Future Generation Computer Systems 32 (2014) 118–127

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

VM consolidation: A real case based on OpenStack Cloud

Antonio Corradi, Mario Fanelli *, Luca Foschini

Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), University of Bologna, Italy

ARTICLE INFO

Article history:
Received 15 November 2011
Received in revised form
4 April 2012
Accepted 17 May 2012
Available online 4 June 2012

Keywords:
Cloud computing
VM consolidation
OpenStack

ABSTRACT

In recent years, Cloud computing has been emerging as the next big revolution in both computer networks and Web provisioning. Because of raised expectations, several vendors, such as Amazon and IBM, started designing, developing, and deploying Cloud solutions to optimize the usage of their own data centers, and some open-source solutions are also underway, such as Eucalyptus and OpenStack. Cloud architectures exploit virtualization techniques to provision multiple Virtual Machines (VMs) on the same physical host, so as to efficiently use available resources, for instance, to consolidate VMs in the minimal number of physical servers to reduce the runtime power consumption. VM consolidation has to carefully consider the aggregated resource consumption of co-located VMs, in order to avoid performance reductions and Service Level Agreement (SLA) violations. While various works have already treated the VM consolidation problem from a theoretical perspective, this paper focuses on it from a more practical viewpoint, with specific attention on the consolidation aspects related to power, CPU, and networking resource sharing. Moreover, the paper proposes a Cloud management platform to optimize VM consolidation along three main dimensions, namely power consumption, host resources, and networking. Reported experimental results point out that interferences between co-located VMs have to be carefully considered to avoid placement solutions that, although being feasible from a more theoretical viewpoint, cannot ensure VM provisioning with SLA guarantees.

1. Introduction

Cloud computing architectures have gained more and more attention in recent years, and several vendors are looking at them for feasible solutions to optimize the usage of their own infrastructures [1,2]. In fact, among several advantages, these solutions offer pools of virtualized computing resources, paid on a pay-per-use basis, and drastically reduce the initial investment and maintenance costs. Hence, the main Cloud strength lies in offering computational resources to third-party service providers that do not intend to build and maintain their own IT infrastructure.

At the current stage, however, several management issues still deserve additional investigations, such as in Cloud architectures for large-scale data centers that introduce high economic investments for the Cloud owner. Above all, the power required by both IT resources (servers and network elements) and cooling systems introduces a significant economic overhead. Hence, also driven by the emerging Green Computing research [3,4], Cloud providers are looking for efficient techniques that dynamically reconfigure the IT infrastructure to reduce the total power consumption of data centers. Toward this goal, careful usage of virtualization techniques attempts to reduce power consumption by consolidating the execution of multiple Virtual Machines (VMs) on the same physical host, often called VM consolidation. The main idea is to execute the VMs on as few physical servers as possible to concentrate the workload and to efficiently limit the number of physical servers powered on. For instance, we can have two physical servers that run one VM each and both are not using their maximum computational capacity; hence, the allocation of both VMs to the same physical server can profitably switch one server off. It goes without saying that the power savings will be more and more effective in large-scale data centers where many physical servers can be managed to save more and more energy.

VM consolidation raises also several management issues because it tends to an optimal exploitation of available resources, while avoiding severe performance degradation due to resource consumption of co-located VMs. On the one hand, in a more formal perspective and with main focus on optimization issues, several works have already addressed VM consolidation by considering local physical node resource constraints, namely CPU and memory sharing, and several algorithms have been proposed to solve the VM consolidation problem with different objectives, such as increasing load balancing among servers, minimizing the number of powered on servers, and so forth [5–7]. On the other hand, in a more practical perspective, the deployment of those studies and results in real Cloud environments introduces new challenges, by calling for new in-the-field studies to verify their usability and

* Corresponding author.
E-mail addresses: antonio.corradi@unibo.it (A. Corradi), mario.fanelli@unibo.it (M. Fanelli), luca.foschini@unibo.it (L. Foschini).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.05.012

core assumptions. For instance, only a few seminal studies of real power-savings obtainable through VM consolidation for different types of services (CPU and network intensive) are now available and, similarly, in-depth studies about networking issues due to heavy communications between different VMs consolidated over the same physical server are also still missing, especially for new Cloud platforms, such as the very recent OpenStack [8].

Along those two directions, the first part of the article provides an updated overview of the current status of VM consolidation issues. It surveys the primary emerging solutions in the literature, and proposes a general architectural model to ease VM consolidation management and to optimize and control different system parameters, such as power, CPU load, and network load. In the second part of the paper, instead, we propose a management solution for VM consolidation for the very recent OpenStack Cloud. Differently from preexisting works, we use a practical perspective for VM consolidation in reducing power consumption by highlighting critical aspects. Then, we show that our envisioned VM consolidation support can greatly reduce power consumption, even for Clouds running on entry-level servers with limited hardware: we evaluate performance degradations and constraints to VM consolidation with a wide set of experimental results from a real OpenStack Cloud deployment. The proposed support components are available as open-source tools for the OpenStack community and, to the best of our knowledge, this is the first presentation of extensive performance results for OpenStack.

The paper is structured as follows. Section 2 gives VM consolidation background and related work; then, Section 3 motivates the need for advanced management solutions and proposes a new architecture for VM consolidation. Section 4 presents the OpenStack platform and its core services, while Section 5 shows extensive experimental results for three different case studies, namely raw VCPU computation, Apache HTTP server provider, and high communication scenarios. Finally, conclusions and directions of our current research end the article.

2. VM consolidation in Cloud computing scenarios: challenges and related work

Cloud computing architectures require complex management infrastructures to transparently deal with the extreme complexity introduced by modern data centers. This section introduces some useful background knowledge for a better collocation of our work. First, Section 2.1 clarifies the main challenges that Cloud management infrastructures must address. Then, Section 2.2 presents an in-depth discussion about the related work in this area, by detailing different VM placement problems and optimization goals, so as to offer a complete overview of the current ongoing research efforts.

2.1. Management challenges

Cloud computing architectures exploit virtualization technologies to support *VM consolidation*, namely the provisioning of multiple VMs on the same physical server. Management infrastructures with the goal of VM consolidation have to introduce and implement proper placement functions in charge of detailing the real VM-to-server mappings. Such placement functions can have different high-level objectives, such as minimizing the number of powered-on servers for the sake of power consumption, increasing provisioned VMs for the sake of economic revenue, and so forth. Although it is possible to highlight different and heterogeneous consolidation goals, at the current stage general and efficient management infrastructures for VM consolidation in Cloud scenarios are still an open issue.

More formally, a Cloud management infrastructure decides which VMs have to be co-located on the same physical server. Given a set of physical servers and a set of VMs, the placement function of the management infrastructure must find proper VM-to-server mappings that minimize an acceptable cost function. At the same time, the placement function has to consider multiple constraints, mainly coming from limited resources in physical servers. Hence, a placement function must face and address two main refinement directions: (1) an *objective function*, to rate how good a VM placement solution is; and (2) *resource constraints*, to avoid too aggressive and unfeasible VM consolidation solutions.

Starting with the objective function, Cloud providers usually require placement functions capable of increasing their economic revenues. Toward this goal, Cloud providers can either reduce the operational costs of the data centers, by reducing power consumption, or increase the number of customers, by increasing the aggregate number of VMs in the data center. Now, a very common goal is to reduce the number of powered-on servers, usually achieved by increasing VM consolidation ratios, so as to eventually increase data center power efficiency. Apart from that, a plethora of different, heterogeneous, and even contradictory goals can be defined. Load balancing between physical servers is important to avoid dangerous hot spots in the Cloud; in fact, overload situations are dangerous since they can easily lead to resource shortage and, at the same time, they can affect hardware lifetime, thus undermining data center reliability. Similarly, for the sake of reliability, a Cloud provider can introduce anti co-location goals to maximize both the spatial distance of the VMs and the use of disjoint routing paths into the data center, so as to prevent a single failure from affecting several VMs belonging to the same Cloud customer. It is clear that both load balancing and anti-co-location goals are partially in conflict with power saving, as they actually try to avoid the usage of high VM consolidation ratios. In general, VM placement has to deal with conflicting goals, combined through different weighting factors according to how the Cloud provider ranks them. Of course, that complicates both the design and the validation of placement functions.

Then, the placement function has to model and introduce resource constraints to reach meaningful solutions. Physical servers have limited CPU, memory, and I/O capacities that have to be carefully considered to avoid placements that may lead to low performance. At the current stage, constraints at host-level granularity are defined for many industrial and research solutions. Then, additional and more complex constraints associated with the data center as a whole may also be required. For instance, due to aggregated power consumption, it could be unfeasible to keep all the servers powered on at the same time. Similarly, from a networking viewpoint, the network topology adopted by the data center, as well as the routing schema, can greatly affect the real bandwidth available between two physical hosts [9]; that further complicates the placement problem since a specific solution, although feasible by considering a host-level view, can lead to link saturations into the network. In addition, since network traffic between co-located VMs is carried out locally, such as by using in-memory message passing mechanisms, it does not introduce real communications, thus saving precious network resources, but that increases CPU and memory overhead, with final runtime effects hard to estimate and dependent on virtualization platforms, device drivers, and so forth.

Apart from that, an important challenge is how the placement function models VM requirements and matches them with server and network capacities. Some works consider VM requirements equal to VM reservation attributes coming from Service Level Agreements (SLAs), that mainly depend on how much the service provider is willing to pay. However, since such static approaches can lead to underuse of physical resources, other solutions

introduce runtime monitoring and profiling of VMs to reach higher consolidation ratios, while minimizing the probability of violating SLAs. Finally, it is worth noting that several placement strategies are based on the assumption that VMs present repeatable patterns on same time scales, i.e., time of the day, period of the year, and so on. Usually, a pre-filtering phase is required to select an important, though reduced, subset of VMs as a locality that presents repeatable patterns and can be easily consolidated with low probability of reaching unfeasible placements due to time-varying resource requirements.

To conclude, VM placement functions for Cloud computing are complex solutions that have to deal with several heterogeneous aspects. In the next section, we introduce related works, with associated main issues and optimization goals, to clarify the current state of the art; this overview will be also useful to present the main aspects considered by our management infrastructure for the OpenStack Cloud, presented in Section 3.

2.2. Related work

VM consolidation has been intensively investigated in the last decade. For the sake of readability, we organized the related work along three main research directions: we start with the works that consolidate VMs for the sake of power consumption reduction; then, we move to the ones that consider objective functions related to CPU and memory capacities; finally, we highlight some works on network constraints and requirements.

It is generally accepted that Cloud power efficiency is an extremely important topic due to both environmental [3] and economic issues [10]: several works strived to reduce the number of powered on servers for the sake of power consumption reduction. [11] is a seminal work that focuses on the provisioning of Cloud resources for real-time services: the authors propose SLAs to drive resource allocation, and increase power efficiency by trading off task completion times and powered on servers. In [12], the authors introduce different VM consolidation techniques to reduce Cloud power consumption, but the proposed algorithms do not consider service provider SLAs, hence the real-world usage of such solutions can lead to performance degradations. In [13], assuming that service workload profiles are available, the authors introduce a mathematical model to find the number of physical servers that ensures power efficiency. The proposed solution considers dynamic CPU frequency scaling and details formulas that supply the value of that parameter, but the model assumes that Cloud jobs can be partitioned between different physical servers: this assumption, viable in specific scenarios, does not fit well the general VM allocation problem. Finally, Mistral is a solution to optimize both VM performance and power consumption, while also considering transient costs associated with runtime reconfigurations [14]. The authors exploit A*-search techniques to find complex reconfiguration actions, and consider indicators on the stability of the next configuration into the decision process; presented experimental results, obtained in a real Cloud testbed, make the proposed solution extremely solid.

Focusing on the second research direction, CPU and memory requirements are also important to optimize runtime performance, and many different works introduce them as placement objectives. Spare capacities along these resource dimensions allow VMs to dynamically change their requests with no need of additional VM migrations. Toward that goal, several research efforts propose to consolidate VMs that present uncorrelated resource demands, namely to consolidate VMs that present high CPU and memory demands at different times. If co-located VMs present such characteristics, they introduce aggregated resource demands that can be far below the sum of their singular requirements. On this subject, [15] introduces a VM placement solution that exploits the notion of equivalent capacity to consolidate an increased number of VMs for each server. The main assumption beyond these works is that VMs, despite requirements supplied by SLAs, do not usually exploit the maximum of their resources; hence, consolidation driven by SLA requirements would probably lead to low resource utilization. Finally, for the sake of placement computation, this first family of works tends to adopt complex stochastic mathematical models, and usually assumes that VM resource requirements are distributed according to a Gaussian random variable: the validity of those assumptions in real Cloud scenarios is still widely unexplored and has to be carefully verified. Other researchers consider VM consolidation driven by SLA reservation requirements. Although these approaches increase placement stability, since the management infrastructure always places only a set of VMs that actually fit physical servers, they can greatly reduce the number of consolidated VMs, by causing underused physical servers and resource waste. Moving to memory requirements, modern hypervisors implement page sharing algorithms between co-located VMs to reduce memory consumption. Hence, a few works in the literature tried to capture and to exploit page sharing characteristics between running VMs, with the main goal of co-locating similar VMs, so as to reduce the total memory footprint. For instance, Memory Buddies exploits information from the hypervisor to extract small footprints of the pages currently used by each VM [16]. Such footprints are sent to a centralized monitoring node that, by computing page sharing scores between running VMs, suggests proper consolidation decisions to reduce the amount of physical memory dedicated to VMs. Sindelar et al. [6] presents a formal analysis of the complexity of the VM page sharing problem, and shows that consolidating VMs with the main goal of increasing the overall page sharing between co-located VMs is an NP-hard problem. Hence, the authors introduce less flexible, but easier to manage, page sharing models based on both operating systems and application templates to deal with problem complexity. Although these works are highly relevant, they should be carefully adopted in real Cloud scenarios: a sudden change in VM read/write access patterns can hinder page sharing, thus leading to higher and perhaps unfeasible memory consumption.

Finally, along the third research direction, several works analyzed the VM placement problem to optimize data center network usage. Network resource constraints are usually introduced at a host-level granularity, by enforcing that the aggregate traffic from/to a physical host does not exceed the maximum capacity of the network interfaces. However, this very simplistic assumption does not consider that data center topologies can be extremely wide-scale hierarchical architectures where pairs of hosts can experience different time-varying bandwidth, according to routing schema and conflicting traffic demands. To the best of our knowledge, [9] is one of the most important works on network-aware VM placement: the authors studied the problem of VM placement with the goal of reducing the aggregate traffic flowing into the data center network. The proposed placement problem is NP-hard, hence, the authors introduce a new heuristic, based on clustering algorithms, to solve it in reasonable time. While this work assumes static and well-defined traffic demands, another important work exploits equivalent capacity notions to consolidate together VMs with uncorrelated traffic demands [17]; it considers network constraints with a host-level granularity, and strives to place VMs by ensuring that the local network capacity is violated with a probability lower than a predefined threshold. Since VM traffic demands are expressed by means of stochastic variables, the proposed problem is a stochastic bin packing, and the authors propose a new heuristic to solve it for reasonable Cloud scenarios. In [18], the authors considered the placement of applications made by a computation and a storage part. They proposed a new optimization

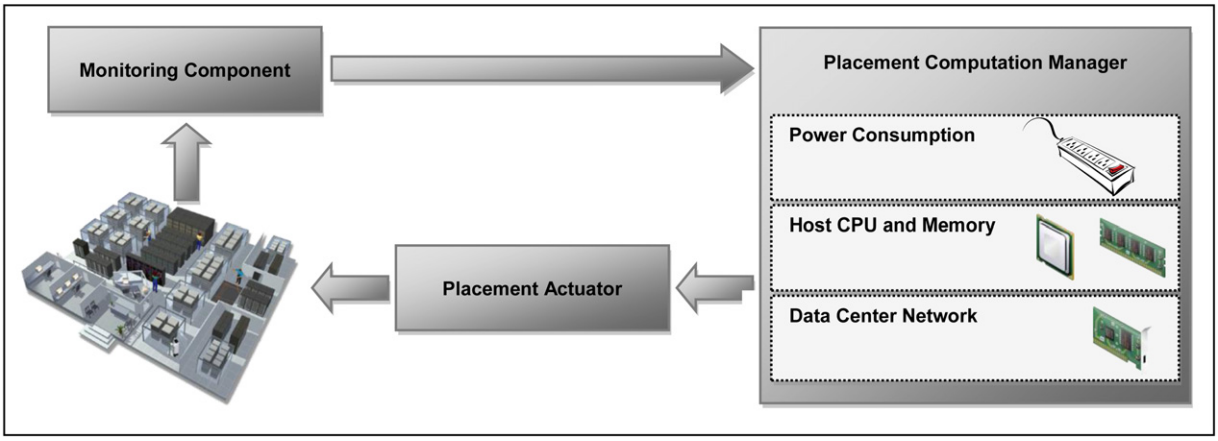

Fig. 1. Logical architecture of the Cloud management infrastructure.

problem to place both application parts, while reducing the runtime traffic on the network. However, they do not consider constraints on the single links, hence, although they reduce the overall traffic into the data center, obtained solutions could be unfeasible. Also, since traffic demands are only considered between the computation and the storage part of each application, and not between parts belonging to different applications, the resulting optimization problem is not as hard as the one considered in [9]. Instead, in [19], we propose a new network-aware VM placement problem, called Min Cut Ratio-aware VM Placement (MCRVMP), that strives to place VMs on physical hosts while reducing the worst-case load ratio over network topology cuts; in this way, our MCRVMP aims to increase the VM placement stability under time-varying traffic demands between VMs. Finally, it is worth remarking that none of the previous network-aware placement problems considers that local communications actually affect CPU and memory load; hence, obtained solutions tend to consolidate VMs by greedily increasing the traffic routed on the same physical host.

In conclusion, several works in the past have already dealt with VM consolidation to optimize particular objectives. Unfortunately, they mainly considered specific optimization goals, by largely leaving out the usage of combined solutions, for instance, dealing with both CPU and networking at the same time. In addition, aforementioned optimization problems do not consider the interferences introduced by VMs co-location; although such interferences would probably require significant efforts to be characterized, we anticipate that, as better detailed in Section 4, they are not negligible and deeply affect the feasibility of obtained placement solutions.

3. Cloud management infrastructure: design guidelines and architecture

Cloud management infrastructures are complex software stack solutions that require a deep understanding of different and heterogeneous aspects, spanning from optimization techniques to virtualization technologies and networking architectures. Currently, many industrial products offer comprehensive Cloud management infrastructures, capable of enabling dynamic resource provisioning into the data center, as well as scaling of Cloud services depending on current user requests. Unfortunately, open-source products, such as Eucalyptus and OpenStack, offer extremely primitive solutions as management infrastructures, and lack of more complex mechanisms to enable dynamic service provisioning and rescaling. Of course, this hinders the widespread adoption of such products in small Cloud environments, since the design and the implementation of new management infrastructures will probably incur unfeasible economic expenditures.

First, the Cloud management infrastructure has to gather a wide set of indicators, spanning from VM load metrics to power consumption of physical servers and network elements. Since data centers can include physical servers belonging to different vendors, it is important to integrate with and use different monitoring protocols. Second, the Cloud management infrastructure must decide the optimal VM placement. This decision is very complex from a computational viewpoint, since it includes the resolution of multi-criterion optimization problems with wide research spaces, limited by user SLAs, server computational power, etc. Finally, if an optimal placement exists, the Cloud management infrastructure has to dynamically reconfigure the data center, by detailing a suitable plan of VM relocation operations, and by taking over hypervisor reconfigurations, network path reconfigurations, etc.

To deal with this complexity, a Cloud management infrastructure usually exploits a three-stage architecture: the first stage collects monitoring data from the Cloud; the second one takes care of using collected data to calculate a new and more suitable VM placement, if available; finally, when it is the case, the third stage calculates the VM relocation plan and applies suggested changes to the Cloud. Hence, the proposed management infrastructure (see Fig. 1) consists of a *Monitoring Component*, a *Placement Computation Manager*, and a *Placement Actuator*, arranged in a pipeline where each stage exploits the output of the previous one. The *Monitoring Component* gathers both system and service level information about used resources, and makes them available to the next stage. The *Placement Computation Manager* exploits both monitoring information and user SLAs to check if a better VM placement exists; as showed in Fig. 1, it contains three main sub-modules to explicitly deal with power, host resources, and network. Finally, if a better placement exists, the *Placement Actuator* takes care of executing the VM migration plan and reconfiguration operations.

For the sake of brevity, and since it is more important in respect of the focus of this work, in the remainder we detail the Placement Computation Manager by introducing additional details about internal software modules. Since VM placement decisions consider three main dimensions, e.g., power consumption, host CPU and memory, and data center network, the following three sub-sections respectively analyze each of them, by introducing common optimization goals and challenges to consider.

3.1. Power consumption

Power efficiency in modern data centers is important for two main reasons. First, from an environmental viewpoint, Gartner remarks that worldwide IT infrastructures are responsible for 2% of the global carbon dioxide emission [3]. Second, from an economic viewpoint, Gartner also estimates that energy-related costs

account for 12% of total data center economic expenditures [10]. Hence, power-aware efficient Cloud management is important for both environmental and economic perspectives.

To implement such a module, the Cloud management infrastructure needs access to both load and power consumption indicators, to be combined by a suitable management policy to reach power efficiency. Of course, the processing power of the data center is inherently related to the power consumption of the physical devices and it increases by powering on additional physical hosts and network elements. Hence, in general, the higher the processing power, the higher the power consumed by the data center will be. However, the Cloud placement function should focus on power efficiency rather than on total power consumption; in particular, we say that a data center A is more power efficient than a data center B if either A can process more workload than B with the same power consumption, or A can process the same workload as B but with less power consumption.

To finely control runtime power consumption, different Cloud management infrastructures exploit advanced management techniques to ensure that each physical server does not exceed a fixed maximum power consumption threshold, such as by dynamically scaling CPU frequency. Since the power consumption can change over time, the processing power of physical hosts also varies sometimes; that can make the current VM placement unfeasible due to the lack of required resources. In addition, VMs introduce time-varying workloads depending on hosted services. On the one side, during peak hours, the workload tends to increase, and the current VM allocation could become unfeasible due to an excess of resource request. On the other side, during night hours, workload tends to decrease, and the data center could end up keeping on unneeded and underutilized physical servers. In summary, since VMs should be dynamically allocated to as few physical servers as possible, VM live migration is fundamental to increase Cloud power efficiency.

3.2. Host CPU and memory

VMs co-located on the same physical host share both CPU and memory resources. Depending on user SLAs, the Cloud management infrastructure is in charge of reconfiguring hypervisors, mainly to avoid VMs proposing more computational load than allowed. The main goal of consolidation is to pack as many VMs as possible on the same physical host, while avoiding resource shortage. However, CPU and memory resources present a few important challenges.

First of all, CPU load measurements are extremely time-varying and affected by errors. The usage of low-pass filters, such as a weighted average upon a limited set of samples, is mandatory to have meaningful load indicators. Similarly, forecasting techniques should be adopted to highlight important trends, in order to avoid management decisions based on temporary load spikes. In addition, there are intrinsic side effects of the VM consolidation that hinder the precise estimation of such load indicators. In fact, as co-located VMs interfere among themselves, it could be the case that profiled CPU loads of VMs are lower than actual requests [7]; this, in its turn, affects VM placement decisions, and can introduce low convergence toward a final placement that effectively meets all the actual demands of the VMs. Second, as stated before, modern hypervisors introduce page sharing facilities among co-located VMs. Although page sharing reduces the physical memory needed to host VMs and can lead to higher consolidation ratios, it is difficult to predict: the shared page set deeply depends on the services currently executed in the VMs, and the fact that two VMs are booted from the same image does not imply that they can share a large set of pages. In addition, the greedy consolidation of VMs sharing a large set of pages increases the probability of incurring unfeasible placements whenever the VM access patterns to the memory change. In fact, this could lead to a much smaller set of shared pages, thus increasing the usage of the physical memory and leading to page thrashing to the disk when the running VMs cannot be accommodated anymore.

3.3. Data center network

Network is intrinsically shared by all the VMs belonging to the same Cloud data center. Although Virtual LANs (VLANs) are usually adopted for security reasons, network paths and switching elements are the same and traffic demands belonging to different Cloud customers interfere among them, thus possibly resulting in reduced service performance. At the same time, both the modeling and the introduction of network constraints are extremely complex tasks to be addressed.

First of all, a first key challenge is to understand which metrics are important from the network point-of-view: bandwidth and latency are two fundamental attributes, with extremely different weights depending on the actual service provisioned. For instance, MapReduce services rate bandwidth as more important since it reduces the time duration of the data shuffling phase [20]; interactive services, instead, prefer low latency to avoid noisy side-effects during user interaction. Meng et al. [9] has shown that, in private data centers, the inter-host latency is almost stable irrespective of physical server location, while the maximum bandwidth available between two servers can deeply depend on the data center network architecture. In fact, data center networks usually exploit hierarchical architectures where, due to physical and economic constraints, links belonging to higher levels cannot carry the worst-case aggregate traffic from one side of the data center to the other. To alleviate such problems, graph-based networks, such as Fat-tree and VL2 [21,22], are preferred to tree-based ones due to increased scalability and reliability; however, even in this case, specific routing schema and link sizing greatly affect available bandwidth.

Although it is possible to have a rough estimation of the maximum bandwidth available between two physical servers, another fundamental challenge arises when we consider how the management infrastructure actually details inter-VM traffic demands. Cloud customers do not have the knowledge and the patience to express expected bandwidth requirements between VMs. Hence, the management infrastructure has to take over network monitoring responsibilities, so as to have a good estimation of the bandwidth required to carry inter-VM traffic demands. In this way, the placement function can consolidate VMs, while avoiding dangerous link saturations. However, the simple monitoring of VM-to-VM bandwidth requirements in a large data center can require not straightforward solutions; since hypervisors usually supply access to aggregate traffic indicators, the monitoring of the traffic routed toward a particular destination requires more sophisticated solutions whose applicability in 1/10 Gbps networks is not easy to assess.

Finally, from the networking viewpoint, unfeasible placements are extremely difficult to recover through VM live migration. In fact, if network links are close to congestion, and the Cloud data center does not have a separate network for VM migration, the introduced overhead can definitely choke close-to-congestion links and networking elements. In addition, a live migration based on the pre-copy approach, carried through a congested network, can result in a very long and useless initial copy phase that is not able to keep up with page dirtying rates [23].

In conclusion, VM consolidation is a fundamental mechanism in modern Cloud data centers but, at the same time, it requires complex solutions usually not available in open-source products. Hence, we decided to focus this article on the runtime side-effects introduced by VM consolidation, by also evaluating them

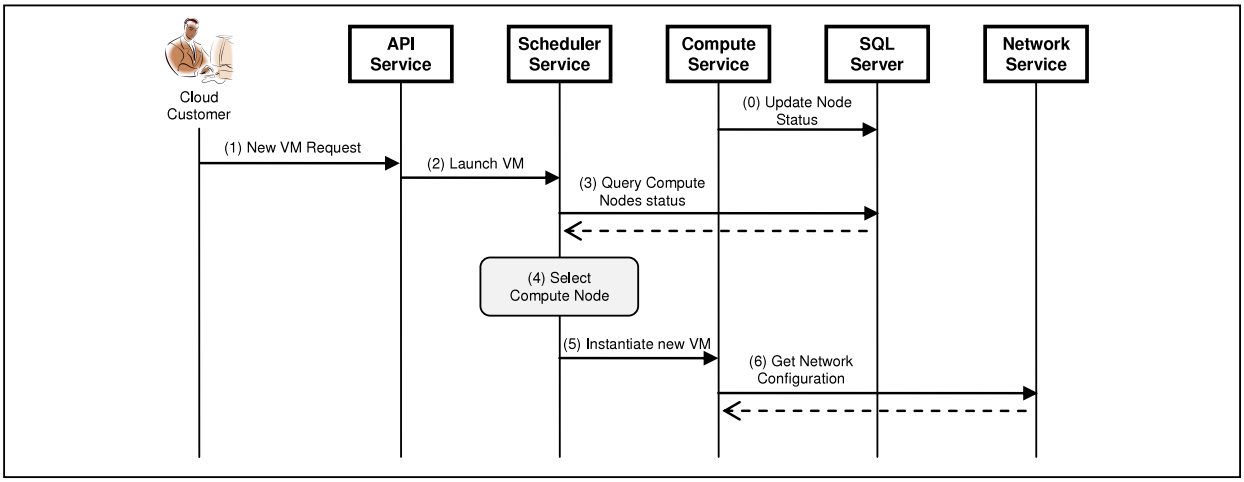

Fig. 2. VM instantiation in OpenStack.

through the OpenStack Cloud. In this way, we hope that both our management infrastructure and experimental results will supply useful hints to the OpenStack community for designing and implementing novel Cloud management mechanisms.

4. OpenStack architecture

OpenStack is an open-source solution for creating and managing Cloud infrastructures [8], originally developed by NASA and Rackspace. The open-sourceness of this package, being free to download, gives to small players the possibility of deploying small Cloud infrastructures. Unfortunately, OpenStack lacks advanced monitoring and reconfiguration mechanisms useful to implement dynamic service scaling facilities; hence, although it is one of the most complete open-source Cloud products, the usage of such a solution in specific scenarios can require extensive tailoring modifications to the management infrastructure.

OpenStack exploits well known open-source components and libraries, and manages both computation and storage resources on the Cloud to enable dynamic allocations of VMs. OpenStack rests upon two different major projects: the first one, called Nova [24], mainly manages computation and network resources, while the second one, called Swift [25], is in charge of the distributed storage into the Cloud. For the sake of clarity, we detail four main services offered by Nova because they are significant for a better understanding of our work.

The *API Service* is the frontend to the OpenStack Cloud: it receives user requests and translates them into Cloud actions. The offered functions span from VM control to authentication and authorization. In particular, this service exports such functionalities through Web Services: user requests are delivered by HTTP messages with XML-based payload, converted into suitable commands, and dispatched to the final internal component. To avoid vendor lock-in, and ease service migration toward different Cloud providers, OpenStack exports a set of functions compliant with the ones offered by different vendors, such as Amazon and Eucalyptus. For instance, to interact with OpenStack, it is possible to use the Euca2tools, a suite of command-line tools of the Eucalyptus project to interact with Web Services compliant with Amazon EC2 and S3 services [26,27].

The *Compute Service*, one for each node belonging to the Cloud, launches and configures VMs within a certain physical server. It mainly handles communications with the local hypervisor, so as to enable VM instantiation and termination, as well as queries to VM load indicators and performance metrics. OpenStack supports multiple hypervisors; at the current stage, KVM is one of the most widespread due to its good performance, as it offers a full virtualization solution on x86 architectures. In addition, OpenStack offers live migration capabilities, but this is possible only with (1) KVM as hypervisor, and if (2) NFS is used for the network storage. To easily manage interactions with virtualized operating systems running on the physical server, OpenStack uses the open-source library libvirt [28], that exposes a set of hypervisor-independent APIs to enumerate and monitor the VMs executed on the current physical server.

The *Network Service* handles all the aspects related to network configuration and communications. In particular, for each server, an instance of this service is in charge of creating virtual networks useful to let VMs communicate between themselves and with the outside of the Cloud. Each VM has a private IP address assigned during boot; then, only if necessary, it is possible to reserve and associate a public IP to make the VM available over the Internet. Of course, VMs co-located on the same physical host communicate through local message passing mechanisms. Since the performance of such primitives greatly depends on the virtualization solution used for I/O devices, we configured OpenStack to use virtio [29], an open-source Linux standard for network and disk drivers where, similarly to what usually happens with para-virtualization, the device driver used by the VM is aware of running in a virtualized environment. In this way, there is a tighter cooperation between device drivers and hypervisor, thus enhancing network and disk performance.

Finally, the *Scheduler Service* decides the node in which a new VM has to be launched. At the current stage, as consequence of a new VM instantiation, the default OpenStack scheduler applies very simple policies, such as randomly selecting an available server, choosing the least loaded server, etc. VMs are never migrated at runtime since OpenStack does not offer mature solutions for dynamic VM consolidation.

To illustrate how the different OpenStack services interact, we briefly introduce the simple use case of a new VM instantiation (see Fig. 2). Above all, each Compute Service periodically sends updates to a SQL Server (step 0) that stores load information about provisioned VMs and physical servers. When a new VM instance is requested (step 1), the API Service sends a request to the Scheduler Service (step 2) that, in its turn, chooses the Compute Service in which the VM should be launched. In this step, the Scheduler Service queries the database to get the list of available servers (step 3), and then chooses one of them (step 4) depending on the adopted policy (random, less loaded, etc.). Finally, it sends a new VM instantiation request to the chosen compute node (step 5) that, in its turn, requests network configuration parameters (IP, gateways, etc.) for the new VM to the Network Service (step 6).

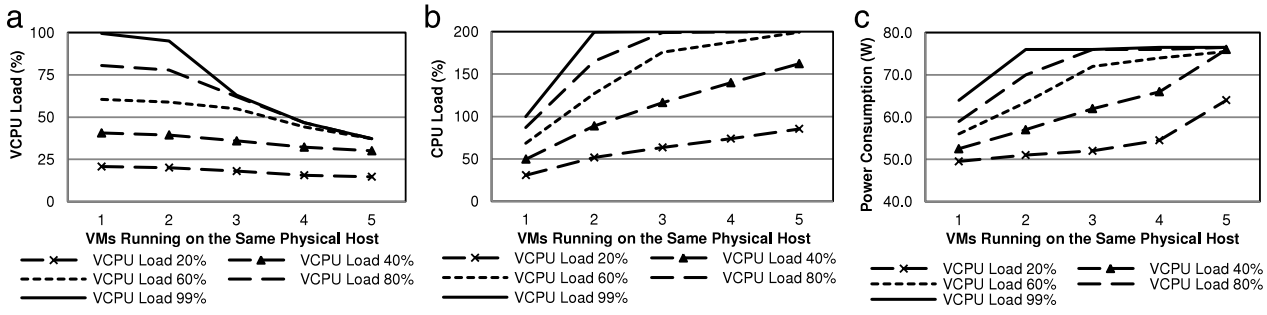

Fig. 3. VCPU load (a), aggregated host CPU load (b), and power consumption (c) for the raw VCPU test.

5. Experimental results

In this section, we present some extensive experimental results to better evaluate the relationships between server consolidation, power saving, and VM performance. For all our tests, we used a testbed made by physical servers with CPU Intel Core Duo E7600 @ 3.06 GHz, 4 GB RAM, and 250 GB HD, and we used KVM as hypervisor, while all the VMs have 1 VCPU and 512 MB RAM. In addition, to monitor power consumption, we employed a wattmeter connected between the server and the power plug. Let us remark that we deployed our infrastructure over traditional personal computers, instead of more powerful and expensive machines, so as to mimic a limited Cloud deployment, like the ones usually available in small industries. Finally, we installed and configured the latest OpenStack release, called Diablo, to better investigate all the current features [8].

For the sake of better readability, we organize the following experiments along three main and exemplar case studies. The first one, *raw VCPU computation*, aims to highlight performance degradation when multiple co-located VMs execute heavy computational tasks. The second one, *Apache HTTP server provider*, considers VMs hosting Apache HTTP servers to remark possible side-effects when many co-located Web servers process multiple incoming requests. The third one, *high network communication*, aims to highlight the performance bottlenecks introduced by virtualization when VMs perform high data transfer, such as when VMs host MapReduce services. Finally, let us remark that, in all shown experiments, we focus on challenging scenarios where the aggregate resource consumption introduced by co-located VMs is higher than the one available at the physical host. We deliberately do not introduce SLA requirements in VM provisioning as our main goal is to draw important guidelines to be used by either the placement process or the SLA negotiation phase. In particular, we focus on highlighting system limitations that may bind SLAs according to different consolidation ratios (from 1 to 5 VMs) and types of services (starting with CPU-bound to conclude with network-bound).

5.1. Raw VCPU computation

In the first set of experiments, we have measured the performance of each single VM running a simple program to impose a particular VCPU load. We have considered an increasing VCPU load of {20%, 40%, 60%, 80%, 99%}, and we compare the current VCPU load reached for different consolidation ratios. The difference between the requested and the current VCPU load implicitly is a measurement of the interferences between VMs; all VMs run on the same one physical host. In particular, we start with an ideal situation where we allocate only one VM to the physical host, and then we progressively increment the number of VMs: all VMs run the same program with the same VCPU load.

Fig. 3(a) shows the VCPU load actually reached by the process according to different degrees of consolidation. For all the tested VCPU loads, since the physical CPU has two cores, performance does not decrease significantly if we consolidate up to two VMs. Instead, when the aggregate VCPU load of all the VMs is higher than 200%, namely the maximum achievable CPU load due to the physical architecture, performance degradations arise. At the same time, even when performance degradations are not expected, such as the case of 3 VMs with 60% VCPU load, we can actually highlight a small degradation due to interferences with the operating system of the physical server. Fig. 3(b) represents the real cumulative load of the physical CPU collected in the standard top tool. By comparing Fig. 3(a) with Fig. 3(b), we remark that VM performance degradations actually start when the physical CPU load reaches 200%. Finally, with a completely different perspective, Fig. 3(c) shows the total power consumption of the physical server depending on different VCPU loads and number of running VMs. For each VCPU load, the higher the number of running VMs, the higher the total power consumption due to increased computation. In addition, with VCPU load equal to 99%, we reach the maximum power consumption of 75 W with only two consolidated VMs. In this case, since no access to disk and network is performed, the power consumption of the physical server is mainly related to the CPU load.

Using above results, we can easily compute the power savings obtained through VM consolidation. One running VM consumes a great part of the whole physical server power, i.e., from about 50 to 65 W, according to the VCPU load. Without consolidation, two VMs would need two physical servers, thus consuming about twice the power required to power on one physical server with two VMs running on it. Of course, the final power savings depend on the user SLAs and on the physical servers. If each VM has tight SLA with a VCPU load of 99%, we can consolidate up to two VMs, thus saving more than 42% of the power required to power on two different servers. If VMs have all best effort SLAs, we can consolidate five VMs on one physical server, thus saving more than 80% the power required to turn on five physical servers. Apart from that, it is important to note that, in this configuration test, VMs are fairly served; this means that, for raw VCPU computation, the Cloud management infrastructure can easily foresee performance degradations as a consequence of consolidation.

5.2. Apache HTTP Server provider

In the second set of experiments, we have considered VMs hosting Apache HTTP Server (configured with the default parameters), and we have stressed them through the Apache HTTP Server Benchmarking Tool (ab) [30]. The main goal of this test is to highlight how server consolidation affects the final performance when we consider programs making network communications. To separate virtualization overhead from side-effects associated with CPU, disk, and network saturation, during this test we always request a small static HTML page of around 50 bytes. The ab client is executed on a different physical server, connected to the one hosting the VMs through a 1 Gbps link. As performance metrics, we consider the average response time and the number of failed requests

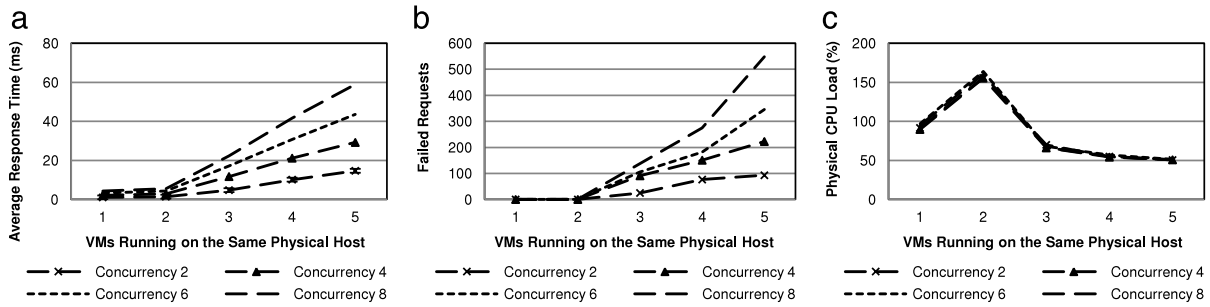

Fig. 4. Average response time (a), number of failed requests (b), and physical CPU load (c) for the Apache HTTP Server test.

as ascertained by ab. For each test, ab performs 100,000 requests with an increasing number of concurrent requests in {2, 4, 6, 8}.

Fig. 4(a) and (b) show respectively the average response time and the number of failed requests according to different degree of concurrent requests. With consolidation ratios higher than 1, these values represent the average between the corresponding values supplied by each ab client, one for each Apache HTTP Server under test. In addition, the average response time shown in Fig. 4(a) is an aggregate indicator referring to batches of concurrent requests: for instance, if we consider a concurrency of eight requests and five VMs running on the same physical server, each VM receives eight requests, and 60 ms is the mean time required to receive all eight responses for the request batches by all VM–ab client pairs. In brief, Fig. 4(a) shows that higher consolidation ratios result in higher average response times, hence, a reduced number of requests/second. From Fig. 4(b), it is possible to remark that increased consolidation ratios, as well as concurrency levels, lead to an increased number of failed requests. Since Apache HTTP has no limitation on the number of processed requests, and ab exploits a large timeout of 30 s before considering a request expired, we remark that the failed requests of Fig. 4(b) are a consequence of errors during TCP connection setup.

Apart from these two indicators, Fig. 4(c) shows the aggregate CPU load experienced by the physical server on which VMs execute (we recall that the highest value is 200% due to the availability of two cores). First of all, ab truly stresses the Web servers by continuously sending requests as soon as possible; in fact, focusing on the case of only one VM hosted on the physical server, all the adopted concurrency levels lead to a physical load of 100%, i.e., one core is fully loaded by the KVM process associated with the VM under test. Second, for different concurrency levels, the CPU load experienced by the physical host does not change in a remarkable way: this is mainly due to the fact that the requested page is a simple HTML page that does not introduce heavy processing while, moving to higher consolidation ratios, performance decreases as expected due to CPU saturation. Fig. 4(c) stresses also other interesting aspects: for a consolidation ratio of 2, the CPU load never reaches 200% because, although VMs require increased computational resources to process incoming requests, it seems that the hypervisor is not able to supply such resources by exploiting all the available physical resources. That is mainly due to the increased synchronization overhead: since more VMs need to perform network communications, the hypervisor has to arbitrate access to network interface resources and device drivers, thus leading to lower performance. This is a fundamental difference from the previous case study, where the usage of raw VCPU computation makes performance degradation more predictable. In addition, with consolidation ratios equal to and higher than 3, the average CPU load decreases due to the self-adaptive ab behavior: in fact, when the number of failures due to TCP connection setup increases, ab automatically reduces the request sending rate, thus decreasing the CPU load; at the same time, we get much longer test execution times, ranging from 50 s for a consolidation ratio of 1, to more than 700 s for a consolidation ratio of 5 (these times are not shown in the figure).

Finally, for the sake of brevity, we do not show the power consumption results for this case study. Since the power consumption is mainly related to CPU load, as shown in the previous case study (see Fig. 3(b) and (c)), the indicator can be easily derived from Fig. 4(c).

5.3. High network communication

As shown in the previous section, network communications increase the overhead on the physical host because of the VMs requiring access to I/O device drivers. This case study completely focuses on the overhead introduced by the hypervisor when hosted VMs exchange huge amounts of data among themselves. To avoid possible side-effects related to CPU saturation, for the following tests, we have adopted two physical hosts with an Intel i7 CPU (four physical cores and eight virtual cores thanks to Hyper-Threading), connected by a 1 Gbps link. In addition, to simulate constant UDP traffic flows with different bandwidth requirements, we used Iperf, a well-known open-source tool useful to estimate real available bandwidth [31].

We considered from two to four pairs of VMs that exchange data through Iperf. For each pair, one VM executes the Iperf server, while the other executes the client; all the VMs hosting the servers (respectively, the clients) are co-located on the same physical host. Fig. 5(a)–(c) represent the maximum UDP bandwidth measured through Iperf when we respectively consider two to four pairs of VMs. For each graph, the *x*-axis represents the rate imposed at the Iperf client, while the *y* value represents the actual bandwidth measured by Iperf. In addition, we introduce two lines; the first one, named "Cumulative BW", is the aggregate bandwidth reached by all the VM pairs involved in the test; the second one, called "Ideal BW", is the ideal aggregate bandwidth we would expect due to rates imposed at the client side (it is limited at 1 Gbps due to physical limitations of the link).

Starting with Fig. 5(a), we note that, for two pairs of VMs, the required bandwidth is almost equal to the experienced one only when each client requires less than 150 Mbps; for higher demands, the experienced bandwidth is always capped around 200 Mbps for each VM pair. Even more interesting, such a threshold remains in all the test cases, spanning from two to four pairs of VMs (see Fig. 5(a)–(c)), as a consequence of the high CPU load introduced by network communications. Although the Iperf client introduces contained overhead for packet generation, our experiments show that, under high communication loads, the KVM process associated with the VM reaches 100% CPU utilization; this means that network communications are extremely expensive and lead to resource saturation of the VM, thus finally reducing the final bandwidth available to communicate. Hence, VMs running network-bound services, such as MapReduce, can experience low performance due to the very high CPU overhead introduced by network communications.

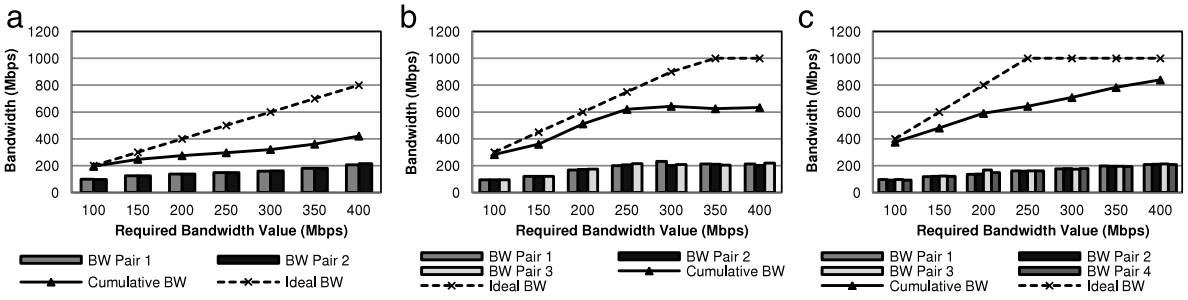

Fig. 5. Maximum UDP bandwidth for two (a), three (b), and four (c) pairs of VMs for the high network communication test.

5.4. Discussions

From the above experimental results, we conclude that VM consolidation introduces novel challenges that have to be considered by the Cloud management infrastructure. Although co-locating VMs on the same physical server is always convenient in terms of power savings, it can lead to performance side-effects that, as presented above, strictly depend on the type of run services. On the one hand, raw computation services are easier to manage since final performance degradation can be easily predicted: as shown in Section 5.1, in the case of CPU overload, all the VMs receive a fair amount of CPU. On the other hand, network-bound services are more difficult to manage as the final performance degradation is a consequence of a mix of CPU and network overheads, where the second one also affects the CPU load of the physical server. Even worse, performance degradation due to networking truly depends on the virtualization platform and on the adopted device drivers. All the above results have been collected in a default OpenStack deployment with KVM and virtio; of course, tuned implementations of both the hypervisor and the device drivers can lead to better results, although high CPU loads are always expected.

Hence, service profiles, describing the main resources used by the service at runtime, could be extremely useful to better guide the VM consolidation process. As pointed out in the related work section, various works in the literature do not consider interference due to VM co-locations; although such side-effects are difficult to measure, the Cloud management infrastructure should strive to reach a balance between CPU-bound and network-bound services on the same host, so as to avoid excessive overhead along any of the resource dimensions. Such profiles will be also useful to speed up the placement computation: in fact, they can introduce significant additional constraints that opportunely further limit the solution space to examine.

6. Conclusion

In this paper, we discussed the problem of VM consolidation in Cloud scenarios, by clarifying main optimization goals, design guidelines, and challenges. To better support our assumptions, we introduced and used OpenStack, an open-source platform for Cloud computing that is now widely adopted both in academia and in the industrial worlds. Our experimental results convinced us that VM consolidation is an extremely feasible solution to reduce power consumption but, at the same time, it has to be carefully guided to prevent excessive performance degradation. By using three significant case studies, very representative of different usages, we have shown that performance degradation is not easy to predict, due to many entangled and interrelated aspects.

At this point, we are interested in going on investigating other important research directions. First, we want to better understand how server consolidation affects the performance of single services and the role of SLAs in the decision process. Our main research goal along that direction is the automatic identification of meaningful service profiles useful to detail introduced workload, e.g., either CPU or network bound, to better foresee VM consolidation interferences. Second, we want to deploy a larger testbed of the OpenStack Cloud, so as to enable and test more complex VM placement algorithms. Third, we want to extend the management infrastructure to perform automatic VM live migration, in order to dynamically reduce Cloud power consumption: our main research guideline is to consider historical data and service profiles to better characterize VM consolidation side-effects.

References

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Elsevier Future Generation Computer Systems 25 (2009) 599–616.

[2] A. Lenk, M. Klems, J. Nimis, S. Tai, T. Sandholm, What's inside the Cloud? an architectural map of the Cloud landscape, in: Proc. of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, 2009, pp. 23–31.

[3] Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO2 Emissions: http://www.gartner.com/it/page.jsp?id=503867, 2011.

[4] S. Murugesan, Harnessing green IT: principles and practices, IEEE IT Professional 10 (2008) 24–33.

[5] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, L. Uyeda, U. Wieder, Validating heuristics for virtual machines consolidation, Technical Report, 2011.

[6] M. Sindelar, R.K. Sitaraman, P. Shenoy, Sharing-aware algorithms for virtual machine colocation, in: Proc. of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures, SPAA'11, 2011.

[7] C. Isci, J.E. Hanson, I. Whalley, M. Steinder, J.O. Kephart, Runtime demand estimation for effective dynamic resource management, in: Proc. of the IEEE Network Operations and Management Symposium, NOMS, 2010.

[8] OpenStack Cloud: http://www.openstack.org/, 2011.

[9] X. Meng, V. Pappas, L. Zhang, Improving the scalability of data center networks with traffic-aware virtual machine placement, in: Proc. of the 29th Conference on Information Communications (INFOCOM'10), 2010.

[10] Gartner Says Energy-Related Costs Account for Approximately 12% of Overall Data Center Expenditures: http://www.gartner.com/it/page.jsp?id=1442113, 2011.

[11] K.H. Kim, A. Beloglazov, R. Buyya, Power-aware provisioning of cloud resources for real-time services, in: Proc. of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, MGC 2009, 2009.

[12] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: Proc. of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010.

[13] H.S. Abdelsalam, K. Maly, R. Mukkamala, M. Zubair, D. Kaminsky, Analysis of energy efficiency in Clouds, in: Proc. of the Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009.

[14] G. Jung, M.A. Hiltunen, K.R. Joshi, R.D. Schlichting, C. Pu, Mistral: dynamically managing power, performance, and adaptation cost in Cloud infrastructures, in: Proc. of the IEEE 30th International Conference on Distributed Computing Systems, ICDCS'10, 2010.

[15] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, K. Yoshihira, Effective VM sizing in virtualized data centers, in: Proc. of the IFIP/IEEE International Symposium on Integrated Network Management (IM), 2011.

[16] T. Wood, G. Tarasuk-Levin, P. Shenoy, P. Desnoyers, E. Cecchet, M.D. Corner, Memory buddies: exploiting page sharing for smart colocation in virtualized data centers, in: Proc. of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2009.

[17] M. Wang, X. Meng, L. Zhang, Consolidating virtual machines with dynamic bandwidth demand in data centers, in: Proc. of the IEEE INFOCOM 2011 MINI-CONFERENCE, 2011.
[18] M. Korupolu, A. Singh, B. Bamba, Coupled placement in modern data centers, in: Proc. of the IEEE International Parallel and Distributed Processing Symposium, IPDPS, 2009.
[19] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, E. Silvera, A stable network-aware VM placement for Cloud systems, in: Proc. of the IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid, 2012.
[20] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, ACM Communications 51 (1) (2008) 107–113.
[21] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: Proc. of the ACM SIGCOMM 2008 Conference on Data Communication, SIGCOMM'08, 2008.
[22] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, S. Sengupta, VL2: a scalable and flexible data center network, in: Proc. of the ACM SIGCOMM 2009 Conference on Data Communication, SIGCOMM'09, 2009.
[23] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proc. of the 2nd Symposium on Networked Systems Design and Implementation, 2005.
[24] OpenStack Nova: http://nova.openstack.org/, 2011.
[25] OpenStack Swift: http://swift.openstack.org/, 2011.
[26] Amazon Elastic: Compute Cloud (Amazon EC2): http://aws.amazon.com/ec2/, 2011.
[27] Amazon Simple Storage Service (Amazon S3): http://aws.amazon.com/s3/, 2011.
[28] The virtualization API (Libvirt): http://libvirt.org/, 2011.
[29] Paravirtualized drivers for kvm/Linux (Virtio): http://www.linux-kvm.org/page/Virtio, 2011.
[30] ab—Apache HTTP Server benchmarking tool: http://httpd.apache.org/docs/2.0/programs/ab.html, 2011.
[31] Iperf: http://sourceforge.net/projects/iperf/, 2011.

Antonio Corradi is a full professor of computer engineering at the University of Bologna, Italy. He graduated from the University of Bologna and received a MS in electrical engineering from Cornell University, USA. His research interests include distributed and parallel systems and solutions, middleware for pervasive and heterogeneous computing, infrastructure support for context-aware multimodal services, network management, mobile agent platforms, and Cloud computing architectures. He is a member of IEEE, ACM, and AICA.

Mario Fanelli received his Ph.D. degree in Computer Science Engineering in May 2012, from the University of Bologna, Italy. From the same university, he received both the Bachelor and the Master degree in Computer Engineering, cum laude, respectively in the 2006 and in the 2008. Now, he is a Post-Doc Research Fellow at the University of Bologna, working on the integration of mobile systems and Cloud solutions for the realization of large-scale context-aware settings. His research interests include distributed systems, context-aware services and middleware for mobile systems, and Cloud computing architectures. He is an IEEE student member.

Luca Foschini is a Research Fellow at the University of Bologna, Italy. In 2007, he received a Ph.D. degree in computer science engineering from the University of Bologna. His research interests include distributed systems and management of Cloud systems, solutions for pervasive computing environments, system and service management, and context-aware services and adaptive multimedia. He is a member of IEEE and ACM.