On Resource Description Capabilities of On-Board Tools for Resource Management in Cloud Networking and NFV Infrastructures

Kurt Tutschku, Vida Ahmadi Mehri, and Anders Carlsson
Blekinge Tekniska Högskola,
Department of Telecommunication Systems,
371 79 Karlskrona, Sweden.
Email: [kurt.tutschku|vida.ahmadi.mehri|anders.carlsson]@bth.se

Krishna Varaynya Chivukula, Johan Christenson
City Network Webbhotell AB,
Borgmästaregatan 18,
371 34 Karlskrona, Sweden.
Email: [varaynya.chivukula|johan]@citynetwork.se

Abstract: The rapid adoption of networks that are based on "cloudification" and Network Function Virtualisation (NFV) comes from the anticipated high cost savings of up to 70% in their build and operation. The high savings are founded in the use of general standard servers, instead of single-purpose hardware, and by efficiency resource sharing through virtualisation concepts.

In this paper, we discuss the capabilities of resource description of "on-board" tools, i.e. using standard Linux commands, to enable OPEX savings. We put a focus on monitoring resources on small time-scales and on the variation observed on such scales. We introduce a QoE-based comparative concept that relates guest and host views on "utilisation" and "load" for the analysis of the variations. We describe the order of variations in "utilisation" and "load" by measurement and by graphical analysis of the measurements. We do these evaluations for different host operating systems and monitoring tools.

I. Introduction

The anticipated high cost savings of up to 70% in build and operation of "cloudified" and *Network Function Virtualisation (NFV)* based networks [12] make their rapid adoption highly probable, although there is significant scientific criticism on the latter concept [11]. The high savings are founded at first in the use of general standard servers, instead of single-purpose hardware. However, major additional savings are expected by the flexibility of these servers and by the efficiency of resource sharing on them through virtualisation concepts [7]. The flexibility and efficiency of the new infrastructure are expected to lead to a faster service deployment, thus generating new revenue streams for operators [3]. Simplified, cloudification and NFV enables network functions, such as HTTP-caches, VPN gateways or concentrators, MMEs (Mobility Management Entities in 4G mobile networks) or BRAS (broadband remote access servers), to be implemented as files rather than in hardware. These files can be executed on standard servers as well as in virtualization environments. So-called *virtual network functions (VNFs)* may now run at arbitrary locations.

Practically, virtualisation is achieved by implementing the VNFs in *virtual machines (VMs; often also denoted as "guests" which are executed on "hosts")* or by *containers*. These VNFs will be operated in large data centres (DCs), which host a very large number of servers. General virtualisation environments for computer systems, like XEN [5] or KVM [21], are used for the execution of VMs. Containers have originated from high performance computing for Cloud services, e.g. Dockers [23]. However, ETSI's Industry Specification Group (ISG) on NFV [13], is currently preferring VMs over containers.

The *economics of scale in DCs* [17] are a key contribution to the cost savings. This feature assumes that it is easy to manage and share computing resources in DCs. The more resources are available in DCs, the more cost-effective the operation becomes. The efficiency of servers in DCs are often described by the terms "load" or "utilisation". These values are readily available as output from standard, so-called "on-board" tools, such as `mpstat` [16] or `top` [22] in modern operating systems. Their availability leads to simple and obviously practical resource management strategies, such as "put as many VNFs on a server until the system reaches a certain load". However, these tools typically haven't been designed with the stringent requirements, e.g. on small time scales, for resource efficiency management.

The aim of the paper is not to explain the nature and fundamentals of load monitoring. The aim is rather on evaluating the applicabilities of standards with respect to load variations resulting from the nature of the tools or of the operating systems. Good overview on load monitoring principles can be found in [27] or [1].

We will show in this paper evidence that a simple understanding of these tools, i.e. without considering their inherent variability, might not be appropriate for efficient resource management in cloudified infrastructures. We will argue that VNFs might require more comprehensive resource usage descriptions, e.g. *VNF profiles* [25]. We also demonstrate that the accuracy of the current on-board tools increases with a high utilisation in guest and host. However, users and performance evaluation typically consider high load as harmful. Thus, VMs are expected to be operated at low load levels and in turn, the accuracy and usefulness of the current load and utilisation monitoring tools in this scenario are in doubt.

We base our discussions on measurements that been carried out in experiments, which are close to the configurations of DCs operated by CityNetwork Webbhotell AB [9]. We will show results that outline the variability of load and utilisation monitoring. We suggest a correlation technique to investigate the variability, which is considered the QoE (as seen by the guest) and the QoS (as seen by the host) [14]. We also do these evaluations for different popular Linux distributions in DCs, such as `Ubuntu` [6] and `CentOS` [26].

We will discuss in Sec. II the time-scale requirements for resource monitoring, possible techniques for resource descriptions and monitoring, a comparative concept to identify load relationships between guests and hosts, and how to generate load for testing. Sec. III details the aims of the experiments and the experiment setup. Sec. IV discusses the results, while Sec. V provides a brief summary and outlook to future work.

II. Requirements and Methods for Resources Descriptions

A. Timescale Requirements for Resource Management

We assume that the success of cloudified and NFV system's origins mainly from their anticipated cost-savings. Costs occurring in networks are usually classified into CAPEX (capital expenditures) and in OPEX (operational expenditures). CAPEX describes mainly the initial non-recurring expenditures in network equipment, infrastructure, hard-, software, buildings, or ground works, which are enjoyed over a long time. CAPEX savings in cloudified and NFV systems are related mainly to the dimensioning of the infrastructure, e.g. the required number of servers to maintain a defined service level. CAPEX typically accounts for 20% of the total costs of networks [8].

OPEX costs are the ongoing costs of running networks. They comprise consumables (incl. spare parts), utilities (power or cooling), labor cost, but also maintenance and facility expenses. OPEX costs might sum up to 80% of the total costs. They occur regularly and might be controlled instantaneously, i.e. on seconds or minutes, e.g. by shutting down un-used servers and saving their utilities costs. Possible OPEX savings in cloudified and NFV systems are diverse. First, the DCs economics of scale lead to a smaller number of administrators, which need less specialised skills and thus, consequently might incur lower labor costs. Second, the independence from execution location permits operation at venues with lower cost, e.g. lowest energy costs. Finally, fine grained resource management in cloudified systems might lead to reduced energy consumption, e.g. VNFs can be moved to servers which are not fully loaded while unused servers are shutdown.

This short discussion shows that the full potential of cost savings in cloudified systems will probably achieved when OPEX reductions are implemented on short time-scales, i.e. seconds or minutes. Hence, the descriptions and monitoring tools for resource consumption to need to obey these time-scales.

B. Resource Descriptions

Next, we will outline which resources should be described and how this should be done. Therefore, we will discuss the type and nature of the resources (i.e their semantics) and formal description concepts (i.e. their syntax). The applied description concept will use statistical notations.

1) Resources Categories and Profiles: We describe the resources by assuming that virtualisation is done by the VMs. Hence, we derive the resource categories from the configuration options of the `virt-install` command, which is a typical tools to create a guest [2]. The configuration options can be classified into: a) CPU, b) memory, c) storage, d) input/ output (I/O), e) network, f) operating system and g) other fields, cf. Table I.

Area	Option	Description	Used here
CPU	`-vcpus`	Number of vCPUs	X
memory	`-ram`	RAM to allocate (MB)	
storage	`-disk`	specify storage media	
	`-filesystem`	export a host directory to the guest	
	`-file`	Installation media location (local install)	
	`-location`	Installation via distribution tree (network install)	
network	`-network`	Host network	
operating system	`-os-type`	OS type	
	`-os-variant`	OS variantversion	
I/O	`-graphics`, `-nographics`	Graphical display method	
Other	`-name`	Virtual machine name	

TABLE I: Resource categories and configuration options in `virt-install`

In this contribution, we will consider only available and used CPU capacity as system state (either by "utilisation" or "load"). Herby, the "host capacity" is the *available physical CPU* on the host running the VM and the "guest capacity" is the assigned number virtual CPUs to a guest. However, it is obvious from Table I that more complex resource descriptions are needed that comprise multiple resource categories. This need has led to the concept of a "VNF profile" [25]. We focus intentionally at CPU only in order to find initial relationships.

2) Sampling Concept and Statistical Characterisations: The requirement for describing the system state in the order of seconds to minutes in cloudified systems sets the scope of the monitoring and sampling concept. We adopt a sampling concept similar to typical steady-state simulation analysis [4]. The sampling concept is depicted in Figure 1. It comprises a start-up phase and stop phase in order to reach a state-steady and to avoid interference with the load generation, cf. Section II-D. The inter-sampling interval Δt is currently chosen as constant, but easily be changed to random and independent intervals [4].

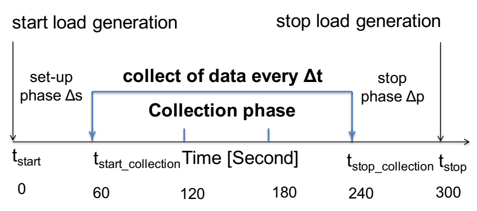

Fig. 1: Sampling concept for CPU utilisation and load

The statistical characterisations for status description are considered initial for this paper and might require further improvements. Due to the objective of simplicity of status monitoring, however, we start with simple but major statical values: average *E[X]*, minimum *Min[X]* and maximum *Max[X]* to quantify the mean and the range of the variations in the sampling interval. Hereby, X is the random variable of the observed state:

$$E[X] = \sum_{i=1}^{N} x_{s,i}/N \quad (1)$$

$$Min[X] = x_{\min} \quad \text{with} \quad \{\forall x_i \in X : x_{\min} \leq x_i\} \quad (2)$$

$$Max[X] = x_{\max} \quad \text{with} \quad \{\forall x_i \in X : x_{\max} \geq x_i\} \quad (3)$$

$$\sigma[X] = \sqrt{E[(X - E[X])^2]} \quad (4) \qquad cov[X] = \frac{\sigma[X]}{E[X]} \quad (5)$$

with x_i is the observed state at $t = t_{\text{start \_collection}} + \Delta t * (i-1)$, $N = \lfloor (t_{\text{stop \_collection}} - t_{\text{start\_collection}})/\Delta t \rfloor$ and $i \in [1, ..., N]$.

C. Tools for Load and Utilisation Monitoring

1) Notions for CPU Utilisation and CPU Load: *CPU load* is the concurrent number of processes using the computational resources on a computer. It is computed by summing up the number of running threads and the number of waiting threads. Typically, the CPU load value should be between 0 and the numbers of available CPU cores. This requirement assures that no resources are wasted by queuing of processes. It should be noted that the CPU load output from monitoring tools, see below, is typically a weighted average of the number of served processes at subintervals during a measurement.

CPU utilisation is denoted by the amount of accumulated time a CPU is busy for handling work during a specific interval. It is reported as a percentage. It can be calculated as the time a CPU is idle, the time a CPU is running a user-level application or the system-level threads for each processor are served, cf. [18].

The CPU utilisation ratio is calculated by dividing amount of accumulated CPU time by the observation time interval of the measurement interval. Almost each tool applies its own calculation and may not necessarily calculate similar to other tools use. For purpose of simplification, we call the CPU utilisation ratio simply as CPU utilisation.

2) Tools:

MPstat: The tool was used for monitoring utilisation and is part of the `sysstat` utilities [16]. It reports individual or combined processor statistics. The used `MPstat` command is: `mpstat -P ALL T1 T2`. The option `-P ALL` specifies that the usage of all processors will be monitored. The value `T1` details the inter-sampling interval and corresponds to the Δt in Figure 1. The parameter `T2` specifies the duration of the measurement. `MPstat` has a multi-column output, comprising the CPU number, percentages of CPU utilisation at `%user/%system/%iowait` levels, and `%idle` ratio. We used in our monitoring the value of $(100 -$ `%idle`$)$, since the `%idle` value was reported in percentages.

top: We used the tool to monitor load and utilisation [22]. It is the default tool in `Linux` distributions for real-time view of a system. It provides a summary and details a list of processes which are handled by the `Linux` kernel. The used `top` command is: `top -b -d T1 -n T2`. The option `-b` starts a batch for monitoring and it is used for sending output to a file. The value `-d T1` provides the sampling interval `T1`, which again corresponds to Δt in Figure 1. The parameter `-n T2` specifies the number of samplings. `top` shows the load average of 1, 5, and 15 minute intervals. We consider the 1-minute-average in our measurements.

D. Load Generation

Imposing load and stress on computer systems has regained popularity with the improved resource sharing capabilities of virtual systems. Such stress tests aim at high loads and can be used for performance fine-tuning, testing of physical and virtual hardware components, or measuring power consumption, cf. [15]

For our investigations will use the `stress-ng` tool, which is a clean room implementation of the original `stress` tool, cf. [20]. It offers a wide range of tests for CPUs, storage and drive systems, I/O syncs, memory and processes.

Since our investigation focuses on load and utilisation of the physical/virtual CPUs or cores, we used the CPU test as defined by the `-c N, -cpu N` options. The used command was: `stress-ng -c N1 -l P -t N2`. The value `N1` is the number of processes, which load the (virtual)CPUs by calculating the computational demanding function `sqrt((double)rand())`. The value `P` specifies the load imposed on the CPU by a process and is given in percentage (%). It ranges between 0 (=sleep) and 100 (=full load). The value `N2` stops the stress test after $N2$ seconds.

E. Comparative Investigation of Load and Utilisation Relationships

A major aim of resource management in cloudified infrastructures is to reduce operational costs by increasing load and utilisation while maintaining expected performance. Levelling resource consumption and performance is already difficult in non-virtualised systems. However, finding such trade-offs in virtualised systems is even more complex due to the higher number of types of resource, their complex interactions (dependencies of network throughput on CPU capacity due virtual hardware) and due to the nesting of virtual resources in physical ones.

In order to make the analysis, independent from the type of the resource, relationships among resource categories and from nesting, we define a *comparative concept* to relate aims and relationships of users and operators in virtual infrastructures. The comparative concept is a kind of "black box" approach [24] for system analysis. Hence, it doesn't require any detailed knowledge about the investigated resource (sharing) mechanisms.

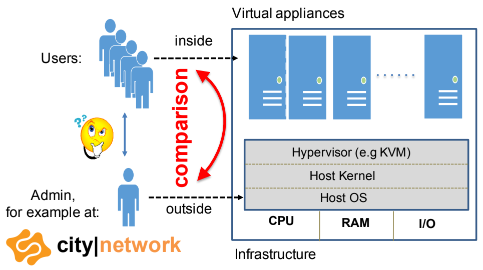

Fig. 2: Comparative Concept to Investigate and Model the Utilisation and Load Relationship in virtual Environments

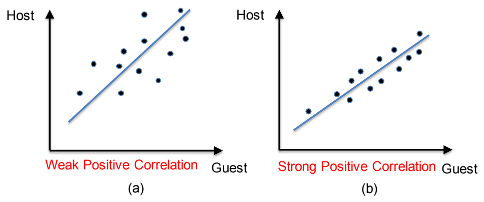

Fig. 3: Scatterplots to Compare Guest and Host Relationships

1) The Comparative Concept: The comparative concept correlates the views of a user of VM (i.e. a person or entity observing the VM's performance) and of the operator (i.e. administrator) surveying the physical infrastructure. We call the view from within the VM the "inside" or "Guest" view and the view on infrastructure as the "outside" or "Host" view. The aim of the concept is to compare the "inside" with the "outside" and to identify correlations between these views. The described the concept is depicted in Figure 2. The compact is a derivative of methods to investigate the QoE/QoS relationship in networks [14], [19].

2) Correlation Analysis Using Scatter Plots: The "inside/outside" relationships researched in this contribution are analysed by simple but powerful *scatter plots* [10]. Scatter plots are two-dimensional diagrams using Cartesian coordinates to display a set of points. These points are defined by a tuple (x_t, y_t), i.e. a pair of concurrently observed variables X and Y at time t.

The tuple $(sample_{\text{inside/ Guest},j}, sample_{\text{outside/Host},j})$ is the pair of simultaneous load or utilisation measurements at guest and host at time j. Examples of correlations identifiable by scatter plots are shown in Figure 3. Figure 3(a) shows a weak positive correlation, i.e. the points are loosely around the increasing line, and Figure 3(b) depicts a strong positive correlation, i.e. the points are quite close to the line.

III. Experiments

A. Aims and Research Questions

The experiments carried out with respect to these questions:

1) Is there a predictable, i.e. strong positive correlation of load and utilisation between guest and host? Here, "scaling" means, whether an increase in guest load results in a predictable increase of the host.
2) How strong do the observed utilisation values vary when observing them during a small sampling interval?

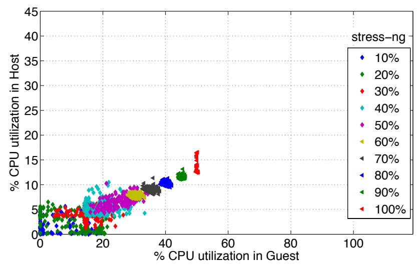

Fig. 4: Relationship for utilisation; one loading process on guest; host: Ubuntu; tool: mpstat

3) Does the accuracy of the prediction depend on guest load levels?
4) Does the scaling depend on the number of stressed vCPUs, i.e. on the resources available to the VM?
5) Does the scaling depend on host operating systems?
6) Are there significant differences between the monitoring tools?

B. Experiment Setup

We carried out the experiments on a single standalone server that is similar to the ones operated in DCs of CityNetwork Webhotell AB. The key host and guest specifications are depicted in Table II. The server was running either `Ubuntu 12.04 LTS Desktop` [6] or `CentOS 6.6 Desktop` [26]. In both cases we used `kvm` [21] as the virtualisation environment. The guests were allocated two virtual CPUs, 2GB of virtual RAM, and 30GB of virtual disk. This assigning is based on the KVM provisioning model, which is based on CPU cores. The guest used `Ubuntu 14.04 LTS Server`.

Criteria	Specification
CPU	Intel ® Xeon ® CPU EC-1230 v2 @ 3.30GHz 8 cores
RAM	8GB
Hard disk	500 GB SAS
Hypervisor	KVM
Host OS	CentOS 6.6 Desktop / Ubuntu 12.04 LTS Desktop
Guest OS	Ubuntu 14.04 LTS server

TABLE II: Specifications for host and guest.

C. Common Experimentation Parameters

Throughout the experiments, we increased utilisation in the guest by using the `stress-ng` command in 10% step from 10% to 100%. Each step was iterated and measured at least 50 times and the load was imposed for 5min in order to have a small time-scale. Finally, the set-up and stop phases Δs and Δp were 1min and the inter-sampling interval Δt was 1sec, cf. Figure 1. Guest and host did no other computing task other than standard OS tasks.

IV. Results

A. Predictability of Scaling

The predictability of the scaling of utilisation was observed using `mpstat` and is depicted in Figures 4 and 5. Figure 4 shows if the VM is loaded with 50% (i.e. 100% load from one

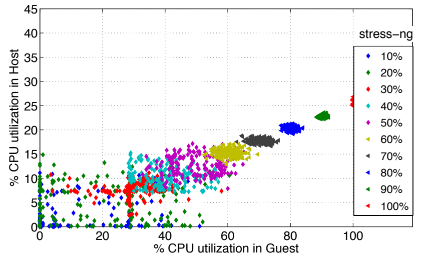

Fig. 5: Relationship for utilisation; two loading processes on guest; host: Ubuntu; tool: mpstat

process) then the host is loaded at about 12%. This behaviour can be expected for a eight core host and for a VM, which is configured with two vCPUs. However, the figures also reveal that the observed utilisations are not deterministic for the load levels and vary strongly. Hence, we conclude that there is predictable scaling, i.e. positive correlation, but this is weak and even weakens when the load on the guest is reduced. Tables III and IV provide the numerical values to the statistical characteristics in Figures 4 and 5. These values shows that `stress-ng` is accurate in generating the expected load E[X] on the guest (in %; E[] is abbreviated by ∅ in the tables) but the *cov* is similarly high for guest and host (above 0.2) for guest loads less than 30%. Hence, we conclude that the prediction accuracy decreases significantly with lower guest load.

	Guest [X]				Host [Y]			
%	∅	cov	min	max	∅	cov	min	max
10	5,20	1,33	0,00	20,6	1,68	1,04	0,00	5,62
20	11,0	0,64	0,00	23,6	2,95	0,62	0,12	6,50
30	15,8	0,24	4,50	26,5	4,11	0,22	1,25	6,25
40	20,5	0,20	14,0	33,3	5,57	0,24	3,63	10,5
50	25,3	0,17	16,0	36,5	6,66	0,18	4,26	10,3
60	30,4	0,04	28,0	33,0	7,79	0,04	7,07	8,70
70	35,5	0,03	32,7	38,2	9,02	0,05	8,08	11,4
80	40,2	0,21	38,3	42,9	10,3	0,04	9,47	11,3
90	45,3	0,10	44,1	46,5	11,5	0,24	11,1	13,2
100	50,1	0,00	49,8	50,5	13,0	0,07	12,5	16,5

TABLE III: Characterisation of utilisation at guest and host; one loading process; host: Ubuntu; tool: mpstat

	Guest [X]				Host [Y]			
%	∅	cov	min	max	∅	cov	min	max
10	10,8	1,33	0,00	51,0	2,85	1,22	0,00	11,1
20	20,4	0,81	0,00	60,5	5,37	0,75	0,00	14,9
30	30,5	0,25	3,5	58,0	7,88	0,20	2,38	11,1
40	40,3	0,19	28,3	57,3	10,4	0,19	7,13	15,3
50	50,3	0,15	32,8	65,8	12,8	0,16	7,88	17,2
60	60,4	0,05	52,5	69,5	15,3	0,05	12,8	17,1
70	70,2	0,04	62,9	76,4	17,8	0,03	16,7	18,9
80	80,2	0,02	77,1	84,0	20,3	0,02	19,1	21,3
90	90,1	0,01	88,3	92,2	22,8	0,01	22,2	23,8
100	100	0,00	100	100	25,3	0,02	25,0	26,9

TABLE IV: Characterisation of utilisation at guest and host; two loading processes; host: Ubuntu; tool: mpstat

The measurements show that monitoring of utilisation is possible on small time-scales only as long as the VM is highly

	Guest [X]				Host [Y]			
%	∅	cov	min	max	∅	cov	min	max
10	8,11	1,01	0,00	28,5	3,96	0,73	0,06	13,6
20	15,5	0,50	0,00	32,8	6,98	0,48	0,81	16,3
30	30,5	0,18	11,3	43,3	9,57	0,20	3,51	17,6
40	40,4	0,12	30,1	54,4	12,0	0,17	8,69	20,2
50	50,4	0,12	34,4	67,8	14,4	0,16	10,1	22,1
60	60,2	0,05	53,3	66,9	15,8	0,06	13,8	20,5
70	70,3	0,09	54,0	84,4	26,2	0,45	16,5	49,9
80	80,2	0,02	75,5	84,5	21,2	0,05	19,8	25,0
90	90,1	0,01	88,4	92,1	23,3	0,03	22,4	25,8
100	100	0,00	100	100	26,7	0,06	25,4	33,4

TABLE V: Characterisation of utilisation at guest and host; two loading process; host: centOS; tool: mpstat

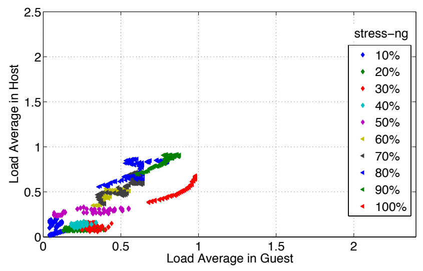

Fig. 6: Relationship for load; one loading process on guest; host: Ubuntu; tool: top

loaded. Moreover, the uncertainty might increase when stressing multiple vCPUs. This is evidenced higher *cov* values on the guest and host for two stressing process and guest utilisation below 20%. This observation shows that the accuracy depends on the guest load level, but also on the type of guest load. Hence, this might indicate that a more detailed description load is needed for describing VNFs. Thus, the suggest VNF profiles [25] need to address the load type as well.

B. *Predictibility of Load*

The predictability of the scaling of load monitored by `top` is depicted in Figures 6 and 7. They confirm a similar weak positive correlation as for utilisation. However, other effects that lead to variations and which are systematic, are also observable. For example, the load at 100% in Figure 6 seems to be asymptotic. These effects need further research.

C. *Dependency on Host OS*

The dependency on the host operating system can be analysis by comparing Figures 5 and 8. Figure 8 and the related statistical characterisations in Table V reveal that `CentOS` has typically higher *cov* at higher utilisation values than `Ubuntu`. Although `CentOS` is typically considered as a better choice as a host operating system, our measurements show that it exhibits a higher variation and in turn a lower accuracy when describing utilisation. Therefore, `Ubuntu` seems to be a better choice when it comes to load optimisation.

D. *Differences Between Tools*

The comparison Figures 4 and 5 for utilisation and Figures 6 and 7 for load indicates that there might be an advantage

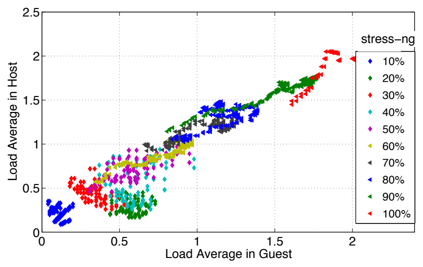

Fig. 7: Relationship for load; two loadings processes on guest; host: Ubuntu; tool: top

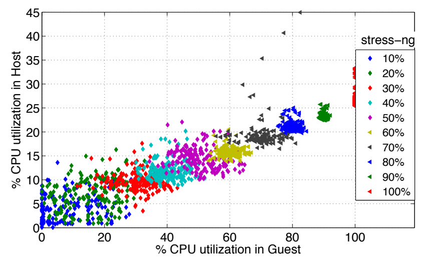

Fig. 8: Relationship for utilisation; two loading processes on guest; host: CentOS; tool: mpstat

for the `top` tool due lower visual variations. However, the advantage might be doubtful due to the apparent systematics in the load relationships. Hence, the comparison of the tools requires further research.

V. Conclusion

This contribution outlines the capabilities of on-board Linux tools for describing of resource utilisation in cloud networking and NFV infrastructures. We argued that resource descriptions, monitoring and management need to address small time-scales in order to obtain high OPEX savings. Furthermore, we introduced for the first time a QoE-based comparative concept to relate guest and host views for utilisation and load.

We have shown by this concept that there is a predictable but weakly correlated relationship for utilisation and load between host and guest VM. Moreover, the accuracy decreases for low guest load levels. Hence, resource utilisation on the host might be overestimated , i.e. the infrastructure might not exhibit the required efficiency. Accurate monitoring on a small, i.e. 5min, time-scale is possible at high load level, but more accurate monitoring is suggested at low load levels. Our measurements outlined significant differences in scaling and predictability between host operating systems and between tools. Hence, `Ubuntu` seems to be a more appropriate choice as host OS when it comes to resource management. The comparison of tools, however, needs further research.

The level, but also the kind of stress, i.e. in terms of stressing processes, determines the utilisation of the infrastructure. Hence, future research should be devoted to identifying what kind of NFV profiles and benchmark loads should be defined!

References

[1] A. Lewis. Understanding Linux CPU Load - When should you be worried? Information at http://blog.scoutapp.com/articles/2009/07/31/understanding-load-averages, 2009.

[2] Anonymous. libvirt – The virtualization API. Information and code available at http://libvirt.org, 2015.

[3] AT&T Inc. AT&T Vision Alignment Challenge Technology Survey – AT&T Domain 2.0 Vision White Paper. Information available at http://www.att.com/, November 2013.

[4] J. Banks, J. Carson II, B. Nelson, and D. Nicol. *Discrete-Event System Simulation*. Prentice Hall, Upper Saddle River, NJ, 2009.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5), 2003.

[6] Canonical Ltd. Ubuntu. Ubuntu. Information at http://www.ubuntu.com/, 2015.

[7] B. Chatras. Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the Application of Different Virtualisation Technologies in the NFV Framework . ETSI GS EVE 004 V0.2.0 – draft available at https://docbox.etsi.org/, 2015.

[8] CISCO Inc. Network Procurement: The Journey from CAPEX through TCO to Business Value. White paper available at http://www.cisco.com/c/dam/en/us/solutions/collateral/executive-perspectives/executive-perspectives/network-procurement-cfo.pdf, 2012.

[9] City Network Hosting AB. Agility Through Open IT-Infrastructure. Information at https://www.citynetworkhosting.com, 2015.

[10] W. Cleveland. *Visualizing Data*. Hobart Press, 1993.

[11] J. Crowcroft. j'accuse NFV. Post to E2E mailing List, Information available at http://http://www.postel.org/pipermail/end2end-interest/2015-April/009289.html, 2015.

[12] ETSI. Network Functions Virtualisation – An Introduction, Benefits, Enablers, Challenges & Call for Action. Information available at http://portal.etsi.org/NFV/NFV_White_Paper.pdf, 2012.

[13] ETSI. Network Functions Virtualisation – Collaborative Portal. Information available at http://portal.etsi.org/portal/server.pt/community/NFV/367, 2013.

[14] M. Fiedler, K. Tutschku, and P. Carlsson. Identification of Performance Degradation in IP Networks Using Throughput Statistics. In *the 18th International Teletraffic Congress - ITC18*, Berlin, Germany, 2003.

[15] V. Gite. How To Stress Test CPU and Memory (VM) On a Linux and Unix With Stress-ng. nixCraft – Linux Howto's Guide – Information available at http://www.cyberciti.biz/faq/stress-test-linux-unix-server-with-stress-ng/, Sep. 2015.

[16] S. Godard. Sysstat Utilities Home Page! Information and code available at http://sebastien.godard.pagesperso-orange.fr/index.html, 2015.

[17] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel. The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, December 2008.

[18] B. Gregg. *Systems Performance: Enterprise and the Cloud*. Pearson Education, 2013.

[19] T. Hossfeld, D. Hock, P. Tran-Gia, K. Tutschku, and M. Fiedler. Testing the iqx hypothesis for exponential interdependency between qos and qoe of voice codecs ilbc and g.711. In *18th ITC Specialist Seminar on Quality of Experience*, volume s. 105-114, 2008.

[20] C. King. stress-ng. GitHub – Information and software available at https://github.com/ColinIanKing/stress-ng, Nov. 2015.

[21] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.

[22] W. LeFebvre. Top. Information and code available at http://sourceforge.net/projects/unixtop/, 2003.

[23] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239), March 2014.

[24] R. Patton. *Software testing*. Sams Pub., 2006.

[25] R. Rosa and C. Rothenberg and R. Szabo. VNF Benchmark-as-a-Service – draft-rorosz-nfvrg-vbaas-00. Information available at https://www.ietf.org/internet-drafts/draft-rorosz-nfvrg-vbaas-00.txt, 2015.

[26] The CentOS Project. CentOS. Information at https://www.centos.org, 2015.

[27] Ray Walker. Examining load average. *Linux J.*, 2006(152), December 2006.