The importance of being OS-aware*

in performance aspects of Cloud Computing research

Tommaso Cucinotta[1], Luca Abeni[1], Mauro Marinoni[1], and Carlo Vitucci[2]
[1]*Scuola Superiore Sant'Anna, Pisa, Italy*
[2]*Ericsson, Stockholm, Sweden*
{*name.surname*}*@santannapisa.it, carlo.vitucci@ericsson.com*

Keywords: Cloud Computing; Operating Systems; Quality of Service Control; Temporal Isolation; Real-Time Scheduling

Abstract: This paper highlights ineffifiencies in modern cloud infrastructures due to a distance between the research on high-level cloud management / orchestration and the research on low-level kernel and hypervisor mechanisms. Our position about this issue is that more research is needed to make these two worlds talk to each other, providing richer abstractions to describe the low-level mechanisms and automatically map higher-level descriptions and abstractions to configuration and performance tuning options available within operating systems and kernels (both host and guest), as well as hypervisors.

1 Introduction

Despite the tremendous growth and transformation undergone by Cloud Computing over the last 10 years, this computing paradigm finds its roots back in the '60s. The paradigm of *utility computing* was mentioned in the MIT centennial speech by John McCarthy in 1961 (Garfinkel, 2011), in its famous statement: "*Computing may someday be organized as a public utility just as the telephone system is a public utility*". Also, machine virtualization, one of the core enablers of cloud computing, had been conceptualized and prototyped already in the '70s (White, 2015), albeit the hardware available at that time was not ready for a widespread use of the mechanism. Furthermore, since the Internet was born, we have seen web-hosting servers on rental, sometimes along with simple accompanying data-base and back-up services. However, the actual possibility for the general public to be able to rent virtual machines (VMs) on-demand, rapidly and in a self-service and completely automated fashion, dates back to year 2006, when Amazon Web Services (AWS) launched its Elastic Compute (EC2) service (Information Resources Management Association, 2012).

At that time, you could rent simple virtual machines choosing among a few instance types (Barr, 2015), using an Infrastructure-as-a-Service model. Later, other players entered the market, and the available services evolved up to the nowadays *native cloud applications* making use of a plethora of services available from a Platform-as-a-Service (PaaS) provider, like security, load-balancing, high-availability, data storage and replication, distributed communication middleware and queueing, big-data processing frameworks, etc. Recently, operating-system (OS) level virtualization, a.k.a., containers, are increasingly replacing machine virtualization in various application domains, finding a natural fit within novel distributed processing architectures based on *microservices* (Brown, 2016).

Academic research has been flanking the emerging industry of cloud computing services, experimenting with novel directions of investigation, mostly inspired to the consolidated areas around distributed computing, fault-tolerance, data-bases, networking protocols, routing, data replication and distributed transactions, security, etc., with many of these research areas overflowing almost naturally from prior, consolidated research lines in the field of grid and distributed computing. Many research groups focused on novel issues coming straight from particular characteristics of the cloud computing paradigm instead, such as the capability to grow and shrink service instantiations by adding or removing VMs at runtime, live-migrate them for re-optimization purposes, as well as the one to create virtual networking overlays on top of the physical infrastructure. This resulted in novel research on scalable resource manage-

*This work was partially funded by Ericsson AB, Stockholm, Sweden.

ment in massively distributed infrastructures, dealing with issues related to monitoring, resource estimation and workload prediction (Cortez et al., 2017), optimal VM placement (Chang et al., 2010; Mann, 2015) including cases of network-aware placement (Alicherry and Lakshman, 2012; Yao et al., 2013; da Silva and da Fonseca, 2016; Steiner et al., 2012) and allocation of data-intensive VMs (Zhang et al., 2015), energy efficiency (Ghribi et al., 2013), elastic applications management and associated performance models (Xia et al., 2015), architectures of cloud *orchestration* layers, and others.

Just recently, further fields of particular success in terms of research productivity have been the ones of cloud federation (Konstanteli et al., 2014) and edge (cloud) computing. The last one, in particular, includes cloudlets (Satyanarayanan et al., 2009) and the cloudification of other IT areas such as those of interest for network operators, with the uprising trend of software defined networking (SDN) and network function virtualization (NFV) (Public, 2016; Networks, 2017) and related technologies (Fiorani et al., 2015). These application scenarios are characterized by tighter constraints concerning low latency and overall Quality of Service (QoS) for mobile applications (Valcarenghi et al., 2017). Moreover, the development of end-to-end SDN-NFV solutions cannot ignore the characteristics and, therefore, the needs of Virtual Functions (VF) that have to be placed in nodes located in different zones of the network, such as the core or the edge. VFs assigned to core nodes present similar needs to the classic cloud infrastructures, while the closer you get to the edge of the network, the more the demand for flexible management of the computing resource becomes critical (Vitucci and Larsson, 2017). These requirements are crucial both concerning the QoS levels and the available resources optimization because, in such contingencies of scarce resources and significant power consumption (SCTE, 2016), a more tailored allocation of energy and resources becomes essential (Gai et al., 2016).

2 The role of the operating system

As due to the intrinsic nature of cloud computing, performance of cloud-hosted applications, services, and internal infrastructure management systems has always been of paramount importance, in the overall picture. Similarly to when one buys physical hardware, renting on-line IaaS facilities needs to respect minimum performance specifications that are usually made more or less explicit by the provider. However, plenty of the mentioned research works tackling performance and QoS control aspects of cloud systems, have been focusing on exploiting *elasticity* to just compensate for the great variability in performance of the rented infrastructure items (unstable VMs processing performance, usually, due to time-sharing of the physical CPUs, but also instability in disk access as well as networking performance). Therefore, one finds an increasing presence of high-level resource management frameworks, orchestration layers and evolved middleware services, dealing with workload monitoring, estimation and prediction, as well as resource management.

One aspect that has not been given sufficient attention, in our opinion, is the bottommost part of the software stack: the *operating system* (OS) and hypervisor layers. Indeed, a significant amount of research in the field focused on the higher levels of the cloud stack, just relying on well-established basic functionalities provided by traditional OSes, first and foremost, the ability to pin down virtual cores onto physical cores of the underlying hardware platform, and in some cases fine-tuning of the hypervisor memory allocation subsystem by exploiting the NUMA capabilities of the hardware. However, OSes and hypervisors have been evolving over time, exhibiting richer and richer functionality for supporting fine-tuning of performance, with reference to achievable latency and throughput within data centre infrastructures, encompassing multiple areas, such as: CPU scheduling, memory bandwidth allocation and management, interrupt dispatching and virtualized interrupt handling, para-virtualization options and ways to access hardware accelerators including GPUs, GP-GPUs, cryptographic accelerators or generic FPGA-based processing elements (Caulfield et al., 2016; Pellerin, 2017). As exemplified in Figure 1, these capabilities are often pre-tuned according to data centre-wide procedures (or just set as found in OS and other software distributions), and from the high-level perspective of infrastructure orchestration layers they are either not made accessible, or subsumed at a too high of an abstraction level, thus resulting in a mismatch with the plethora of heterogeneous requirements coming from different applications, resulting in a poor effectiveness in infrastructure management.

However, it is our belief that the novel panorama of heavily distributed, elastic, fault-tolerant and often interactive, computing applications and services calls for novel OS services and low-level middleware, and that these need to be better connected to higher-level cloud management and orchestration layers. Using more advanced functionalities provided by modern OSes can significantly improve the performance,

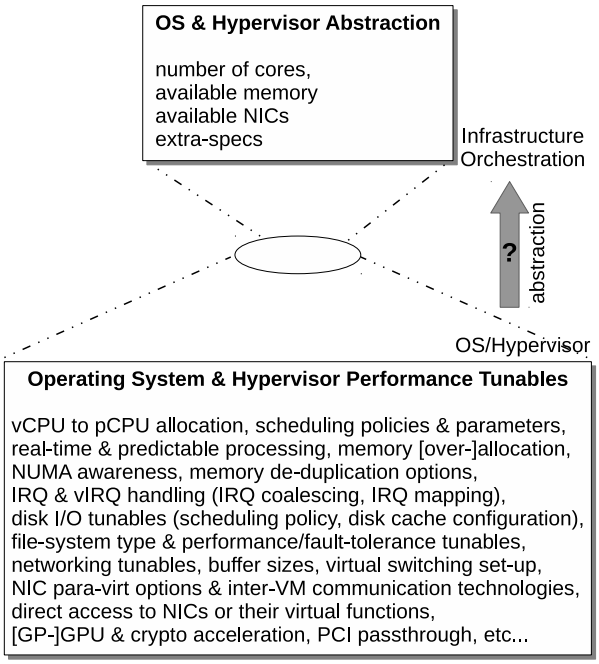

Figure 1: Limits of high-level infrastructure orchestration layers in handling the plethora of low-level performance tunables available in nowadays OSes and hypervisors.

scalability, predictability thus usability of cloud services, making them even more attractive.

3 Low level support

This section briefly introduces some of the advanced features that can be provided at the lower-levels of the cloud stack, and that could be exploited by higher level cloud control infrastructures to improve performance.

3.1 CPU scheduling

Management of the CPU resources in cloud infrastructures follows basically two main patterns: the best VM instance types are mapped 1-to-1 so that virtual CPUs run onto dedicated physical CPUs. On the other hand, the cheapest VM instance types are mapped onto clusters of physical machines where there is an over-allocation of virtual CPUs, compared to the available physical CPUs. Finally, the most expensive and best isolated instance types are those relying on entire physical machines dedicated to a single tenant's instance, either in virtualized or also in bare-metal form.

Focusing on physical machines shared among multiple tenants, the dedicated 1-to-1 virtual to physical core mapping is the set-up that tries to achieve the best temporal isolation among different VMs. However, this is not sufficient to isolate data-intensive applications, such as high-definition multimedia (video) processing workloads, big-data analytics and others. These, even when deployed on dedicated cores, cause quite a lot of interference to processing tasks deployed on other physical cores, due to interferences at the memory access bandwidth and cache-level interferences. Works aimed at limiting said phenomenon can be found (Yun et al., 2013), specifically confining a precise budget in memory access for individual physical cores, that, once exhausted, would cause the VM to be throttled till replenishment of the budget at the next period. Mechanisms have been studied also to limit and keep under control interferences at the cache level on big multi-core machines, where there is a sufficiently big last-level cache (LLC). In such systems, it is both possible to employ software-only cache partitioning schemes based e.g., on *cache colouring* (Kim and Rajkumar, 2016), or to leverage hardware technologies, such as the recent Intel Cache Allocation Technology (CAT) (Corbet, 2016).

Whenever virtual CPUs are over-allocated to physical ones, the way multiple VMs compete among each other for running on the physical CPUs when active at the same time is left to heuristics of the hypervisor scheduler, which unfortunately falls short in its attempt to satisfy requirements of a multitude of application domains, without knowing much about what kind of service/application is running within what VMs. Normally, the temporal interferences a VM gets from other co-scheduled VMs are quite high, making the performance of each hosted VM quite unstable. In this area, prior research works focused on the Xen (Barham et al., 2003) hypervisor, comparing various scheduler types (Cherkasova et al., 2007) and its impact on the performance of the hosted workload.

Other research works focused on co-scheduling virtual CPUs of VMs on the same physical CPUs by employing a special scheduler within the hypervisor, that allows for a reservation-based paradigm. Namely a virtual core may be allotted a budget (a.k.a., *runtime*) of Q time units every period of P time units, so that the underlying hyperivosr (or host OS) scheduler guarantees Q time units every period of duration P, and optionally it also constraints the VM to be throttled once said budget is exhausted. This is the example of the IRMOS real-time scheduler for Linux (Cucinotta et al., 2010; Checconi et al., 2009), or the similar recent hierarchical extension of the `SCHED_DEADLINE` scheduler of Linux (Balsini et al., 2017). Similar efforts have been attempted on Linux/KVM also by exploiting existing in-kernel

features, so that EDF-based scheduling capabilities could be injected into the kernel at run-time, without any need for static code modifications to the kernel code (Aesberg et al., 2011).

3.2 Networking

The service performance are affected not only by the configuration of the used VM, but also by the OS and kernel mechanisms used to relay data among VMs, between host and guest applications, etc... For example, it has been shown (Abeni et al., 2015) that, when using Linux as host OS and KVM as hypervisor, simply changing the packets forwarding mechanism from the default bridge + `tap` to `macvtap` can greatly improve the performance of a virtual router. In particular, the rate of small packets the virtual router can forward has been shown to increase of nearly 50%. Other configuration details (such as the queue length of the virtual interfaces, the scheduling priority and interrupt affinity, etc...) that are often ignored in high-level descriptions also affect the performance and the ability to sustain a high forwarding rate in a stable way.

Of course, many of the low-level details are OS or VM specific (for example, the possibility to use `macvtap` or `vhost-net` are Linux-only features; Xen provides a completely different way to forward packets - with different configuration options - etc....). This is why many cloud providers often rely to default settings, that do not require any knowledge of the OS, kernel, and VM details, resulting, in much worse performance and usability.

As an additional example, high-performance networking in cloud services can often be obtained by using some form of user-space networking (even arriving to user-space network drivers). In this regard FreeBSD provides netmap (Rizzo, 2012a; Rizzo, 2012b), that marked a significant milestone in the area of packet processing in general-purpose operating systems, showing how well-engineered optimizations of existing drivers, coupled with proper novel user-space/kernel-space interfaces for networking, can lead to a great increase in processing throughput of small packets. On different operating systems, similar results can be obtained by using different mechanisms; for example, on Linux-based systems it is possible to use DPDK[2], that has been shown to greatly improve the performance (especially in software defined networks) (Pongrcz et al., 2013). Virtual bridges between different VMs also can be setup in different ways, using different technologies that depend on the used network drivers; for example, if netmap is used then the VALE virtaul switch (Rizzo and Lettieri, 2012) is the best choice for high performance communications among VMs on the same host, reaching up to 2 million packets per second without hardware assistance.

Since it is clear that an adminitrator in charge of setting up a cloud cannot have all of the information needed to properly select the most appropriate technologies and configurations, it becomes clear that it is of vital importance to automate this process by properly mapping high-level descriptions understandable by the administrator to low-level features. To do this, it is fundamental to develop a way to make the low-level virtualization infrastructure attributes directly visible to the higher levels of the cloud management stack, or exploit richer and more complex abstract interfaces, that allow for properly tuning / configuring / using all the available options (even if they are not standard).

3.3 Computing *and* networking

Some authors tried to build mechanisms to improve the responsiveness of networking applications, in presence of concurrently running CPU-bound activities. This is the case of the work in (Kim et al., 2009), where a heuristic identifies I/O tasks within a VM, and boosts their scheduling priority (tuning parameters of the Xen credit-based scheduler) whenever there are pending incoming packets to be received by that task in the VM. A similar approach has been attempted on KVM-based systems (Cucinotta et al., 2011), where a VM being scheduled with a reservation-based scheduler has been made able to dynamically switch to a finer-grain scheduling deadline whenever there were packets pending to be processed by particular services running in the VM at higher priority.

4 Exploiting the low-level features

When the performance of the cloud services becomes relevant, it is important to properly map high-level performance requirements of the cloud application or service to low-level virtualization infrastructure configuration and tuning parameters.

4.1 Adoption of reservation-based approaches

As an example, consider a specific area of system support (CPU scheduling): one of the obstacles into getting reservation-based scheduling widely adopted, is

[2] `http://www.dpdk.org`

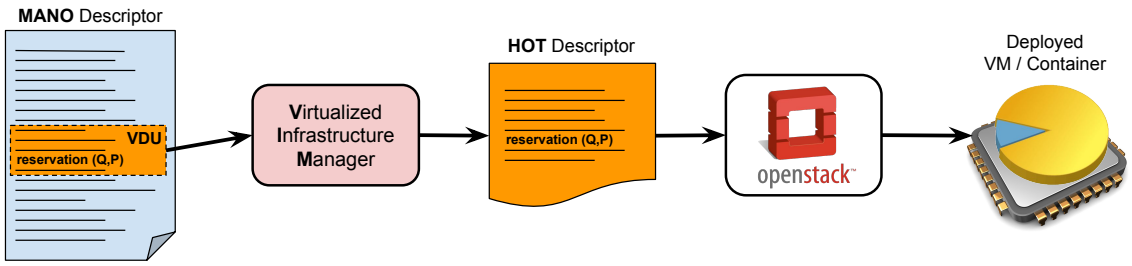

Figure 2: Propagation of the enriched descriptors from the MANO specification to the kernel level.

that software design needs a shift from tuning priorities, to tuning budgets and periods, a change that is not trivial at all. While a reservation period is easily found considering activation and latency constraints of an application/service, the budget/runtime is more difficult to find out, also due to its inherent dependency on the specific platform the software runs atop.

An interesting line of work has been proposed in the RT-Xen project (Xi et al., 2011), investigating on the use of various *server*-based scheduling algorithms for Xen domains, with scheduling parameters set according to sound arguments from hierarchical real-time theory literature (Feng and Mok, 2002). RT-Xen, like many of the research works mentioned above, only focused on low-level kernel or hypervisor mechanisms, implementing features that do not result to be available to higher levels of the cloud management stack. The only work that tries to address this concern is RT-OpenStack (Xi et al., 2015), that tries to integrate the analysis within the OpenStack Nova scheduling algorithm. While a lot of work still needs to be done in this area (for example, RT-OpenStack is strictly dependent on RT-Xen and cannot take advantage of the SCHED_DEADLINE (Lelli et al., 2016) policy that is currently available on vanilla Linux kernels), the RT-Xen / RT-OpenStack work shows a possible strategy for exploiting advanced features provided by the lower levels of the cloud stack.

The next subsection shows how a similar approach can be extended to better integrate with higher level abstractions and to consider other lower level features.

4.2 Bridging gaps in abstraction levels

One of the interesting results shown by RT-OpenStack is that to make a real and practical use of all the OS-level improvements described in the previous section, it is crucial to export all the available features and tuning points to have them available during the placement.

To reach such a result, the model of the underlying infrastructures has to be enhanced, and the corresponding descriptors must be enriched to describe such an enhancement. As an example, consider OpenStack and the Heat Orchestration Template (HOT) used by the OpenStack orchestration engine (Heat) to describe the system resources. The level of details currently described by the template has been defined focusing on homogenous cloud infrastructures and is unable to exploit hybrids clouds and modern features provided at the OS level in each physical node. This lack concerning representativeness leads to a suboptimal use of resources also at higher levels of the stack.

For example, Tacker (Orchestration, 2017), the NFV orchestrator, that is in charge of managing NFV component cannot utilize specific OS features in the MANO (ETSI, 2014) descriptors because they are not exported by the Virtualized Infrastructure Manager (VIM) (aka OpenStack). This could jeopardize the possibility to satisfy the strict latency requirements of this kind of applications.

To address this issue, the interfaces among the different layers must be enriched allowing to describe the QoS requirements regarding the computational capacity at all levels, as shown in Figure 2.

Within the MANO descriptor of each NFV component, we can find the definition of the required Virtual Deployment Units (VDU) needed to deploy that service. Each of them is characterized by a specification of the number of required CPUs, which should be extended to include the capability to declare CPU fractions defined by budget and period or by utilization and maximum service delay, in order to exploit reservation-based real-time scheduling features. This enriched VDU descriptors would be processed by the VIM to produce the corresponding orchestration templates (HOT), equivalently enhanced, required by the Heat orchestration engine. With such an approach, a real-time container or VM could be deployed according with the specified scheduling parameters, after proper admission control done at the orchestration layer, exploiting the kernel extensions and modern scheduling algorithms, such as SCHED_DEADLINE.

4.3 Guest architecture

Although most of the previous discussion focused on the issues in mapping high-level abstractions to the host OS and VMM configurations, similar opportunities for improvements can be encountered in the configuration of the guest system.

For example, many of the currently used cloud management stacks end up installing a complete general-purpose OS (that can run in containers or in more isolated VMs) for every cloudified service. However, better performance (with smaller resource requirements) can be achieved by properly configuring the guest OS to take advantage of the virtualization features or even using specialized guest OSes. Of course, this requires some visibility of the virtualization infrastructure from the guest.

Although Unikernels (Madhavapeddy et al., 2013; Kantee, 2015) have been proposed as a way to easily specialize the guest OS for specific services and specific virtualization technologies, many fo the of the most commonly used cloud management stacks are not able to properly exploit this possibility.

Again, an appropriate mapping of abstract service descriptions to guest system details (and a better description of the host OS / hypervisor capabilities) is needed to allow the automatic configuration and deployment of virtualized services with better performance.

5 Conclusions

A major drawback of existing research efforts in performance control and isolation of multiple VMs when co-located on the same physical hosts resides in their lack of direct and easy applicability in the context of real industrial cloud providers. For example, with a few exceptions, the above works provide prototype implementations in the form of hacks to the original open-source software ensemble (e.g., Xen, Linux kernel, KVM sources), rather than well-engineered solutions that can readily be leveraged in higher-level infrastructure management stacks.

Indeed, industrialization of prototypes as described in a research paper can be something totally non-trivial. As it was the case for `SCHED_DEADLINE`, having a prototype non-standard mechanism landing into the mainline Linux kernel can cost several months of development effort, with continuous interactions with core kernel developers and maintainers. Therefore, we have several of these custom approaches, patches and hacks, that are rarely made available and released in the public in a usable state, with the natural side-effect that it is overly difficult to design higher-level cloud and NFV orchestration layers that can rely on such experimental low-level features. On the side of OS vendors and maintainers, it is always difficult to incorporate novel non-standard features, without a clear idea of the need for said features by at least a community of users. This chicken-and-egg situation impairs availability of advanced performance control features in orchestration engines, which, coupled with the lack of exposure of the plethora of capabilities and performance tunables already available in nowadays OS and hypervisors, leads to the consequence that it is quite difficult for real cloud and NFV providers to leverage solid and important research results in these areas, as well as existing advanced performance control mechanisms.

In our opinion, an important step in this regard is to increase exposure of low-level performance tunables to the high-level layers of the infrastructure orchestration suite, in order to allow for a proper matchmaking process that considers also high-level application/service requirements. Therefore, we are working on further research along these lines.

REFERENCES

Abeni, L., Kiraly, C., Li, N., and Bianco, A. (2015). On the performance of kvm-based virtual routers. *Computer Communications*, 70(Supplement C):40 – 53.

Aesberg, M., Forsberg, N., Nolte, T., and Kato, S. (2011). Towards real-time scheduling of virtual machines without kernel modifications. In *ETFA2011*, pages 1–4.

Alicherry, M. and Lakshman, T. V. (2012). Network aware resource allocation in distributed clouds. In *2012 Proceedings IEEE INFOCOM*, pages 963–971.

Balsini, A., Parri, A., and Abeni, L. (2017). RT bandwidth constraints enforced by hierarchical DL scheduling. https://lkml.org/lkml/2017/3/31/658.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 164–177, New York, NY, USA. ACM.

Barr, J. (2015). AWS News Blog – EC2 Instance History. `https://aws.amazon.com/blogs/aws/ec2-instance-history/`.

Brown, K. (2016). Beyond buzzwords: A brief history of microservices patterns. `https://www.ibm.com/developerworks/cloud/library/cl-evolution-microservices-patterns/index.html`.

Caulfield, A. M., Chung, E. S., Putnam, A., Angepat, H., Fowers, J., Haselman, M., Heil, S., Humphrey, M., Kaur, P., Kim, J. Y., Lo, D., Massengill, T., Ovtcharov,

K., Papamichael, M., Woods, L., Lanka, S., Chiou, D., and Burger, D. (2016). A cloud-scale acceleration architecture. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13.

Chang, F., Ren, J., and Viswanathan, R. (2010). Optimal resource allocation in clouds. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 418–425.

Checconi, F., Cucinotta, T., Faggioli, D., and Lipari, G. (2009). Hierarchical Multiprocessor CPU Reservations for the Linux Kernel. In *Proceedings of the 5th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2009)*.

Cherkasova, L., Gupta, D., and Vahdat, A. (2007). Comparison of the three cpu schedulers in xen. *SIGMETRICS Perform. Eval. Rev.*, 35(2):42–51.

Corbet, J. (2016). Controlling access to the memory cache. https://lwn.net/Articles/694800/.

Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., and Bianchini, R. (2017). Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 153–167, New York, NY, USA. ACM.

Cucinotta, T., Checconi, F., and Giani, D. (2011). Improving Responsiveness for Virtualized Networking Under Intensive Computing Workloads. In *Proceedings of the 13th Real-Time Linux Workshop*, RTLSWS.

Cucinotta, T., Checconi, F., Kousiouris, G., Kyriazis, D., Varvarigou, T., Mazzetti, A., Zlatev, Z., Papay, J., Boniface, M., en Berger, S., Lamp, D., Voith, T., and Stein, M. (2010). Virtualised e-Learning with Real-Time Guarantees on the IRMOS Platform. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010)*, pages 1–8, Perth, Australia.

da Silva, R. A. C. and da Fonseca, N. L. S. (2016). Topology-aware virtual machine placement in data centers. *Journal of Grid Computing*, 14(1):75–90.

ETSI, N. (2014). Network functions virtualisation (NFV); management and orchestration. http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf.

Feng, X. and Mok, A. K. (2002). A model of hierarchical real-time virtual resources. In *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.*, pages 26–35.

Fiorani, M., Skubic, B., Mårtensson, J., Valcarenghi, L., Castoldi, P., Wosinska, L., and Monti, P. (2015). On the design of 5g transport networks. *Photonic Network Communications*, 30(3):403–415.

Gai, K., Qiu, M., Zhao, H., Tao, L., and Zong, Z. (2016). Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. *J. Netw. Comput. Appl.*, 59(C):46–54.

Garfinkel, S. (2011). MIT Technology Review – Intelligent Machines – Computing Pioneer Dies. https://www.technologyreview.com/s/425913/computing-pioneer-dies/.

Ghribi, C., Hadji, M., and Zeghlache, D. (2013). Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 671–678.

Information Resources Management Association (2012). *Grid and Cloud Computing: Concepts, Methodologies, Tools and Applications*. IGI Global, Hershey, PA, USA.

Kantee, A. (2015). The rise and fall of the operating system. *USENIX login*, 40(5).

Kim, H., Lim, H., Jeong, J., Jo, H., and Lee, J. (2009). Task-aware virtual machine scheduling for i/o performance. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, pages 101–110, New York, NY, USA. ACM.

Kim, H. and Rajkumar, R. (2016). Real-time cache management for multi-core virtualization. In *2016 International Conference on Embedded Software (EMSOFT)*, pages 1–10.

Konstanteli, K., Cucinotta, T., Psychas, K., and Varvarigou, T. A. (2014). Elastic admission control for federated cloud services. *IEEE Transactions on Cloud Computing*, 2(3):348–361.

Lelli, J., Scordino, C., Abeni, L., and Faggioli, D. (2016). Deadline scheduling in the linux kernel. *Software: Practice and Experience*, 46(6):821–839.

Madhavapeddy, A., Mortier, R., Rotsos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., and Crowcroft, J. (2013). Unikernels: Library operating systems for the cloud. *SIGARCH Comput. Archit. News*, 41(1):461–472.

Mann, Z. A. (2015). Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. *ACM Comput. Surv.*, 48(1):11:1–11:34.

Networks, J. (2017). Network transformation with NFV and SDN – a journey toward sustainable competitive advantage. http://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000628-en.pdf.

Orchestration, T. O. (2017). https://wiki.openstack.org/wiki/Tacker. Last accessed on Dec 6, 2017.

Pellerin, D. (2017). Accelerated Computing on AWS – Applications for GPUs and FPGAs. www.asapconference.org/slides/amazon.pdf.

Pongrcz, G., Molnr, L., and Kis, Z. L. (2013). Removing roadblocks from sdn: Openflow software switch performance on intel dpdk. In *2013 Second European Workshop on Software Defined Networks*, pages 62–67.

Public, C. (2016). Align operations for network functions virtualization environments. https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/network-functions-virtualization-nfv/white-paper-c11-737101.pdf.

Rizzo, L. (2012a). netmap: A novel framework for fast packet i/o. In *2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 101–112, Boston, MA. USENIX Association.

Rizzo, L. (2012b). Revisiting network i/o apis: The netmap framework. *Queue*, 10(1):30:30–30:39.

Rizzo, L. and Lettieri, G. (2012). Vale, a switched ethernet for virtual machines. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 61–72, New York, NY, USA. ACM.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.

SCTE (2016). Scte analysis of available energy 2020 participating mso data. Brochure.

Steiner, M., Gaglianello, B. G., Gurbani, V., Hilt, V., Roome, W., Scharf, M., and Voith, T. (2012). Network-aware service placement in a distributed cloud environment. *SIGCOMM Comput. Commun. Rev.*, 42(4):73–74.

Valcarenghi, L., Giannone, F., Manicone, D., and Castoldi, P. (2017). Virtualized enb latency limits. In *2017 19th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4.

Vitucci, C. and Larsson, A. (2017). Flexible 5G Edge Server for Multi Industry Service Network. *International Journal on Advances in Networks and Services*, 10(3-4). ISSN: 1942-2644.

White, R. (2015). Introduction to Virtualization. `http://www.vm.ibm.com/devpages/bitner/presentations/virtualb.pdf`.

Xi, S., Li, C., Lu, C., Gill, C. D., Xu, M., Phan, L. T. X., Lee, I., and Sokolsky, O. (2015). Rt-open stack: Cpu resource management for real-time cloud computing. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 179–186.

Xi, S., Wilson, J., Lu, C., and Gill, C. (2011). Rt-xen: Towards real-time hypervisor scheduling in xen. In *2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT)*, pages 39–48.

Xia, Y., Zhou, M., Luo, X., Pang, S., and Zhu, Q. (2015). Stochastic modeling and performance analysis of migration-enabled and error-prone clouds. *IEEE Transactions on Industrial Informatics*, 11(2):495–504.

Yao, Y., Cao, J., and Li, M. (2013). A network-aware virtual machine allocation in cloud datacenter. In *Proceedings of the 10th IFIP International Conference on Network and Parallel Computing - Volume 8147*, NPC 2013, pages 71–82, New York, NY, USA. Springer-Verlag New York, Inc.

Yun, H., Yao, G., Pellizzoni, R., Caccamo, M., and Sha, L. (2013). Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms. In *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 55–64.

Zhang, J., Wang, M., Luo, J., Dong, F., and Zhang, J. (2015). Towards optimized scheduling for data-intensive scientific workflow in multiple datacenter environment. *Concurr. Comput. : Pract. Exper.*, 27(18):5606–5622.