OpenStack extensions for QoS and energy efficiency in edge computing

Alessandro Carrega, Matteo Repetto and Giorgio Robino
S3ITI National Lab, CNIT
Via Opera Pia 13,
16145 Genoa, Italy
Email: {alessandro.carrega, matteo.repetto, giorgio.robino}@cnit.it

Giancarlo Portomauro
DITEN, University of Genoa
Via Opera Pia 11,
16145 Genoa, Italy
Email: giancarlo.portomauro@edu.unige.it

***Abstract*—Mobility has always been a big challenge in cellular networks, because it is responsible for traffic fluctuations that eventually result into inconstant resource usage and the need for proper Quality of Service management. When applications get deployed at the network edge, the challenges even grow because software is harder to hand-over than traffic streams. Cloud technologies have been designed with different specifications, and should be properly revised to balance efficiency and effectiveness in distributed and capillary infrastructures.**

In this paper, we propose some extensions to OpenStack for power management and Quality of Service. Our framework provides additional APIs for setting the service level and interacting with power-saving mechanisms. It is designed to be easily integrated with modern software orchestration tools and workload consolidation algorithms. We report real measurements from an experimental proof-of-concept.

***Index Terms*—Energy efficiency, QoS, mobile edge computing**

I. Introduction

Edge computing is an essential element in the evolution towards 5G infrastructures, bringing an effective solution for a number of challenging requirements that cannot be satisfied by the legacy cloud paradigm (e.g., geographical distribution, computing proximity, transmission latency). In this respect, effective management of quality of service is essential to guarantee the required performance levels during the whole service lifetime: placement, CPU time, and network bandwidth are key factors to effectively support smart manufacturing, e-health, energy, automotive, and media and entertainment industries [1].

With the grow of user-centric services, characterized by interactivity and strict constraints on latency, the workload is expected to follow user distribution and mobility patterns, hence leading to uneven, unsteady, and non-uniform distribution within the network. Consequently, the usage of peripheral installations is expected to dynamically vary with hourly, daily, weekly, and seasonal periodicity.

Telecommunication infrastructure have always suffered from large variations of traffic streams. Overprovisioning has been the common 'least-effort' relief for long, but it leads to poor utilization of the overall system, raises power consumption and crumbles the overall efficiency, hence it is no more acceptable today in its current form, both for environmental and economical reasons [2]. The real problem behind overprovisioning is the need to run idle devices, which results in poor efficiency in terms of power consumption per actual computation load [3]. Such inefficiency is also intrinsic in the cloud paradigm, but it is far more severe in edge computing due to the variable workload within the system and the large number of installations.

In this paper we address the design of virtualization frameworks for edge computing, by focusing on the need to balance Quality of Service (QoS) with Energy Efficiency (EE). We propose a set of functional and semantic extensions to OpenStack, a well-known cloud management software, which enable better management of QoS and EE. We extend our previous work on power measurements and monitoring [4], by including external APIs for controlling the network behavior and power-saving mechanisms available in compute and network nodes. We also report a larger set of measurements and trials over the experimental infrastructure.

The paper is organized as follows. We review related work in Section II. We briefly discuss the main limitations of current cloud software for edge computing in Section III. We then describe the implementation of the energy-efficient infrastructure, which provides support for QoS and power management in Section IV. We report evaluation and numerical analysis from both the testbed and simulations in Section V. We briefly discuss the usefulness of our work for advanced consolidation strategies in Section VI. Finally, we give our conclusions and plans for future work in Section VII.

II. Related work

The need for more power efficiency in ICT infrastructures largely comes from telecommunications, where capillary installations are ineffectively used by periodic traffic patterns [2]. Fog and mobile edge computing will raise the same issue for cloud installation as well, since the workload for peripheral installation will be much more variable and unpredictable than in big data centers.

Even though many power-saving mechanisms (e.g., voltage and frequency scaling, low-power idle, and stand-by states) have been available for long, their effectiveness in ICT systems is limited if not properly integrated in system management. Workload consolidation, packet re-routing, and network connectivity proxying [3], [5]–[7] are the most effective strategies

978-1-5386-5896-3/18/$31.00 ©2018 IEEE

to tackle the well-known non-linearity between performance and power consumption of both computing and networking devices of an entire infrastructure in a coordinated way [8].

Despite the availability of several algorithms for workload consolidation, both taking into account computing resources (e.g., [9]–[11]), telecommunication networks (e.g., [12]–[14], or the combination of both computing and networking devices (e.g., [15]–[19], small effort has been devoted to design energy efficient infrastructures that provide required functionality to apply such algorithms, mainly limited to networking devices and protocols [4], [20], [21]. For what concerns edge computing, we can refer to existing cloud management software. VMware vSphere include both host and system-wide power management, which can be used alone or in combination [22], [23]. The former saves energy by placing certain parts of a computer system or device into a reduced power state when the system or device is inactive or does not need to run at maximum speed; the latter redistributes virtual machines among physical hosts in a cluster to enable some hosts to be powered off completely. No power management interfaces are currently available in the official OpenStack release.

Energy efficiency cannot be reduced just to the lowest power consumption: it is a trade-off between energy and performance, the latter being directly related to Quality of Service and Quality of Experience for the users. In light of the expected convergence between cloud and wide-area networking for building mobile edge computing, on-demand provisioning with guaranteed service level agreements and quality of service has been extensively studied [24]–[27]. VMware vSphere filters and tags IP packets at virtual switches [28], while OpenStack also provides rate limiters and bandwidth shapers in the hypervisor. In both cases, there is no direct interaction with networking devices for resource reservation.

III. Limitations of existing technologies

While static provisioning and manual configuration might still be an acceptable administrative and technical practice in many data centers with specific business models (e.g., tailored and high-performance cloud services), they do not fit the dynamicity and unpredictability of mobile edge computing. As a matter of fact, mobility patterns of users are expected to translate into dynamic workloads for latency-sensitive and bandwidth-hungry applications and services deployed at the network edge. Overdimensioning each edge installation will be unavoidable but, given the capillarity and the extension of such infrastructure, efficient resource usage will be crucial for both environmental and economical sustainability of this paradigm. Several frameworks for edge computing are currently porting existing cloud management software to edge installations as well, mainly to maintain compatibility and exploiting well-tested tools. Unfortunately, most automated tools treat uniformly all service instances, without the possibility to specify and guarantee QoS constraints. As a matter of fact, a past study has revealed that performance variance in cloud services (IaaS) is at an unacceptable level particularly when deploying performance-sensitive applications [29].

Recent software-development paradigms build on modular design, micro-services architectures, and software orchestration to dynamically scale services to the current workload, both for cloud applications and network function virtualization [30]–[34]. Though this approach is a basic requirement for efficient usage of virtual and physical resources, it still needs proper support from the underlying infrastructure to really result into energy saving and effective QoS management. Indeed, such design envisions the creation of multiple software components, each with its specific functionality, that can be easily replicated in an automated way by the orchestration process to achieve horizontal scalability, high-availability, and resilience. Provisioning such instances in real-time is not currently possible (though unikernel technologies leave this possibility open for the future), because of technical and administrative issues; indicatively, it usually takes from minutes to days to have a VM ready. Hence, the alternative is either reactive provisioning after need, with a delay that might not be acceptable for most demanding applications, or proactive provisioning, which implies to run idle instances that might even be never used.

Commercial cloud management software already provides specific features for workload consolidation and power management (see Section II), but do not provide an effective solution for the dynamic provisioning problem described above. This is because it does not know what each software instance is meant for, i.e., if it is actually running or it is a spare component to be used in case of sudden peaks of requests, failure of the primary components, movement of users to a different location, etc. We argue that energy efficiency could be largely improved by including such information in the virtualization model.

Another typical limitation of existing cloud installations is the loose integration between networking and computing. Networking-as-a-Service functions create virtual instances of switches, routers, firewall, etc. They usually provide overlay links for each tenant, but require manual and static configuration of the underlying physical devices (e.g., VLANs, provider networks, routing, QoS paths), apart some plugins designed for specific vendors (e.g., OpenStack ML2 plugins for Brocade and Nexus devices). There is no way to automatically map QoS requirements from applications into network configuration, in order to check if enough resources (link bandwidth and switching capacity) are available to guarantee specific throughput, latency, and jitters. This limitation is mainly due to the large heterogeneity and lack of standardization of interfaces and APIs for configuring network devices; however, network controllers and software-defined protocols are now available to program the underlying network infrastructure.

IV. Energy-efficient infrastructure

To properly balance service levels and energy efficiency in high-mobility environments like edge computing, we have designed and set up an energy-efficient virtualization infrastructure that addresses the main issues discussed in Section III. It is conceived to extend existing virtualization paradigms

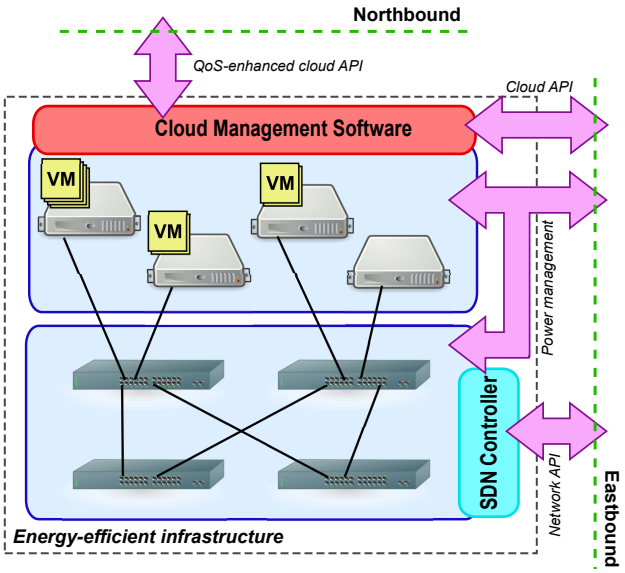


Fig. 1: Architecture of the energy-efficient infrastructure.

by allowing better control on network traffic and power management. The energy-efficient infrastructure is specifically conceived to be used by advance consolidation algorithms to achieve optimum balance between QoS and Energy Efficiency [35]; some indicative examples of its usefulness and usage are briefly described later in Section VI.

The energy efficient infrastructure is a mix of hardware and software components that includes:

- computing hardware and hypervisors, with power-saving mechanisms;
- cloud management software (CMS) that implements the Infrastructure-as-a-Service (IaaS) model;
- software-defined network (SDN), fully programmable for optimizing network paths according to bandwidth requests and with power-saving mechanisms;
- monitoring, which collects measurements about power consumption and CPU utilization from all servers and network devices;
- management interfaces, which allow interaction both for using and managing the infrastructure.

Fig. 1 shows the architecture of our energy-efficient infrastructure, which can be used for each edge installation. It provides Infrastructure-as-a-Service and is designed with two kinds of interfaces: North and East. The North interface is conceived for users (final users or service providers), and provides traditional APIs enhanced with QoS properties. It could be used by human staff or by automatic orchestration engines. The East interface is used for control and management by infrastructure providers (i.e., to retrieve data, apply configurations, and trigger power-saving mechanisms), allowing the integration of pluggable logics; this includes workload consolidation algorithms [35], as well as integration with WAN interfaces and other NFV elements.

A. Computing servers

Computing servers support ACPI power states and optionally other power-saving mechanisms (dynamic voltage/frequency scaling, low-power idle). Currently, we only use ACPI S3 state (aka 'Standby', 'Sleep', or 'Suspend-to-RAM'). This is one of the most useful power state, since it almost cuts off all power, while allowing to restore full operation in a few hundred milliseconds. We plan to include additional controls over other power-saving saving mechanisms in future updates. While lacking broad support for specific interfaces[1], a custom interface is currently used to change the power state from ACPI S0 to S3 (i.e., to put the device to sleep), whereas Wake-on-Lan is used to resume it back to full operation (from ACPI S3 to S0).

Our installation uses QEMU/KVM hypervisor, which is the default choice for OpenStack (see below).

B. Cloud management software

We use the open-source OpenStack framework as cloud management software. Our installation includes authentication (Keystone), computing (Nova), networking (Neutron), image (Glance), storage (Cinder), and telemetry (Ceilometer) modules. The Neutron ML2 plugin is used for virtual networking, together with openvswitch software-based switch. VLANs encapsulation is used for tenant networks, because this is the only tunneling protocol managed by OpenFlow in physical switches. The network node is installed on a dedicated machine. A shared file system is available on all compute nodes, so that live-migration of VMs is possible among hypervisors.

C. Software-Defined Network

The Software-Defined Network is composed of OpenFlow software switches, running openvswitch. We preferred software-switches running on commercial desktops over commercial hardware because the latter does not provide ACPI power states, so cannot be used to conduct live tests on power consumption. In addition to OpenFlow, network devices also expose the GAL interface (*Green Abstraction Layer* [38]), a recently introduced standard to report and modify energy-related characteristics (power states, power/performance relationship, etc.). The GAL is used to collect the power profile of each device (e.g., the description of how energy consumption changes with different utilization levels), and to change the power state from active (ACPI S0) to sleep (ACPI S3). Devices are resumed back to full operation (from ACPI S3 to S0) by the Wake-on-Lan protocol.

OpenDayLight is used as network controller. We use the l2switch feature to provide full connectivity among all nodes, while explicit flows are configured between VMs with bandwidth requirements. This approach limits the number of OpenFlow rules to specific flows with QoS requirements (which will be mapped to higher priority traffic classes), while falling back to best effort flooding behavior for ancillary traffic (e.g.,

[1]A Power State Management Profile is already available in the DTMF Common Information Model [36] and mapped to IPMI [37], but it is not available in most commercial devices.

DNS queries and DHCP). Configuration of QoS parameters (priority queues, traffic shapers, scheduling disciplines) in the physical switches is not implemented yet, but it is already on the development roadmap.

D. Monitoring framework

The main objective for the monitoring component is to collect measurements from heterogeneous devices, hence retrieving data from different meters, with different communication protocols, and with different data formats. To this aim, we use Kwapi [39], a modular framework for monitoring power consumption and publishing data in OpenStack Ceilometer. We extended Kwapi to collect data about CPU usage of the hypervisors , so we can build the power profile of each device, i.e., the relationship between power consumption and performance [4].

Kwapi includes drivers to query commodity power meters over SNMP, serial/usb wattmeters, and the IPMI interface available in many recent computing boards[2]; this covers most hardware that could be deployed in edge sites, including legacy servers with a single motherboard as well as blade servers.

E. North and East interfaces

The North interface is the OpenStack API. Currently, QoS support in OpenStack is very limited, and only covers a few configurations in the hypervisor, but none in the physical network. We envision a richer semantics that let users specify QoS constraints. QoS information is not directly used within the energy-efficiency infrastructure, which does not include any specific logic, but is exposed to external management software (e.g., QoS/Consolidation algorithms) via the East interface. For example, when using the energy-efficient infrastructure in an advanced consolidation framework [35], users are expected to provide the following information:

- *active/paused* state. VMs can be in different steady states[3], roughly corresponding to ACPI power states. We expect that users make use of the PAUSED state to indicate those VMs that are not currently used (i.e., spare components for horizontal scaling or backup), hence implicitly requesting (temporary) low service levels;
- *bandwidth* requirements. These are properties in the form $\langle key, value \rangle$ associated to each VMs and stored in the metadata server[4].

When using software orchestration tools, it is easy to include QoS characteristics in development models like TOSCA [40] and ETSI NFV [41].

The East interface is a collection of control and management APIs to exploit the unique features of this kind of infrastructure. From top to bottom on the right side of Fig. 1, we find the cloud API, the power management interface, and the network API.

The *cloud API* is the standard OpenStack interface, used in admin mode to retrieve information about the virtualization system (running VMs, state, tenant networks, VLAN ids, etc.) and to trigger migration of VMs. It is also used to detach hypervisors from the OpenStack controller before putting them to sleep, so to avoid any unreachability error.

The *power management* interface is used to:

- retrieve information about current power consumption;
- building power profiles for servers and network devices, by correlating power consumption with different load conditions (CPU usage);
- change the power state of servers and network devices.

The first two operations rely on the monitoring framework described in Sec. IV-D, whereas the last one uses the GAL and custom interface, and the Wake-on-Lan protocol (see Sec. IV-A and IV-C).

Finally, the *network API* allows programmatic configuration of the communication infrastructure. We use RESTCONF [43], available as OpenDayLight feature, which provides high-level network abstraction through the Yang model. This interface can be used to set flows in the underlying datapath, according to the specific strategies computed by management algorithms.

V. Evaluation and numerical results

A. Experimental setup

We deployed a working testbed as depicted in Fig. 2. It is a small testbed, but it is rather representative of edge installations, which are not expected to have the same size as modern data centers. It is composed of:

- three compute nodes (Intel NUC6i7KYK, i7-6770HG 2.60GHz 4 cores, 4GB RAM);
- four software switches that interconnect compute and network nodes (data network[5]), running on a heterogeneous set of low-end commercial desktops (Atom, Pentium, Phenom; 2-4 cores; around 1.1GHz);
- openstack controller, network node, and opendaylight controller, deployed as virtual machines on a high-end server;
- management network on a separate infrastructure.

The installed software is the Newton release for OpenStack, and Carbon for OpenDayLight. We used a mix of bash scripts and simple java programs for evaluating our APIs and QoS parameters.

B. Network configuration

Migration of VMs will requires change in the network configuration, to re-route packets. In our testbed, network configuration is updated by installing OpenFlow flows, and this may result in packet losses, jitter variations, broken connectivity. To understand the potential impact of network reconfiguration, we generated a UDP traffic stream from VM2

[2]The Intelligent Platform Management Interface (IPMI) provides management and monitoring capabilities of computer systems over the network, without support from the firmware or the Operating System

[3]OpenStack documentation – Server states. URL: https://developer.openstack.org/api-guide/compute/server_concepts.html.

[4]Metadata can be inserted when the VM is created, but can also be updated later. See: https://wiki.openstack.org/wiki/NovaImageCreationAPI.

[5]The communication infrastructure for hosted VMs is denoted as "guest" network in latest OpenStack releases.

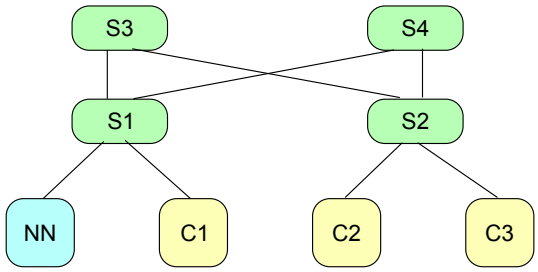


Fig. 2: Experimental testbed. NN=Network Node, C=Compute node (server), S=Switch.

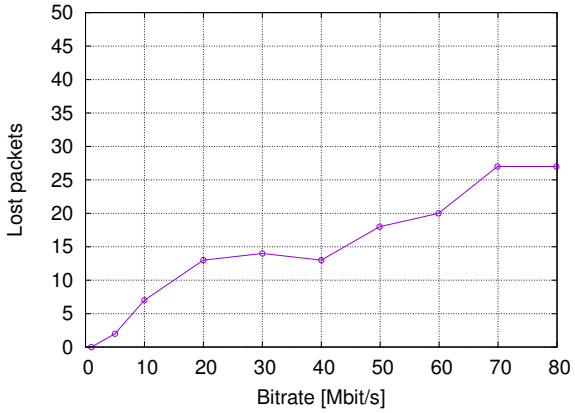


Fig. 3: Packet losses during flow re-configuration, for different stream sizes.

(hosted on C2) and VM1 (hosted on C1), while switching between two different paths: S1-S3-S2 and S1-S4-S2.

Fig. 3 reports packet losses registered with different stream sizes. We notice very good performance, even for 80 Mb/s (which is close to the upper limit for our set up). Jitter are almost constant for every bitrate, as shown in Fig. 4. Indeed, we can explain such good performance by recalling that the underlying forwarding behavior of l2switch in OpenDayLight is flooding with loop avoidance. This means that when no specific OpenFlow rules are configured for the UDP traffic stream, it get anyway forwarded to the destination. We think that this behavior is very good for service continuity, while traffic burst are only generated occasionally in the network during re-configuration.

On the contrary, if no specific path is configured in the network, the behavior of l2switch generates congestion (see next Section) and packet losses quickly raises to unacceptable levels, as shown in Fig. 5.

C. Network congestion

While the default l2switch forwarding behavior may be enough with low traffic, flooding packets in the entire network is likely to cause congestion in realistic scenarios. Our framework allows to set up specific flows between each pair of VMs, hence saving bandwidth and avoiding congestion.

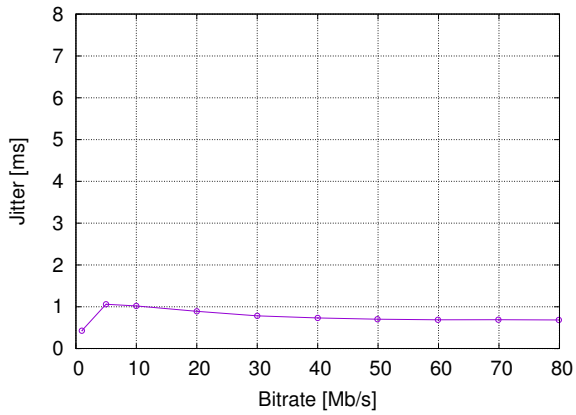


Fig. 4: Packet jitter during flow re-configuration for different stream sizes.

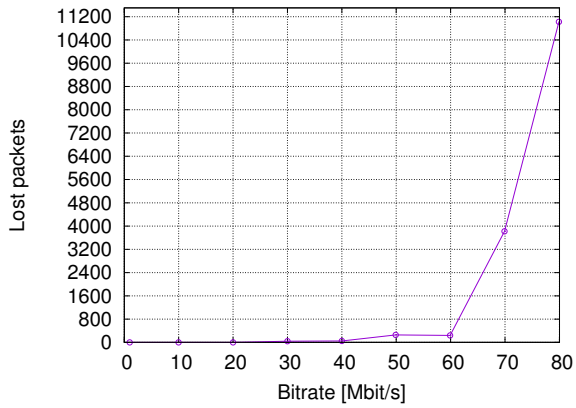


Fig. 5: Packet losses with plain l2switch behavior (no flow configured), for different stream sizes.

To understand the benefit of our approach, we measured bandwidth utilization under these two different conditions:

- *l2switch* flooding behavior, where all packets are replicated on all ports but the ones inhibited by the loop avoidance mechanism;
- *unicast flow* for packets exchanged between VMs, and flooding for other traffic (DNS, DHCP, ARP, etc).

Fig. 6 shows that without unicast flows the bandwidth usage almost double for the entire network. We can see that link s2-s3 is empty because of loop avoidance.

D. Power consumption vs performances

Figs. 7 and 8 show the relation between power consumption and performances measured for servers and software switches, respectively. We call this relationship the “power profile” of the device; performances are expressed in terms of CPU load for computing servers and packet throughput for network devices.

We note that this information does not follow a straight relationship as often assumed in analytic simulations. We also argue that the curves substantially depend on the active power

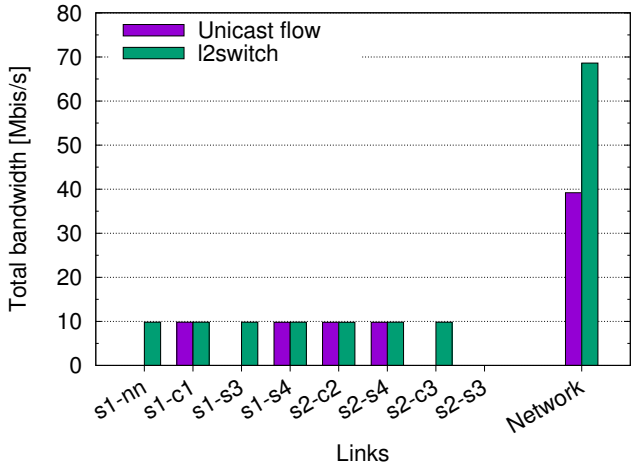


Fig. 6: Total bandwidth per link and for the whole network.

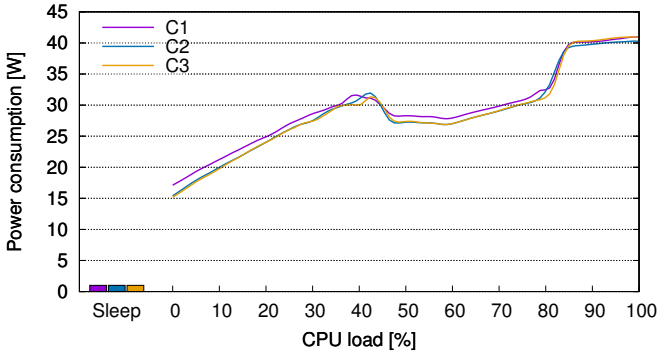


Fig. 7: Computed power profiles for servers.

saving mechanisms of the devices (low-power idle, dynamic voltage and frequency scaling, as we measured in different set-ups), and the external temperature (that affects the activation of cooling fans). This motivates further the validation of consolidation algorithms in real testbeds with real data.

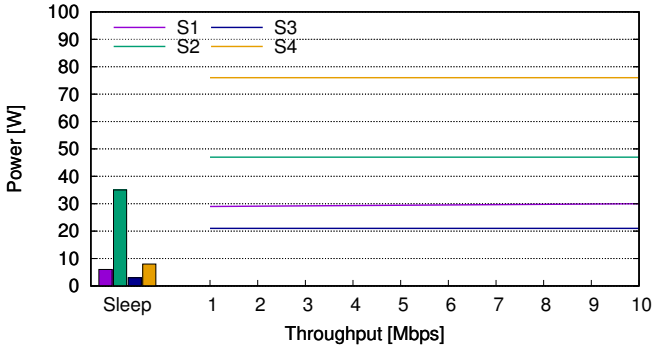


Fig. 8: Computed power profiles for software switches.

VI. Examples of usage

Additional components and enhanced APIs that we have developed in our energy-efficient infrastructure are particularly suited to be used by workload consolidation algorithms. Indeed, our work was originally motivated by the need for an experimental testbed where the effectiveness and efficiency of workload consolidation algorithms could be demonstrated and validated in real environments, beyond software simulations [42]. We believe that this step is indeed very important, since simulations cannot account for critical factors as real power consumption of devices, delays in changing power states, service disruptions while changing the network configuration, impact of VM migrations, and so on.

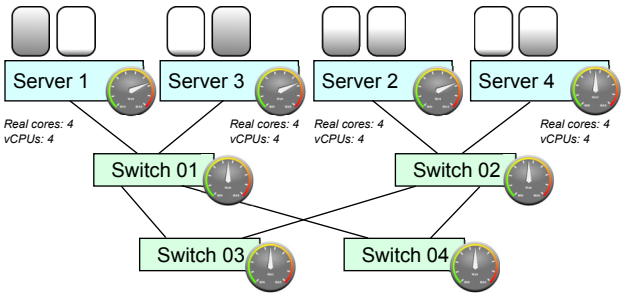
In this Section, we pictorially describes how our energy-efficient infrastructure enables advanced consolidation strategies that support automatic horizontal scaling of modular applications while saving energy. For more rigorous dissertation and mathematical details we refer the reader to our previous work [35].

Let us consider a simple IaaS scenario that may be representative of small edge computing installations. For the sake of simplicity, let us suppose the infrastructure as four servers, with four cores each, and four switches, which may represent a single pod in a three-tier architecture with full bisectional bandwidth. We represent the current usage level of each VM by the extension of the internal coloration. We consider 8 VMs, with 2 vCPUs each; two of them are just pre-provisioned for later use (e.g., backup or scaling), so will be idle most of time. Fig. 9 shows typical behaviors under different consolidation strategies.

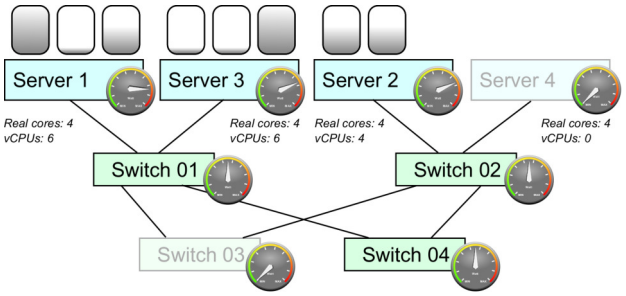
The first case under consideration is a business-as-usual scenario with no consolidation (Fig. 9a). In this situation, VMs are usually placed uniformly among servers, by caring that vCPUs do not exceed the overprovisioning thresholds on each server, but without taking into account QoS constraints.

The second case considers ‘blind’ consolidation. Fig. 9b depicts the expected behavior for typical algorithms that take into account both servers and network devices. VMs are placed on the smallest number of servers, according to CPU, RAM, and bandwidth constraints; idle devices are put to sleep, hence saving energy. Though this approach largely improves the overall efficiency with respect to the previous scenario, there are still a couple of critical issues. First, it does not distinguish mission-critical applications, so overprovisioning may lead to service violation, while VM migration may lead to service disruption. Second, idle machines represent a problem for both users and infrastructure providers: users have to pay for the two additional machines, and the infrastructure provider likely wastes energy to run more servers than strictly needed. We note that also network traffic is consolidated and some network devices could be put to sleep.

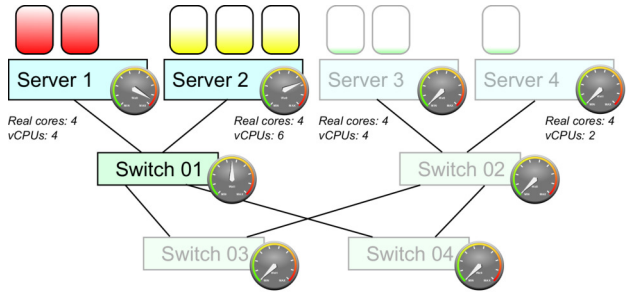
Finally, the third scenario uses a very simple model, where a limited number of different colors (red, yellow, and green) correspond to different QoS constraints:



(a) Business-as-usual



(b) Legacy consolidation



(c) QoS-aware consolidation

Fig. 9: Placement of VMs and power consumption with different consolidation strategies.

- *red*, for mission-critical applications which requires strict allocation of CPU, memory, and bandwidth requirements;
- *yellow*, for working instances that tolerate some degree of overprovisioning on CPU and bandwidth requirements;
- *green*, for unused and idle instances, which only requires to be available within a maximum delay.

The colors can be easily mapped to specific interfaces of our energy efficient infrastructure. Red/yellow VMs set their color and QoS requirement in OpenStack metadata, whereas green VMs are in PAUSED state (see Section IV-E). The main advantage of this model is that green VMs are explicitly identified as "unused", so they could be grouped together. Servers hosting only this kind of VMs could be put to sleep, but they can be resumed quickly when one of their VMs changes to the yellow/red status. In addition, red VMs could be grouped together without overprovisioning, so to minimize the number of migrations. Fig. 9c shows the expected result from this kind of consolidation strategy. We note that a larger number of servers is now sleeping (hosting green VMs), which in turn enables to shut down even a large part of the network.

We conclude by arguing that our testbed can be used for both kinds of consolidation, even if we consider the last option the most interesting and promising one for modular applications in cloud/edge computing.

VII. Conclusions

In this paper, we have presented our energy-efficient virtualization infrastructure that extends plain OpenStack with additional features to support workload consolidation and power management. Our work explicitly targets mobile edge computing and next-generation networks, where efficiency will have to be combined with high-variable workloads.

Functional validation and performance evaluation have demonstrated that our approach can effectively support QoS-constrained environments. Though these are only preliminary results, we plan to carry out additional measurements and to include evaluation for real-world applications under different workload conditions (e.g., steady-state, peaks, ramp-up, ramp-down). This will also include estimation of power saving compared to the business-as-usual scenario when using different consolidation strategies.

Future extensions to our testbed will bind QoS constraints requested by applications to priority mechanisms in the network: queue scheduling, rate limiting, traffic shaping.

Acknowledgment

This work was supported in part by the European Commission under the projects ARCADIA (contract no. 645372) and MATILDA (contract no. 761898).

References

[1] "5G empowering vertical industries," Whitepaper from the 5G-PPP, ERTICO, EFFRA, EUTC, NEM, CONTINUA and Networld2020 ETP, February 2016. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2016/02/BROCHURE_5PPP_BAT2_PL.pdf

[2] R. Bolla, F. Davoli, R. Bruschi, K. Christensen, F. Cucchietti, and S. Singh, "The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization?" *IEEE Communications Magazine*, vol. 49, no. 8, pp. 80–86, August 2011.

[3] R. Bolla, R. Khan, and M. Repetto, "Assessing the potential for saving energy by impersonating idle networked devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1676–1689, March 2016.

[4] D. Kanapram, G. Lamanna, and M. Repetto, "Exploring the trade-off between performance and energy consumption in cloud infrastructures," in *2nd IEEE International Conference on Fog and Edge Mobile Computing (FMEC 2017)*, Valencia, Spain, May, 8th–11th 2017, pp. 121–126.

[5] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in data center networks," *Computer Communications*, vol. 40, pp. 1–21, March 2014.

[6] M. Polverini, A. Cianfrani, A. Coiro, M. Listanti, and R. Bruschi, "Freezing forwarding functionality to make the network greener," *Computer Networks*, vol. 78, pp. 26–41, February 2015.

[7] C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed," *International Journal of Network Management*, vol. 15, no. 5, pp. 297–310, Sep.-Oct. 2005.

[8] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, December 2007.

[9] A. Beloglazov and R. Buyya, "OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 5, pp. 1310–1333, April 2015.

[10] F. Ahmad and T. N. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems (ASPLOS XV)*, Pittsburgh, PA, USA, Mar.13–17, 2010, pp. 243–256.

[11] G. A. Geronimo, J. Werner, C. B. Westphall, C. M. Westphall, and L. Defenti, "Provisioning and resource allocation for green clouds," in *The Twelfth International Conference on Networks (ICN 2013)*, Seville, Spain, Jan. 27–Feb. 1, 2013.

[12] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP network energy cost: formulation and solutions," *IEEE/ACM Transactions on Networking*, 2012.

[13] A. Cianfrani, V. Eramo, M. Listanti, M. Polverini, and A. Vasilakos, "An OSPF-integrated routing strategy for QoS-aware energy saving in IP backbone networks," *IEEE Transactions on Network and Service Management*, vol. 9, no. 3, pp. 254–267, September 2012.

[14] A. A. Kist and A. Aldraho, "Dynamic topologies for sustainable and energy efficient traffic routing," *Computer Networks*, vol. 55, no. 9, pp. 2271–2288, June 2011.

[15] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, January 2013.

[16] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, "Joint virtual machine assignment and traffic engineering for green data center networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 3, pp. 107–112, December 2013.

[17] H. Shirayanagi, H. Yamada, and K. Kono, "Honeyguide: A VM migration-aware network topology for saving energy consumption in data center networks," in *IEEE Symposium on Computers and Communications (ISCC)*, Cappadocia, Turkey, Jul. 1–4, 2012, pp. 460–467.

[18] A. Carrega and M. Repetto, "Exploiting novel software development paradigms to increase the sustainability of data centers," in *Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2016)*, Shanghai, China, Dec. 6^{th}–9^{th}, 2016.

[19] D. Kliazovich and P. Bouvry, "DENS: Data center energy-efficient network-aware scheduling," in *IEEE/ACM Int. Conf. on Green Comp. and Comm. & IEEE/ACM Int. Conf. on Cyber, Physical and Social Comp.*, Hangzhou, China, Dec. 18–20, 2010, pp. 69–75.

[20] R. Bolla, C. Lombardo, R. Bruschi, and S. Mangialardi, "DROPv2: Energy efficiency through network function virtualization," *IEEE Network*, vol. 28, no. 2, pp. 26–32, March-April 2014.

[21] G. Yan and J. Yang, "OSPF extensions for MPLS green traffic engineering," IETF Internet-draft, October 2013. [Online]. Available: https://tools.ietf.org/html/draft-li-ospf-ext-green-te-01

[22] Q. Ali, "Host power management in vmware vsphere 5.5 – performance study," Whitepaper, August 2013. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/hpm-performance-vsphere55-white-paper.pdf

[23] [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/Distributed-Power-Management-vSphere.pdf

[24] L. M. Contreras, V. López, O. González de Dios, A. Tovar, F. Muñoz, A. Azañón, J. P. Fernández-Palacios, and J. Folgueira, "Toward cloud-ready transport networks," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 48–55, September 2012.

[25] H. Puthalath, J. Soares, A. Sefidcon, B. Melander, J. Carapinha, and M. Melo, "Negotiating on-demand connectivity between clouds and wide area networks," in *IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, Paris, France, Nov. 28–30, 2012, pp. 124–130.

[26] L. Velasco, L. M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, and J. P. Fernández-Palacios, "A service-oriented hybrid access network and clouds architecture," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 159–165, April 2015.

[27] A. Sanhaji, P. Niger, P. Cadro, C. Ollivier, and A.-L. Beylot, "Congestion-based API for cloud and WAN resource optimization," in *IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, South Korea, Jun. 6–10, 2016, pp. 141–145.

[28] [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/vmware-whats-new-in-vmware-vsphere-networking-white-paper.pdf

[29] J. Schad, J. Dittrich, and J.-A. Quiane-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *Proceedings of the VLDB Endowment*, vol. 3, no. 1–2, pp. 460–471, September 2010.

[30] N. Ferry, H. Song, A. Rossini, F. Chauvel, and A. Solberg, "CloudMF: applying MDE to tame the complexity of managing multi-cloud applications," in *7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014)*, London, UK, Dec. 8th-11th, 2014, pp. 269–277.

[31] B. Karakostas, "Towards autonomic cloud configuration and deployment environments," in *International Conference on Cloud and Autonomic Computing (ICCAC)*, London, UK, Sep., 8th-12th 2014.

[32] P. Gouvas, C. Vassilakis, E. Fotopoulou, and A. Zafeiropoulos, "A novel reconfigurable-by-design highly distributed applications development paradigm over programmable infrastructure," in *Proceedings of the 28th International Teletraffic Congress (ITC 28)*, Wuerzburg, Germany, Sep. 12-16, 2016.

[33] P. Bellavista, L. Foschini, R. Venanzi, and G. Carella, "Extensible orchestration of elastic IP multimedia subsystem as a service using Open Baton," in *5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, San Francisco, CA – USA, Apr., 6th-8th, 2017, pp. 88–95.

[34] J. F. Riera, J. Batallé, J. Bonnet, M. Días, M. McGrath, G. Petralia, F. Liberati, A. Giuseppi, A. Pietrabissa, A. Ceselli, A. Petrini, M. Trubian, P. Papadimitrou, D. Dietrich, A. Ramos, J. Melián, G. Xilouris, A. Kourtis, T. Kourtis, and E. K. Markakis, "TeNOR: Steps towards an orchestration platform for multi-PoP NFV deployment," in *IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, South Korea, Jun. 6th–10th, 2016, pp. 243–250.

[35] A. Carrega and M. Repetto, "Energy-aware consolidation scheme for data center cloud applications," in *First International Workshop on Softwarized Infrastructures for 5G and Fog Computing (Soft5 2017)*, Genoa, Italy, Sep., 4th–8th 2017.

[36] DTMF, "Power state management profile," Specification DSP1027, December 2009, version: 2.0.0. [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP1027_2.0.0.pdf

[37] J. Hass, "Ipmi cim mapping guideline," June 2006, document Revision 0.60. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/cim-mapping-guideline-.6.pdf

[38] "Green Abstraction Layer (GAL); power management capabilities of the future energy telecommunication fixed network nodes," ETSI ES 203 237, March 2014, version 1.1.1.

[39] F. Rossigneux, J.-P. Gelas, L. Lefèvre, and M. D. de Asunção, "A generic and extensible framework for monitoring energy consumption of openstack clouds," in *4th IEEE International Conference on Sustainable Computing and Communications (SustainCom)*, Sydney, Australia, Dec. 3–5, 2014.

[40] "Topology and orchestration specification for cloud applications," OASIS Standard, November 2013, version 1.0. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf

[41] "Network functions virtualisation (nfv); management and orchestration," ETSI GS NFV-MAN 001, December 2014, v1.1.1. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[42] R. Bolla, L. Sambolino, D. Tigano, and M. Repetto, "Enhancing energy-efficient cloud management through code annotations and the green abstraction layer," in *First International Workshop on Sustainable Data Centres and Cloud Computing (SD3C)*, Limassol, Cyprus, Dec. 7, 2015.

[43] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF protocol," RFC 8040, January 2017. [Online]. Available: https://tools.ietf.org/html/rfc8040