difference" between the theoretical and the real ladders is the "Stressor Efficiency".
- In order deliver a more accurate result, the tests are performed ten times and then averaged, with a true mean probability of 95%.

- Beside of the $StressorEfficiency[\%]$ (Eq. 3.3), the $vIMSEfficiency$, based on Eq. 3.4) is now being taken into account. It is represented by:

$$vIMSEfficiency[\%] = \frac{SuccessfulCalls}{TrulyGeneratedTransactions} * 100 \tag{4.1}$$

The $SuccessfulCalls$ are those completed in $sipp - gym$ after receiving the "200 OK" message during the $runtime$ period. Delayed and failed calls are the difference between $TrulyGeneratedTransactions$ and $SuccessfulCalls$:

$$TrulyGeneratedTransactions = SuccessfulCalls + FailedCalls + DelayedCalls \tag{4.2}$$

- Even knowing the importance of running tests with different call scenarios, besides of the simple two-way SIP default registration, it was not considered due to the focus on the framework itself and the basic correlated results, such as in (CAO *et al.*, 2016). Anyway, the "SIPpProber" is fully able to receive new calls scenarios [3] as input parameter (Task) and then to stress vIMS with them.

Interval time: one challenge concerning valid data collection is determining the time between each of the ten cycles (rounds) of collection, ensuring a True Mean Probability of 95%. This interval time is require to ensure that all buffers from Clearwater vIMS and OS network stack buffers and memory caches are integrally cleaned (since queues from a previously cycle could affect the next one). When there is no bottleneck (processing, memory or network), the vIMS efficiency is around 90% (between 86% and 94% in this collection), leading to a time between cycles of 200 seconds, a number empirically determined for the necessary buffers/caches emptying time. As can be seen on the table below, the VNF-FG when deployed over $m1.small$ instances cannot achieve around 90% due to CPU limitation, as can be seen in the Figure 4.8 (a).

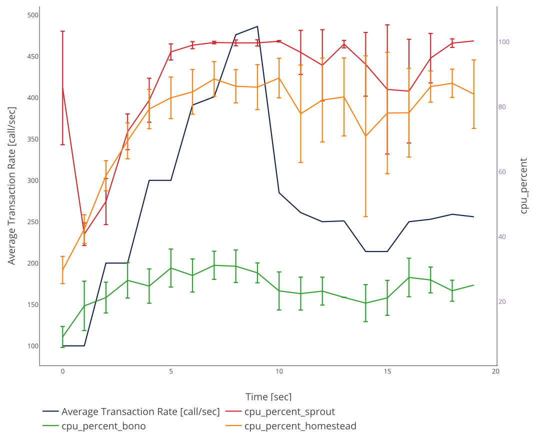
Table 4.3 – vIMS Efficiency vs. Time Between Data Collection Cycles.

| **Time Between** | **vIMS Efficiency over OpenStack** | | |
|---|---|---|---|
| **Data Collections** | **m1.small** | **m1.medium** | **m1.large** |
| 50 | 40% | 43% | 44% |
| 100 | 46% | 52% | 66% |
| 200 | 77% | 92% | 88% |

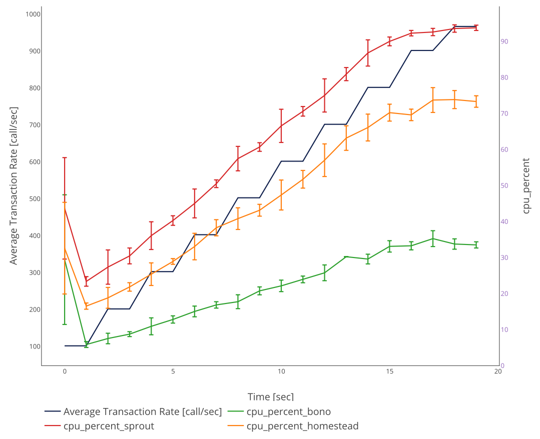
What results in a $TransactionsAmountLimit$ (Eq.3.1) of 11,000 calls, a $RunTime$ (Eq.3.2) of 20 seconds and a "Stressful Ladder" of 10 steps. In order to ensure a good

[3] http://sipp.sourceforge.net/doc/reference.html#Create+your+own+XML+scenarios

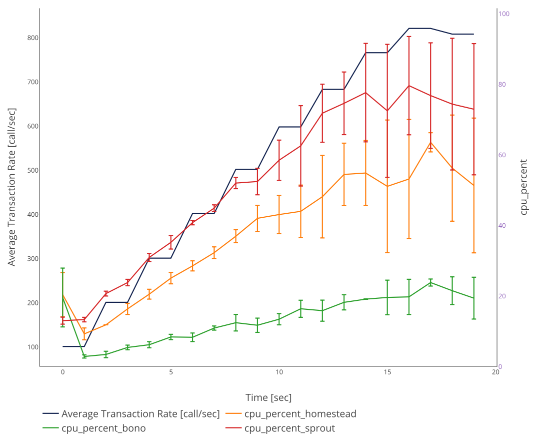
confidence interval, ten collection cycles/rounds and a true mean probability of 95% are taken respecting the interval of 200 seconds, previously determined, and a guard time of 10 seconds after the tests finalization (each 20-seconds-round) for the results collection from Manager, what results in a total runtime of 38 minutes and 20 seconds.



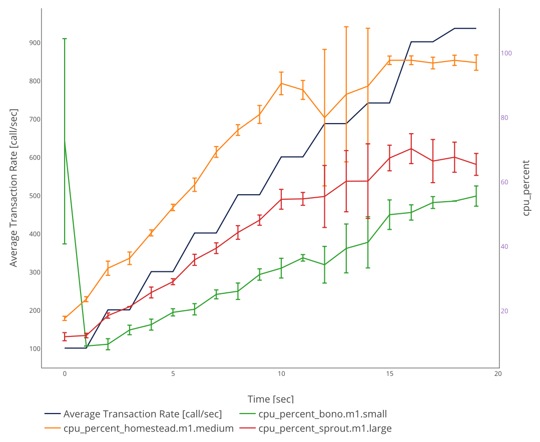
(a) vIMS VNFs over *m1_small* flavor.



(b) vIMS VNFs over *m1_medium* flavor.



(c) vIMS VNFs over *m1_large* flavor.



(d) vIMS VNFs over heterogeneous* flavors

Figure 4.8 – Average Transaction Rate (calls/sec) vs. CPU Usage (%).

*Bono as *m*1.*small*, Homestead as *m*1.*medium* and Sprout as *m*1.*large*.

Just for comparison or benchmarking, Figure 4.8 (c) shows the same task but stressing a *m*1.*large* vIMS VNF-FG. After observing results with an homogeneous VNF-FG, i.e., composed by VNFs with the same flavorl it can be visually concluded that for the same input of the test above (maximum 1,000 calls/second etc.), a *m*1.*small* Bono, a *m*1.*large* Sprout and a *m*1.*medium* Homestead (an heterogeneous VNF-FG) could support the given traffic demand instead of bigger ones. The larger the homogeneous VNF-FG flavor the easier drawing this conclusion

VNF-FGs *m*1.*small* reaches out a transaction rate of almost 500 calls/second, while $m1.medium$ has a peak of more that 700 calls/second and $m1.large$ reaches out more than 800 calls/second. All respecting a vIMS efficiency of around 90%. With all data read is possible to identify which metric and which VNF are the bottleneck that limits the whole system capability. Also anomalies and faults can be found through the outputs brought by Gym framework and its extensions.

The next outputs are taken from this "optimized" VNF-FG, now taking a safer interval between cycles of 300 seconds, instead of 200, in order to show, besides of the already known CPU Usage, other metrics such as $\imath memory\_virtual\_percent \jmath$, $\imath network\_counters\_eth\_0\_packets\_sent \jmath$, $\imath storage\_io\_counters\_write\_bytes \jmath$ and $\imath storage\_io\_$
that are also target of this study.

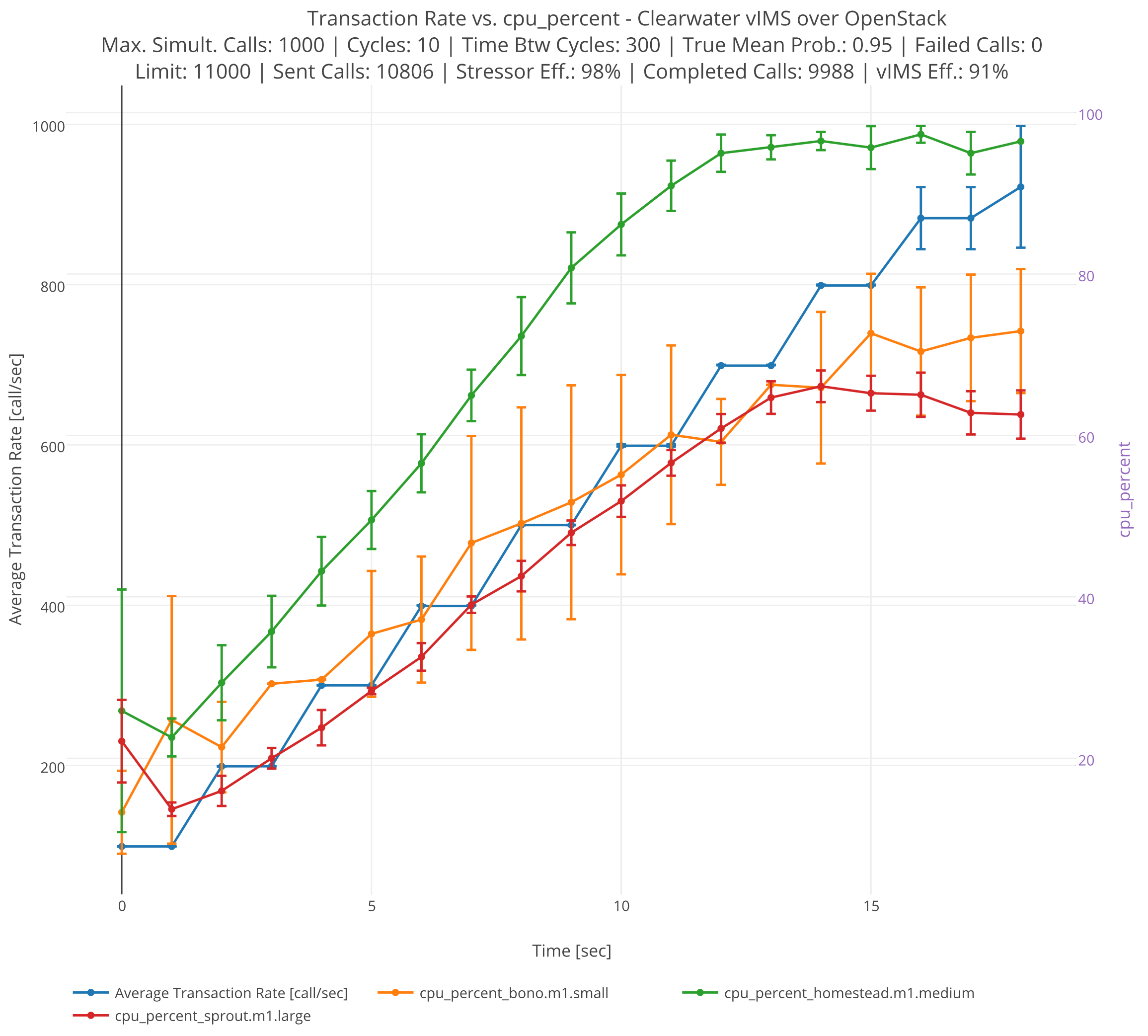


Figure 4.9 – SIP Registration Rate vs. CPU Usage.
Interactive scatter-plot available at <https://plot.ly/~bertoldo/976>.

Figure 4.10 shows the difference between VNFs on regard to RAM utilization. While Sprout and Bono have a low utilization (less than 1 GB), Homestead has a mid one (almost 2 GB). On the other hand, when Bono and Homestead have a stable utilization between collections cycles, Sprout has more an unstable one.

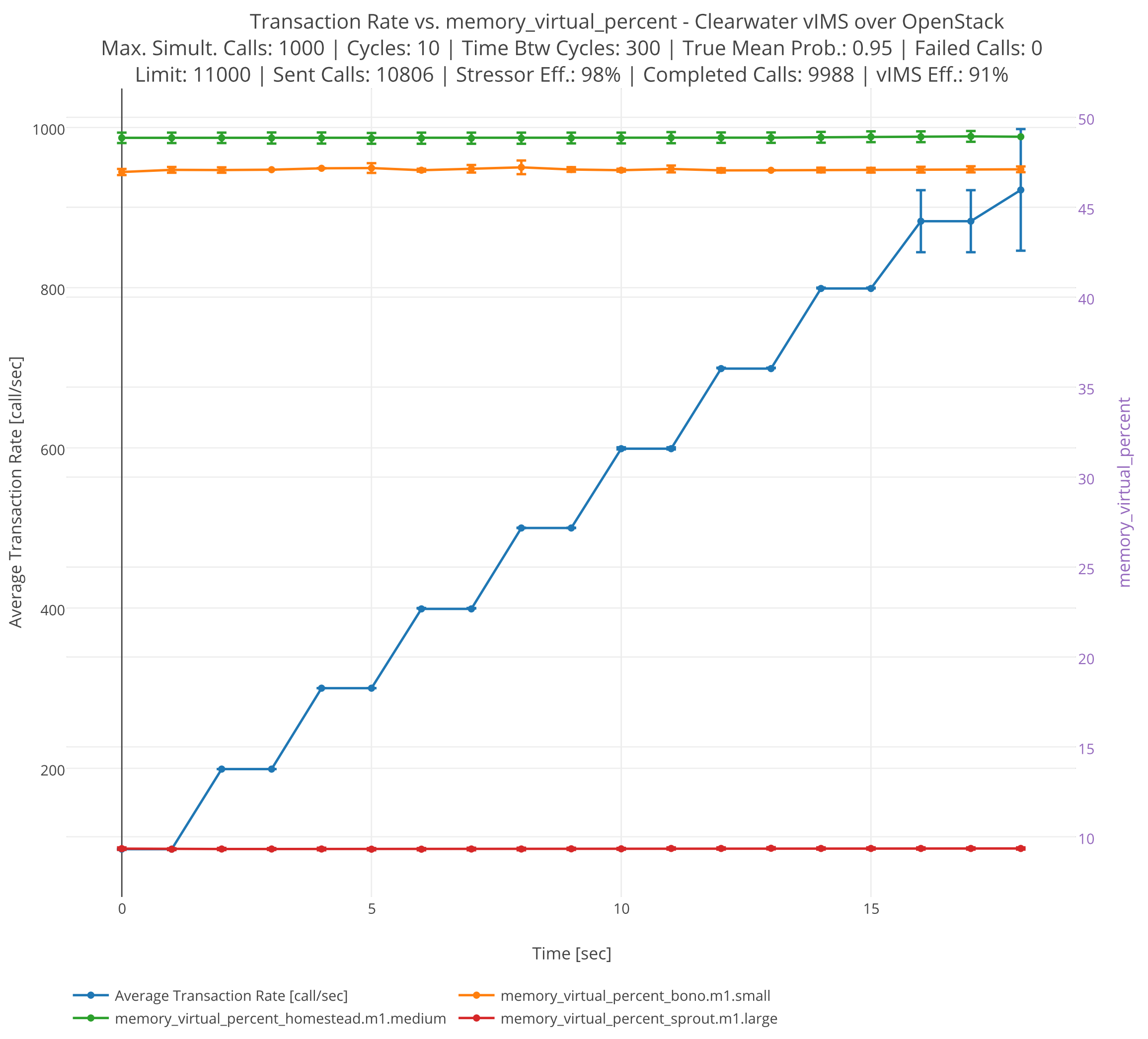


Figure 4.10 – SIP Registration Rate vs. RAM Usage.
Interactive scatter-plot available at <https://plot.ly/~bertoldo/978>.

Since all request and response packets of each two-way SIP registration procedure go through Bono and Sprout equally and the number of subscribers queries from Sprout to Homestead is also the same, all curves grow proportionally, as shown in Figure 4.11. These metric's values are incremented cycle by cycle, what justify the huge error bars.

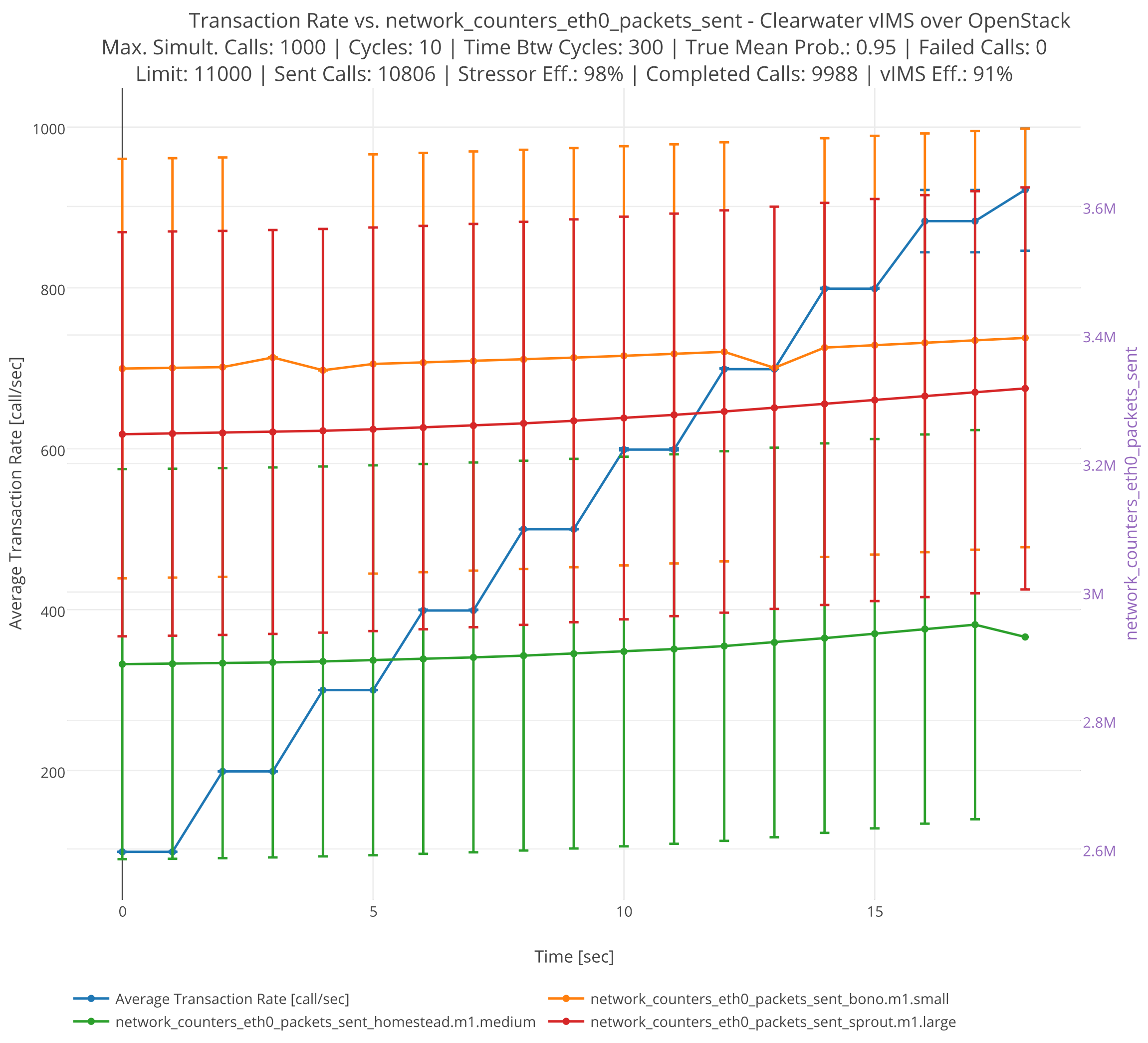


Figure 4.11 – SIP Registration Rate vs. Network Sent Packets.
Interactive scatter-plot available at <https://plot.ly/~bertoldo/980>.

Both following metrics (Figures 4.12 and 4.13) are related to disk and due to the characteristics of the VUTs, they do not bring relevant conclusions.

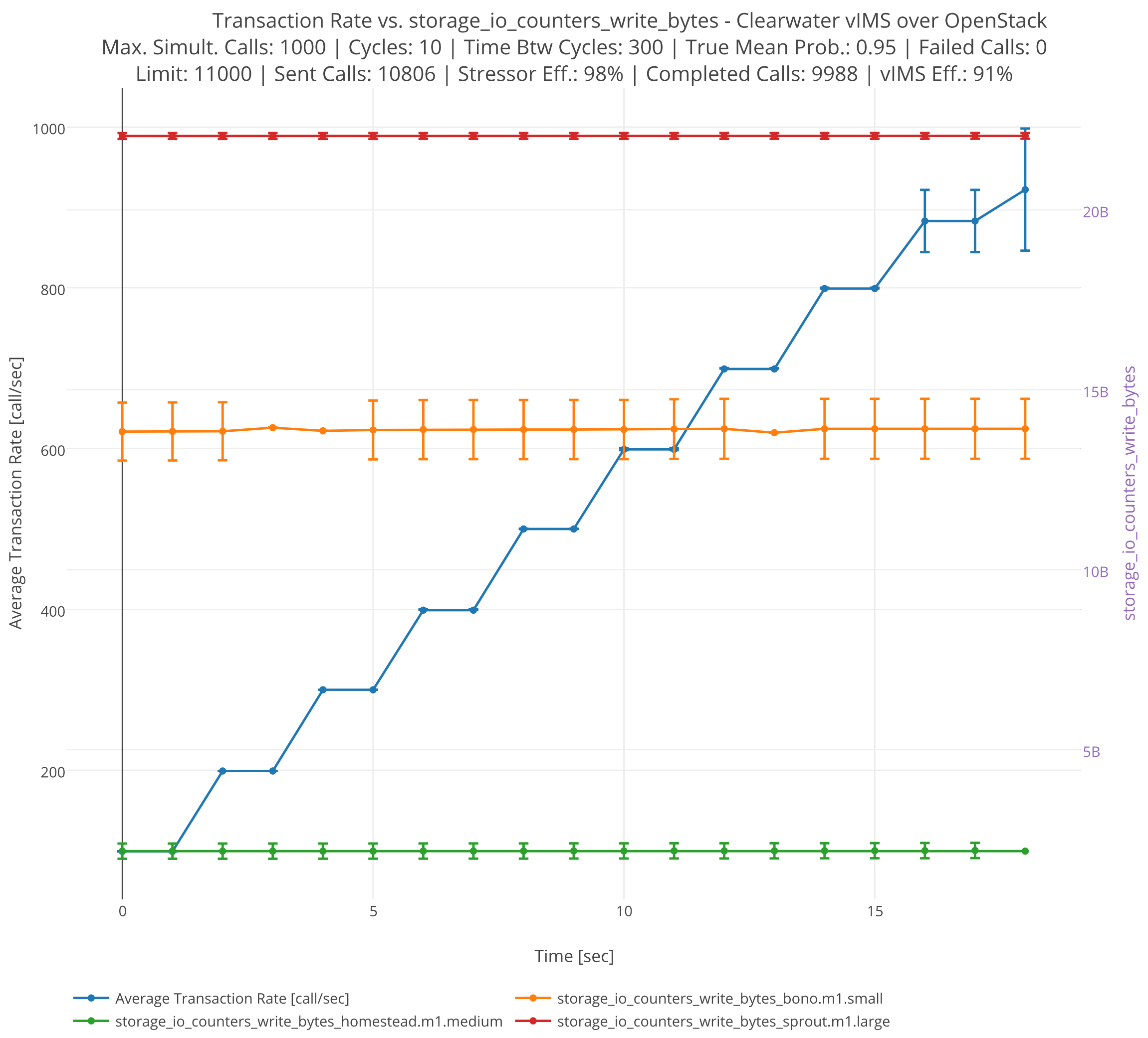


Figure 4.12 – SIP Registration Rate vs. Disk Writing.
Interactive scatter-plot available at <https://plot.ly/~bertoldo/982>.

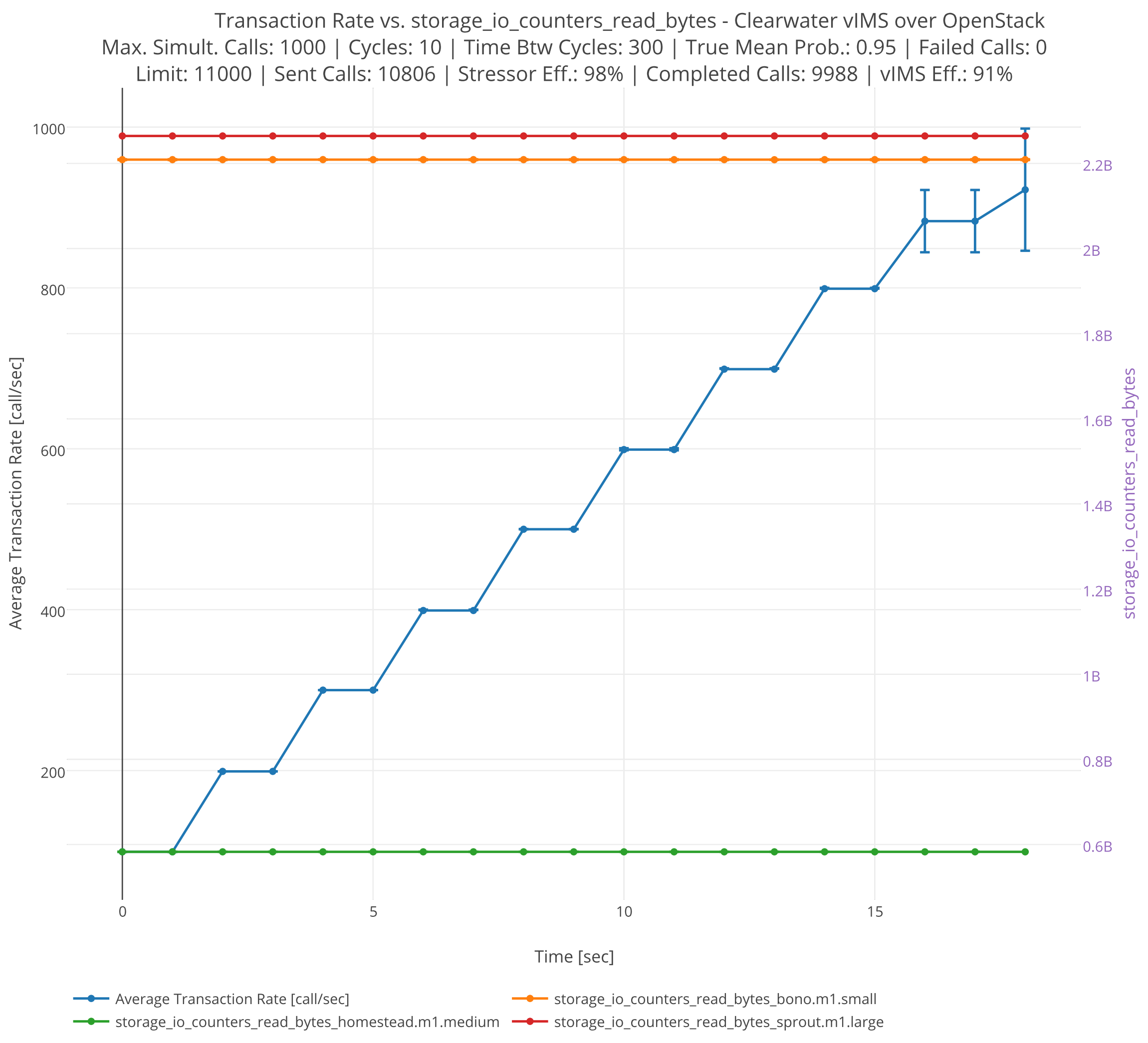


Figure 4.13 – SIP Registration Rate vs. Disk Reading.
Interactive scatter-plot available at <https://plot.ly/~bertoldo/984>.

## 4.4 Discussion and Lessons Learned

The results obtained, independently of the method, manual or automated, contribute to the knowledge on performance evaluation of the physical computational infra, that due to the cutting-edge technologies for virtualization is very close to the performance delivered to its upper layers in the context of virtualization. From this, this research has taken a new direction and started investigating the correspondence between the performance of the core activity of a certain virtualized function and the computational capacity of its virtualized underlying infrastructure, which is practically the same as the physical infrastructure, as seen, cooperating with the development a framework (Gym) and extension tools that shorten the path between the evaluation objective and the evaluation/test/benchmarking itself. The results presented previously speak volumes and are intuitive: the higher the workload demand, the greater the work/capacity required for the infra that supports it. But the central objective is to know, regardless of whether there are or not transparent horizontal scaling techniques, the limits of the infra in terms of high-level metrics, such as the given SIP registration rate, and the ideal resource allocations for the different VNF that are part of a given VNF-FG, i.e., it shall not occur oversizing or undersizing each VNF in terms of its configuration (size or flavor), regardless of it is a VM or a container.

From the perspective of the Gym framework and the methodology for raising requirements through the assumption of future needs of NFV operators and developers, through collection, parsing and outputting manually and prototypes for automation of all processes for the presentation of these correlated data (initial central problem), the difficulties that have become challenges and that have been and still are being surpasses, are basically summarized in questions such as:

- The opening/receptivity of new technologies or platform under test, since the particularities introduced by different information and/or communication protocols naturally make it difficult to create generic interfaces in the context of the proposed framework, with emphasis on the generation of the desired outputs, as presented in this chapter;
- Issues related to flow control and buffering of the VNFs or VNF-FGs that directly affect the results between the data collection cycles: explored to generate highly reliable outputs - the results are the mean of the results of at least ten repetitions and for the divergences of data between these repetitions, a.k.a. margin of errors, is considered a 95% confidence interval. Therefore, the open challenge that remains is about discovering what is the best strategy for each technology/platform. To force buffers and caches emptying, to perform a cold reset in the whole infra or simply to wait for an empirically determined period (as was done in this work).

In any case, the results obtained are compatible with those expected. Considering the given UC, the metric “SIP registration rate” represents a basic and initial activity in an IMS network and does not represent a real-world traffic pattern, which is composed of several random and complex methods, naturally, such as group calls with media transfer.

An important pain point raised initially through the non-automated experiments is the trade-off about having an implemented dynamic resource allocation algorithm, that is positive and the essence of virtualization, and then the difficult of benchmarking, i.e., results with different levels of workload from different VNFs sharing the same physical resources, vs. having a static resource allocation algorithm implemented, e.g. CPU Pinning[4], that is often under or over-dimensioned, and then be able to not worry about concurrent load.

In short, through the collected values for generic (SIP Registration Transactions) and specific metrics (CPU and SIP Proxy's process RAM Usage) is possible to determine whether the NUT is suitable for the target VNF. The test were performed with no concurrency on the NFVI, so, as shown through the generic benchmarkings, the performance of a VM when its vCPU quantity is greater than or equal to the pCPU quantity, is equal to the performance of its bare metal host. It means that in presence of concurrent load, e.g. another VNFs running on the same host, and concerning the OpenStack's default computation dynamic resource allocation algorithm, the capacity of a NFVI to host a VNF is directly affected. This phenomenon does not occur whether there was an algorithm for static allocation of computation resources. I.e., the dilemma of having or not a deterministic environment touches the dynamic vs. static allocation trade-off. In other words, the non-determinism occurs due to the unpredictable concurrent load. If the goal is really to guarantee predicable performance after VNF deployment, techniques such as CPU Pinning should be considered.

Through the lessons learned, concerning the challenges of non-determinism of performance in a NFV environment, the next steps comprise in seeking solutions to estimate concurrent workload on the same physical host in order to reproduce the real world and to consider it into the VNF-specific and NFVI-generic results correlation. Additionally, the need of provisioning a specific test scenario for each network function or function chain benchmarking is an issue that can be remains open, allowing the operators/developers to implement their own test scenario according to the target VUT, as well as is being done through the proposed UC.

Through this assessment phase and UC, besides of demonstrating the under-development Gym framework efficiency, aimed to evaluate/correlate the performance results between VNF (specific) and NFVI (generic) allowing to:

1. Better understand the infra and then to perform human or machine-triggered optimization and then to reevaluate;
2. Compare different cloud providers or hardware/virtualization platform, including the internal one (private cloud) in concern to cost, flexibility, performance itself etc. (decision support);
3. Support selection of the environment where the incoming VNF fits better, based on its performance history, required/negotiated SLA etc.

[4] https://specs.openstack.org/openstack/nova-specs/specs/juno/approved/virt-driver-cpu-pinning.html

4. In order to monitor and/or perform benchmarking of each function of a system, such as the decomposed Clearwater vIMS, the “All-in-One” approach does not work. In this case, it was necessary to deploy a function per VM and then to evaluate it individually. It was done when the automated process was adopted, with Gym.

The NFVI has consolidated metrics that are already employed since the emergence of cloud computing. The application/VNF, in turn, suggests specific metrics, which in the majority of cases are variations of throughput and latency of transactions that it performs. One of the most relevant challenges of this ongoing work is the diversity of specific performance metrics for VNF assessment due to the diversity of network functions candidates to the virtualization, such as those one that compound an IMS, demanding from framework its adaptability for evaluating entities still unknown. The choice of vIMS as an UC was broadly motivated by CIMI (2014), Hiray (2014), Cotroneo *et al.* (2015), Carella *et al.* (2014) and Thißen *et al.* (2009).

# 5 Conclusions and Future Work

This work has fulfilled its main objective, contributing to the development of a specific framework for evaluating virtualized environments for network functions, besides of proving its concept through an UC applied to a technology widely used nowadays with the advents of the NGMNs. The results obtained has been making possible the design, implementation, adaptation and optimization of the Gym framework itself and also could bring several insights on the behavior of the adopted VNFs and its underlying NFVI, besides of the challenges that remain open, as per cited in the Results Analysis of the previous chapter (Chapter 4) and that will be treated as future work. The proposed methodology and the prior visualization of the expected results, i.e., the "top-down approach", actually facilitated the development cycle of the framework, showing how to get there. The various candidate tools have converged to a few, which has been fulfilling the role of collecting generic performance data in-loco, supporting the so-called Gym-MO plug-ins. The decomposition of vIMS into "One-VM-per-Function", instead of the "All-in-One" approach, brought better conclusions about the relationships between the different VNFs that are part of the system under test. Challenges such as time synchronization, since all VNFs, Gym-MO and Gym-MN's clocks shall be aligned, the need of segmenting virtual data and control networks and the parallel collection of generic performance data through multiple Gym-MOs have been overcome. Considering the final results, it was possible, for instance, to identify the upper limit, in transactions per unit of time, for different configurations, in terms of size or flavor, of the VNFs or the VNF-FG and also identifying the best configuration (heterogeneous) for the given stressing workload.

As future work, the study of elastic and parallelized VNFs assessment can be considered, e.g. stressing a cluster/VNF-FG and then observing elastic triggers working concerning new physical infrastructures - nodes, racks or even data centers. Still, more complex scenario tests, such as dummy subscribers interaction with media transfer will be exploited just creating and pointing out new scenarios files (xml) through the task (Manager's API) and implementation of real-world network issues, such as bandwidth limitation, high latency and packet loss through Netem[1]. But the most relevant work will be implementing machine learning to learn real-world specific traffic patterns, and then using it to stress a given target system and to show the visual and/or the calculated correlation between itself and the performance of its underlying infrastructure. Even as a value proposition, but in a future work context, the support for smart self-planning/configuration/optimization - Self-Organizing Networks (SON), based on bench-

[1] https://wiki.linuxfoundation.org/networking/netem

marking over candidate server nodes. In a “Cloud Radio Access Network (C-RAN) plus virtualized Evolved Packet Core (EPC)” scenario, as e.g. an eNodeB selecting the best server System Architecture Evolution Gateway (SAE-GW) based on delay and bandwidth, where benchmarkings are instantiated just before the auto-configuration step. It is noteworthy that the UC here proposed just exploits the IMS Control Plane. Considering an implementation of all IMS functions in future works, such as that one proposed in Taylor (2014), the implementation of Session Border Controller (SBC) is required to exploits RTP media (Data Plane). It is interesting the assessment of Control Plane due to the packets sizes, most often small ones, that pass through its interfaces, since the gap of virtual interfaces (vSwitches) is to keep a high throughput around the line rate one, due to the issue of processing these small packets. Another target to be explored by Gym framework in a mobile network technology context are the several possibilities brought by the OpenAirInterface[2] initiatives.

[2] http://www.openairinterface.org/

# Bibliography

ALCATEL-LUCENT. The Journey to Packet Core Virtualization. 2014. Citado na página 22.

ALOMARI, J. Performance Evaluation of Virtualization Solutions for Real-time Applications. *2015 4th Int. Conf. Adv. Comput. Sci. Appl. Technol. Perform.*, 2015. Disponível em: <http://www.editorialmanager.com/tels/download.aspx?id=1204&guid=3656fa0c-43f7-44dc-aad5-f884dfcb8c6a&scheme=1>. Citado na página 26.

BAI, X.; LI, M.; CHEN, B.; TSAI, W. T.; GAO, J. Cloud testing tools. In: *Proceedings - 6th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2011*. [S.l.: s.n.], 2011. p. 1–12. Citado na página 23.

BOTTA, A.; DAINOTTI, A.; PESCAPÉ, A. A tool for the generation of realistic network workload for emerging networking scenarios. *Comput. Networks*, v. 56, n. 15, p. 3531–3547, 2012. Disponível em: <http://wpage.unina.it/a.botta/pub/COMNET_WORKLOAD.pdf>. Citado na página 35.

CAO, L.; SHARMA, P.; FAHMY, S.; SAXENA, V. NFV-VITAL: A framework for characterizing the performance of virtual network functions. *2015 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Network, NFV-SDN 2015*, n. 3, p. 93–99, 2016. Citado 2 vezes nas páginas 27 and 52.

CARELLA, G.; CORICI, M.; CROSTA, P.; COMI, P.; BOHNERT, T. M.; CORICI, A. A.; VINGARZAN, D.; MAGEDANZ, T. Cloudified IP Multimedia Subsystem ( IMS ) for Network Function Virtualization ( NFV ) -based architectures. *2014 IEEE Symp. Comput. Commun.*, p. 1–6, 2014. Citado 3 vezes nas páginas 40, 41, and 61.

CIMI. *The CloudNFV Proof-of-Concept Was Approved by the ETSI ISG!* 2014. Disponível em: <http://blog.cimicorp.com/?p=1553>. Citado 3 vezes nas páginas 28, 40, and 61.

COOPER, B. F.; SILBERSTEIN, A.; TAM, E.; RAMAKRISHNAN, R.; SEARS, R. In: *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*. [S.l.: s.n.], 2010. Citado na página 21.

COTRONEO, D.; De Simone, L.; IANNILLO, A. K.; LANZARO, A.; NATELLA, R. Dependability evaluation and benchmarking of Network Function Virtualization Infrastructures. In: *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015. p. 1–9. ISBN 978-1-4799-7899-1. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7116123>. Citado 4 vezes nas páginas 27, 40, 41, and 61.

CUNHA, M.; MENDONCA, N.; SAMPAIO, A. A declarative environment for automatic performance evaluation in IaaS clouds. In: *IEEE International Conference on Cloud Computing, CLOUD*. [S.l.: s.n.], 2013. p. 285–292. Citado 3 vezes nas páginas 21, 24, and 25.

DIN, G. An IMS performance benchmark implementation based on the TTCN-3 language. *Int. J. Softw. Tools Technol. Transf.*, v. 10, n. 4, p. 359–370, 2008. ISSN 14332779. Citado na página 37.

DOYLE, L. *Forecasting the NFV Opportunity.* 2013. Disponível em: <http://www.lightreading.com/forecasting-the-nfv-opportunity/a/d-id/705403>. Citado na página 19.

EL-REFAEY, M. A.; RIZKAA, M. A. CloudGauge: A dynamic cloud and virtualization benchmarking suite. In: *Proc. Work. Enabling Technol. Infrastruct. Collab. Enterp. WETICE.* [S.l.: s.n.], 2010. p. 66–75. Citado na página 21.

FERDMAN, M.; ADILEH, A.; KOCBERBER, O.; VOLOS, S.; ALISAFAEE, M.; JEVDJIC, D.; KAYNAK, C.; POPESCU, A. D.; AILAMAKI, A.; FALSAFI, B. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. *SIGARCH Comput. Archit. News*, v. 40, p. 37–48, 2012. Disponível em: <http://dl.acm.org/ft_gateway.cfm?id=2150982&type=pdf>. Citado na página 21.

GAO, J.; BAI, X.; TSAI, W. Cloud testing-issues, challenges, needs and practice. *Softw. Eng. An Int. J.*, v. 1, p. 9–23, 2011. Disponível em: <http://www.seij.dce.edu/Paper1.pdf>. Citado 2 vezes nas páginas 21 and 23.

HEWLETT-PACKARD. Seagull - Open Source tool for IMS testing. 2006. Citado na página 35.

HIRAY, S. *Network Function Virtualization.* 2014. Citado 4 vezes nas páginas 22, 40, 41, and 61.

INTEL. Intel® Open Network Platform Server Reference Architecture: SDN and NFV for Carrier-Grade Infrastructure and Cloud Data Centers. 2014. Disponível em: <http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/open-network-platform-server-paper.pdf>. Citado na página 41.

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* [S.l.: s.n.], 1991. v. 491. 685 p. ISBN 0471503363. Citado na página 34.

JAYASINGHE, D.; SWINT, G.; MALKOWSKI, S.; LI, J.; WANG, Q.; PARK, J.; PU, C. Expertus: A generator approach to automate performance testing in IaaS clouds. In: *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012.* [S.l.: s.n.], 2012. p. 115–122. Citado 3 vezes nas páginas 21, 24, and 25.

KANGARLOU, A.; GAMAGE, S.; KOMPELLA, R. R.; XU, D. vSnoop: Improving TCP Throughput in Virtualized Environments via Acknowledgement Offload. In: *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis.* IEEE, 2010. p. 1–11. ISBN 978-1-4244-7557-5. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5645469>. Citado 2 vezes nas páginas 15 and 16.

LI, A.; YANG, X.; KANDULA, S.; ZHANG, M. CloudCmp: Comparing Public Cloud Providers. *Proceedings of the 10th annual conference on Internet measurement - IMC '10*, ACM Press, p. 1, 2010. Disponível em: <http://portal.acm.org/citation.cfm?doid=1879141.1879143>. Citado na página 21.

LI, Z.; O'BRIEN, L.; ZHANG, H.; CAI, R. On a Catalogue of Metrics for Evaluating Commercial Cloud Services. In: *2012 ACM/IEEE 13th International Conference on Grid Computing*. IEEE, 2012. p. 164–173. ISBN 978-1-4673-2901-9. ISSN 1550-5510. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6319167>. Citado 2 vezes nas páginas 23 and 33.

LYNCH, P.; HAUGH, M.; KURTZ, L.; ZETO, J. *Demystifying NFV in Carrier Networks*. [S.l.: s.n.], 2014. ISBN 978-1500269814. Citado na página 19.

MATSUMOTO, C. *NFV Market Size: How's $2B for a First Guess?* 2014. Disponível em: <http://www.sdncentral.com/news/nfv-market-size-2b-first-guess/2014/04/>. Citado na página 18.

MINDCOMMERCE. *Network Functions Virtualization (NFV) Market: Business Case, Market Analysis and Forecasts 2015 - 2020*. [S.l.], 2015. Disponível em: <http://www.mindcommerce.com/network_functions_virtualization_(nfv)_market_business_case_market_analysis_and_forecasts_2015___2020.php>. Citado na página 18.

RIMAL, B. P.; CHOI, E.; LUMB, I. A taxonomy and survey of cloud computing systems. In: *NCM 2009 - 5th International Joint Conference on INC, IMS, and IDC*. [S.l.: s.n.], 2009. p. 44–51. Citado 2 vezes nas páginas 13 and 41.

RIUNGU, L. M.; TAIPALE, O.; SMOLANDER, K. Research issues for software testing in the cloud. In: *Proc. - 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2010*. [S.l.: s.n.], 2010. p. 557–564. Citado na página 21.

ROSA, R.; SIQUEIRA, M.; BAREA, E.; MARCONDES, C.; ROTHENBERG, C. Short course on NFV. Network Function Virtualization: Perspectivas, Realidades e Desafios. In: . Florianópolis, Brazil: XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 2014. Citado 2 vezes nas páginas 13 and 30.

ROSA, R. V.; ROTHENBERG, C. E.; SZABO, R. VBaaS: VNF Benchmark-as-a-Service. In: *Fourth Edition of the European Workshop on Software Defined Networks (EWSDN)*. Bilbao, Spain: [s.n.], 2015. Citado 2 vezes nas páginas 13 and 30.

SCHEUNER, J.; LEITNER, P.; CITO, J.; GALL, H. Cloud WorkBench - Infrastructure-as-Code Based Cloud Benchmarking. *2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom)*, p. 246 – 253, 2014. Citado 2 vezes nas páginas 21 and 24.

SHEA, R.; WANG, F.; WANG, H.; LIU, J. A deep investigation into network performance in virtual machine based cloud environments. In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. IEEE, 2014. p. 1285–1293. ISBN 978-1-4799-3360-0. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6848061>. Citado 2 vezes nas páginas 15 and 26.

SILVA, M.; HINES, M. R.; GALLO, D.; LIU, Q.; RYU, K. D.; Da Silva, D. CloudBench: Experiment automation for cloud environments. In: *Proceedings of the IEEE International Conference on Cloud Engineering, IC2E 2013*. [S.l.: s.n.], 2013. p. 302–311. Citado 2 vezes nas páginas 21 and 24.

SOBEL, W.; SUBRAMANYAM, S.; SUCHARITAKUL, A.; NGUYEN, J.; WONG, H.; PATIL, S.; FOX, A.; PATTERSON, D. Cloudstone: Multi-Platform, Multi-Language Benchmark and Measurement Tools for Web 2.0. *Benchmarking*, 2005. Citado na página 13.

SPIRENT. The Ins and Outs of Cloud Computing and Its Impact on The Network. 2010. Disponível em: <http://www.spirent.com/White-Papers/Broadband/PAB/Cloud_Computing_WhitePaper>. Citado na página 21.

TAYLOR, M. 2014. Disponível em: <https://privatewiki.opnfv.org/_media/wiki/vsbc_test_proposal_for_opnfv_13-nov-2014.pdf>. Citado na página 63.

THISSEN, D.; MIGUEL, J.; CARL, E. Evaluating the Performance of an IMS / NGN Deployment. *Proc. 2nd Work. Serv. Platforms, Innov. Res. new Infrastructures Telecommun. SPIRIT*, 2009. Disponível em: <http://subs.emis.de/LNI/Proceedings/Proceedings154/gi-proc-154-224.pdf>. Citado 3 vezes nas páginas 28, 36, and 61.

XILOURIS, G.; TROUVA, E.; LOBILLO, F.; SOARES, J. M.; CARAPINHA, J.; MCGRATH, M. J.; GARDIKIS, G.; PAGLIERANI, P.; PALLIS, E.; ZUCCARO, L.; REBAHI, Y.; KOURTIS, A. T-NOVA: A marketplace for virtualized network functions. In: *2014 European Conference on Networks and Communications (EuCNC)*. [s.n.], 2014. p. 1–5. ISBN 978-1-4799-5280-9. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6882687>. Citado 4 vezes nas páginas 29, 30, 34, and 35.

YIGITBASI, N.; IOSUP, A.; EPEMA, D.; OSTERMANN, S. C-Meter: A framework for performance analysis of computing clouds. In: *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009*. [S.l.: s.n.], 2009. p. 472–477. Citado na página 21.

# Glossary

**3GPP** 3rd Generation Partnership Project.

**API** Application Programming Interface.

**ASIC** Application-Specific Integrated Circuit.

**AWS** Amazon Web Services.

**BSS** Business Support System.

**C-RAN** Cloud Radio Access Network.

**CAGR** Compound Annual Growth Rate.

**CapEx** Capital Expenditure.

**CLI** Command Line Interface.

**COTS** Comercial Off-The-Shelf.

**CPU** Central Processing Unit.

**DCCP** Datagram Congestion Control Protocol.

**DPI** Deep Packet Inspection.

**DSL** Domain-Specific Language.

**DSP** Digital Signal Processor.

**EMS** Element Manager System.

**EPC** Evolved Packet Core.

**ETSI** European Telecommunication Standard Institute.

**eUTRAN** Evolved Universal Mobile Telecommunications System Radio Access Network.

**FOSS** Free Open Source Software.

**FPGA** Field-Programmable Gate Array.

**GUI** Graphical User Interface.

**HSS** Home Subscriber System.

**IaaS** Infrastructure-as-a-Service.

**IDS** Intrusion Detection System.

**IMS** IP Multimedia Subsystem.

**IP** Internet Protocol.

**ISG** Industry Specification Group.

**ISV** Independent Software Vendor.

**IT** Information Technology.

**KPI** Key Performance Indicator.

**LTE** Long Term Evolution.

**MANO** Management and Orchestration.

**MME** Mobile Management Entity.

**MNO** Mobile Network Operator.

**MVNO** Mobile Virtual Network Operator.

**NAT** Network Address Translation.

**NFV** Network Functions Virtualization.

**NFVI** Network Functions Virtualization Infrastructure.

**NGMN** Next Generation Mobile Network.

**NGN** Next Generation Networks.

**NPU** Network Processing Unit.

**NUT** NFVI Under Test.

**O&M** Operation & Maintenance.

**OpEx** Operational Expenditure.

**OS** Operation System.

**OSS** Operation Support System.

**OTT** Over-The-Top.

**P-GW** Packet Data Network Gateway.

**PoC** Proof of Concept.

**PoP** Point of Presence.

**QoE** Quality of Experience.

**RTP** Real Time Transfer Protocol.

**S-GW** System Architecture Evolution Gateway.

**SAE** System Architecture Evolution.

**SAE-GW** System Architecture Evolution Gateway.

**SBC** Session Border Controller.

**SCTP** Stream Control Transfer Protocol.

**SIP** Session Initiation Protocol.

**SLA** Service Level Agreement.

**SON** Self-Organizing Networks.

**TaaS** Testing-as-a-Service.

**TCP** Transmission Control Protocol.

**TEM** Telecommunications Equipment Manufacturer.

**TS** Tecnical Specification.

**UC** Use Case.

**UDP** User Datagram Protocol.

**VBaaS** Virtualized Network Function Benchmarking-as-a-Service.

**VIM** Virtualized Infrastructure Manager.

| | |
|---|---|
| **VM** | Virtual Machine. |
| **VNF** | Virtualized Network Function. |
| **VNF-FG** | Virtualized Network Function Forwarding Graph. |
| **VNFaaS** | Virtualized Network Function-as-a-Service. |
| **VoLTE** | Voice over LTE. |
| **VUT** | VNF Under Test. |